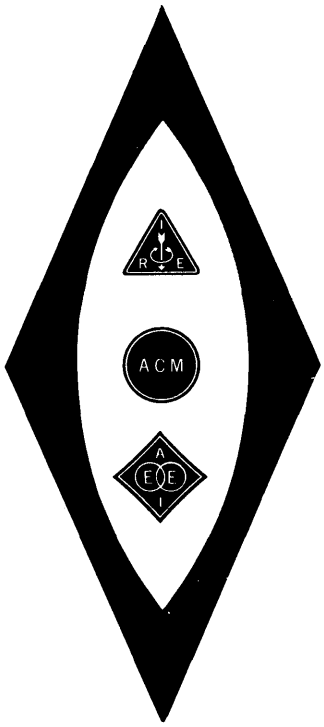
Proceedings of the

WESTERN JOINT COMPUTER CONFERENCE

May 3-5, 1960 San Francisco, Calif.



Sponsors:

THE INSTITUTE OF RADIO ENGINEERS

Professional Group on Electronic Computers

THE AMERICAN INSTITUTE OF ELECTRICAL ENGINEERS

Committee on Computing Devices

THE ASSOCIATION FOR COMPUTING MACHINERY

Vol. 17

Price $3.00

PROCEEDINGS OF THE WESTERN JOINT COMPUTER CONFERENCE

PAPERS PRESENTED AT
THE JOINT IRE-AIEE-ACM COMPUTER CONFERENCE
SAN FRANCISCO, CALIF., MAY 3-5, 1960

Sponsors

THE INSTITUTE OF RADIO ENGINEERS
Professional Group on Electronic Computers

THE AMERICAN INSTITUTE OF ELECTRICAL ENGINEERS
Committee on Computing Devices

THE ASSOCIATION FOR COMPUTING MACHINERY

Published by

WESTERN JOINT COMPUTER CONFERENCE

ADDITIONAL COPIES

Additional copies may be purchased from the following sponsoring societies at $3.00 per copy. Checks should be made payable to any one of the following societies:

INSTITUTE OF RADIO ENGINEERS
1 East 79th Street, New York 21, N. Y.

AMERICAN INSTITUTE OF ELECTRICAL ENGINEERS
33 West 39th Street, New York 18, N. Y.

ASSOCIATION FOR COMPUTING MACHINERY
2 East 63rd Street, New York 21, N. Y.

Manufactured in the U.S.A. by the National Press, Palo Alto, California

LIST OF EXHIBITORS

Aeronutronic Division of Ford Motor Co.	Newport Beach, California
AMP Inc.	Harrisburg, Pennsylvania
Ampex Corp.	Redwood City, California
Anelex Corp.	Boston, Massachusetts
Automatic Electric Sales Corp.	Northlake, Illinois
Beckman/Berkeley Division	Richmond, California
Bendix Computer Division	Los Angeles, California
Benson-Lehner Corp.	Santa Monica, California
Bryant Computer Products Division	Springfield, Vermont
Burroughs Corp., Electrodata Division	Pasadena, California
C-E-I-R, Inc.	Arlington, Virginia
C. P. Clare & Co.	Chicago, Illinois
Clary Corp.	San Gabriel, California
Computer Control Co.	Los Angeles, California
Computer Systems, Inc.	New York, New York
Computronics, Inc.	Denver, Colorado
Consolidated Electrodynamics Corp.	Pasadena, California
Control Data Corp.	Minneapolis, Minnesota
Cook Elec. Co., Data-Stor. Division	Skokie, Illinois
Datamation, Magazine of	Los Angeles, California
Di/An Controls, Inc.	Boston, Massachusetts
Digital Equipment Corp.	Maynard, Massachusetts
Digitronics Corp.	Albertson, L. I., New York
Electronic Associates, Inc.	Long Branch, New Jersey
Electronic Engineering Co. of California	Santa Ana, California
Fairchild Semiconductor Corp.	Mountain View, California
Ferranti Electric, Inc.	Hempstead, New York
Friden, Inc.	San Leandro, California
Industrial Tubular Equipment Co.	North Hollywood, California
International Business Machines Corp.	New York, New York
Laboratory for Electronics, Inc.	Boston, Massachusetts
Librascope, Inc.	Glendale, California
Minneapolis-Honeywell Regulator Co., Datamatic Division	Newton Highlands, Mass.
F. L. Moseley Co.	Pasadena, California
Moxon Electronics Corp.	Beverly Hills, California
The National Cash Register Co., Electronics Division	Hawthorne, California
North American Aviation, Inc.	El Segundo, California
Pacific Telephone Co.	San Francisco, California
Packard-Bell Computer Corp.	Los Angeles, California
Philco Corp.	Philadelphia, Pennsylvania
Potter Instrument Co., Inc.	Plainview, L. I., New York
Raytheon Mfg. Corp.	Waltham, Massachusetts
Remington-Rand	New York, New York
Royal McBee Corp.	Port Chester, New York
Stromberg-Carlson Co.	San Diego, California
System Dev. Corp.	Santa Monica, California
Tally Register Corp.	Seattle, Washington
Telemeter Magnetics, Inc.	Culver City, California
Teletype Corp.	Chicago, Illinois
Thompson-Ramo-Wooldridge, Inc.	Canoga, Park, California
John Wiley & Sons, Inc.	New York, New York
Wright Engineering Co.	Pasadena, California

NATIONAL JOINT COMPUTER COMMITTEE

Chairman

H. H. Goode
Bendix Systems Division
Ann Arbor, Michigan

Vice-Chairman

P. Armer
RAND Corporation
Santa Monica, California

Secretary-Treasurer

Miss M. R. Fox
National Bureau of Standards
Department of Commerce
Washington 25, D. C.

IRE Representatives

H. H. Goode
Bendix Systems Division
Ann Arbor, Michigan

F. E. Heart
Lincoln Laboratories
Lexington, Massachusetts

W. H. Ware
RAND Corporation
Santa Monica, California

W. Buchholz
IBM Corporation
Poughkeepsie, New York

AIEE Representatives

R. A. Imm
IBM Corporation
Rochester, Minnesota

R. R. Johnson
General Electric Co.
Phoenix, Arizona

C. A. R. Kagan
Engineering Research Center
Western Electric Co., Inc.
Princeton, New Jersey

S. Rogers
c/o Convair, Mail Zone 6-156
San Diego, California

ACM Representatives

P. Armer
RAND Corporation
Santa Monica, California

W. M. Carlson
E. I. Du Pont de Nemours
Wilmington 98, Delaware

J. D. Madden
System Development Corp.
Santa Monica, California

H. R. J. Grosch
Corp. for Economic and
Industry Research, Inc.
Los Angeles 46, California

Ex-Officio Representatives

R. W. Hamming(ACM)
Bell Telephone Laboratories
Murray Hill, New Jersey

M. Rubinoff (AIEE)
Philco Corporation
Government and Industrial Div.
Philadelphia 44, Pennsylvania

R. O. Endres (IRE)
Rese Engineering Inc.
Philadelphia, Pennsylvania

Headquarters Representatives

J. Moshman
Corp. for Economic and
Industry Research, Inc.
Arlington 2, Virginia

R. S. Gardner
American Institute of Electrical Engineers
New York 18, New York

L. G. Cumming
The Institute of Radio Engrs.
New York 21, New York

WESTERN JOINT COMPUTER CONFERENCE COMMITTEE

General Chairman	R. M. Bennett, Jr., IBM Corp., San Jose, California
Vice-Chairman	G. A. Barnard, Ampex Corp., Redwood City, California
Secretary-Treasurer	J. P. Fernandez, Chairman, IBM Corp., San Jose, California D. E. Eliezer, IBM Corp., San Jose, California
Technical Program	H. M. Zeidler, Chairman, Stanford Research Institute, Menlo Park, California J. E. Sherman, Lockheed M. S. D., Sunnyvale, California
Publications	D. D. Willard, Chairman, IBM Corp., San Jose, California E. T. Lincoln, IBM Corp., San Jose, California
Publicity	C. Elkind, Chairman, Stanford Research Institute, Menlo Park, California N. S. Jones, Friden, Inc., San Leandro, California
Exhibits	H. K. Farrar, Chairman, Pacific Telephone Co., San Francisco, California J. W. Ball, Pacific Telephone Co., San Francisco, California
Registration	H. N. Wells, Chairman, General Electric Co., General Electric Computer Lab, Palo Alto, California J. E. Stokdyk, Hewlett-Packard Co., Palo Alto, California
Printing and Mailing	R. A. Isaacs, Chairman, Philco Corp., Palo Alto, California R. E. Wye, Philco Corp., Palo Alto, California
Women's Activities	Miss Mary Fraser, Chairman, IBM Corp., San Jose, California Mrs. Eleanor Schmidt, IBM Corp., San Jose, California
Local Arrangements	G. E. Morrison, Smith-Corona Marchant, Inc., Oakland, Calif.

TABLE OF CONTENTS

**Participation by Simulation Councils, Inc.

TABLE OF CONTENTS, continued

**Participation by Simulation Councils, Inc.

FOREWORD

The 1960 Western Joint Computer Conference met in San Francisco for the third time. Interest in these conferences has been increasing steadily along with the impressive growth of the computer industry in the San Francisco Bay area. Manufacturing representation in the form of exhibits was the greatest in the history of the Western Joint Computer Conference.

In keeping with the theme of the 1960 Conference, "The Challenge of the Next Decade," the papers presented in the Proceedings were selected on the basis of trends in techniques and applications rather than descriptions of existing or about-to-be announced equipment. The philosophy of this year's conference committee was that of "taking stock" and attempting to "look ahead" to determine which way this dynamic industry is going. We have essentially reached a plateau where technology has given us a feel for the tremendous impact that this industry will have upon our everyday life. It seems desirable, occasionally, to pause and evaluate and, indeed, this in itself can justify the value of a conference.

This publication contains the papers presented at the 1960 Western Joint Computer Conference, and was available at the time of the Conference.

R. M. Bennett
General Chairman
1960 Western Joint Computer Conference

THE HISTORICAL DEVELOPMENT, AND PREDICTED STATE-OF-THE-ART OF THE GENERAL-PURPOSE DIGITAL COMPUTER

By Charles P. Bourne and Donald F. Ford
Computer Techniques Laboratory
Stanford Research Institute
Menlo Park, California

Summary

Some of the important characteristics of all the general-purpose digital computers that have ever been built, or are in the process of being built, have been collected together in order to show the changes in performance and characteristics during the passage of time. These collected data, as well as information regarding recent development work, have been used to extrapolate the characteristics and performance figures into the 1960-1965 era. The report considers such characteristics as add and multiply times, memory characteristics, pulse repetition rates, and internal system parameters.

The collected data seems to suggest that the majority of the computers developed between now and 1965 will show very little change in performance from that which was obtained during the last five years. However, a few research machines will definitely advance the technology, possibly as much as one order of magnitude for some of the characteristics. In appendix B there is a listing of approximately 300 different computers, in an attempt to provide an initial directory of the world's computers.

Introduction

This paper presents the results of a study that was conducted to answer the question, "What can be learned from a historical study of the development of general-purpose digital computers?" It was of interest to re-trace, in summary fashion, the development of the digital computer. It was also of interest to see if the development of any of the machine characteristics followed a pattern which would allow an extrapolation into the future. The study was restricted to the characteristics which describe the memory and central processor or arithmetic units, and did not consider input-output features.

The characteristics were examined from the viewpoint of a machine user, and not a machine designer or component specialist. That is, an attempt was made to describe the operational performance which the computer system provides the user, instead of concentrating on the details of the manner in which the logic is accomplished. For example, a comparison of typical execution times was studied, instead of looking at the switching times for the individual circuits. In a few instances, a study was made of some of the more hardware-oriented characteristics such as the pulse repetition rate or "clock" rate, and the type of high-speed memory component used.

The machine characteristics were obtained from several summary publications[1,2], as well as numerous journal articles and the literature of manufacturers. The data for approximately 180 of the machines was checked by letters of verification from the equipment designers or manufacturers.

The machine characteristics which were examined are listed below. The definitions for these characteristics are included in Appendix A.

Number of addresses per instruction
Number of index registers
Number of decimal digits per instruction
Number of decimal digits per data word
Number of binary bits per instruction
Number of binary bits per data word
Internal clock rate
Fixed point add time
Fixed point multiply time
Floating point add time
Floating point multiply time
Number of memory speed levels
Types of high speed memory
High speed memory size
High speed memory access time
Total internal memory size

Summary Comparisons of Computers to Date

Each of the machine characteristics was plotted against a horizontal calendar scale in order to observe the changes and rates of change for the parameters. A single point was plotted for each different machine type to show the earliest date that the system performed with the described characteristics. Points were plotted for all the available data, however, in many cases, various parameters were not described in the literature. For this reason, all the plots do not contain the same number of points.

The actual growth curve to show the rate at which new computer models have been developed is shown in Fig. 1. Fig. 2 shows the cumulative number of different computers which have been developed. In addition to the computers which are shown on Fig. 1 and 2, approximately 65 machines were not plotted, since an accurate operational date could not be determined. It can be seen that new computer models are being developed in continually increasing numbers, and that this trend will probably continue into the future. The increasing number of new computer models is due to several factors, but it would appear that the major reasons are those which are listed below:

1. Parallel efforts by commercial organizations acting in the spirit of free enterprise, with each organization competing for a share of the market with its own particular model. This is fostered by the increasing market for computers.

2. Continued marketing pressures to improve the performance characteristics of a particular computer model, or to produce more competitive equipment.

3. Efforts by universities and some industrial concerns to build their own computers in order to develop a technical competence in this field, or to obtain an inexpensive computing facility.

4. The emergence of a computer industry in areas which were late in starting in this field. For example, Japan built no digital computers before 1953; since that time however, she has developed 29 different computers, and shows every indication of developing more. The European countries are also developing rapidly as computer manufacturers.

Internal Characteristics

There has been very little uniformity or indication of trends for such internal characteristics as the number of addresses per instruction and the number of characters per data word or instruction. There have been some slight trends noticed in the number of index registers and the internal clock rate.

1. Number of Addresses per Instruction - Figure 3 shows that the single-address instruction is used more than any other type, although a large number of systems have used two-and-three-address systems. The four-address systems are definitely a minority; however, there is one instance of a five-address system. Future computers will probably follow the same pattern, with predominantly single-address format.

2. Number of Index Registers - Figure 4 shows that there is a definite trend toward providing index registers (or "B-boxes") in increasing numbers for each computer. The use of a single index register is first noted in 1951. The first system with multiple index registers is noted in 1954. From that time on, systems became available with a greater number of registers, and in some cases this went as high as 64, 99, and 1024 index registers per machine. A large percentage of the future computers will probably have at least one index register. Index registers have proven to be effective for applications in both business and scientific computations, and will probably appear more frequently on both types of systems.

3. Decimal Digits per Instruction - Figure 5 shows the data for the non-binary machines, to indicate the number of characters (decimal, octal, or alphabetic) per instruction. There does not appear to be any significant degree of uniformity, although more systems use 10 digits than any other number. Future computers will probably show the same large variety and lack of uniformity.

4. Decimal Digits per Data Word - Figure 6 shows a large spread of values running from 4 to 24 digits. However, data words with 10, 11, or 12 digits appear to be the predominant choice. In 1953 the concept of a variable-length data word was introduced, and several systems utilized this feature after that. Future computers will probably continue to show a variety of values, but may use the 10, 11, or 12 digits more often than any other choice. There will probably be an increasing number of machines with variable-length data words.

5. Binary Bits per Instruction - Figure 7 shows the binary bits per instruction ranging from 4 to 68. There is a decided lack of uniformity, and no strong tendency toward a particular value. Future computers will undoubtedly follow the same large variety and lack of uniformity.

6. Binary Bits per Data Word - Figure 8 shows the binary bits per data word ranging from 4 to 72. However, the majority of the systems have ranged between 30 and 50 bits per data word. Future computers will probably follow the same large variety and lack of uniformity.

7. Internal Clock Rate - Figure 9 shows the internal clock rate or pulse repetition rate. This is not a very good measure of the speed or power of a particular computer. This is because the great variety of ways in which the logic can be implemented (serial, parallel, and various combinations of serial-parallel) can provide a large range of effective operating speeds. There are several examples of machines with nearly equal arithmetic speeds in spite of the fact that one of the machines has a clock rate which is five times slower than the clock rate of the other machine. Conversely, it might be noted that all of the IBM-700 series machines from 1953-1957 used a 1.0 Mc clock, even though there were marked differences in execution times. The clock rate data was included here to give an indication of the speeds of operation of the internal circuits. The fastest clock rate (10.0 Mc) is currently credited to the IBM-STRETCH, or Los Alamos computer, although hardware has been developed[3,4] which operates at speeds up to 500 Mc. The data shows, among other things, that people like round numbers, as indicated by the large number of 100 Kc and 1 Mc systems. The clock rates ranged from 200 cycles per minute to 10.0 Mc.

There are a moderate number of asynchronous systems (at least 28) which date back to the earliest machines. Because the asynchronous systems offer an inherent speed advantage over the clocked systems, they will probably find increasing use as better design techniques become available. There will be an increasing number of systems which use clock rates of 1 Mc or higher. There are already at least 47 systems which operate with pulse rates of 1 Mc or higher. These internal speeds can now be achieved fairly easily with current transistor and diode circuitry. However, because of the variety of possible applications, there will continue to be a large number of new systems with moderate internal speeds (100 - 500 Kc).

Arithmetic Speeds

As defined in the Appendix, the execution times for the add and multiply operations are the effective operating times which a programmer would use in estimating the running time of a particular program. It can be seen from the pertinent figures (Fig. 10, 11, 12, 13) that there continues to be a wide range of execution times for all computers. This wide range of machine speeds, and characteristics, does not solely reflect the state-of-the-art or its trends. This range of values more accurately reflects the policy decisions and compromises which were made by each computer builder in his attempt to aim for a particular part of the commercial market, or produce (at minimum cost) a technical staff or computing facility. It is probably reasonable to state that not every builder tried to advance the state-of-the-art. Because of the fact that the collected data represented a heterogenous collection of intents and compromises, it was felt that it would be misleading to represent this data in some mathematical notation such as a yearly average or polynomial approximation.

The computers which have served to extend the state-of-the-art were primarily research and development machines, and were not designed primarily for commercial exploitation. The distinction in performance between systems which were built for research and systems which were built for commercial exploitation is shown in Fig. 14. This figure shows the fixed point multiply times for the machines in these two categories, and indicates that the state-of-the-art improvements were furnished entirely by the research machines. However, the commercial machines usually caught up with the fastest research machines in a matter of approximately five years.

Many builders have claimed the title of "world's most powerful computer" for their particular machine. It is impossible at this date to define accurately and unambiguously "computer-power". However, in an attempt to find out how long a machine could expect to retain this title, it was assumed that "computer-power" was proportional to the multiply time. With this assumption, it was a simple matter to determine the ten most powerful computers in each year from 1944 to the present. It was indeed an interesting observation to note that, on the average, a computer did not remain in the "top-ten" for more than three years.

For many reasons, there will be a continued effort to develop computers with faster execution speeds. However, it appears that faster speeds will not be achieved by extensions of the current hardware practices and techniques. It appears that new components and approaches must be developed if these speeds are to be attained. Meagher[5] sums this up very nicely in the following paragraph.

> "Let us first examine the need for new techniques which has resulted from our desire for higher speed. The existing circuits, with separate resistors, diodes, capacitors, and transistors, have a physical size which requires at least one cubic inch for a logical element. One such circuit has within itself a loop which constitutes an inductance with a shunt capacitance in the switching element. This LC circuit exhibits resonance. If the loop is about one-half inch in diameter, the inductance would be about 0.06 microhenry, and further, if the capacitance is 5 micro-micro-farad (both reasonable minimum values), the resulting resonant frequency would be about 300 Mc. Clearly it would be difficult to operate this circuit at an information frequency of more than about one-fifth the resonant frequency, or in other words, 60 Mc. We are already close to this frequency in some present computer circuits. Thus, faster circuits require either smaller size for "lumped-constant" techniques, or, alternatively, "distributed-constant" techniques. Low temperature circuit elements offer one possibility for extremely small size. Micro-wave techniques, the subject for the Symposium, offer the possibility of the distributed-constant approach."

To date, all of the circuits for storage and logic in the operating computers have employed a frequency band for pulse rates which starts at or near zero and extends to some upper limit. The upper limit for this base-band type of system appears to be about 50 Mc (see reference[8]). It would appear that some new techniques would have to be devised to achieve information pulse rates of greater than 50 or 100 Mc.

Microwave circuits for storage and logic functions have been demonstrated by Sterzer[15] which operate at pulse rates of 100 Mc and a carrier frequency of 2,000 Mc; and by Ortel[4] which operate with pulse rates of 500 Mc and a carrier frequency of 11,000 Mc. Ortel demonstrated a serial multiplier operating at a clock frequency of 160 Mc which multiplied two 8-digit binary numbers to form a 16-digit product in 1.6 microseconds. It was stated that it would be feasible to use the same microwave circuit with a clock frequency of 640 Mc to obtain a multiplication

time of 0.4 microseconds.

Billing[6] and Rudiger have stated that the use of the nonlinear capacitance of semiconductor diodes in parametron circuits also appear promising for high frequency operation. Experimental work has been conducted at frequencies up to 450 Mc, and theoretical studies seem to indicate that frequencies in the neighborhood of 30 Kmc should be possible.

Aside from economic problems, one of the major technical problems in achieving very high circuit speeds is that the time allocated to switch a signal through the circuit approaches the propagation time for any electrical wave. This is stated very well by Leas[7] in the following paragraphs.

> "To improve substantially upon the present computing speeds, manipulative elements for the basic functions of gating and storing binary signals are required which have extremely fast physical response, reckoned in nanoseconds (10^{-9}) rather than microseconds (10^{-6}). Furthermore, because large numbers of such elements must be used in systems sufficiently comprehensive to make significant use of their speed, these elements must be physically small in order that the machine itself be small enough not to cause prohibitive delays due to the finite propogation velocity of electrical signals.
>
> "For example, during one nanosecond electrical signals in free space travel one foot (30 cm), and in most solid materials only 6 to 8 inches (15 to 20 cm). This means that two circuits must be only fractions of an inch apart if the delay between them is to be negligible. Furthermore, the whole computer size is limited since the performance of an instruction requires information to reach and be returned from all parts of the computer. To obtain memory cycles of 10 milli-microseconds, the computer can not be more than 18 to 24 inches (45 to 60 cm) in diameter."

In addition to the problem of physical size, Leas also mentions the problem of power consumption of the individual logic elements.

> "This limitation in size also sets limitations on the power consumption of individual elements. For a computer of minimum size, at least 5000 logical elements and 10,000 memory elements will be required, implying power consumptions below 500 mw for each logic element, and 50 mw for each memory element, if normal forced air cooling is used. It is highly desirable that these figures be reduced to increase the capability of the computer. Satisfactory values would be 15,000 logic elements, each taking 150 mw, and 2.5 x 10^5 memory elements, each taking 2 mw."

One additional drawback to improving the execution times is the lack of a very high speed random access storage. Unless some "look-ahead" or sophisticated control techniques are used, the lack of a high speed memory will prevent the effective execution speeds from being achieved. Temporary solutions may be obtained by using a small amount of very fast memory in a multi-level memory. Further impediments to widespread use of high speed operation are the lack of inexpensive instrumentation equipment, the scarcity of technical and professional manpower with training in this area, and the lack of production and testing experience.

1. Fixed Point Add Times - The fastest fixed point add time (0.57 microsec) is currently credited to the University of Illinois ILLIAC-2. It might also be noted that after 1950 all of the computers that claimed the title of "world's fastest adder" were binary machines. A computer with an add time of less than 0.1 microseconds will probably not be built before 1965, even though the hardware techniques are already available to build adders to operate at even faster speeds. Logic circuits and adders have been operated at clock rates of 50 Mc, and even up to 500 Mc using radio frequency carrier techniques[9], but the components and circuitry are currently so expensive and bulky that they will probably not be used as part of a complete computer system.

2. Fixed Point Multiply Times - The fastest multiply time (1.4 microseconds) is credited to the IBM-STRETCH, or Los Alamos machine. As with the add times, the fastest machines are predominantly binary machines. A computer with a multiply time of less than 0.5 microseconds will probably not be built before 1965. Prototype multipliers have operated at 50 and 160 Mc clock rates[8,4], but these devices will probably not be incorporated into a complete computer system for some time.

3. Floating Point Add Times - Figures 11 and 13 illustrate the systems which are known to utilize floating point hardware (binary or BCD). After 1958, more and more systems incorporated floating point arithmetic circuitry, and the execution speeds increased in the same manner as for the fixed point operations. The fastest floating point add time (0.8 microsec) is currently credited to the IBM-STRETCH machine. Floating point arithmetic has proven to be a useful feature both for scientific and business computers. There will be an increasing need for floating point hardware in business computers as more users employ mathematical tools and operations research techniques for business applications. The instances where a single machine is used for both business and

scientific computations will also press the need for floating point hardware in any general-purpose machine at least as an optional feature. For future computers, it is expected that a greater percentage will incorporate floating point hardware and the speeds will generally increase, but a computer with a floating point add time of less than 0.1 microseconds will probably not be built before 1965.

4. Floating Point Multiply Times - Figure 13 shows the comparative floating point multiply speeds. The wide pattern with some increasing speeds is generally the same as for all the other execution speeds. The fastest floating point multiply time (1.4 microsec) is credited to the IBM-STRETCH machine. A computer with a floating point multiply time of less than 0.5 microseconds will probably not be built before 1965.

Internal Memory

1. Number of Memory Speed Levels - In many computers the internal working memory consists of a combination of different types of memory devices with different speeds of access. For example, there are machine designs in which a small amount of core storage is backed up by additional drum storage, or a few high speed drum bands are backed up by a large number of slower bands. In many cases, the slower storage media is not directly addressable, and provisions are made to transfer data to and from the high speed storage in some modular quantity such as 20 or 100-word blocks. A homogenous, directly-addressable memory is desirable from a programmer's viewpoint, since the programs which use multi-level storage are necessarily more complex and less efficient than programs which utilize single-level storage. However, for large memories, the multi-level storage does provide a reasonable compromise between the expense and performance of high-speed and low-speed systems. Figure 15 illustrates the number of systems which have used 1-, 2-, 3-, or 4-level memories. Single level systems are more numerous than any other type. For future machines, it is expected that the single level machines will continue to dominate the scene, but there will still be many multi-level systems.

2. Types of High Speed Memory - Figure 16 describes the types of devices which have been used for the high speed memory. A definite pattern of emergence and de-emphasis can be noted for most of the devices. The various types of high speed memory--relay, vacuum tube, delay line, drum, cathode ray tube, magnetic core, disk, and diode-capacitor--made their operational appearance in that order.

   It would appear that the relay, vacuum tube, delay line, and cathode ray tube memories are on the way out. From now until 1965, the drum and core memories will be the devices which will appear most frequently as the high speed memory devices. There will be an increased application of thin magnetic films[10] for memory applications in the immediate future and some systems with large (1000-5000 words) thin film memories should be operating before 1965. A small section (32 words) of film memory has already been used with the Lincoln Lab TX-2 computer[11, 12, 13], but it has been used as part of the control circuitry and not as part of the addressable memory.

3. High Speed Memory Size - In order to examine and compare the memory sizes of the single-level along with the multi-level memory systems, it was decided to look at the size of the highest speed memory as well as the size of the largest possible total internal memory size. Figures 17 and 18 illustrate the size of the high speed memory, which is always less than or equal to the total internal memory size. Figure 17 distinguishes between binary and non-binary memories, and Figure 18 distinguishes between the types of device used as the memory element. The maximum or extreme size of high speed memory is generally increasing. However, if the top and bottom extreme points are removed from either of these figures, the range of values actually shows very little change over the last decade. Another feature which is changing (see Fig. 18) is the type of device used for the high speed memory. This is in agreement with the data from the preceding section on types of high speed memory. Future computers will probably fall in the same range described in the figure, but there may be an occasional system which will show a larger high speed memory size, perhaps to 20 million bits. This figure is a little misleading since this size of memory could almost be achieved by a single magnetic drum. However, there will be an increasing number of systems with large, directly-addressable high speed memories.

4. High Speed Memory Access Time - Figure 19 shows the high speed memory access times, and distinguishes between the types of devices used as the memory element. This figure shows that with a few exceptions, the state-of-the-art of memory access time has not shown as rapid a development as some of the other operating speeds. Access times on the order of 10 microseconds have been achieved with computers since 1950. The credit for the fastest memory access time currently belongs to the University of Illinois ILLIAC-2, which achieves a 0.2 microsecond access time with a small section of high speed transistor flip-flop buffer memory. Most of the fast access times have been achieved with magnetic cores. The probable limit[12] to core switching speeds is 0.1 microseconds, and although the probably limit to thin film switching speeds is 0.01 microseconds, future computer systems

will probably not achieve an access time of less than 0.05 microseconds before 1965. The higher memory speeds in the immediate future will probably be obtained with thin magnetic films, transistor flip-flops, and diode-capacitor memories. Tunnel diodes have also been mentioned[14] as devices for high speed access (10^{-8} to 10^{-7} sec), but they will probably not be available in an operational computer before 1965.

5. Total Memory Size - Figure 20 shows the total amount of internal memory which is available for each computer. There is a large spread of values, and a general increase in the maximum sizes, but the average size of the memory has remained about the same for the last ten years. Future systems will probably fall in the same range described by the figure. There will be an increasing use of large, random-access devices such as the disk files or drum files. A computer system which utilizes photoscopic storage for high density permanent storage (such as the glass disk memory developed for the Air Force machine translation projects) will probably be demonstrated before 1965.

Discussion

An examination of many of the computer characteristics failed to disclose any significant trends. The number of bits or digits per instruction or data word is an example of this lack of definite trend. For many characteristics, it appears that the majority of machines have utilized approximately the same state-of-the-art, or have copied each other to a large extent, and that a relatively small group of machines have provided the significant technical advances. It is also interesting to note that very few of the machines which were developed for commercial exploitation really provided any major advances in the characteristics studied in this paper. It would appear that if there is an interest only in predicting the extensions of the state-of-the-art, then this can probably be accomplished primarily by looking at the small number of developmental and feasibility machines in addition to studying the progress which is being reported on circuit and component development.

The collected data seems to suggest that the majority of the computers developed between now and 1965 will show very little change in performance from that which was obtained during the last five years. However, a few research machines will definitely advance the technology, possibly as much as one order of magnitude for some of the characteristics, such as multiply times, memory sizes and memory access times.

References

1. "A Survey of Automatic Digital Computers," 1953, Office of Naval Research, Washington, D.C.

2. "A Second Survey of Domestic Electronic Digital Computing Systems," M. H. Weik, June 1957, Ballistic Research Lab Report No. 1010, Aberdeen Proving Ground, Md. (also distributed by OTS as Report No. PB-111996R)

3. H. S. Sommers, Jr., "Tunnel Diodes as High-Frequency Devices," Proc. I.R.E., Vol. 47, No. 7, pp. 1201-1206, July 1959.

4. W. C. G. Ortel, "Nanosecond Logic by Amplitude Modulation at X Bands," I.R.E. Transactions on Electronic Computers, Vol. EC-8, No. 3, pp. 265-271, September 1959.

5. R. E. Meagher, "History and Introduction--Microwave Techniques for Computers," I.R.E. Transactions on Electronic Computers, Vol. EC-8, No. 3, pp. 263-265, September 1959 (includes 21 references).

6. H. E. Billing, A. D. Rudiger, "The Possibility of Speeding up Computers Parametrons," International Conference on Info. Proc., Paris, June 1959.

7. J. W. Leas, "Microwave Solid-State Techniques for High Speed Computers," International Conference on Info. Proc., Paris, June 1959.

8. R. M. Walker, D. E. Rosenheim, P. A. Lewis, A. G. Anderson, "An Experimental 50-Megacycle Arithmetic Unit," I.B.M. Journal of Research and Development, Vol. 1, No. 3, pp. 257-278, July 1957.

9. D. J. Blattner, F. Sterger, "Fast Microwave Logic Circuits," I.R.E. National Convention, New York, 1959, Part 4, pp. 252-258. (This paper was also published in the I.R.E. Transactions on Electronic Computers, Vol. EC-8, No. 3, pp. 297-301, September 1959.)

10. W. E. Proebster, S. Methfessel, C. O. Kinberg, "Thin Magnetic Films", International Conference on Info. Proc., Paris, June 1959. (22 references)

11. A. J. Kolk, J. T. Doherty, "Thin Magnetic Films for Computer Applications," Datamation, pp. 8-12, September 1959. (includes a list of 26 references)

12. D. O. Smith, "Thin Magnetic Films for Digital Computer Memories," Electronics, pp. 44-45, June 26, 1959. (includes a list of 9 references)

13. J. I. Raffel, D. O. Smith, "A Computer Memory Using Magnetic Films," International Conference on Info. Proc., Paris, June 1959. (19 references)

14. J. A. Rajchman, "Solid State Microwave High Speed Computers," Eastern Joint Computer Conference, Boston, December 1959.

15. F. Sterzer, "Microwave Parametric Subharmonic Oscillators for Digital Computing," Proc. I.R.E., Vol. 47, No. 8, pp. 1317-1324, August 1959.

Appendix A

Definitions

Computer

This is the name of the computer model being described. If significant changes have been made in the original model, then the new computer configuration is described as an additional entry (e.g., IBM 705 Mod. 1, IBM 705 Mod. 2, IBM 705 Mod. 3). Whenever possible, the manufacturer's or builder's name precedes the computer's common name.

Operational Date

This is the date on which the computer model was operating as a complete system. This may not be a clearly defined date, but represents an approximation to the date at which a system with the described parameters was shown to be an actuality.

Number of Addresses per Instruction

This indicates the number of addresses which are included in a single instruction. This includes addresses of the next instruction, as well as addresses for the data being operated upon, and in some cases the addresses of addressable registers.

Binary (decimal) Digits Per Instruction (data word)

This indicates the number of functional digits per instruction and/or data word. The sign digit is included in this count, but not parity digit. If the data word can have 2 or 3 different sizes, the maximum size is considered.

Number of Index Registers

This is the number of special ("B-box") registers which can perform automatic address modification to the command which is currently being interpreted and executed.

Clock Rate

This is the internal pulse repetition rate or clock rate of the computer.

Add Time

This is the time required to execute an add command under optimum programming conditions. The time required for memory accesses are included in this number. Ten-digit operands were chosen for those machines which used a variable-length data word. If a range of speeds was given with no other explanation, an arithmetic mean of the range was chosen as the add time. It was assumed that no indirect addressing or address modification occurred at the time that the command was being executed.

Multiply Time

This is the time required to execute a multiply command under optimum programming conditions. The time required for memory accesses are included in this number. Ten-digit operands were chosen for those machines which used a variable-length data word. A word of "fives" was chosen for those cases where the execution time was a function of the value of the operand digits. If a range of speeds was given with no other explanation, an arithmetic mean of the range was chosen as the multiply time. It was assumed that no indirect addressing or address modification occurred at the time that the command was being executed.

Floating Point Add (Multiply) Time

This is the time required to execute a floating point command with floating point hardware. The execution times are not included for the machines which have to perform floating point arithmetic by subroutine. The general assumptions listed for the add and multiply times are also applicable here. If the floating point times are the same as the fixed point times, it is probably because the machine is basically a floating point machine and an entry was made in both the fixed and floating columns.

Number of Speed Levels of Memory

This is the number of different speeds of internal memory which are present in a computer. This number does not include auxiliary storage (magnetic tapes, drums, etc.) which is not essential to the operation of the basic machine. The amount and access time of the highest speed memory of each machine are described in separate tabulations.

High Speed Access Time

This is the time required to read a word (10 BCD characters if not otherwise specified) from the highest speed memory of the machine under optimum programming conditions. If a range of access times is quoted with no further explanation, then an arithmetic mean of the access time was used.

Type of High Speed Memory

This is the type of the highest speed memory used for each machine.

Total Memory

This is the total amount of available memory (bits for binary machines, characters for BCD, bi-quinary, or alpha-numeric machines) for all levels of directly-addressable memory, exclusive of auxiliary storage (magnetic tapes, drums, etc.)

Size of High Speed Memory

This is the total amount of available memory with the fastest access time.

General-Purpose, Stored-Program Computer

This is a machine which has a stored program (by plugboard, pinboard, or internal memory), and can perform program iterations and modify its own stored commands. (A few machines which do not completely satisfy this requirement are included in this report because of their historical or technological significance.)

APPENDIX B

AIL, INC., MODAC 5014
AIL, INC., MODAC 404
AIL, INC., MODAC 410
AIL, INC., MODAC 414
AIR FORCE CAMBRIDGE ABC
ALWAC 800
ALWAC 3-E
ARB. ELEK. RECH. PERM
ARGONNE LAB. AVIDAC
ARGONNE LAB. GEORGE
ATOM. EN. RES. EST. HARWELL
AUTONETICS RECOMP 1
AUTONETICS RECOMP 2
AUTONETICS VERDAN
BELGIAN BELL IRSIA-FNRS
BELGIAN BELL TEL. BELL
BELL TEL. MARK-22 *MOD-4*
BELL TEL. MODEL-6
BELL TEL. MODEL-5
BELL TEL. LEPRECHAUN
BENDIX G-15
BENDIX G-20
BIRBECK COLLEGE ARC
BIRBECK COLLEGE SEC
BIRBECK COLLEGE APERC
BIRBECK COLLEGE APEXC
BRITISH TAB. HEC-1
BRITISH TAB. HEC-2
BRITISH TAB. HEC-2M
BRITISH TAB. HEC-4 *1201*
BRITISH TAB. 555
BRITISH TAB. HEC 1202
BRITISH TAB. 1400
BROOKHAVEN MERLIN
BRUSH ELECT. ENG. CO., LTD.
BULL GAMMA-3
BULL GAMMA 60
BURROUGHS E101
BURROUGHS 205
BURROUGHS 220
BURROUGHS LAB COMPUTER
BURROUGHS UDEC-2
BURROUGHS UDEC-1
CAL INST. TECHNOL. MINAC
CAMBRIDGE UNIV. EDSAC-1
CAMBRIDGE UNIV. EDSAC-2
CLARY DE-60
COMP. RES. CORP. CADAC-102A
COMP. RES. CORP. CRC-107
COMP. RES. CORP. CADAC-102
CONSOL. ENG. CORP. 36-101
CONTROL DATA CORP. 1604
CONTROL DATA CORP. 160
CORBIN CORP. CORBIN
C.S.I.R.O. MARK-1
DANISH BOARD DASK
DARMSTADT DERA
DIG. EQUIP. CORP. PDP
REF. RES. TEL. EST. COMPUTER
EIDGEN, TECHN, HOCH. R4S
ELEC. LAB. TOKYO MARK-1
ELEC. LAB. TOKYO MARK-2
ELEC. LAB. TOKYO MARK-3
ELEC. LAB. TOKYO MARK-4
ELEC. LAB. TOKYO MUSASINO-1
ELEC. LAB. TOKYO SENAC-1
ELEC. LAB. TOKYO MARK-4A
ELEC. LAB. TOKYO MARK-5
ELLIOTT BROS. NICHOLAS
ELLIOTT BROS. 403
ELLIOTT BROS. 404
ELLIOTT BROS. 402E, F
ELLIOTT BROS. 402
ELLIOTT NRDC 401 MARK-1
E.M.I. ELECT. EMIDEC-1100
E.M.I. ELECT. EMIDEC-2400
ENG. ELEC. CO. DEUCE MK-1
ENG. ELEC. CO. DEUCE MK-2
ENG. ELEC. CO. DEUCE MK-2A
ENG. ELEC. CO. KDP-10
ERA LOGISTICS
FACIT INC. EDB
FERRANTI ARGUS
FERRANTI ATLAS
FERRANTI MANCHESTER MARK-1
FERRANTI MARK 1*
FERRANTI MERCURY (MARK 2)
FERRANTI ORION
FERRANTI PEGASUS-1
FERRANTI PEGASUS-2
FERRANTI SIRIUS
FERRANTI PERSEUS
FLAC
FORD INST. DATAKEEPER-1000
FUJI PHOTO FILM CO. FUJIC
FUJI COMMUN. FACOM-138
FUJI COMMUN. FACOM-1288
FUJI COMMUN. PC-2
FUJI COMMUN. FACOM-212
FUJI COMMUN. FACOM-222
GENERAL ELECTRIC 150
GENERAL ELECTRIC 210
GENERAL ELECTRIC 312
GENERAL ELECTRIC 100
GENERAL ELECTRIC OMIBAC
GENERAL ELECTRIC OARAC
GOETTINGEN G-3
GOETTINGEN G-1
GOETTINGEN G-2
GOETTINGEN G-2
HARVARD MK 2 *DAHLGREEN*
HARVARD MARK-3 *ADEC*
HARVARD MARK-4
HASLER ERMETH
HITACHI LTD. HIPAC-1
HITACHI LTD. HITAC-102
HITACHI LTD. HIPAC-101
HITACHI LTD. HITAC-301
HITACHI LTD. HITAC-1
HOGAN LABS CIRCLE COMP.
HOKUSHIN ELECT. H-1
HONEYWELL 800
HONEYWELL DATAMATIC 1000
HUGHES DIGITAC *AIRB*
HUGHES DIGITAIR *AIRB*
IBM 604
IBM 607
IBM 608
IBM 305 RAMAC
IBM 650 MOD. 1

IBM 650 MOD. 2
IBM 650 MOD. 4
IBM 701 *CRT MEMORY*
IBM 701 *MC MEMORY*
IBM 702 MOD. 1
IBM 702 MOD. 2
IBM 704
IBM 705 MOD. 1
IBM 705 MOD. 2
IBM 705 MOD. 3
IBM 709
IBM 7080
IBM 7070
IBM 7090
IBM 1401
IBM 1620
IBM STRETCH *LOS ALAMOS*
IBM AN/FSQ-7 *SAGE*
IBM AN/FSQ 7A *SAGE*
IBM NORC
IBM DINABOC *AIRBORNE*
IBM COMAR-1 610 PROT
IBM ASEC *HARVARD MK.1*
IBM DATA COORD/RETRIEVER
INFO. SYSTEMS 609
IN. P.M. RECHENAUTOMAT-IPM
JACOBS INSTR. JAINCOMP-A
JACOBS INSTR. JAINCOMP-B
JACOBS INSTR. JAINCOMP-C
JAPAN TEL. & TEL. M-1
KIDRICH INST. COMPUTER
LEEDS & NORTHRUP LN-3000
LEO COMPUTERS LTD. LEO-1
LEO COMPUTERS LTD. LEO-2
LEO COMPUTERS LTD. LEO-2C
LFE DIANA
LIBRASCOPE ASN-24 *AIRB*
LIBRASCOPE LGP-30
LIBRASCOPE LIBRATROL-500
LIBRASCOPE LIBRATROL-1000
LIBRASCOPE RPC-4010
LIBRASCOPE FAA-ARTC COMP.
LIBRASCOPE ASROC COMP.
LIBRASCOPE CP-209
LINCOLN LAB TX-0
LINCOLN LAB MEMORY TEST
LINCOLN LAB TX-2
LINCOLN LAB CG-24
LINCOLN LAB TX-3
LOS ALAMOS MANIAC-1
LOS ALAMOS MANIAC-2
MAGNAVOX MAC-3
MARCHANT MINIAC C,2
MATH CENTRUM ARMAC
MATH CENTRUM ARRA
MATSUSHITA COMM. DM-8001A
MELLON INST. COMPUTER
METR. VICK. METROVICK-950
METR. VICK. A.E.I. 1010
M.I.T. WHIRLWIND-1
MINISTRY OF SUPPLY MOSAIC
MIT WHIRLWIND-1 *MC MOD.*
MONROE MONROBOT-3
MONROE MONROBOT-5
MONROE MONROBOT-6
MONROE MONROBOT-MU
MOORE SCHOOL EDVAC
MOORE SCHOOL ENIAC
MOORE SCHOOL MSAC
NAREC
NATIONAL-ELLIOTT 802
NATIONAL-ELLIOTT 405M
NATIONAL-ELLIOTT 405
NAT. PHYS. LAB. PILOT ACE-1
NAT. PHYS. LAB. PILOT ACE-2
NAT. PHYS. LAB. ACE
NAVAL RES. LAB NAREC
NBS SWAC
NBS SEAC *DL MEMORY*
NBS SEAC *DL & CRT*
NBS DYSEAC
NBS PILOT *PRIMARY COMP.*
NBS PILOT *SECOND. COMP.*
NCR 102D
NCR 304
NCR 102A
NEPA/USAF FAIRCHILD COMP.
NIPPON ELECT. NEAC-1101
NIPPON ELECT. NEAC-1103
NIPPON ELECT. NEAC-2201
NIPPON ELECT. NEAC-1102
NIPPON ELECT. NEAC-2203
NO. AMER. NATPAC
N.V. ELECTROLOGICA X1
OAK RIDGE ORACLE
OKI ELECT. OPC-1
OLIVETTI PISA
ONR RELAY COMPUTER
PACKARD BELL PROCESS CONTROL COMP. 3/60
PENN ST. UNIV. PENNSIAC
PHILCO COMPAC
PHILCO BASIPAC
PHILCO C-1102
PHILCO S-1000 *SOLO*
PHILCO S-2000
POWERS-SAMAS PCC
PRAGUE SAPO
PRINCETON INST. AD. ST. IAS
PTT PTERA
RAND CORP. JOHNNIAC
RAYTHEON RAYCOM
RAYTHEON RAYDAC
RCA 110 *PROCESS CONTROL*
RCA 502
RCA 503
RCA BIZMAC 2
RCA 501
RCA BIZMAC 1
RCA 504
REA CO. READIX
REM-RAND UNIVAC 1102
REM-RAND UNIVAC-1
REM-RAND UNIVAC 1101
REM-RAND BINAC
REM-RAND UNIVAC-2
REM-RAND UNIVAC-3
REM-RAND UNIVAC-1105
REM-RAND FILE COMP. MOD. 0
REM-RAND FILE COMP. MOD. 1
REM-RAND TRANSTEC-2
REM-RAND SS-80
REM-RAND SS-90
REM-RAND ATHENA
REM-RAND UNIVAC 1103

REM-RAND AFCRC MAG. COMP.
REM-RAND X-308 *BOGART*
REM-RAND LARC
REM-RAND M-460 *COUNTESS* 5/58
RICE INSTITUTE COMPUTER
RPC-4000
RW-40 *AN/FSQ-27*
RW-300
RW-400
RW-30 *AIRBORNE*
SAAB CO. SARA
SHIBAURA ELECT. TAC
SHIBAURA ELECT. TOSBAC-3
SIEMENS 2002
SOC. DELECT. AUTO. CAB-500
SOC. DELECT. CAB-2000
SOC. NOUV. DELECT. KL-901
SOC. DELECT. CUBA
SOC. DELECT. AUT. CAB-3000
SOC. DELECT. AUT. CAB-3030
STAND. TEL. STANTEC ZEBRA
STANDARD ELECTRIC STANLEC
STAND. ELEK. LORENZ ER -56
SWEDISH BOARD BARK
SWEDISH BOARD BESK
SYLVANIA MOBIDIC
SYLVANIA UDOFT
TBA-1
TECHNITROL 180
TEL. RES. EST. TRE COMPUTER
TELEFUNKEN TR-4
TOKYO SHIBAURA CO. TAC
TOKYO UNIV. PC-1
TOKYO UNIV. MUSASINO-1
UNDERWOOD ELECOM 100
UNDERWOOD ELECOM 120
UNDERWOOD ELECOM 125
UNDERWOOD ELECOM 200
UNIV. OF CALIF. CALDIC
UNIV. OF LUND SMIL
UNIV. OF TORONTO UTEC
UNIV. OF ILL. ORDVAC
UNIV. OF ILL. ILLIAC-1
UNIV. OF ILL. ILLIAC-2
UNIV. OF MICH. MIDAC
UNIV. OF WISC. WISC
UNIV. OF MICH. MIDSAC
USSR ARAGATZ
USSR EREVAN
USSR M-3
USSR VOLGA
USSR MESM
USSR RAZDAN
USSR STRELA *ARROW*
USSR BESM-1
USSR M-2
USSR URAL
USSR BESM-2
USSR KIEV
USSR LEM-1
USSR SETUN
VIENNA TECH URRI
VIENNA TECH. MAILUFTERL
WEGEMATIC 1000
WEIZMAN INST. WEIZAC
WESTINGHOUSE NORDIC-2
WHARF ENG. LAB. MAC
WARF ENG. LAB. M-2
WILLOW RUN RES. CEN. MIDAC
ZEISS RELAY COMP. OPREMA
ZEISS ZRA-1
ZUSE CO. Z-4
ZUSE CO. Z-11
ZUSE CO. Z-5
ZUSE CO. Z-22

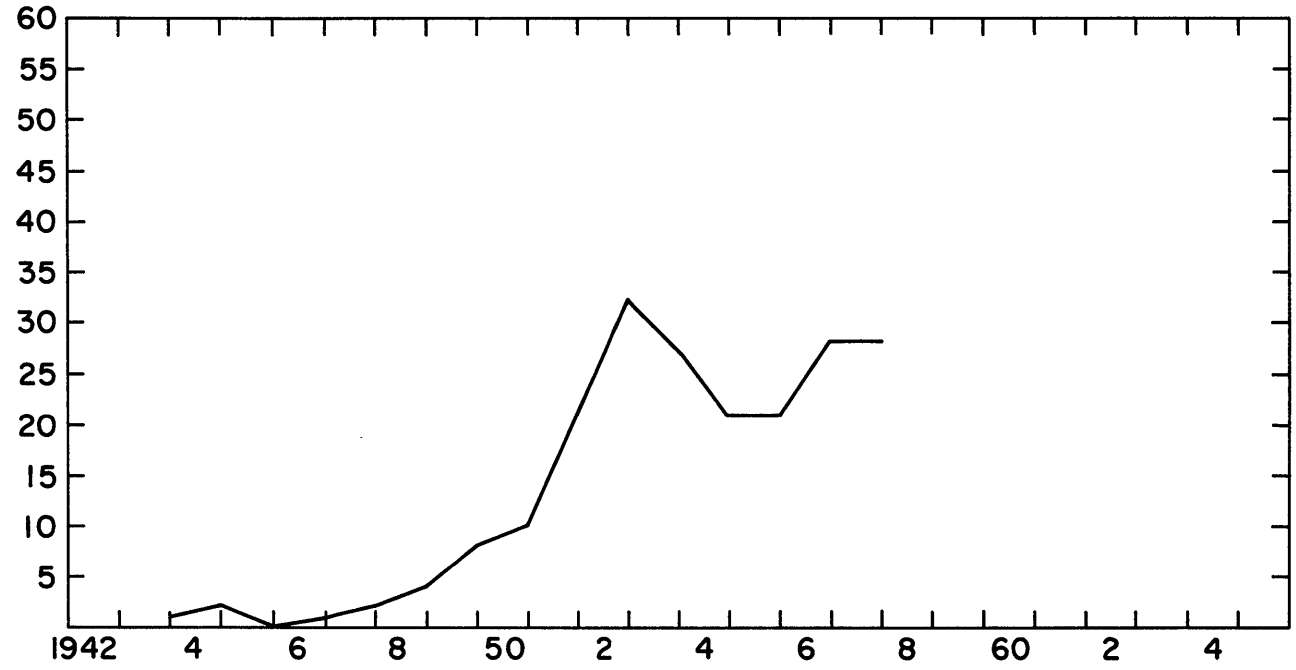


FIG. 1 NUMBER OF DIFFERENT TYPES OF COMPUTERS BUILT PER YEAR

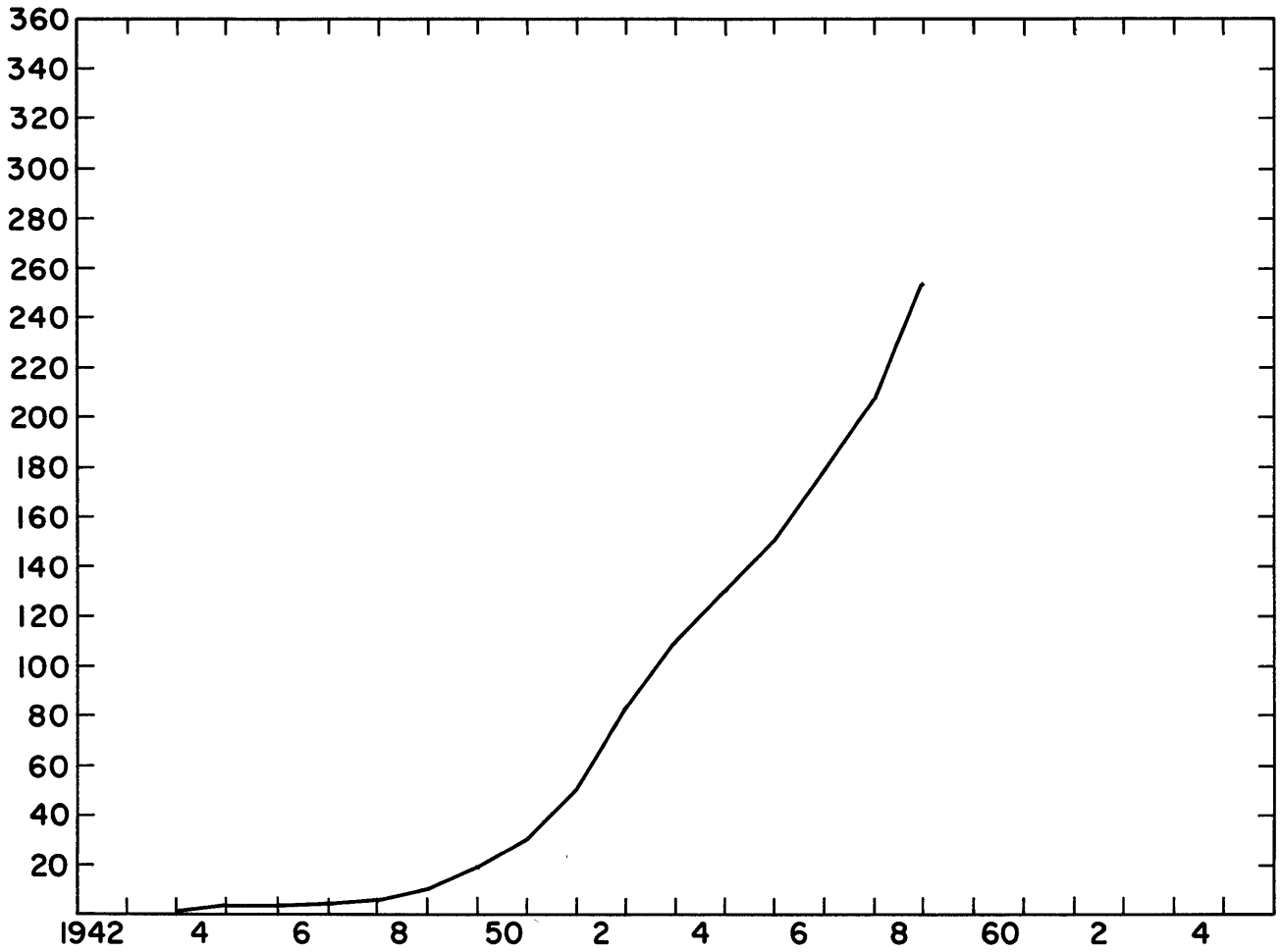


FIG. 2 TOTAL NUMBER OF DIFFERENT TYPES OF COMPUTERS DEVELOPED TO DATE

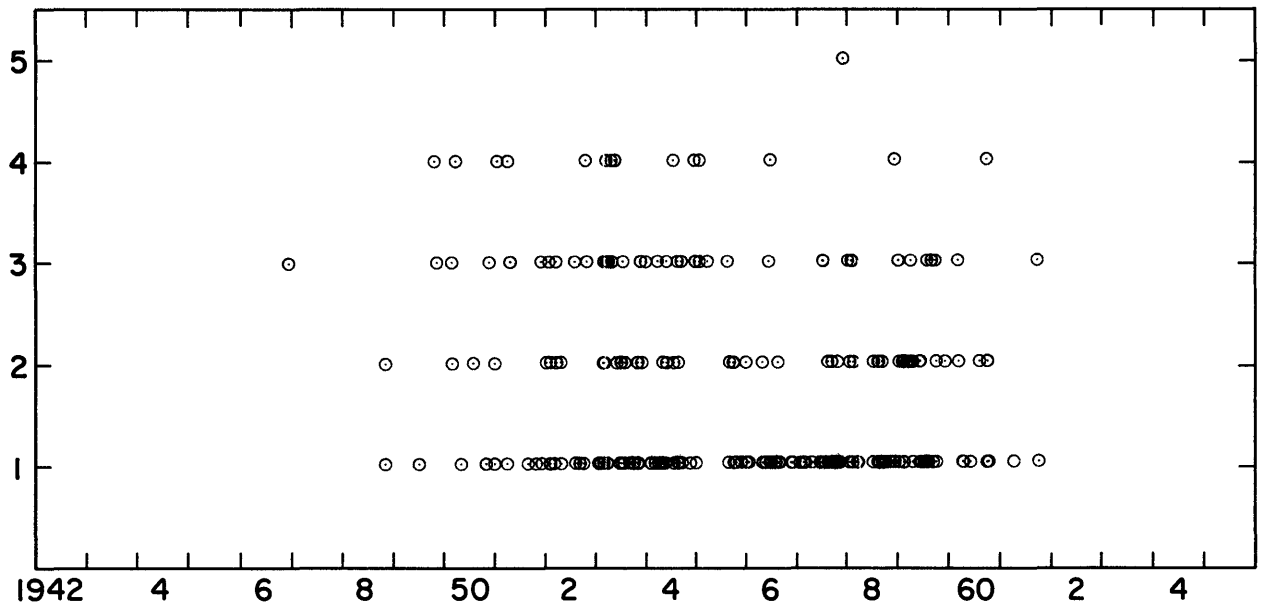


FIG. 3 NUMBER OF ADDRESSES PER INSTRUCTION

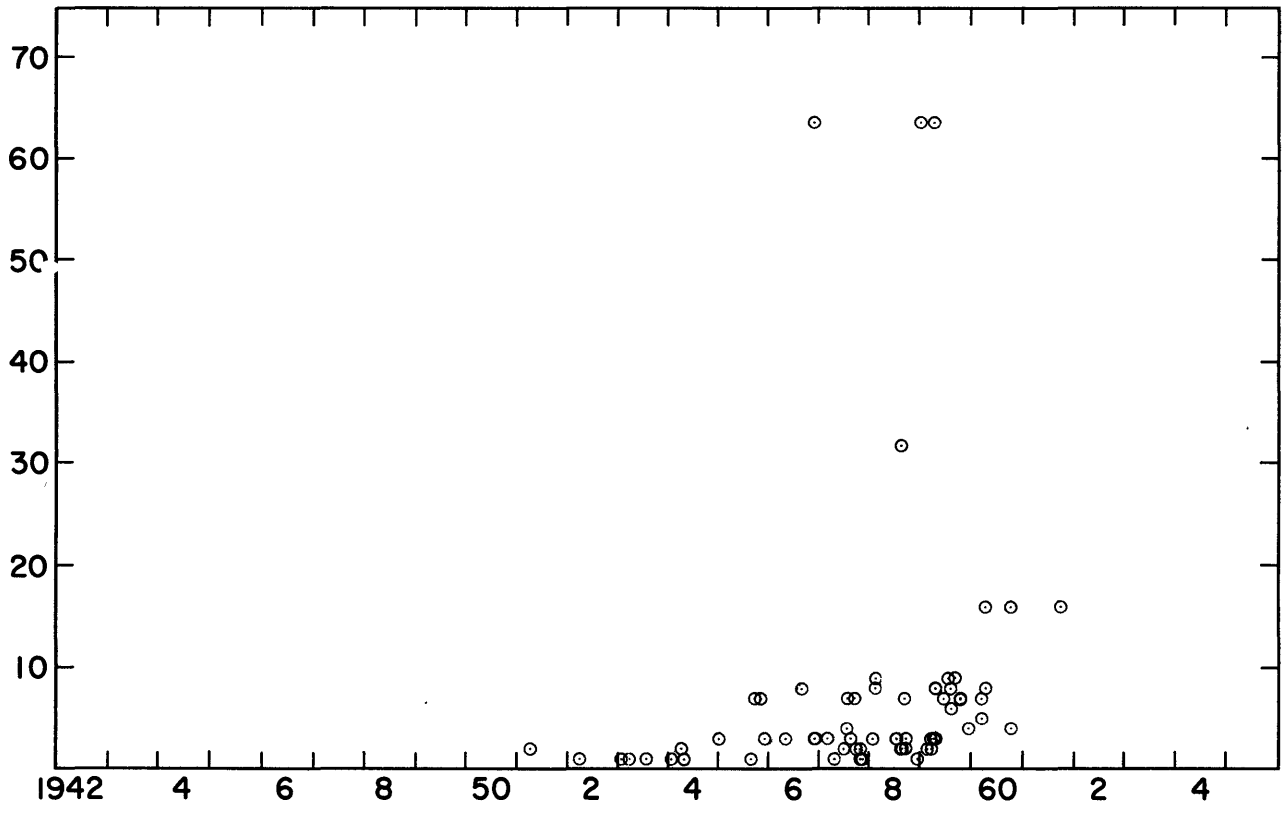


FIG. 4 NUMBER OF INDEX REGISTERS PER MACHINE

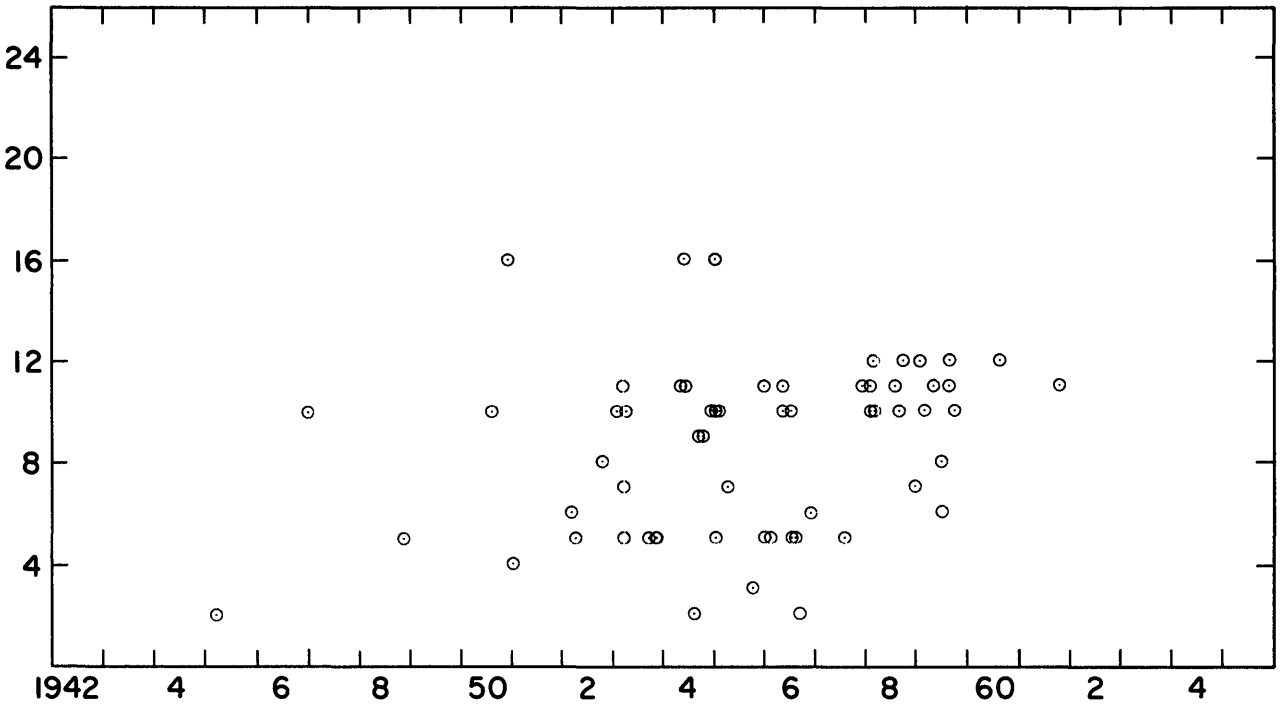


FIG. 5 NUMBER OF DIGITS PER INSTRUCTION

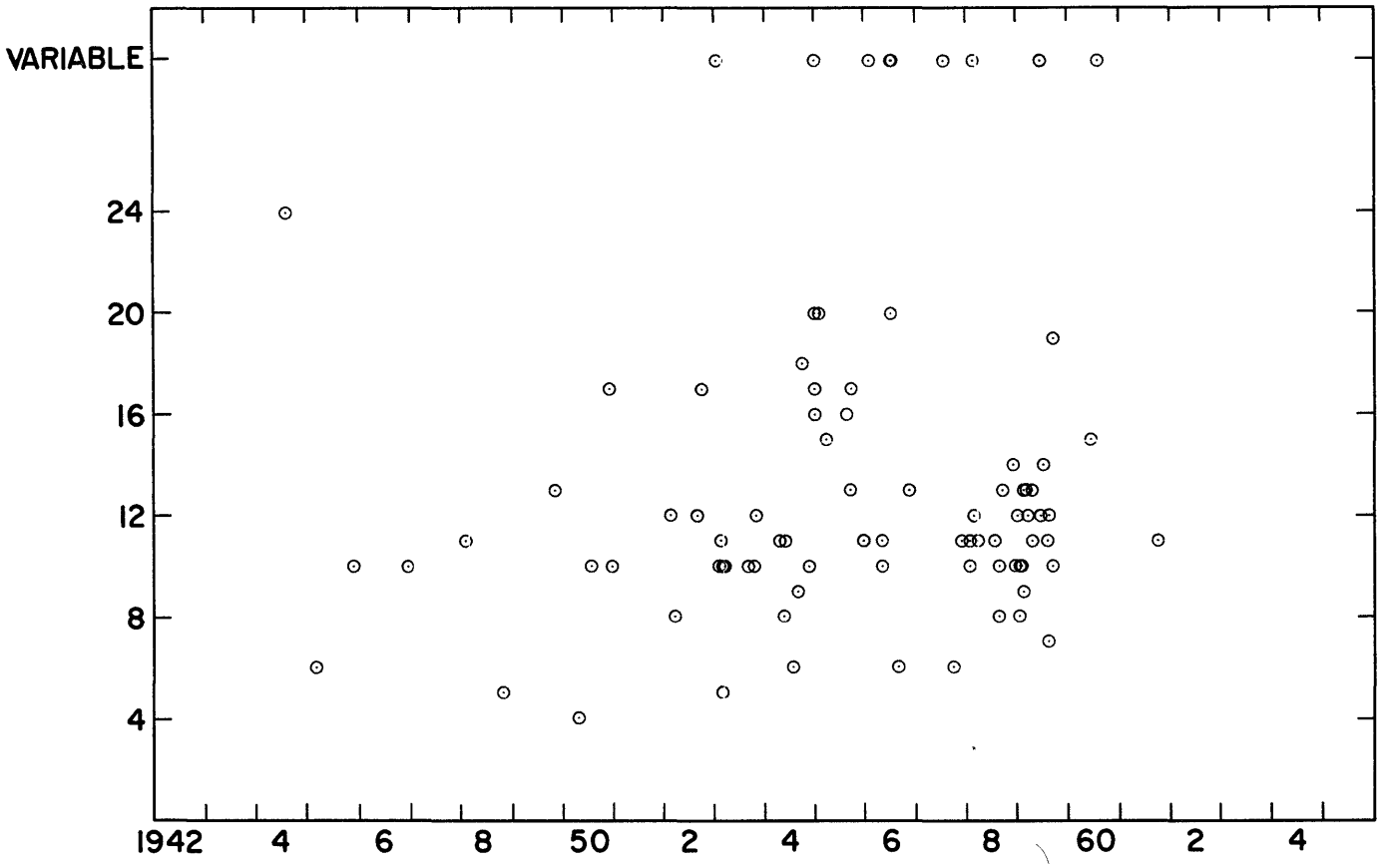


FIG. 6 NUMBER OF DIGITS PER DATA WORD

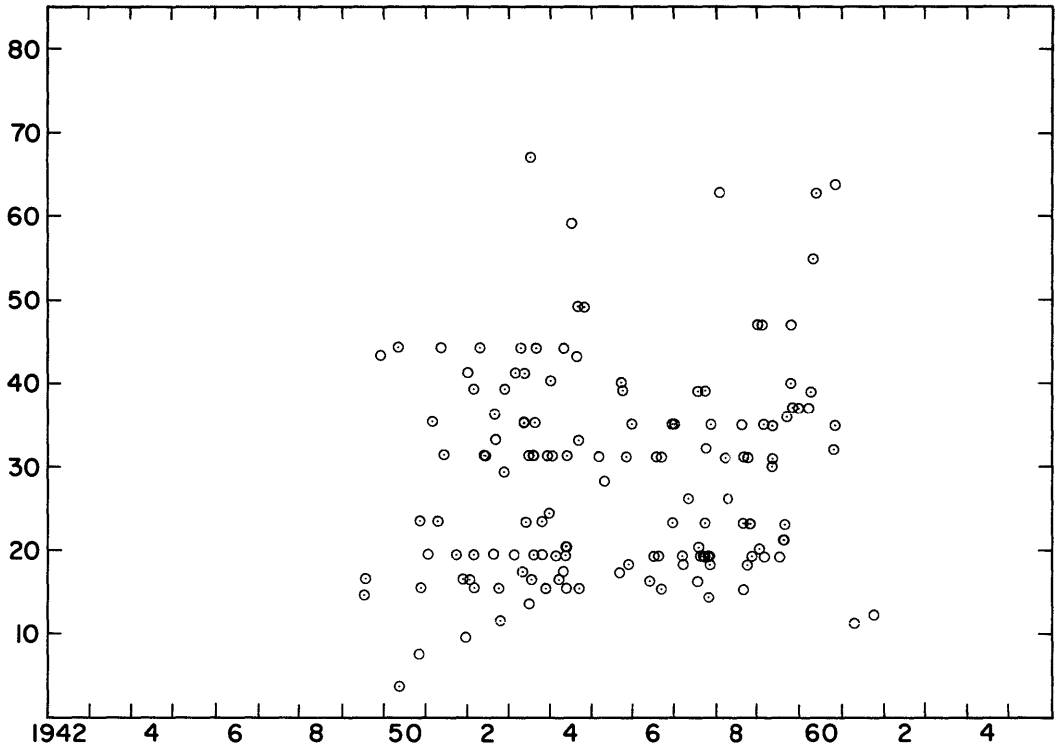


FIG. 7 NUMBER OF BINARY BITS PER INSTRUCTION

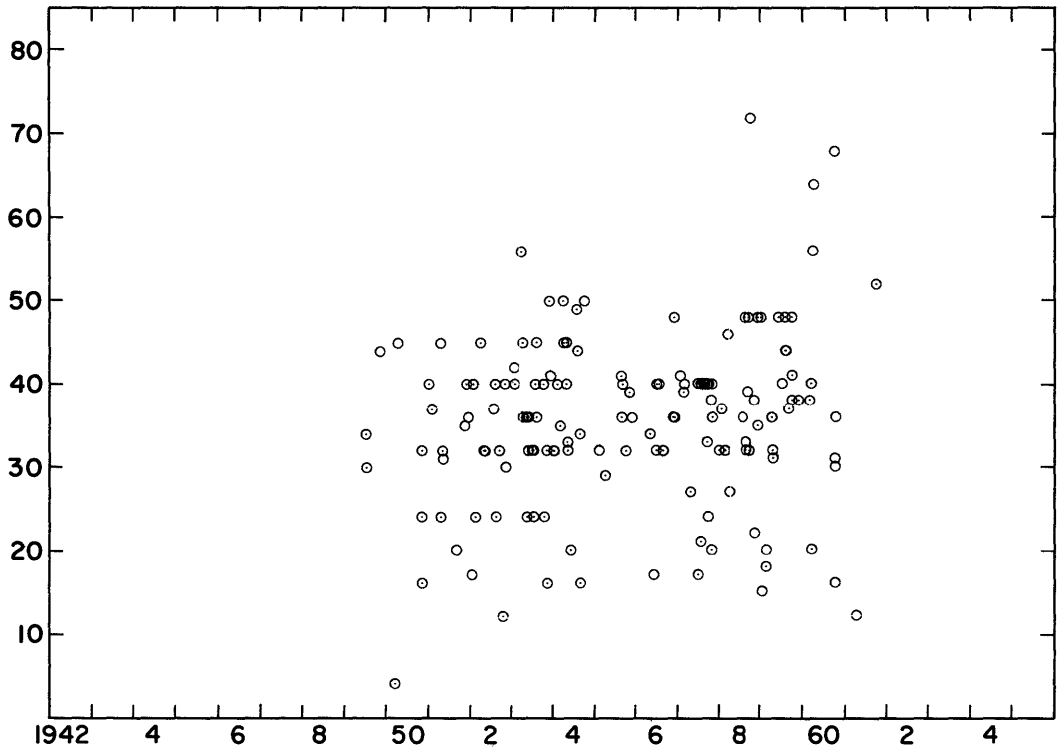


FIG. 8 NUMBER OF BINARY BITS PER DATA WORD

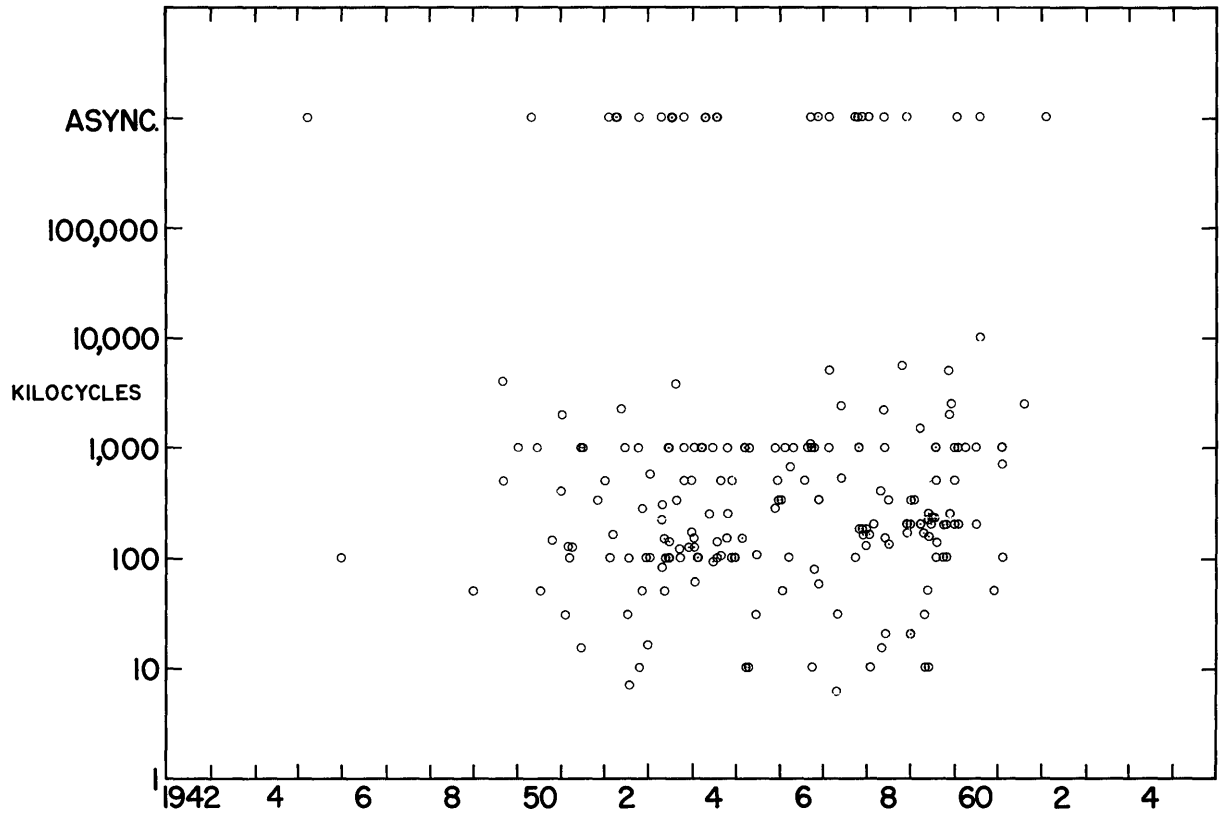


FIG. 9 INTERNAL CLOCK RATE

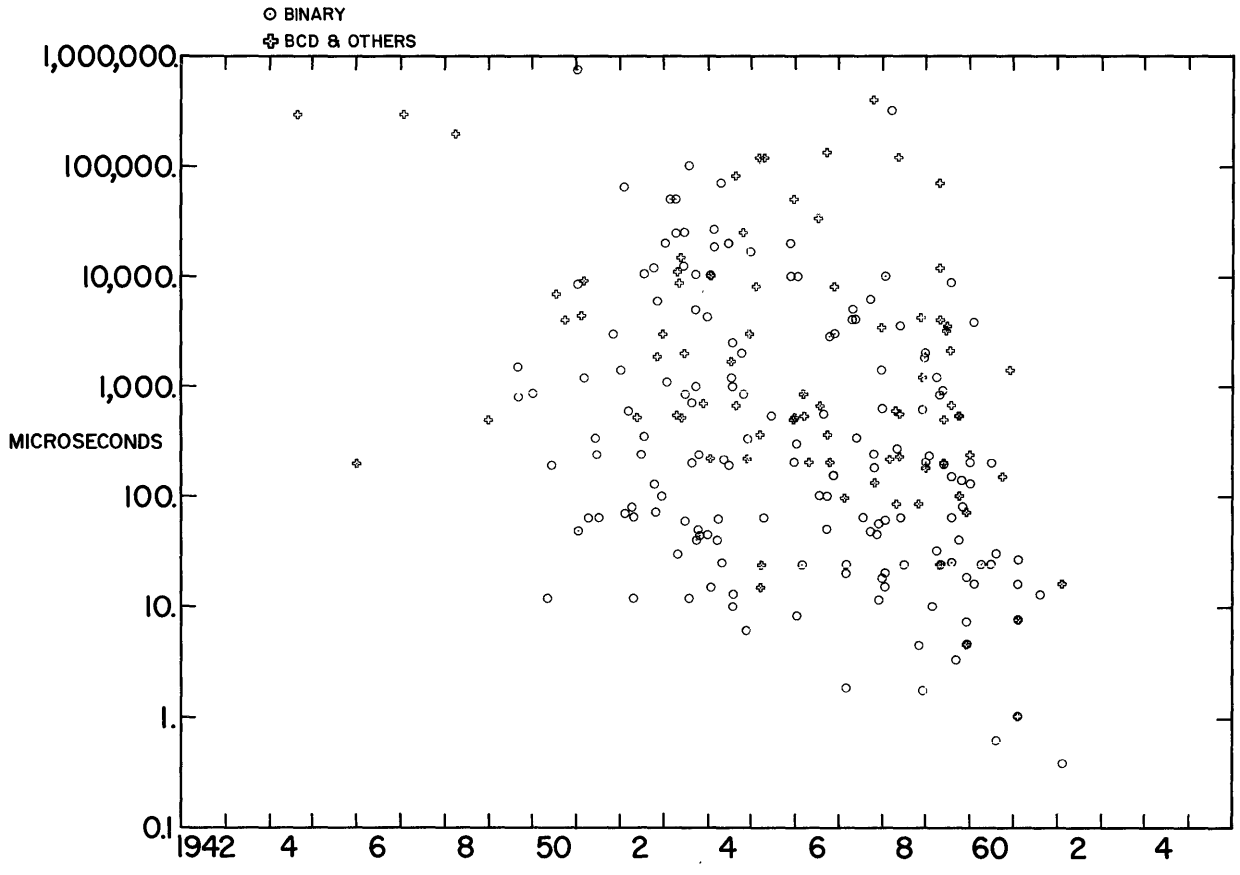


FIG. 10 FIXED POINT ADD TIME

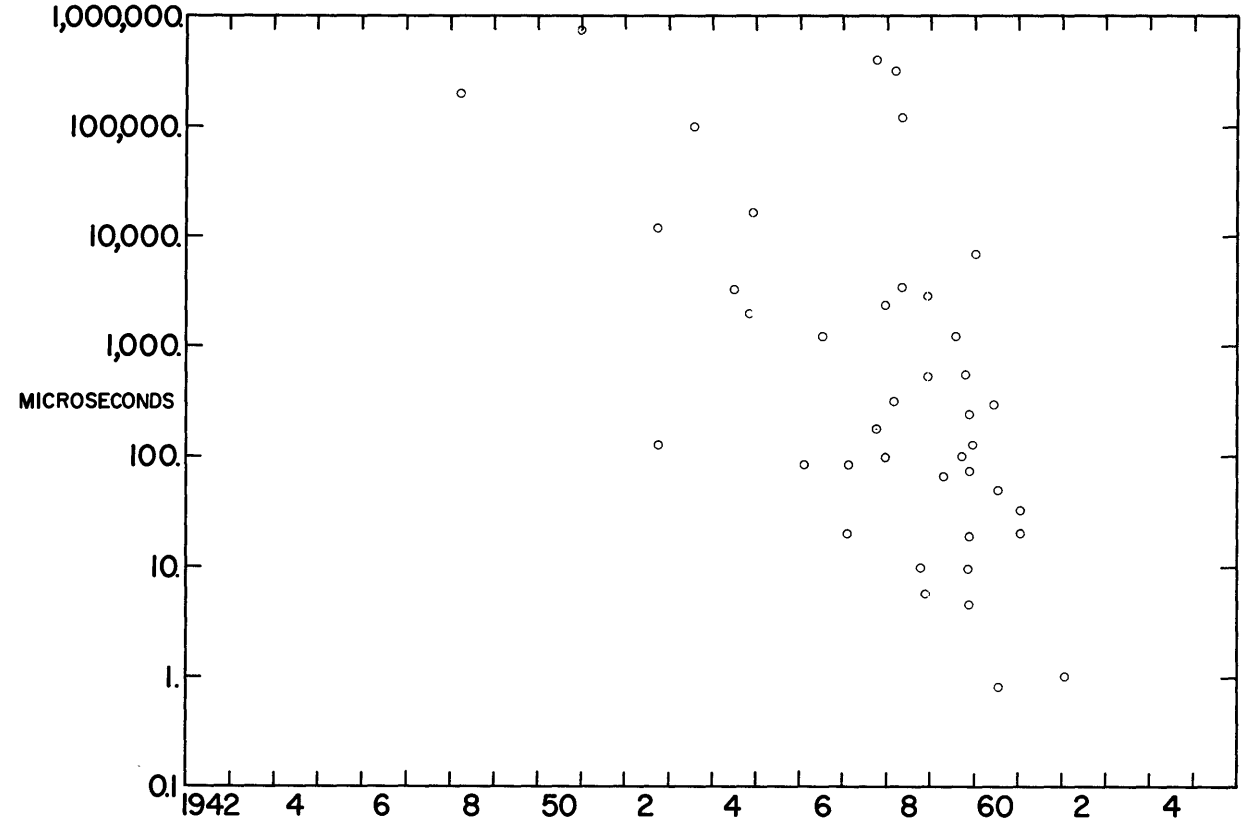


FIG. 11 FLOATING POINT ADD TIME

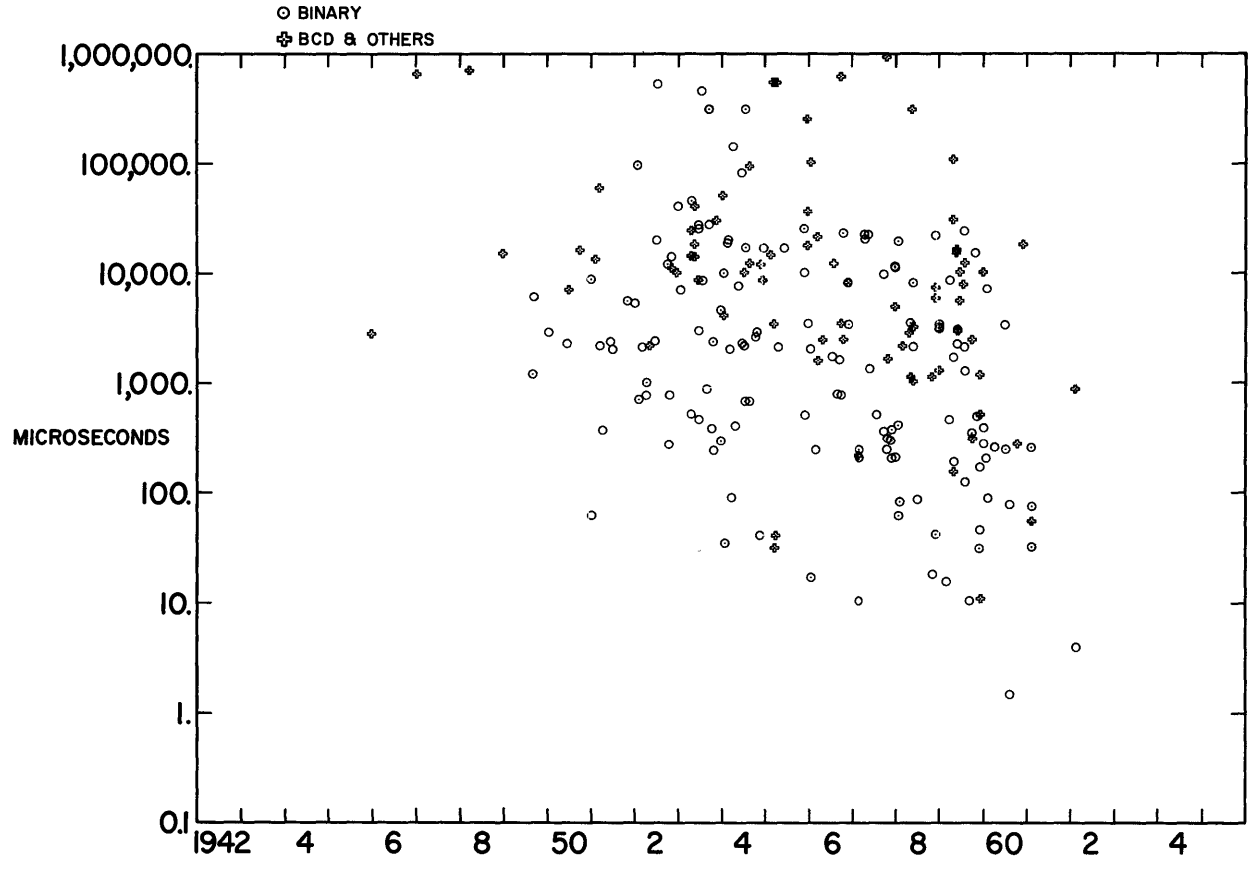


FIG. 12 FIXED POINT MULTIPLY TIME

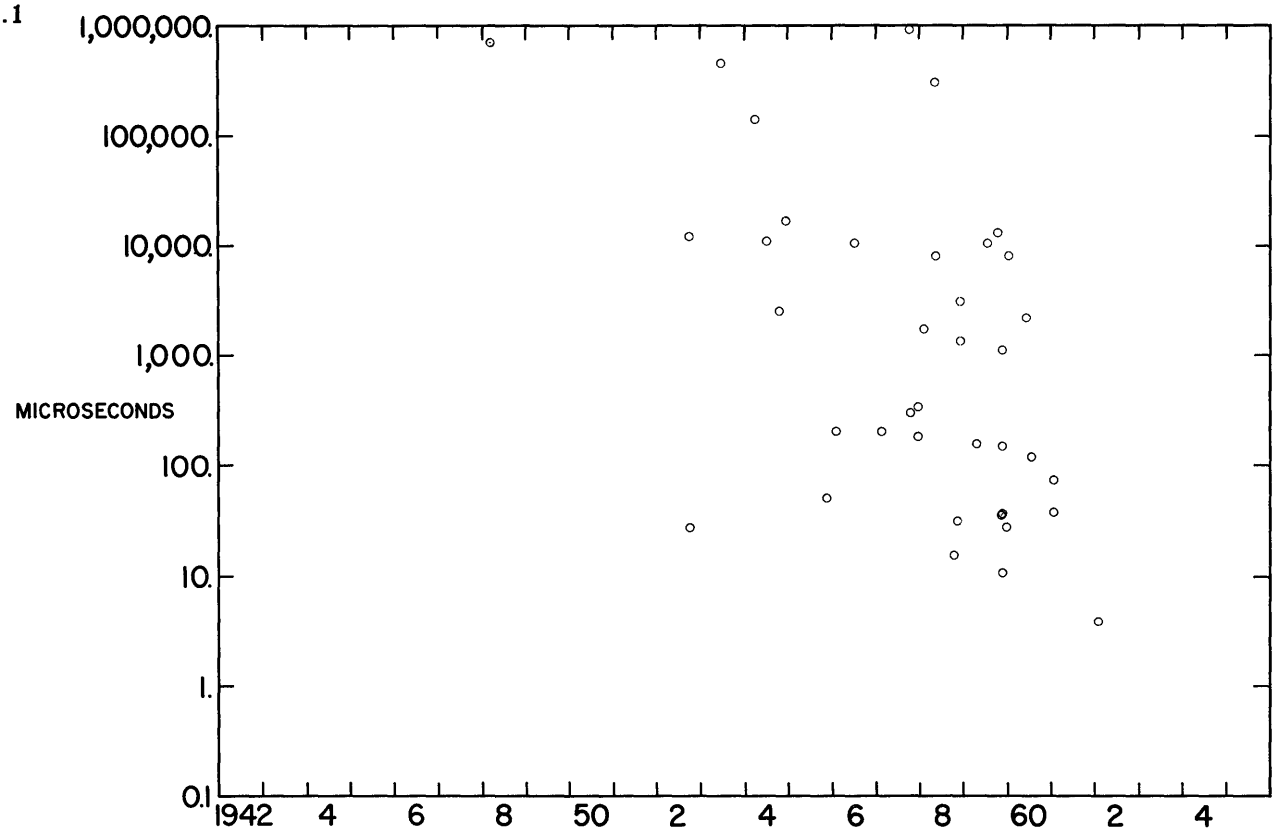


FIG. 13 FLOATING POINT MULTIPLY TIME

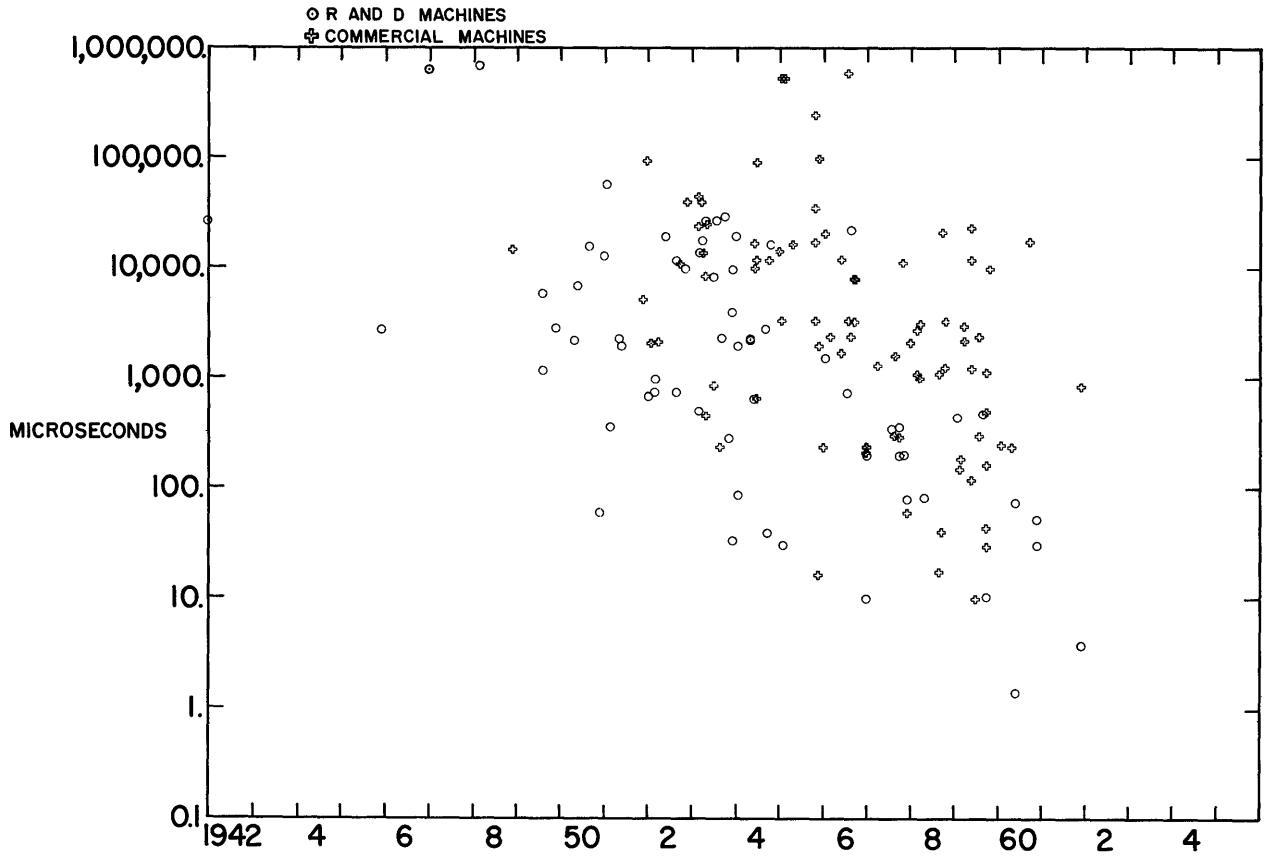


FIG. 14 FIXED POINT MULTIPLY TIME - BY COMPUTER TYPE

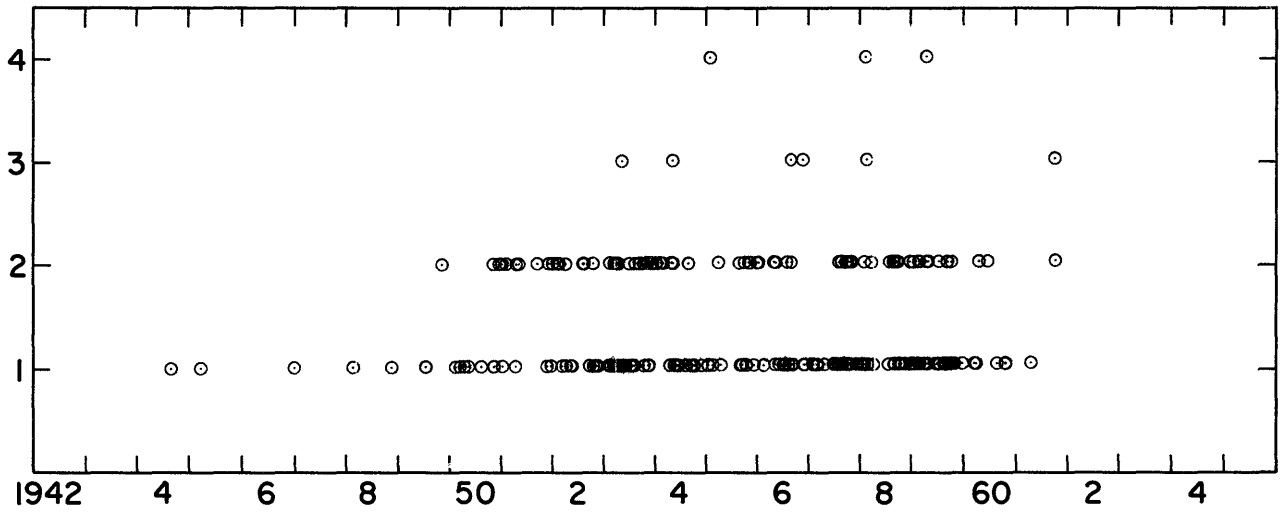


FIG. 15 NUMBER OF MEMORY SPEED LEVELS

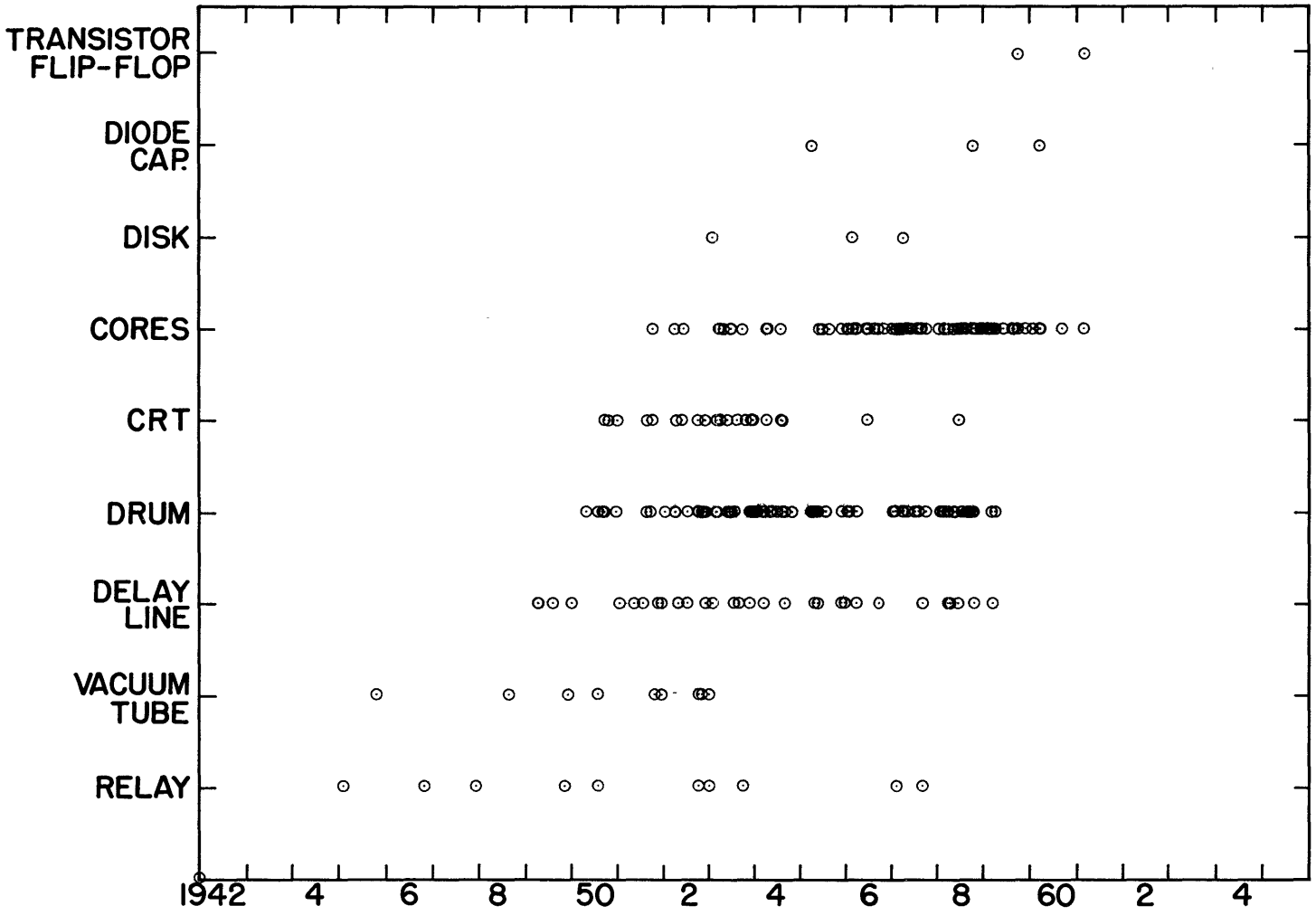


FIG. 16 TYPES OF HIGH SPEED MEMORY

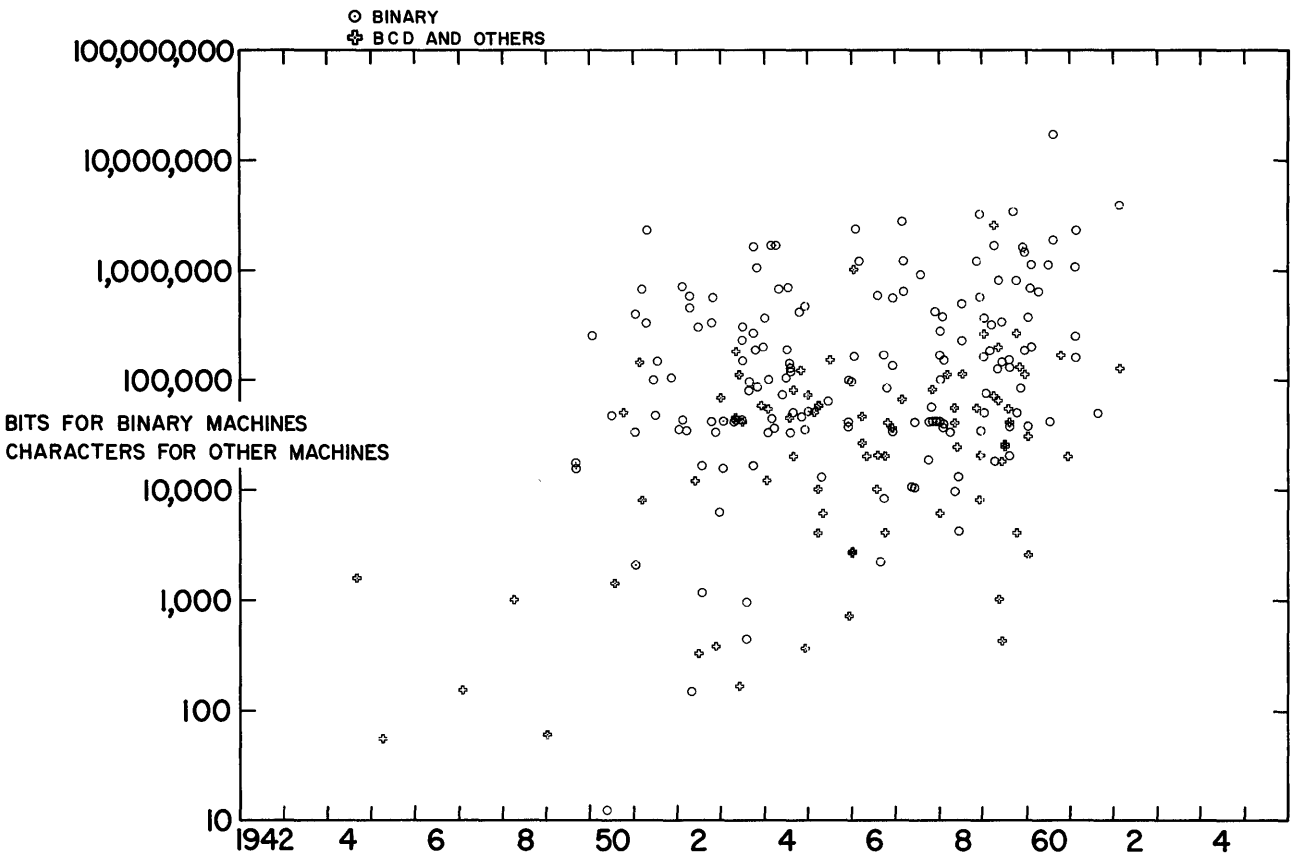


FIG. 17 HIGH SPEED MEMORY SIZE - BY CODE

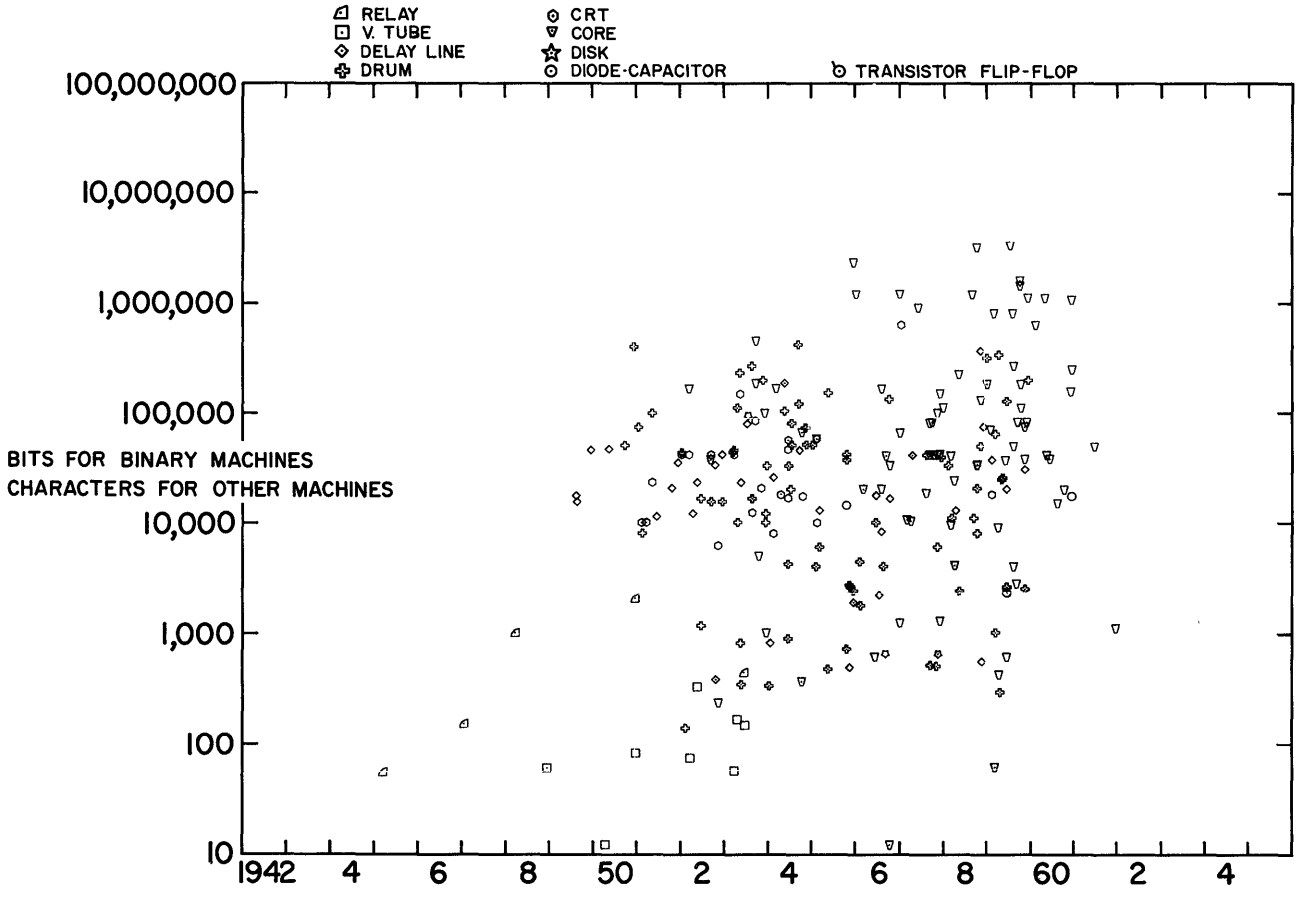


FIG. 18 HIGH SPEED MEMORY SIZE - BY TYPE

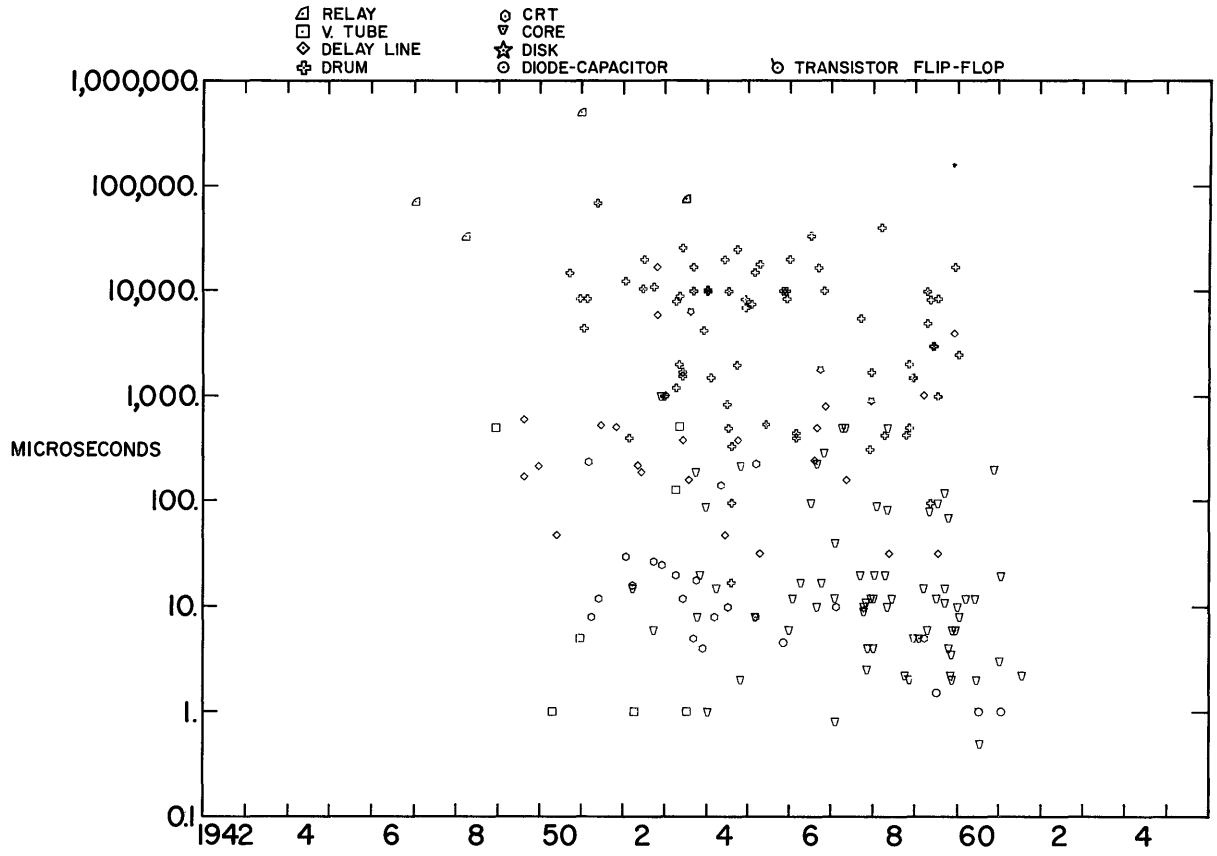


FIG. 19 HIGH SPEED MEMORY ACCESS TIME

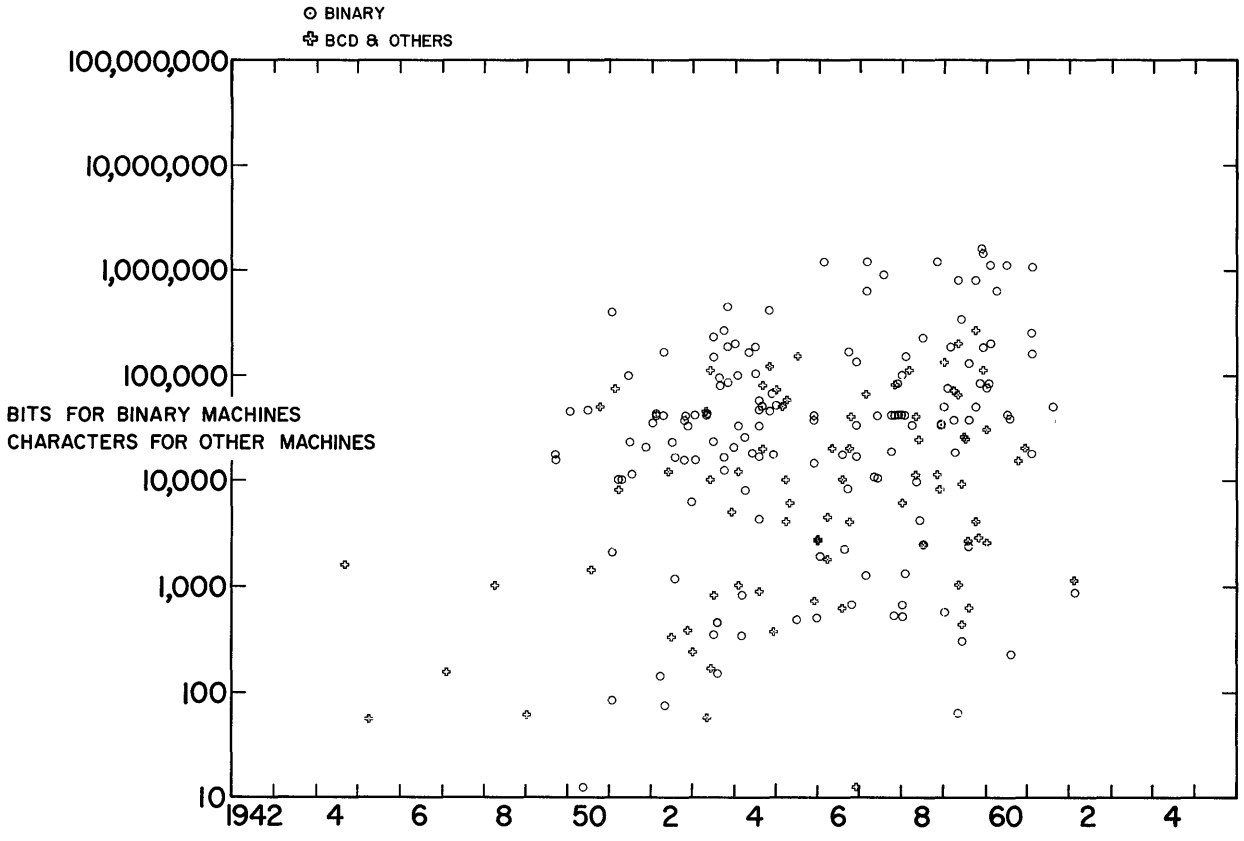


FIG. 20 TOTAL MEMORY SIZE

THE HARVEST SYSTEM

Paul S. Herwitz and James H. Pomerene

Product Development Laboratory, Data Systems Division
International Business Machines Corporation
Poughkeepsie, New York

Summary

The Harvest System is a large-scale data processor designed for maximum performance in handling extremely large amounts of data in primarily non-arithmetic operations. It is being built by IBM under contract with the Government and incorporates both a new system organization and a stored program concept of macro-instructions which directly implement many useful data manipulating subroutines. Design features include a very high processing rate and an on-line table lookup facility for effecting very general transformations.

Introduction

Most of the information generated and handled in our society is either non-numeric or developed within areas supported by little or no theoretical structure. A general-purpose system for processing this type of information must be organized very differently from more conventional scientific computers. Many of these areas are in the early stages of scientific development where ordering and classification are predominant activities and where the major problem is to uncover patterns and trends upon which theory can be built. There are few rules for determining the relevancy of information so that often enormous amounts must be examined to achieve significant results. Comparatively simple operations and data transformations are the most useful, and the output desired may frequently be some statistical characterization of the input.

The Harvest System is a large scale computer intended for maximum performance in this broad area of information processing. It was defined and is being built under study and development contracts between IBM and the Government. Harvest comprises a major data processing system reflecting the above considerations and includes an IBM Stretch computer for high performance on conventional operations (Figure 1). Stretch has been described elsewhere; only the special data processing portion is discussed here.

Harvest Organization

The Streaming Mode

Processing in Harvest is parallel by character, represented by a quantity of 8 bits or less. This quantity is called a byte and is the basic information unit of the system. The streaming mode is primarily a design attitude whose aim is to select bytes from memory according to some preassigned pattern and to keep a steady stream of such selected bytes flowing through a designated process or transformation and thence back to memory. Emphasis is on maximum flow rate so that the typical large volumes of information can be processed in minimum times. Processing time per byte is held to a minimum by prespecifying byte selection rules, processing paths, and even methods for handling exception cases, so that decision delays are suffered but once for a long sequence of bytes rather than being compounded for each byte. There is a functional analogy to plugboard machines except that the plugging can be changed at high electronic speeds and much of it on the basis of data encountered.

Stream Formation

The bytes which are selected to form the stream are taken from memory according to

either simple or complicated patterns which are chosen by the programmer. For technical reasons memory is organized into 64-bit words, but this artificial grouping is suppressed in Harvest so that memory is handled as a long string of bits any one of which can be addressed for selection. Up to 2^{18} words of memory can be directly selected, and since the word size is exactly 2^6 a Harvest address consists of 24 bits: 18 to select the word and 6 to select the bit within the word.

Data are transferred to and from memory as 64 parallel bits; selection down to the bit is accomplished by generalized operand registers called stream units (Figure 2). Each stream unit contains also a switching matrix which allows a byte to be selected out with minimum delay, starting at any bit position within the register. To handle cases where a byte overlaps from one memory word into the next and to minimize waiting time for the next needed word from memory, each stream unit is actually 2 words (i.e., 128 bits) long. The byte output of the stream unit is fed into the processing area through a bit-for-bit mask which enables the programmer to pass any subset of the 8 bits, including non-consecutive combinations. The selection of these bytes is controlled by the low order 6 bits of a stream of 24-bit addresses generated by the pattern selection units. There are two source stream units which feed operands into the processing area and one sink stream unit which accepts results from the processing area.

Pattern Selection

The data input to Harvest may be highly redundant to any particular problem, and so a powerful mechanism is provided for imposing selection patterns on the data in memory. It is assumed that the very effective input-output control in the basic Stretch system has grossly organized the contents of memory. For example, various characteristics may be recorded for a population and recorded in uniform subdivisions of a file. A particular problem may be concerned with only a certain characteristic drawn from each record in the file. Again, for example, data may be stored in memory in matrix form and the particular problem may require the transpose of the matrix.

Pattern selection in Harvest resembles indexing in other computers, except that in Harvest the programmer determines the algorithm which generates the pattern rather than listing the pattern itself. Each stream unit has its independent pattern generating mechanism which is actually an arithmetic unit capable of performing addition, subtraction, and counting operations on the 24-bit addresses. The programmer specifies patterns in terms of indexing levels. Each level consists of an address incrementing value I which is successively added to the starting address value S until N such increments have been applied, after which the next indexing level is consulted to apply a different increment. The programmer may then choose that incrementing continue on this level or that the previous level be resumed for another cycle of incrementing.

Many other indexing modes are provided to permit almost any pattern of data selection. Particular attention has been given to direct implementation of triangular matrix selection and to the iterative chains of any formal inductive process, however complex.

In general, the pattern selection facilities completely divorce the function of operand designation from that of operand processing, except that pre-designated special characteristics of the operands may be permitted to change the selection pattern in some fashion.

Processing Facilities

The pattern selection units determine the movement of data between the stream units and memory and, together with the stream units, determine the byte flow in the processing area. The processing facilities, together with the selection facilities, have been designed to give a flow rate of approximately 4 million bytes per second. Source stream units to the processing area are stream units P and Q while the sink stream unit is stream unit R.

Transformation Facilities

Two facilities are provided for the transformation of data (Figure 3). Extremely general operations on one or two input variables can be accomplished with the on-line table lookup facility. Simpler operations can be done directly by the logic unit without involving memory lookup. The logic unit also provides a choice of several one-bit characterizations of the input bytes (such as Byte from P > byte from Q). These one-bit signals can be used to alter the stream process through the adjustment mechanism.

The table lookup facility consists of two

units. The more important logically is the Table Address Assembler which accepts bytes from one or two sources and from them forms the lookup addresses which are sent to memory (Figure 4). The other is the Table Extract Unit which permits selection of a particular field within the looked-up word. Both units have their own indexing mechanisms and together they permit the programmer to address a table entry ranging in size from one bit up to a full word and starting at any bit position in memory. This freedom is abridged only by considerations of the table structure chosen by the programmer.

The table lookup facility also provides access to the memory features of Count and Existence. Under instruction from the Table Address Assembler the medium speed memory can use the assembled address to logically OR a one into the referenced bit position. The referenced word as it was just before the ORing can be sent to the table extract unit. In the high-speed memory a one may be either ORed or added into the referenced bit position with the same provision for sending the word before alteration to the table extract unit. The ability to add ones into high-speed memory words permits use of these words as individual counters. Several counter sizes can be specified.

Statistical Aids

The table lookup facility may be used to associate statistical weights with the occurrence of particular sets of bytes. For example, the occurrence of a byte P_i in the P stream together with a byte Q_j in the Q stream may be assigned a weight W_{ij}, which would be stored in a table and referenced by an address formed from both P_i and Q_j. Alternatively a memory counter may be associated with each pair P_i, Q_i and stepped on the occurrence of each such pair.

A Statistical Accumulator (SACC) is provided (Figure 5) either to sum the weights W over a succession of sets of bytes or to provide a key statistical measure of the counting results. In addition, SACC can be used for many other accumulating purposes.

A Statistical Counter (SCTR) provides a way of counting the occurrences of any of a large number of events during a stream. In particular, SCTR can be designated to count the number of weights W which have been added into SACC.

The Stream Byte-by-Byte Instruction

The table lookup unit, the logic unit, and the statistical units can be connected into the processing stream in various ways by the programmer. Like a class of analog computers, these connections reflect the structure of a problem and are the electronic equivalents of a plugboard. The hookup chosen by the programmer then applies the same processing to each byte or pair of bytes sent through it; this very general processing mode is called the Stream Byte-by-Byte instruction. The connections, indexing patterns, and special conditions described below all form part of a pre-specified setup which can be considered as a macro-instruction putting the computer into a specific posture for a specific problem.

Monitoring for Special Conditions

The concept of a streaming process determined by a flexible and extensive setup specification is most meaningful when applied to a large batch of data which is all to be treated the same way. In any particular streaming process, special conditions may arise within the data being processed which call for either momentary or permanent changes in the process. For example, the transformation being performed may be undefined for certain characters so that these must be deleted at the input; or a special character may be reserved to mark the end of a related succession of bytes after which the process or the pattern of data selection must be altered.

Special conditions can be monitored in several ways (Figure 5). Special characters can be detected by Match Units, each of which can be assigned a special 8-bit byte to match. There are four Match Units; W, X, Y, and Z, which can be connected to monitor the stream at several different points. When a match occurs, the Match Unit can directly do one of several operations on the stream and can also emit a one-bit signal indicating the match.

A large number of other one-bit signals are generated by the various stream facilities to mark key points in their respective processes. These one-bit signals, collectively called stimuli, can be monitored to accomplish specific operations, such as stepping SCTR or marking the end of an indexing pattern. They can also be used to accomplish a much wider

range of operations through the adjustment mechanism.

Up to 64 stimuli are generated by the various processing, indexing, and monitoring functions in Harvest. For any particular problem those stimuli which represent the key or significant properties of the data being streamed can be chosen. To each stimulus or coincident combination of stimuli the programmer may associate one or more of a large number of reactions on the stream; its data, its process, or its indexing. These stimulus-reaction pairs are called adjustments. The adjustment mechanism gives the programmer a direct way of picking out those elements of the stream which are different from the general run. These different elements may provide the key to the pattern being sought, either because they are particularly relevant or because they are distinctly irrelevant.

Programming Harvest

The Instruction Set

Conventional arithmetic and scientific computational processes and all input-output operations for Harvest are performed in the Stretch computer which is part of the system. When Stretch instructions are used, the system operates in the arithmetic mode; when streaming mode control is imposed, the unusual instructions unique to Harvest are available. There are about 85 instruction families (i.e., instruction types plus associated operation modifiers) in Stretch alone. The Harvest stream instructions add a variety of extremely powerful data processing tools to the several thousand basic Stretch operations. Throughout the system the instruction formats vary in length: single-address instructions are 32 bits long, two-address instructions and instructions that operate on variable length fields are 64 bits long, and stream instructions have an effective length of 192 bits.

Streaming mode instructions are very much like built-in subroutines or macro-instructions. Just as it is necessary to initialize a programmed subroutine, it is also necessary to initialize or set up the Harvest processor. Roughly about 150 parameters and control bits may influence streaming. Harvest is set up by loading values of some of these parameters into and setting the desired control bits in specific, addressable setup registers prior to the execution of a stream instruction. Certain changes in the parameter values or control bit settings generate stimuli which may be used to terminate or cause automatic adjustments to be made to the stream, or to cause a change to the arithmetic mode of operation. The adjustment operations essentially constitute a second level of stored program and are used most generally to handle the exception cases that occur during processing operations. Thus the programmer sets up Harvest to execute a stream instruction; execution begins and is automatically modified as changing data or setup dictates; much routine bookkeeping is done automatically by the several independent pattern generating (indexing) mechanisms; changing values of parameters are always available for programmed inspection if automatic inspection is not sufficient for the particular operation being performed.

While most of the programming in the streaming mode of operation is centered around the Stream-Byte-by-Byte instruction, a number of other instructions derive from the unique organization of Harvest. The arrangement of the Harvest data paths and processing units facilitates the inclusion of operations that perform within one instruction each many of the routine collating functions such as merging, sorting, and file searching and maintenance that are so common to commercial data processing. Facilities of the table lookup unit are used extensively in these as well as in several other instructions designed primarily for the logical manipulation of data.

Since such extensive use is made of tables of parameters, table transformations, and other data arrays in the Harvest approach to data processing, a special Clear Memory instruction is available for clearing large blocks of memory in minimum time and with minimum programming effort. A single execution of this instruction will clear as few as 64 consecutive words or as many as 2048, depending on the programmer's wishes. The execution time in the latter case is less than 3-1/2 μsec, and only one instruction access from memory is made. A full memory complement of 2^{18} words can be cleared in less than one millisecond.

Collating Operations

Generally speaking, in order to perform merging, file searching, and other such collating operations, it is necessary to specify a number of parameters such as record length file length, control field length and position, etc. For Harvest, these parameters need

only be tabulated in proper order by the programmer. They are then utilized by the indexing mechanisms to cause data to be picked from and later be stored into memory according to the patterns that naturally occur in such data.

The Merge instruction family actually contains eight independent control sequences that may be used to merge files or even to completely sort blocks of records with but a single instruction access from memory. The particular option chosen by the programmer depends upon whether files are to be arranged in ascending or descending order, whether or not the record block can be contained in at most half the available memory, and whether the control field heads the record or is offset.

As an indication of the processing speed of Harvest, in the most favorable case (one-word records with control fields at the beginning of records) a block of 30,000 records already in memory can be completely sorted in 1-1/4 seconds or less.

The Search instruction complex consists of twelve control sequences, each of which facilitates abstracting from a master file all records whose control fields bear any specific one of six possible relationships to the control field of each record of a detail file. The possible relationships are the six standard comparison conditions $<, \leq, >, \geq, =, \neq$. If it is not desired to move the records that meet the search condition, it is possible to tabulate their addresses automatically.

Another complex called Select permits the programmer to select from a file that record having the least or greatest control field.

For the purpose of facilitating file maintenance operations, Harvest includes a collating instruction complex called Take-Insert-Replace. When the operation is executed under "instruction control," then whenever master and detail record control fields match, either the master record is taken out of (deleted from) the master file or is replaced by the detail record. Under "data control" the action taken whenever control fields match is indicated by the contents of a special control byte in the detail record. The masters can be deleted or replaced, or the detail record can be inserted in the master file; or under certain circumstances the maintenance procedure can be interrupted when master records with special characteristics are located, and resumed with a minimum of programming effort when desired.

Instructions such as the collating operations described above lead to a considerable reduction in the length of the generalized report generators, file maintenance routines, sorting and merging programs, etc., that might be expected to become associated with such a computer system.

Table Lookup Operations

It is often desired to be able to obtain data from or store data at an address that is not directly dependent on the data itself. The Indirect Load-Store instruction complex permits wide latitude in the formation of such addresses and in the subsequent manipulation of the original data. In essence the operation is as follows: parameters from one of the source stream units are used in the formation of an address in the table lookup unit; either this primary address itself or either of the addresses found in the word at the memory location specified by the primary address becomes either the origin of a field of data to be entered via the other source stream unit or the location at which the data field is to be stored by the sink stream unit; the data is moved from source to sink and the entire cycle is repeated. The counting and ORing features of the table lookup unit are available to the programmer as modifications of the basic instruction control sequence.

The second instruction complex built around the table lookup unit is called Sequential Table Lookup. An extremely powerful but conceptually simple instruction, it permits a class of transformations to be performed that may best be described as data dependent. The instruction causes a series of table references to be made, each successive reference after the first being made in a table whose address is extracted automatically from the previously referenced table entry. Also, as each reference is completed, a variable amount of data may be extracted from the table entry. Moreover, the indexing of the input or output data may be adjusted according to the contents of the table entry (similar to the operation of the Turing machine). The applications of Sequential Table Lookup are manifold; the editing for printing of numerical data, the transliteration of Roman numerals to Arabic, and the scanning of symbolic computer instructions for assembly and compilation purposes are but a few.

The extensive use of tables in problem solution typifies the different approach the programmer will take with Harvest. The

problem of transliteration of Roman numerals to Arabic illustrates the power of the method. Several simplifying assumptions have been made so that the flow chart is easier to follow. First, the data - a set of numbers expressed in Roman numerals, each number separated from the next by a blank (B) - is assumed to be perfect, and only the characters I, V, X, L, C, D, and M are used. Second, the set of numbers is terminated by two blanks, Third, the use of four successive identical characters (as Roman IIII for Arabic 4) is outlawed. Finally, the numbers to be transformed are all assumed to lie on the range 1 to 1000, inclusive.

The flow chart (Figure 6) shows the eighteen tables (consisting of a total of eighty-two memory words) used. Under each table heading a two-part entry is shown, the parts separated by a colon. On the left of the colon is the argument being looked up, followed in parentheses by an indication of the range on which the number or digit that will eventually result must lie. On the right of the colon the parameters of the table word corresponding to the argument are indicated symbolically: e.g., RO-1B (meaning "read out the integer 1 followed by the character for blank") or NRO (meaning "no readout"). This is followed by an integer in parentheses indicating what data byte is the next argument (0 means same byte, 1 means next byte, etc.); the arrow indicates the table in which the next argument is looked up.

As an illustration, consider the transliteration of DCLXXVIII:

1. D is looked up in the First Table; the number must be on the range 500 through 899. No digit is read out. The next argument is the next data byte.

2. C is looked up in the D_1 Table; the range must be 600 through 899. No readout. The next argument is the next data byte.

3. L is looked up in the DC_1 Table; the range is 650 through 689. Read out 6. The next argument is the next data byte.

4. X is looked up in the L_1 Table; the range of the unknown part of the number is 60 through 89. No readout. The next argument is the next data byte.

5. X is looked up in the LX_1 Table; the range is reduced to 70 through 89. No readout. The next argument is the next data byte.

6. V is looked up in the LX_2 Table; the range is now 75 through 79. Read out 7. The next argument is the next byte.

7, I is looked up in the V_1 Table; the range of the final digit is 6 through 8. No readout. The next argument is the next data byte.

8. I is looked up in the V_2 Table; the final digit is 7 or 8. No readout. The next argument is the next byte.

9. I is looked up in the V_3 Table; the final digit is 8. Read out 8 B. The next argument is the second following byte (the next byte is a B), i.e., the first byte of the next number to be transliterated, and is looked up in the First Table.

The process just described yielded the number 678 for DCLXXVIII. Only one instruction was accessed from memory (the Sequential Table Lookup) and, in fact, this single access served to transform the entire set of numbers. The process terminated when the character B was looked up in the First Table.

Clearly the decision logic for the problem is incorporated in the structure of the tables. However, in constructing these tables the programmer must concentrate on precisely this logic; most of the bookkeeping and other peripheral programming considerations are automatically taken care of. Wherever it was possible, this philosophy guided the systems planning of Harvest.

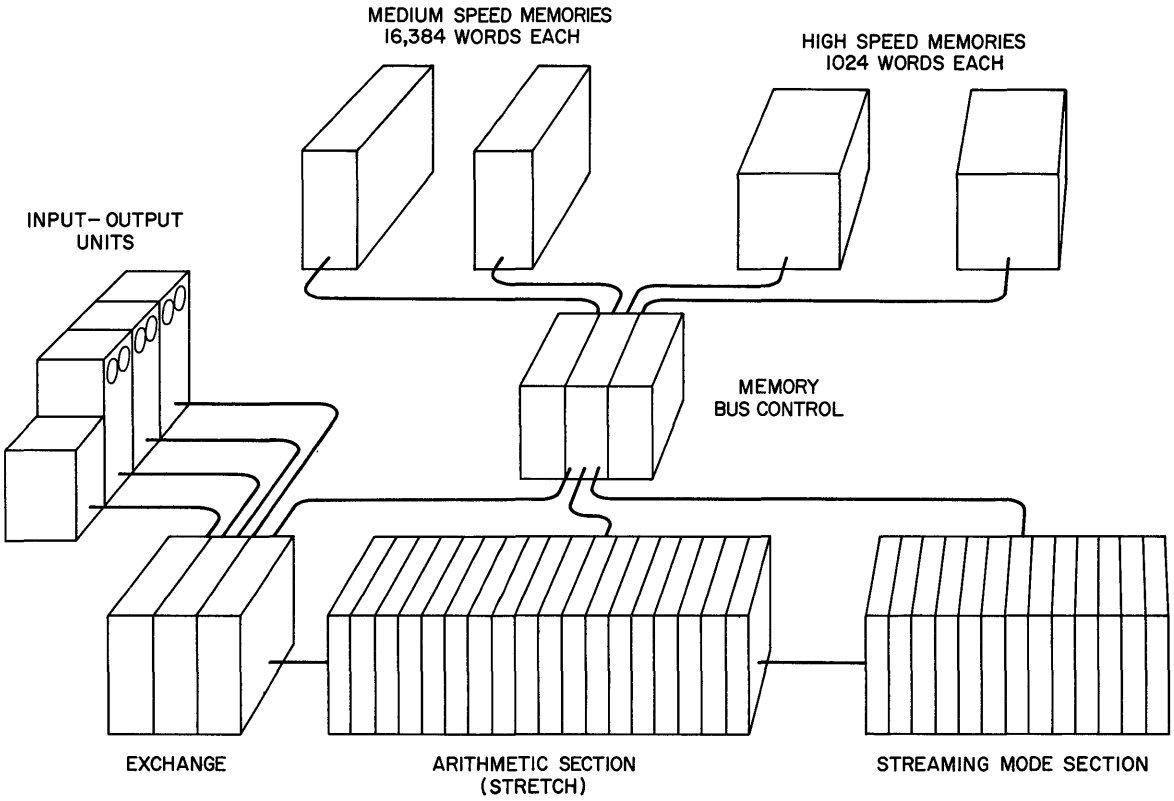


Fig. 1. The Harvest System.

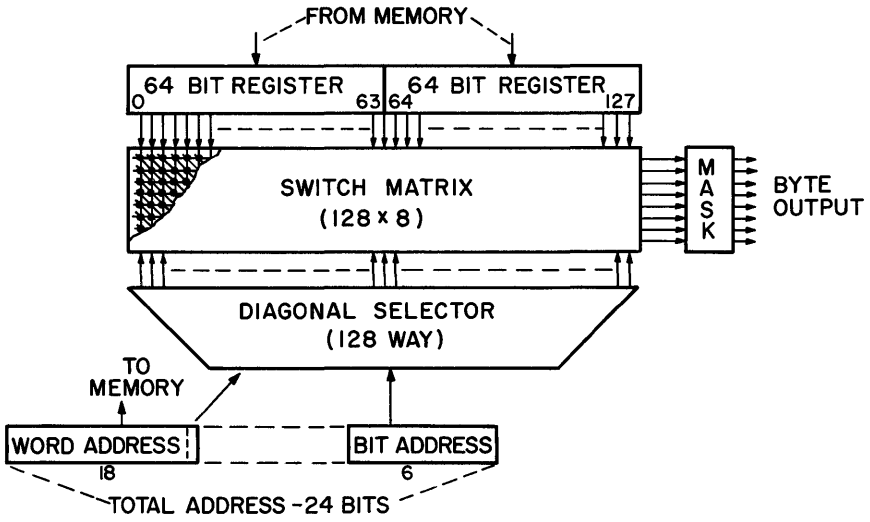


Fig. 2. The stream unit.

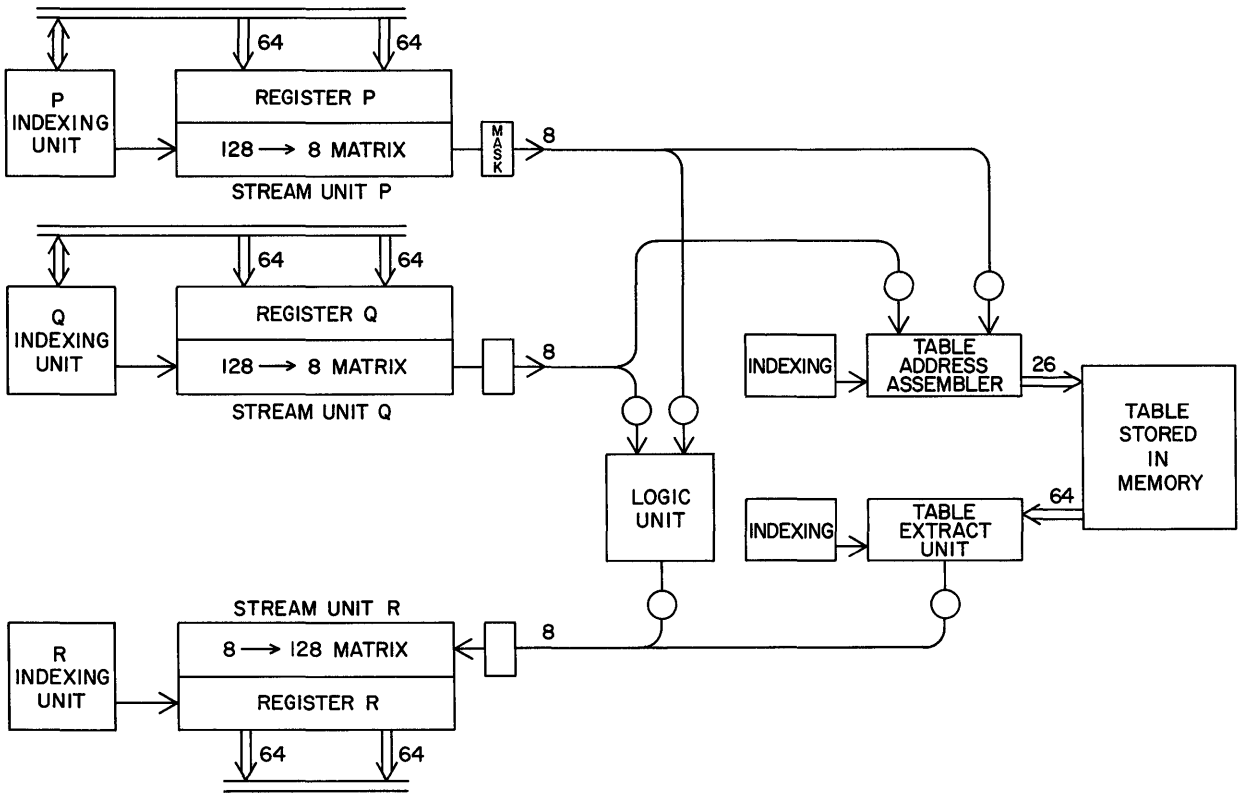


Fig. 3. Transformation facilities.

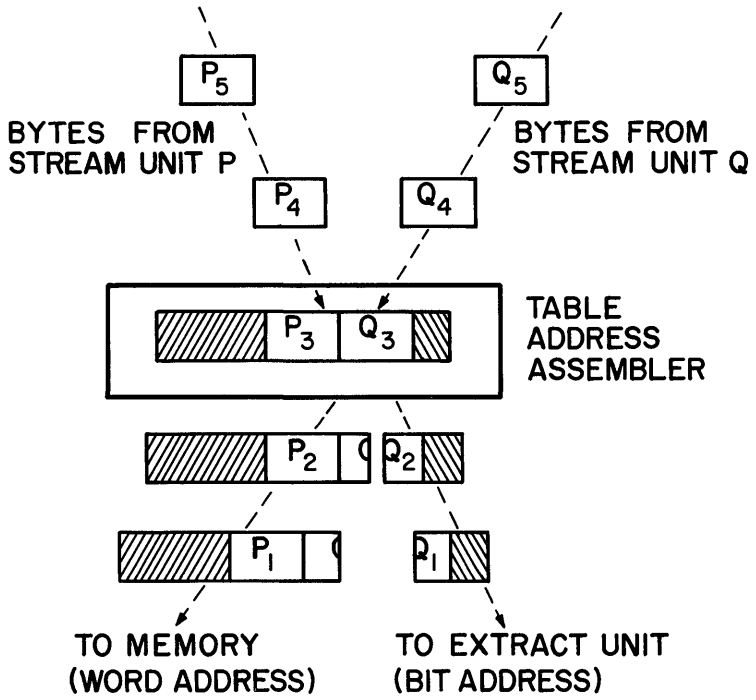


Fig. 4. Formation of lookup address.

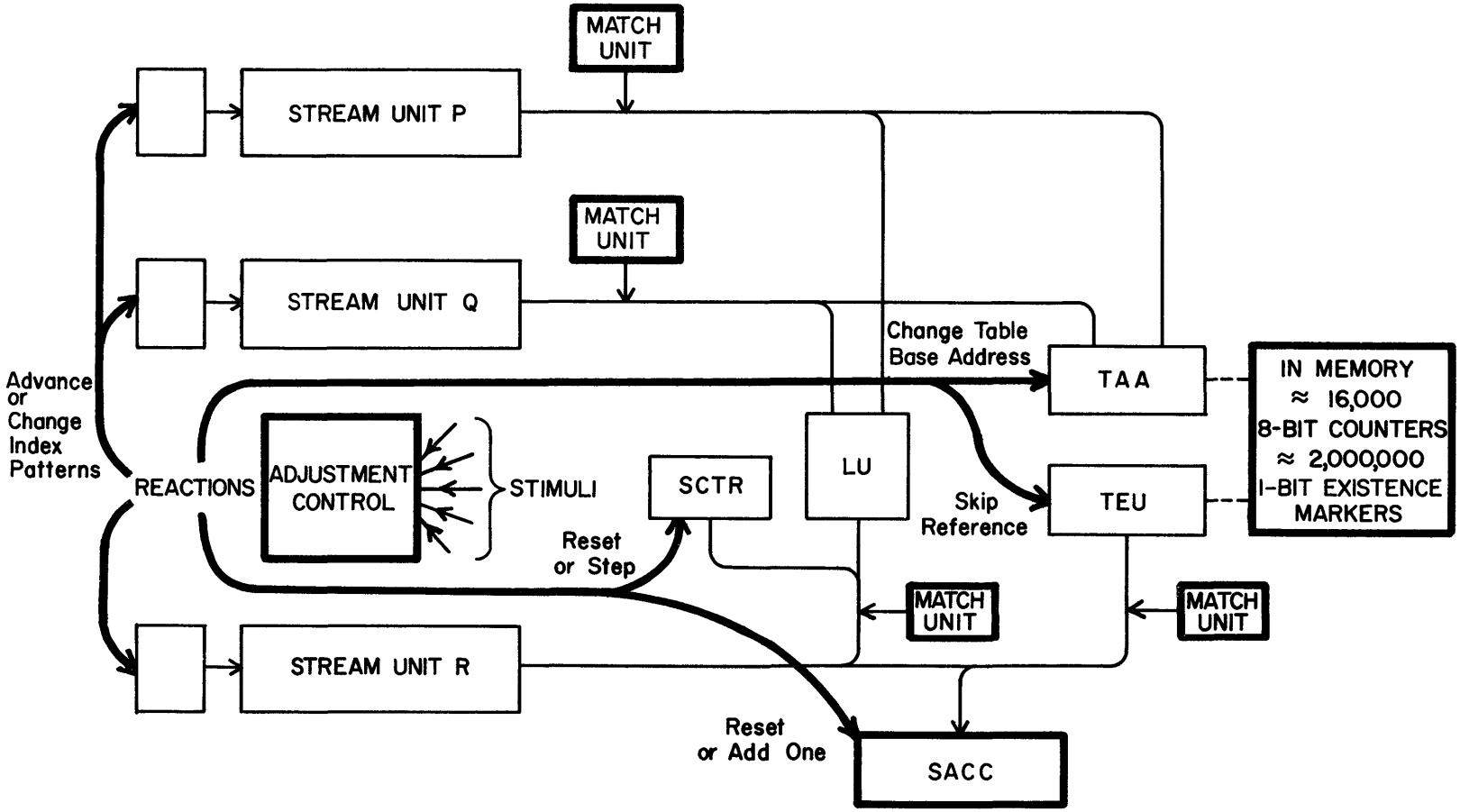


Fig. 5. Monitoring and statistical features with typical adjustment reactions.

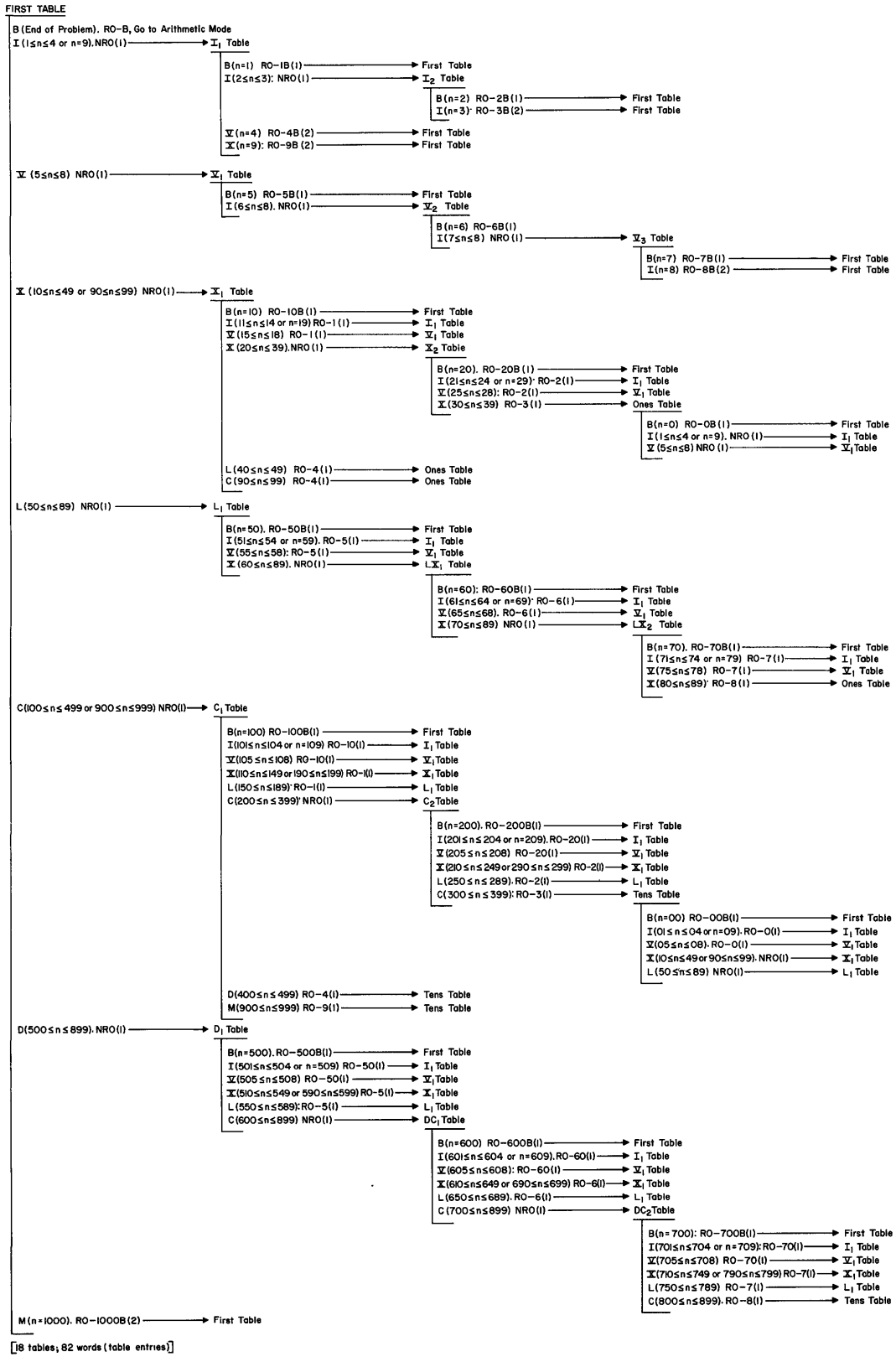

FIRST TABLE
B (End of Problem). RO-B, Go to Arithmetic Mode
I (1≤n≤4 or n=9). NRO(1) → I1 Table
B(n=1) RO-1B(1) → First Table
I(2≤n≤3): NRO(1) → I2 Table
B(n=2) RO-2B(1) → First Table
I(n=3): RO-3B(2) → First Table
X(n=4) RO-4B(2) → First Table
X(n=9): RO-9B(2) → First Table
X (5≤n≤8) NRO(1) → X1 Table
B(n=5) RO-5B(1) → First Table
I(6≤n≤8). NRO(1) → X2 Table
B(n=6) RO-6B(1)
I(7≤n≤8) NRO(1) → X3 Table
B(n=7) RO-7B(1) → First Table
I(n=8) RO-8B(2) → First Table
X (10≤n≤49 or 90≤n≤99) NRO(1) → X1 Table
B(n=10) RO-10B(1) → First Table
I(11≤n≤14 or n=19) RO-1(1) → I1 Table
X(15≤n≤18) RO-1(1) → X1 Table
X(20≤n≤39). NRO(1) → X2 Table
B(n=20). RO-20B(1) → First Table
I(21≤n≤24 or n=29): RO-2(1) → I1 Table
X(25≤n≤28): RO-2(1) → X1 Table
X(30≤n≤39) RO-3(1) → Ones Table
B(n=0) RO-0B(1) → First Table
I(1≤n≤4 or n=9). NRO(1) → I1 Table
X(5≤n≤8) NRO(1) → X1 Table
L(40≤n≤49) RO-4(1) → Ones Table
C(90≤n≤99) RO-4(1) → Ones Table
L(50≤n≤89) NRO(1) → L1 Table
B(n=50). RO-50B(1) → First Table
I(51≤n≤54 or n=59). RO-5(1) → I1 Table
X(55≤n≤58): RO-5(1) → X1 Table
X(60≤n≤89). NRO(1) → LX1 Table
B(n=60): RO-60B(1) → First Table
I(61≤n≤64 or n=69): RO-6(1) → I1 Table
X(65≤n≤68). RO-6(1) → X1 Table
X(70≤n≤89) NRO(1) → LX2 Table
B(n=70). RO-70B(1) → First Table
I(71≤n≤74 or n=79) RO-7(1) → I1 Table
X(75≤n≤78) RO-7(1) → X1 Table
X(80≤n≤89): RO-8(1) → Ones Table
C(100≤n≤499 or 900≤n≤999) NRO(1) → C1 Table
B(n=100) RO-100B(1) → First Table
I(101≤n≤104 or n=109) RO-10(1) → I1 Table
X(105≤n≤108) RO-10(1) → X1 Table
X(110≤n≤149 or 190≤n≤199) RO-1(1) → X1 Table
L(150≤n≤189): RO-1(1) → L1 Table
C(200≤n≤399): NRO(1) → C2 Table
B(n=200). RO-200B(1) → First Table
I(201≤n≤204 or n=209). RO-20(1) → I1 Table
X(205≤n≤208) RO-20(1) → X1 Table
X(210≤n≤249 or 290≤n≤299) RO-2(1) → X1 Table
L(250≤n≤289). RO-2(1) → L1 Table
C(300≤n≤399): RO-3(1) → Tens Table
B(n=00) RO-00B(1) → First Table
I(01≤n≤04 or n=09). RO-0(1) → I1 Table
X(05≤n≤08). RO-0(1) → X1 Table
X(10≤n≤49 or 90≤n≤99). NRO(1) → X1 Table
L(50≤n≤89) NRO(1) → L1 Table
D(400≤n≤499) RO-4(1) → Tens Table
M(900≤n≤999) RO-9(1) → Tens Table
D(500≤n≤899). NRO(1) → D1 Table
B(n=500). RO-500B(1) → First Table
I(501≤n≤504 or n=509) RO-50(1) → I1 Table
X(505≤n≤508) RO-50(1) → X1 Table
X(510≤n≤549 or 590≤n≤599) RO-5(1) → X1 Table
L(550≤n≤589): RO-5(1) → L1 Table
C(600≤n≤899) NRO(1) → DC1 Table
B(n=600) RO-600B(1) → First Table
I(601≤n≤604 or n=609). RO-60(1) → I1 Table
X(605≤n≤608): RO-60(1) → X1 Table
X(610≤n≤649 or 690≤n≤699) RO-6(1) → X1 Table
L(650≤n≤689). RO-6(1) → L1 Table
C(700≤n≤899) NRO(1) → DC2 Table
B(n=700): RO-700B(1) → First Table
I(701≤n≤704 or n=709): RO-70(1) → I1 Table
X(705≤n≤708) RO-70(1) → X1 Table
X(710≤n≤749 or 790≤n≤799) RO-7(1) → X1 Table
L(750≤n≤789) RO-7(1) → L1 Table
C(800≤n≤899). RO-8(1) → Tens Table
M(n=1000). RO-1000B(2) → First Table
[18 tables; 82 words (table entries)]

Fig. 6.

ORGANIZATION OF COMPUTER SYSTEMS -- THE FIXED PLUS VARIABLE STRUCTURE COMPUTER[‡]

by Gerald Estrin
Department of Engineering and Department of Mathematics
University of California, Los Angeles
Los Angeles, California

I. Introduction

The past decade has seen the development of productive fast electronic digital computers. Significant problems have been solved and significant numerical experiments have been executed. Moreover, as expected, a growing number of important problems have been recorded which are not practicably computable by existing systems. These latter problems have provided the incentive for the present development of several large scale digital computers with the goal of one or two orders of magnitude increase in overall computational speed.

The following discussion attempts to characterize aspects of digital technology related to extending the bounds of practicable computability and a new concept of computer system organization is proposed.

Component Technology

During the past semi-decade advances in reliability and speed have been produced by the development of magnetic core memories, new semi-conductor switching elements, improved vacuum tubes and superior input-output media. Further speed increases are promised by magnetic thin films, low temperature devices, semiconductors, microwave and microminiaturization techniques. The fastest single component switching speeds being discussed are about 10^{-9} second. For any significant parallel information transfers most researchers observe a loss of a factor of 10^{2} or 10^{3} bringing basic parallel computer operations to 10^{-6} or 10^{-7} second. These speed losses are a function of driving circuit limitations, length of lines, requirements for coincidence of independent applied forces, propagation time of output signals, amplification delay, destructive observation of memory cells, and requirements for simultaneous observations of the results of information transfers.

One cannot expect to do very much better with the populations of switching elements without a change in the method of organizing them. We may expect significant effects of the great effort being put into "microminiaturization" techniques. Any method which can accomplish more complex digital operations in a smaller volume will significantly affect overall speed if the basic element switching speed and the digital nature of the operation are not lost in the process.

Embryonic Systems

The most powerful digital computers either new-born or in development or under study are universal stored program machines which provide parallelism of operations[1-13]. This parallelism is introduced not just at the level of the bits in a number but rather by preparing new operands and instructions while arithmetic and logical operations are executed; by processing input and output data simultaneously with operation of the high speed machine; by providing more than one arithmetic unit; and by controlling the parallel operation of more than one machine.

Practicable Computability

In a finite system practicable computability is a function of such mundane factors as cost of computing and total delay from formulation of a problem to interpretation of results. These factors must of course be added to round-off, truncation, problem reformulation and machine reliability, all of which determine whether there is a relation between the symbols coming out of a computer and the answer to the original question posed.

The most significant advances are to be expected from the formulation of better methods of solving problems, of organizing physical elements for problem solution and the interaction between these two approaches. Any factors which enhance mathematical experimentation and man-machine learning are important system characteristics. With existing computing systems man-machine learning has been exercised in the past by the growth in libraries of subroutines and automatic programming methods, by growth in skill of the expert programmer, and by accumulated experimental results concerning the success and failure of various numerical methods and deficiencies in the machine design. There has been a particularly dramatic change in the language required to communicate with machines.

Structural Rigidity

In the process of design of any general purpose computer, a command set is selected on the basis of past experience and analysis of some limited problem set. With a universal computer it is known that it is always possible to program any special command. The compromise between simplicity, cost and speed resulted in short command lists in the first generation of machines. As speed and ease of programming became more significant, command lists lengthened. The "microprogramming"[14-17] approach made it feasible to change the command list by resequencing of a defined set of microoperations.

‡ The preparation of this paper was sponsored, in part by the Office of Naval Research, and the Atomic Energy Commission.

II. Design Objectives

The primary goal of the Fixed Plus Variable Structure Computer is:

(1) To permit computations which are beyond the capabilities of present systems by providing an inventory of high speed substructures and rules for interconnecting them such that the entire system may be temporarily distorted into a problem oriented special purpose computer.

In the following the above design objective will be expanded in an attempt to remove obvious ambiguities.

(1a) "...computations which are beyond the capabilities of present systems..."

The problems considered here are those for which some model exists -- that is, a set of rules according to which independent variables interact to produce change in dependent variables. Given such a model and given a question concerning the behavior of the dependent variables, then if existing systems require too much time to provide an answer to the question in a usable form, the problem is considered beyond their capabilities.

The least ambiguous cases are those for which there is an explicit time limitation -- commonly called real time problems. In such cases computations are often required to predict changes in a process and prescribe correcting forces according to chosen criterion functions. The absolute time available for computation is determined by the process to be controlled and may vary from the order of 10^{-9} second to the order of a year. In the former case the operation of a single binary element is just beyond the horizon. In the latter case it is possible to carry out about 10^{12} sequential operations during that time interval. It is of course possible to remove the real time pressure and test the predictive power of a model for one stage of a process. However if the effect of the corrective force is to be determined through several stages it is necessary to either be constrained by the time restrictions of the process or be capable of simulating the process on an expanded time scale.

Most problems have more implicit constraints on computability. Thus if a particular budget is allowed for the solution of a given problem, the cost per hour and computation time will determine computability.

In some cases, the value of a solution may be very high because of widespread influence on future technology but it may take so long to conduct numerical experiments that the interaction between investigators and experiment is essentially vitiated. In these cases user personality may determine computability.

Some of the problems which exist as a constant challenge to computer designers and mathematicians are the numerical solution of partial differential equations in four independent variables, nonlinear partial differential equations of lower dimensionality, mechanical vibrations in solids, truly three dimensional problems in elasticity and plasticity, multiple integration problems arising in connection with the equation of state, radiation diffusion, complex design problems such as reactor design, combinatorial problems and problems associated with experiments in number theory.

One may also consider the aggregate of smaller problems as a distinct problem type. Most computer manufacturers are in fact concentrating their attention on computer organizations which can process several independent computations in parallel so as to maintain a high duty factor in all elements of the system and reduce the cost per operation. Although the variable structure computer will consider such a property as a design factor, major emphasis will be placed on overall computation time of individual problems which cannot be handled by existing systems.

(1b) "...an inventory of high speed substructures..."

The Fixed Plus Variable Structure Computer will consist of a set of independent digital complexes ranging in size from individual switching elements or flip-flops to, say, shift-registers and counters to a high speed general purpose computer. The latter element is "fixed" in the sense that the machine user is presented with some minimum vocabulary and some minimum set of machine characteristics which do not change rapidly with time and which do not require the specification of some variable set of interconnections for each problem.

The existence of the "fixed" general purpose computer as an element in the system is considered essential to the evolution of higher languages for man-machine communication. If either the instruction code were always changing or the meaning of the instructions were always changing, any tendency for humans to invest effort in the definition of more complex instructions would be quickly damped despite the ability to make use of the variable structure inventory to effect higher language operations. The items in the inventory list are described by their inputs, outputs, logical functions, permissible loading, speed of operation, noise sensitivity, etc. The manner in which the initial inventory of substructures is determined is discussed in the second part of this paper.

(1c) "...rules for interconnecting them such that the entire system may be distorted into a problem oriented special purpose computer..."

The items listed in the inventory are not

in spatial positions relative to each other. Every effort is to be made to minimize parasitic resistance, capacitance and inductance in the interconnections such that the circuits may operate at speeds near those of equivalent fixed circuits. Wherever possible programmable electronic switching from one configuration to another will be considered but manual switching (e.g. soldering iron) will be considered if it leads to improved performance. It should be observed that substructures may of course operate in parallel.

(2) To permit computations which are beyond the capabilities of the initial system by establishing characteristics such that the inventory of substructures and the rules for interconnection may be changed as a function of past experience and new technological advances.

For example if it is found that an assembly of a number of the inventory items is used very frequently, then it may be desirable to introduce that assembly as a separate inventory item with speed optimized fixed wiring. In fact over longer periods, it may be observed that some substructures are used so frequently that they should be part of the "fixed" structure general purpose computer so that there is no need to explicitly specify the substructure each time it is used.

Figure 1 illustrates the relation between the Fixed Machine, the Variable Structure part, the Input-Output, Supervisory Control and the Humans.

III. Scope of Study for $\left[F + VS\right]_o$

Investigators at UCLA are now in the process of studying problems and compiling information which will permit a first design of the variable structure computer system.

Major emphasis is being placed on the production of a library of substructures. There will be two primary sources of that library. The first is the design of configurations for frequently used subroutines utilized by existing computer centers. The second is the design of configurations for particular complex problems of the type listed in the first part of this paper. In the latter problems many subroutines (substructures) must be linked in in the integrated process of problem solution.

In each case the computational method is analyzed; other computational methods are studied for possible increased overall speed; the computational methods are described in an essentially sequential form, i.e. the minimum number of elementary operations which must occur in sequence even if available equipment is unlimited (excluding total table lookup); and a recommended special purpose computer organization is defined assuming some reasonable equipment constraint.

The simple process of polynomial evaluation is one which is fairly efficiently handled by the conventional computer. However its discussion below serves to highlight the type of evaluation which may precede the design of any particular special purpose substructure. Consider the polynomial,

$$y = C_o + C_1x + \ldots + C_nx^n \tag{1}$$

The most common method called nesting or Horner's method effects the calculation as

$$y = \left(\ldots\left((C_nx + C_{n-1})\,x + C_{n-2}\right)x + \ldots + C_o\right) \tag{2}$$

If one assumes that "multiply-accumulate" is included in the set of elementary machine operations then the highly sequential graph of Fig. 2a demonstrates the computational flow. If there are no constraints on equipment, a tree structure of parallel arithmetic operations may be used to minimize the number of sequential steps as illustrated in Fig. 2b. An intermediate sequence suggested by K. Ralston makes efficient use of two multipliers falling just shy of doubling the speed of function evaluation. The process which consists of essentially separating odd and even terms, factoring, and using Horners method on the half length polynomials is illustrated in Fig. 2c for

$$y = C_o + C_1x + C_2x^2 + \ldots + C_nx^n \tag{3}$$

$$y = C_o + C_2x^2 + \ldots + C_{2m}x^{2m} + x(C_1 + C_3x^2 + \ldots + C_{2m+1}x^{2m})$$

$$y = \left(\ldots\left((C_{2m}x + C_{2m-2})\,x + C_{2m-4}\right)x + \ldots + C_o\right) + x\left(\ldots\left((C_{2m+1}x + C_{2m-1})\,x + C_{2m-3}\right)x + \ldots + C_1\right) \tag{4}$$

The simple nesting procedure represented by Fig. 2a, requires a single multiplier. At each interior time step the result of a previous multiply-accumulate operation becomes a multiplicand with the multiplier, x, unchanged and a new constant additively introduced. This is a very efficient operation requiring four initial memory accesses for the instruction and the first three operands followed by n multiplication times and n-1 access times for the additive constants. In a machine utilizing an anticipatory control the latter constants might be obtained while multiplication was proceeding. If such a machine provided a multiplication time which was a function of the operands one would obtain the time for polynomial evaluation,

$$t_{pe} \approx 4t_{access} + \sum_{i=1}^{n} t_{x_i}$$

where t_{x_i} = the time required for the i^{th} multiplication

For n = 2m or 2m-1, the process represented by Fig. 2b uses m+1 arithmetic units during the first step with the required number of units reduced according to the same relationship at each of the I steps where

1.3

$$I \leq \log_2 n < I + 1$$

There is always a final step utilizing a single arithmetic unit and resulting in a total of I + 1 sequential multiply-accumulate operations. If only a single arithmetic unit were available the operations implied by Fig. 2b could be sequentially effected in (n + I) multiply accumulate operations. Two arithmetic units would reduce the number of steps to approximately (n/2 + I). It is also of some significance to observe that the increased parallelism reduces the effect of information in the operands. The time required up to any node in the tree is determined by the longest computation time in the branches leading into that node.

The process represented by Fig. 2c uses two arithmetic units and if we assume two independent memories the process requires a time,

$$t_{pe} \approx 3t_{access} + M_{ax}\left[\sum_{i=o}^{m} t_{x_{2i+1}}, \sum_{j=o}^{m} t_{x_{2j}}\right] + t_{x_{in}} + t_{x_f}$$

where $t_{x_{in}}$ is the time required for the initial multiplication forming x^2 and t_{x_f} is the time required for the final multiplication. Thus this process is more efficient than that of Fig. 2b for most cases in which there is a constraint on the number of available arithmetic units. With two arithmetic units only m+2 steps are required. What is the criterion by which one may select among these various methods? The focus of this computer system is on increasing the computability of each problem formulated for it. The criterion for deciding between alternate sets of substructures is the estimated time for a given amount of computation or equivalently the amount of computation achieved in a given time. The alternate substructures themselves are constrained by the total available inventory and the inventiveness of the designer. Referring back to the simple polynomial evaluation, if a problem requires the independent evaluation of many polynomials, it might be more efficient to use independent substructures each executing the simple sequential version of Horner's method. On the other hand if the quantities entering into one polynomial are dependent upon the calculation of the previous one, the more parallel substructures will probably be more efficient. Finally it would be desirable to compare all of the above with substructures whose design is focussed on reducing the time of any single multiply-accumulate operation.

In fact, applications for which the variable structure computer exhibits its greatest strength are those which may utilize abnormal number representations such as multiple precision, partial precision or multi-radix representations.

For example, consider the problem of generating the permutations of n unlike objects, where the objective is actually evaluating some function of the permutation. There are n! such permutations and so purely the process of systematically generating them is limited to fairly small values of n.

However, an algorithm was devised by C. B. Tompkins which made use of a multi-radix representation of a number representing each permutation. Some twenty students were asked to program the generation of these permutations on eleven different modern computers for n = 10, i.e. $n! \approx 3.6 \times 10^6$. The time estimates ranged from 7 to 34,000 hours. An extremely simple special purpose accumulator effecting variable radix addition reduced the estimated time to about 8 seconds.

As a result of design efforts like these a library of substructures will be produced and then overlayed to aid the first specification of the variable structure inventory.

A number of new high speed computers are being evaluated as possible candidates for the fixed structure computer. In addition to summarizing their primary characteristics, the instruction lists of the several machines are being described in a common notation in order to make comparison easier and to intelligently process the following typical questions.

> "Which instructions must be included in "F" to enhance the overall system without deteriorating the performance of "F" alone."
>
> "How simple is it possible to make "F" because "V" is included in the system?"
>
> "Does "F" permit extension of the normal word length by "V"?"
>
> "How easy is it for "V" to operate in parallel on partial word lengths, i.e. how fast can packed words be transferred from "F" to "V"?"
>
> "How much memory should be shared by "F" and "V"?"

These questions are primarily related to "F" — another set is typical of the questions pointed toward the variable structure computer.

Assuming that the first estimate of the inventory of modules in "V" is determined from results of the above studies, from the overall experience of the designers and with a necessary dollar constraint,

> "What is the character of modules which will permit physical re-organization in a short time without need for determining resonances and removing common line interactions?"
>
> "Given a user with a problem what is the procedure which will lead to the design of a special purpose computer, to the specification of modules, their relative position and interconnection?"
>
> "What procedure can be established to check such organization rapidly?"
>
> "How can each new structure be described

such that learning operates in the man-machine system, i.e. the next time similar substructures are needed it requires much less time from problem statement to use?"

"How can we design a "cleared" state of the variable structure computer such that the variable inventory is interconnected like a second "F"--since there are many problems for which the best we can do is to provide two computers operating in parallel on different sets of data?"

"How can the supervisory control exercise its power without affecting the speed of operation of "F" and "V"?"

"How can the computer system be used to aid the design of new organizations?"

This list of questions can grow very long but it serves to display the directions of our thinking

One might expect to find questions of interlocks and queueing appearing strongly in a parallel system like this. However it should be repeated that the focus of this system will be extending computability of individual large problems by detailed reorganization of the computer inventory into parallel special purpose substructures. Hence the parallel processing is a much more deterministic arrangement than those systems which seek to optimize the handling of many independent problems.

IV. Conclusion

The UCLA development described above is at an early stage. However we hope to present a first design in one year so that we may make a proper target for criticism. We are firmly convinced that when a special purpose configuration may be accomplished using available facilities, a new level of inventiveness will be exercisable.

References

1. E. Bloch, "The Engineering Design of the Stretch Computer," IBM Tech. Publ. TR 00.01000, 702, Dec. 1, 1959.

2. S. W. Dunwell, "Design Objectives of the IBM Stretch Computer," Proc. EJCC, 1956, AIEE, N.Y., 1957, pp. 20-22.

3. J. P. Eckert, "Univac-Larc, The Next Step in Computer Design," Proc. EJCC, 1956, AIEE, N.Y., 1959, pp. 16-20.

4. UNIVAC-LARC SYSTEM, GENERAL DESCRIPTION, Sperry Rand Corporation, 1959.

5. P. Dreyfus, "System Design of the Gamma 60," Proc. WJCC, May 6-8, 1958, pp. 130-132.

6. A. L. Leiner, W. A. Notz, J. L. Smith, A. Weinberger, "Pilot--A New Multiple Computer System," JACM, Vol. 6 #3, July 1959.

7. R. E. Porter, "The RW-400 -- A New Polymorphic Data System," Datamation, January/February 1960, p. 8-14.

8. W. A. Clark, "The Lincoln TX-2 Computer Development," Proc. WJCC, Feb. 1957, pp. 143-145.

9. W. F. Bauer, "Computer Design from the Programmers' Viewpoint," EJCC, Dec. 4, 1958.

10. PHILCO-TRANSAC S-2000, Philco Corporation, Government and Industrial Division.

11. CHARACTERISTICS OF THE MODEL 1604 COMPUTER, Control Data Corporation, Publication No. 018, November 11, 1959.

12. C. V. L. Smith, "Electronic Digital Computers", McGraw-Hill, N.Y., 1959, Chapt. 17.

13. Rex Rice, "Computers of the Future," IBM Research Report RC-151, April 20, 1959.

14. M. V. Wilkes and J. B. Stringer, "Micro-programming and the Design of the Control Circuits in an Electronic Digital Computer," Proc. Cambridge Phil. Soc., Vol. 49, pt. 2, p. 230, 1953.

15. M. V. Wilkes, W. Renwick and D. J. Wheeler, "The Design of the Control Unit of an Electronic Digital Computer," IEE Tech. Paper M 2365, June, 1957.

16. H. Billing and W. Hopman, "Mikroprogramm-Steuerwerk," Elektron, Rundschau, Vol. 10, pp. 349-353, Oct. 1955.

17. R. J. Mercer, "Micro-Programming," ACM, Vol. 4, pp. 157-171, April, 1957.

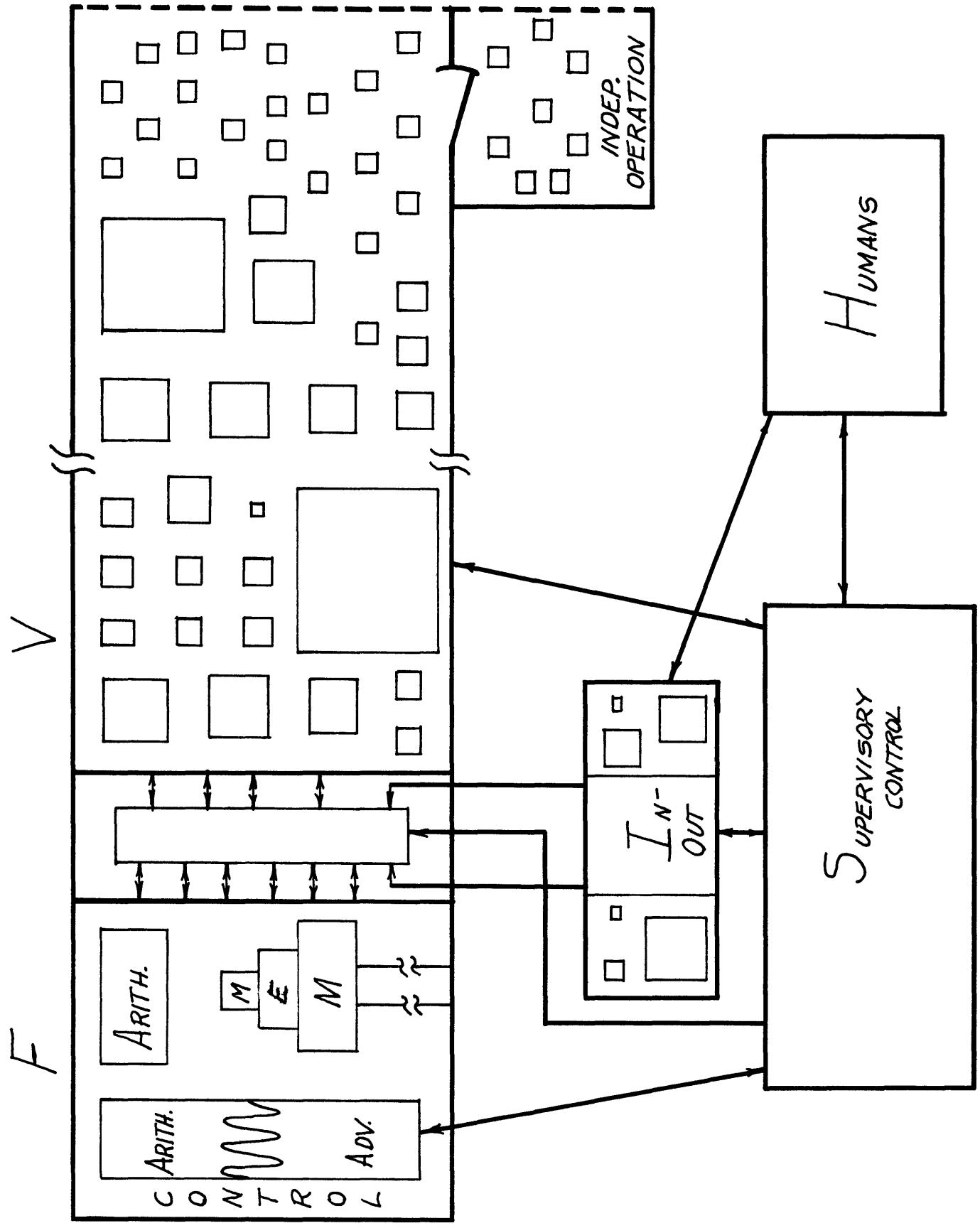

F
V
ARITH.
M
E
M
CONTROL
ARITH.
ADV.
IN-
OUT
INDEP.
OPERATION
HUMANS
SUPERVISORY
CONTROL

Figure 1

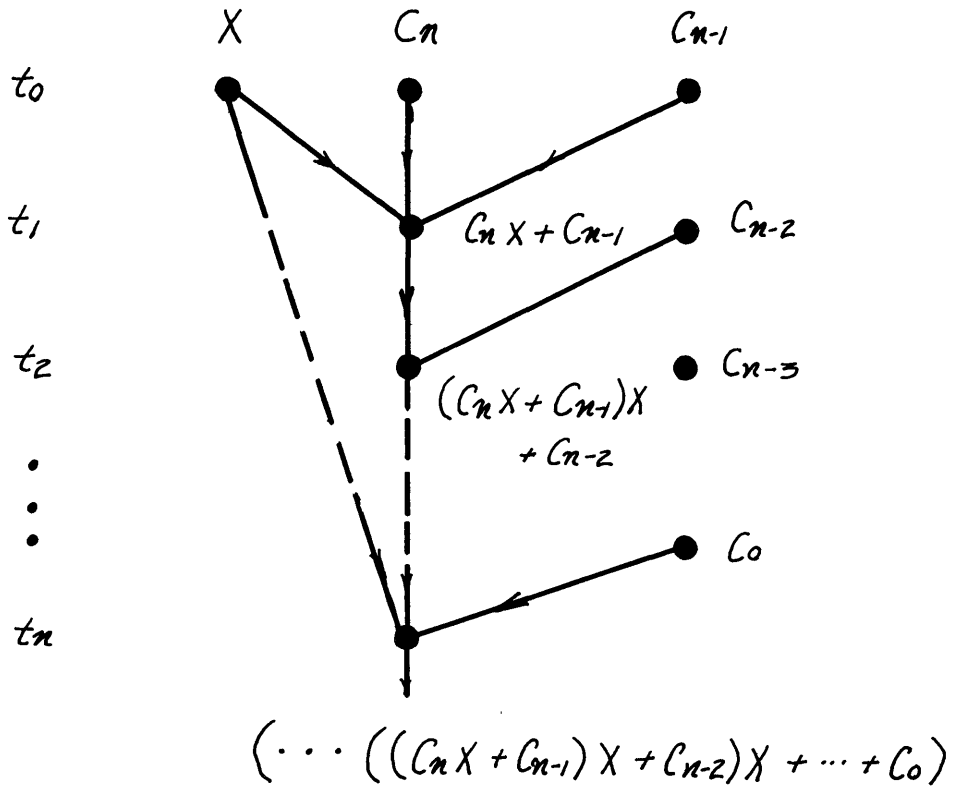


Figure 2 a

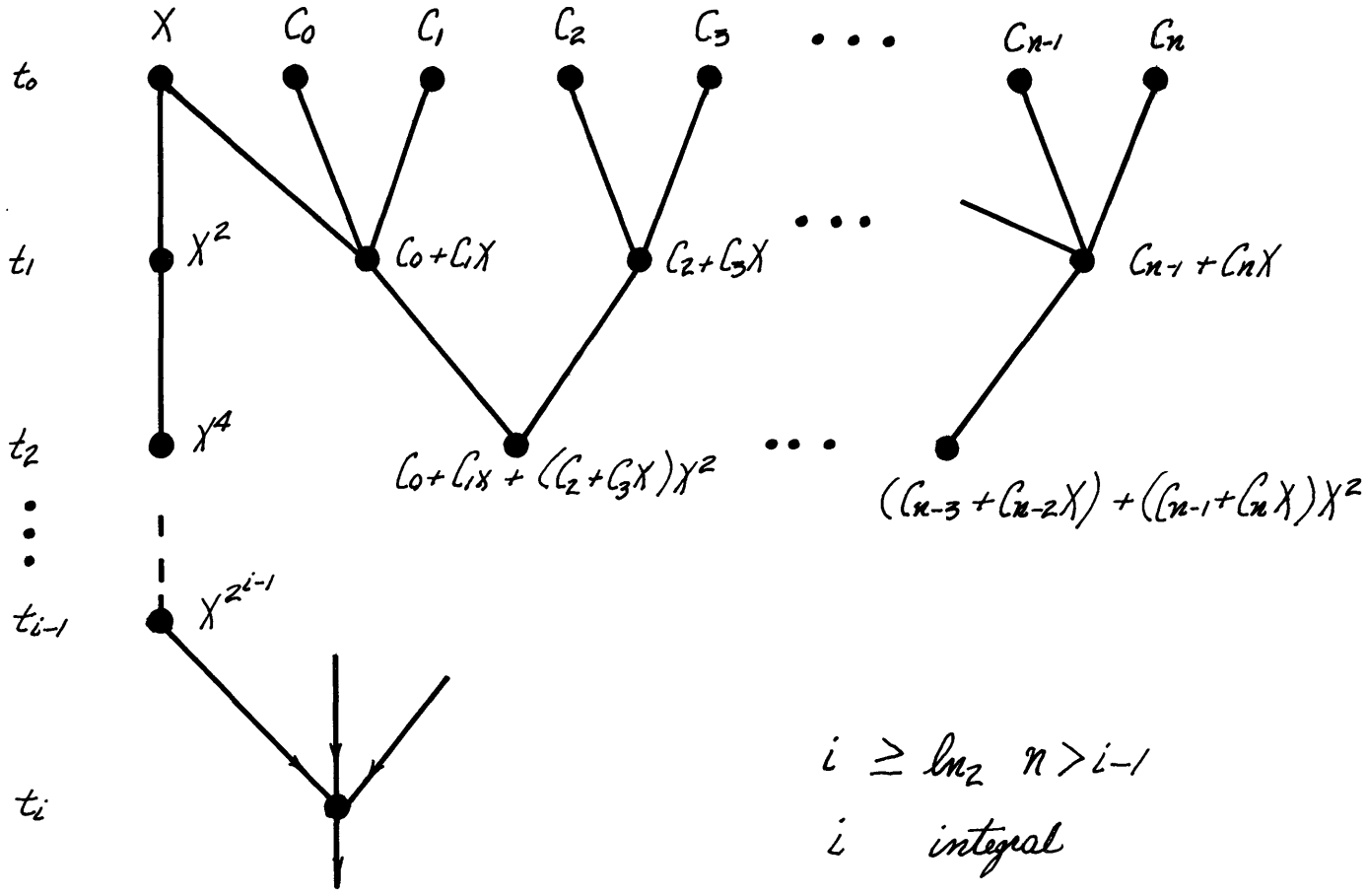


Figure 2 b

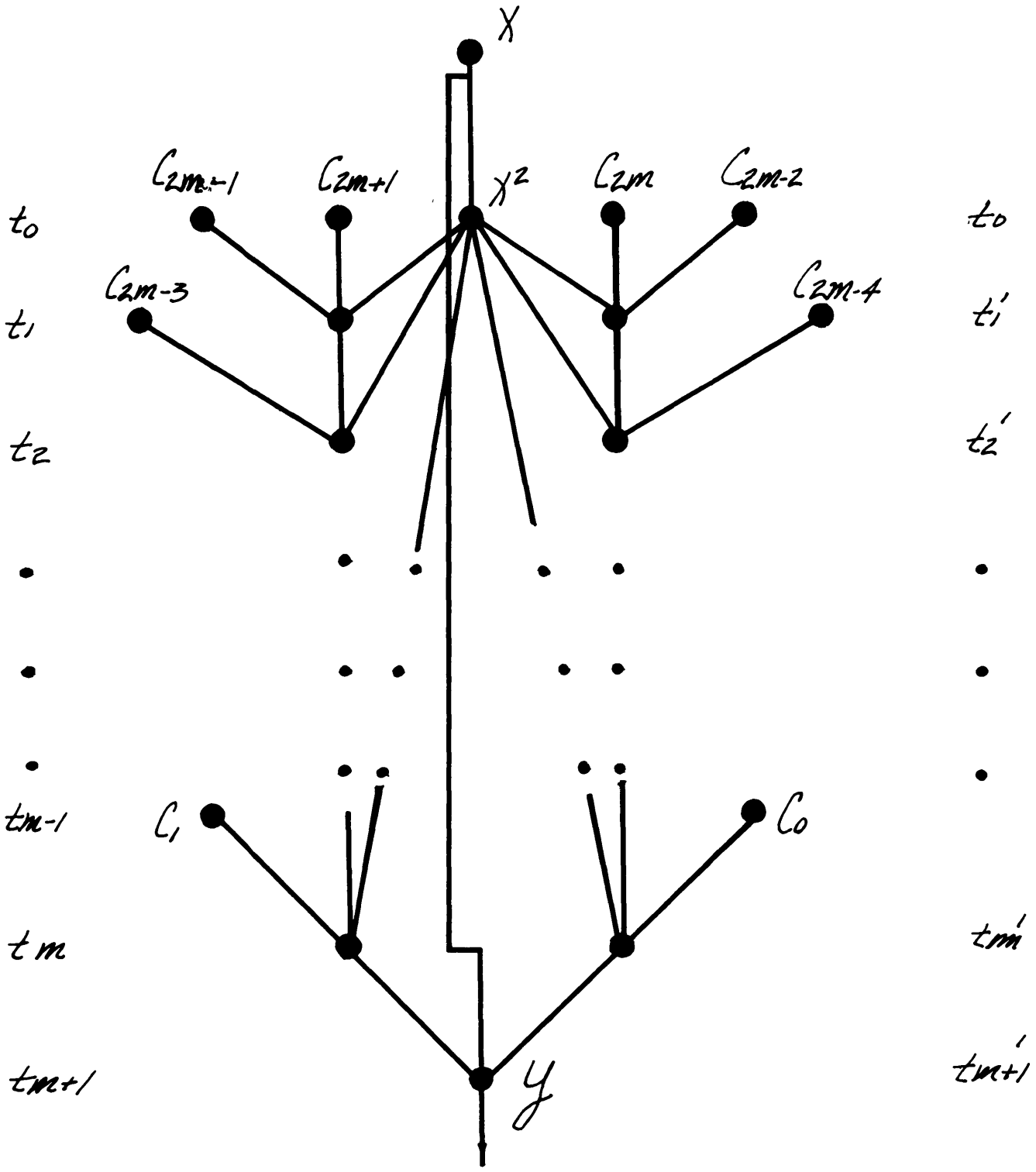

X
C_{2m-1}
C_{2m+1}
X^2
C_{2m}
C_{2m-2}
t_0
C_{2m-3}
C_{2m-4}
t_1
t_2
t_{m-1}
C_1
C_0
t_m
t_{m+1}
Y
t_0
t_1'
t_2'
t_m'
t_{m+1}'

Figure 2 c

HORIZONS IN COMPUTER SYSTEM DESIGN

By
Walter F. Bauer
Intellectronics Laboratories
Ramo-Wooldridge
Canoga Park, California

Summary

"Computer system design", or using the new phrase "computer organization design", has not received nearly the attention that the design of circuitry and components has received. Circuitry and component development tend to provide the user with higher operating speeds at lower cost but do not necessarily provide the user with a system easily adaptable to his particular needs. There is a great need for computers which can be adapted, as a total system, to the particular needs of the user. The changing needs of the user result from considering the computer as one part of a larger information handling system where man-machine communication is of great importance. There are a number of trends in the organization of computers such as concurrent operation, modularity, control hierarchy, and integral interrogation and display, which help meet the needs of these more sophisticated applications. Analysis shows some of the benefits of modular design. In addition to the design of the computers themselves, the analysis techniques for investigating the application, are appearing. These involve queuing theory and the development of methodologies for information systems analysis and the evaluation of computer applications.

Introduction

It is interesting to note the phrase "computer organization" in the title of this session. For some time now, those of us interested in the broader systems aspects and user aspects of computer design, have been using the phrase "computer system design". To many, however, the phrase "computer system design" still has the connotation of circuitry and component design or even that branch referred to as "logical design". Although Webster defines "organization" and "system" with phrases that have identical meanings, to many the phrase "computer organization" will have a different meaning than "computer system". The former clearly refers to broader systems aspects of computers such as instruction repertoire, communication with magnetic tapes, systems expandability, etc. Perhaps "computer organization" will stick; at any rate, the word "system" is slightly less overworked.

The thesis of my talk is that insufficient time and energy has been devoted to the analysis of computer organization; the developments in organization have been overshadowed by developments in circuitry and components. There have been some recent developments, however, in computer organization which indicate growing emphasis in this direction. It is my plan to review some of the changing needs, and the trends in computer organization to meet these needs, and to further discuss certain techniques in the analysis of information handling systems.

The Status of Computer Organization

In the general design of computer systems, three areas of activity come to mind: circuits and components, programming, and systems design or computer organization. There have been noteworthy developments in circuitry and components, many of which could be classified as breakthroughs: the development of large-scale magnetic core memories, the development and use of the transistor, the use of printed circuitry, and computer automated design techniques. We are in the midst of still higher speed circuitry developments with microwave techniques and nanosecond speeds becoming discussed more frequently. Similarily, there have been somewhat less noteworthy, but significant, developments in programming: sophisticated compilers, formula translators, complete programming systems for automatic computer operation; more recently we have seen the development of problem oriented languages such as COBOL (Common Business Oriented Language) and programs for

translating between various computer languages.

But the area which has, from my point of view, received far too little attention, and the area which we are discussing here this morning, is computer systems design or computer organization. Until approximately 1958 there was little progress in system design. There had been a gradual improvement in instruction logic with index registers becoming common and certain concepts like the indirect address, coming into use. The idea of buffering data between the input - output was advanced and the interrupt was adopted as a powerful technique for matching the computer approach to asynchronous external devices. While these advances are significant, they do not in any sense compare with the giant steps taken in circuitry and component development.

Happily, during the period since 1958, a number of significant advances have been made. The idea which now appears as obvious, of having a computer operate in a truly parallel fashion was first embodied in the GAMMA 60 and the LARC computers. The designers of these machines saw the possibilities of increasing speed through having large parts of the computer operating simultaneously rather than, or perhaps in addition to, increasing circuitry speeds. While it cannot be proven that parallelism or concurrent operation is more economical than higher speed circuitry, I believe that few would argue that increased parallelism can bear fruit and that it has not been pushed far enough or fast enough.

We are on the verge of other breakthroughs in computer organization. The concept of modularity coupled with sophisticated intramachine communication, such as that found in the Ramo-Wooldridge RW-400 computer, will undoubtedly receive a great play in the future.

Application Horizons

For the purposes of our discussion here, consider the applications field as being divided into three parts: computing systems, control systems, and information systems. "Computing systems" here refers to the mechanization by electronic computers of processes which are being done or can be done semi-automatically or manually. For example, partial differential equations were solved for many years by manual means. Similarly, business data processing was performed by a variety of machines before the main bulk of the activity was taken over by the electronic digital computer. Computing systems constituted the first application phase of electronic computers; this application will continue to grow and flourish and is, and will remain for many years, the major application area of digital computers in terms of dollars spent.

The "control systems" applications area of electronic computers is a much newer one. Perhaps the first notable control system was the Air Force's SAGE System for Air Defense. Other systems such as refinery process control are appearing. This applications area is possibly best characterized by saying that it provides closed loop control. It frequently requires specialized input - output for the computer, and seldom requires human intervention -- the human only serves to monitor the process. Indeed, an important characteristic of control systems application of computers is that the human can not carry out the process through manual or semi-automatic means because of accuracy requirements, speed requirements, and the requirement for reliability in spite of monotony.

But the newest and most exciting applications area for electronic computers is the so-called "information systems". The lines of definition between the three applications areas are not clear cut and it is difficult in some cases (and unimportant) to decide in which area a particular application belongs. However, information systems are probably best characterized as follows:

1. The computer is imbedded in a system involving data receipt from many sources, information processing, information display, and information dissemination. The computer itself is a relatively small part of the operation.

2. Automatic data inputs from remote sources are part of the system. Data may be control data or data to be processed. The familiar digital data link involving wire communications or radio communications is employed and speeds up to 5500 bits per second are achieved.

3. Man-machine relationships are of the greatest importance. Equipments for the display of information are an integral part of the system as well as equipments to interrogate the memory of the computer. Virtually all the memory of the computer is available to analysts or operators, and many analysts can consult the memory or use the computer facility simultaneously.

4. Processing is characterized by the processing of independent requests for service. The ratio of tolerable delay to processing time is low as compared to most applications of computers for scientific and business purposes.

5. The requirements for reliability are very severe. Electronic failures which cause the complete outage of the system, are intolerable.

6. The information system must have the provision for growing as an integral system. This growth must take place in an orderly way and without lengthy periods of inoperability during change over.

7. The system must respond to a large variety of requests and it must adapt itself to the particular processing requirements of the instant.

Any information system may have these characteristics to varying degrees. In some cases, some of the characteristics are absent completely. For example, the requirement for absolute round-the-clock reliability may be absent. Another comment is in order: frequently these so-called information systems perform closed loop control of devices in addition to providing information to which humans react. In other words, the definitions for the various applications areas here are suggested to provide general guide lines and not absolute lines of demarcation.

Needless to say, information systems is the newest and most sophisticated applications area for electronic computers. Consider for example the use of interrogation devices and display devices in connection with computers. They are almost non-existent today. However, they are beginning to appear as standard pieces of computers and the possible uses of these devices are appearing in numerous places, frequently in unexpected areas.

Quite possibly the market for information systems will dwarf all other market areas eventually. The road ahead, however, is not without its serious obstacles. Programming techniques must be developed much further to allow the efficient communication between man and machine that is made possible, at least, by hardware inquiry stations and display devices. The automatic report generator techniques in computer programming are a beginning step in this direction. Also considerably more work must be done in the overall systems analysis to make good use of computers. It has been frequently stated that computers attached with automatic data input equipment, display and interrogation equipment can be used by corporate top management to conduct their daily business and more specifically, make computer-aided decisions. Operations analysis and decision theory must be developed considerably further before such uses of the equipment can be made.

Computer Organization Trends

Computer Adaptability

Information systems, however, give rise to a different set of requirements for the customer than has been previously seen. Almost all of the needs can be summed up in one short comment: There is a need for computers which can adapt to problems. Up to this time, computer customers have found it necessary to adapt their problems to the computers; the computer is purchased as an entire off-the-shelf system and the customer does as best he can to make his problem conform; gradually the problem fills the computer and the customer starts all over again worrying about the next phase.

In the never ending search for speed, many of the requirements of the customer -- many of which now seem obvious and fundamental -- which would make the computer a saleable item, were neglected. Everybody wants to sell computers but it seems that nobody wants to design computers attractive to buyers.

As stated above, the computer must be adaptable; it must be adaptable to the particular

characteristics of the application, it must be adaptable to the requirements which may change from minute to minute or from millisecond to millisecond, and it must be adaptable to changes which occur throughout the years.

Adaptability can undoubtedly be best achieved through modularity; the two thoughts are almost synonomous. Having modular components which can be added (or deleted) as the application changes provides the long term adaptability. Adaptability on a micro-second-millisecond basis is provided by connecting the modules by high speed switching equipment.

Instruction Repertoires

Adaptability to the requirements of the user can appear in many places. One of these places is in the instruction repertoire. Some advances in computer instruction repertoire have been made during the last few years as I noted previously. However, techniques do not exist today to determine in some quantitative way, the power of a set of instructions as applied to a given information processing problem. Perhaps in this connection one should borrow an idea from the mathematical physicist. Consider treating a given application as an abstract vector space which is "spanned", as the mathematical physicists say, by the set of computer instructions. The set of instructions can be considered to be "independent" according to some definition of "independence" and as is done in theoretical physics in the case of a set of vectors. The set of independent vectors is "rotated" to minimize the "energy" required to "span the (problem) space". In other words the set of instructions should be chosen to be minimal from the standpoint of the processing required and with respect to a particular problem application.

A possibility in instruction logic is the idea of building computers which can change their form through changing the set of instructions they provide. This technique is frequently referred to as "micro-programming" and has fallen into a state of considerable disrepute. However, there are many good reasons to exhume the remains. The idea will look promising with the following provisos: if the programming is not burdened unduly by a new set of details, and if the computer can operate at nearly the same speeds as its wired-in counterpart, and if criteria can be established for the intelligent choice of computer instructions such as is suggested in the above paragraph. A possibly attractive idea is to build a basic computer package for marketing, and design it so that the buyer or seller can "particularize" it for a given application.

In a session[1] similar to this one at the EJCC in late 1958, the fact was deplored that instruction repertoires were not oriented toward automatic programming and translation between computer languages. Unfortunately nothing has happened to make that indictment less justified.

Information Display and Interrogation

Man-machine communication is undergoing important changes. These changes will probably result in most large scale computers being designed from the outset to include display and interrogation consoles. Our first computers, and most of our present ones for that matter, allow the display of perhaps 100 lights to signify the state of the computer or to provide information to the user-operator. In most cases, approximately the same number of on/off switches are available to communicate with the computer. Of course, these computers were not specifically designed with sophisticated man-machine communication in mind. Our modern computers designed for a tight man-machine communication loop involve the presentation of thousands of bits to the user-operator in assimilable form while allowing him communication switches in roughly equal numbers. With most systems currently in use, information in the form of printed copy comes from the machine; the user must usually wait hours -- and frequently days -- before the new set of information, which he requests as a result of what he has just previously learned, can be made available. Not so, of course, with the new display-interrogation operator consoles such as those found on the RW-400 computer.

Computer Control Hierarchy

Figure 1 depicts a technique increasing in popularity in computer design and computer use -- that of establishing a hierarchy of computer control. A possible result of this "status seeking" of computer operations is

shown in the figure. The Systems Management function is the highest level of control; it would be responsible for the systems and procedures of the operation and the problem priority requirements. If, for example, a certain class of problems were not getting the required service, this level of control would change the priority of that problem class. The Systems Management control level would perform utilization analyses which would provide information as to possible equipment changes, and it would provide summaries and records of the way in which the computer system was used. Under this hypothetical model, System Control would be the next control level and would be responsible for the assignment of equipments to do a specific problem and for executing the priorities which were imposed at the higher control level. It would react to stimuli from external sources such as digital data links and the depression of keys at an analyst's console, it would initiate much of the processing required when problems are interrupted, and it would terminate problems as required. At the Problem Control level, macro-instructions would be interpreted and the management of memory would be performed; that is, decisions would be made here as to where data is to be stored, and how and where it is to be transferred. At the Problem Control level, the reactions to internal stimuli would be handled such as the interruption of the problem due to lack of data or the onset of an overflow condition. In addition, at this level the management, programming logistics and subroutines would be performed. The last two levels of control, Programming Logistics and Data Processing and Equipment Control, are the usual kinds of computer control which we have been doing many years. It might be added that a still lower level of control could be identified, namely, "micro programmed control".

Whether the control as indicated here would be carried out by hardware or through programming is uncertain. Probably most of the control would be effected through programming although this control is made possible through hardware. For example, the interrupt signal enables the reaction of the computer to external asynchronous occurrences. It behooves the computer designer to seek ways to make this control hierarchy easy to achieve through programming.

The functions of this hypothetical control model could be carried out in a time-shared way by one serially operating computer or, on the other hand, by computing and control elements working simultaneously and to the extent indicated, independently. Probably the latter technique will be found to be more effective; there will be a trend toward physically separate units performing the indicated control functions.

The importance of a hierarchy in computer control is that it gives the computer a self-organizing character. With such a hierarchy the computer can be introspective much as the human is, and can continually monitor the course of its business to optimize its operations.

Modularity and Reliability

Some of the benefits of a modular-designed computer have been referred to above in discussing computer adaptability. Another advantage of modularity is that it brings about greater reliability at a lower cost.

In Figure 2 there is shown a diagram showing the relationship between modularity and reliability. On the abscissa is the degree of modularity of a computer system defined as 100 minus the percentage of the total represented by the largest module. If, for example, the computer was made up of 10 modules, each of which was 10% of the total, the degree of modularity in this case would be 90%. The ordinate here refers to the extra equipment which is needed to provide a level of reliability equal to that of duplicating an entire computer system. The solid curve shows the amount of extra equipment needed to provide the same degree of reliability as obtained by doubling up on the amount of equipment in the case of the computer with zero modularity. It is assumed that all modules have the same reliability.

There are many vagaries in interpreting a graph of the kind shown in Figure 2, and the results there are meant to be more qualitative than quantitative. However, it can be shown through simple mathematics that the computer 90 per cent modular requires only 30% extra equipment to get the same reliability as duplicating the entire single computer with no modularity, where the

assumption is made that the modules are each no more reliable than the entire computer without modularity. The dashed lines showing the total equipment refer to a rough estimate of the equipment needed plus a rough estimate of the additional equipment needed for communication among the modules. It seems clear that as the modularity proceeds closer to 100% -- that is, closer to the transistor or diode level -- that the equipment needed for communication becomes very great, and possibly unbounded in the limit. Experience plus some analysis shows that there is probably a minimal point lying somewhere between 80 and 95% degree of modularity.

Computer reliability, modularity, and control hierarchy, incidentally, team together in an important way. One of the higher levels of control, probably System Control of Figure 1, monitors equipment failures. When failure occurs the module not in use is switched in. Or, if all modules are in use, the lower priority problems are temporarily put aside. In other words, catastrophic failure does not occur; the system automatically adapts itself to carry on the processing with only slightly degraded performance.

Computer Memories

Progress in the system organization and use of high speed memories is particularly lacking. We have made memories bigger and faster, but that is about all. Some discussion has taken place of the virtues of small ultra-high speed memories for "scratch-pad" use, and it has been adopted in a few cases. A few possibilities for new uses have been advanced[1] but there has been no embodiment of anything organizationally or logically new and, more surprisingly, no analysis has been performed.

Referring to Figure 3, there is a remark that could be made about the duty cycle of large scale memories. Probably the curve of memory usage vs memory size for large scale memories is similar to that shown in the figure. It certainly is true that the absolute requirement for 30,000 words of storage occurs far less frequently than the requirement for 4,000 words of storage. It is realized of course, that this is a function of the problem and a function of the programmer's tolerance of red tape programming logistics. However, since 32,000 words of memory are on hand in the hypothetical example, there is an extra capacity as represented by the part of the rectangle above the curve. This extra capacity is roughly proportional to the excess capacity for the memory device in excess of the "theoretical" amount needed as delineated by the curve. Therefore, it seems that if the computer were designed to share this memory with other problems and if a system could be devised to switch memory assignments, the duty cycle of the memory could be greatly increased and efficiency would accrue. The curve and the large area above the curve suggest that a considerable fraction of the cost of the memory could be spent in the hardware to effect a sharing of the memory and still a saving could be achieved.

Information Systems Analysis

There was reference above to the need of computers which adapt to problems. The question immediately arises, however, as to deciding what the needs of the problems are. It is clear that if information systems are to take their respected place in our scientific world, we must develop ways of analyzing them.

Queuing Theory

One of the disciplines which might be applied is queuing theory. It was stated above that information systems are characterized by a low ratio of tolerable delay to processing time for a service request. It is this characteristic which makes a queuing theory approach possible. A non-mathematical statement of the problem would go something like this: service requests for the processing of data arrive on a random basis and it is desired to know the minimum equipment necessary to provide the servicing of these requests while meeting system performance specifications stated in terms of maximum tolerable delay or average delay.

Figure 4 shows a curve which relates the processing speed of a computer system to the average delay in processing a request. The curve is similar to that given by Ackley[2] but the variables have been normalized so as to make the results generally applicable. An important implication of this curve is the following: the average service demand rate

in almost all applications goes up as the system develops. If the processing speed does not at the same time go up, the ratio given by the abscissa is reduced and the average delay increases as the curve shows. Obviously then, there is a requirement to have the computer designed so that its processing speed can change according to the demand for service so as to keep the average delay within tolerable bounds and still not have a period of time when the system has excess capability and excess cost. Modular computers provide exactly this ability to increase their processing speed. Computing elements can be added to the system to keep pace with the demand and to keep, consequently, the average delay bounded.

Information Systems Design Methodologies

In the design of information systems it is important to develop techniques for systematically fitting the computer requirements to the operation requirements of the system.

Figure 5 shows a method for carrying out such a design. The operational requirements give rise to the identification of certain data types, volumes of data, the processing required, and the distribution of requests for service. Together they define a problem. If an initial assumption is made as to the speed for processing each one of these types of data, the entire process can be analyzed. Probably the technique used here is simulation by large scale digital computer. This gives rise to information on the average length of delays which, when compared with the delay requirements imposed by the operational characteristics of the system, will determine whether the service is tolerable or not. If the service is intolerable, then assumptions must be changed as to the speed with which the processing is performed. Having determined that the service is tolerable, and having determined the speed requirements, an analysis can be made to determine the complement of equipment necessary. After this is done, the total system is analyzed in view of programming requirements to determine whether the design is in all respects acceptable. Possible imbalances in the service or in the amount of equipment can result in a further analysis and a change in the equipment complement.

An analysis was performed at Ramo-Wooldridge similar to this and was reported by Rothman[3]; the results are shown in Figure 6. The problem mix gives the characteristics of the problem. For example, 50% of the problem required servicing requests which had to be completed on an average of 0.5 minutes and with a total processing time of 1.5 minutes. Poisson distributions were assumed and the frequency of arrivals is given essentially by the abscissa in terms of the average number of requests arriving translated to the number of computers to handle this average load. The zero per cent curve shows that if the number of computers equals the average load, then there is a zero probability of servicing the problem mix. As an excess of computers is applied, service improves as the results indicate.

In many cases, however, the queuing theory approach is not applicable. There is a need for a more general approach to evaluating the applicability of a computer to a given problem. What is sorely needed then is a technique for developing the important and significant characteristics of problems and, likewise, the important and significant characteristics of computers.

The general thought is portrayed in Figure 7 where, starting with the general problem characteristics and the general computer characteristics, a problem model of general applicability and a computer model of general applicability, results. (The author is indebted to S. Rothman of Ramo-Wooldridge for stimulating discussions on the subject.) If a specific problem is described in terms of this general problem, and a specific computer is described in terms of this general computer module, the theoretical performance of that computer involving the particular problem would be obtained. If the model were deemed sufficiently valid, the results then at this point would be useful. In the development of the methodology however, it would probably be desirable to compare the theoretical performance with the actual performance, providing that the specific problem had been already solved by a specific computer. By making this comparison one determines what problem model changes and what computer model changes are necessary. These then can be sent back to improve the problem model and the computer model.

Conclusion

That there has been little progress in computer system design or organization stems from the fact that computer technology is relatively new. That excuse, of course, ages rapidly. A contributing factor is that most commerical computer manufacturers are inherently conservative, and see increased circuit speed as the only solution. Unfortunately other groups which are in a position to promote progress, have not always been sufficiently perceptive. However, "right is might", and the fruits of these approaches will most likely become increasingly manifest.

References

1. "Computer Design from the Programmer's Viewpoint", W. F. Bauer, Proceedings Eastern Joint Computer Conference, December 1958.

2. "The Multi-Sequence Computer as a Communications Tool", J. N. Ackley, Proceedings of the Eastern Joint Computer Conference, December 1959.

3. "The RW-400 Data Processing System", S. Rothman, Proceedings of the Auto-Math Conference, International Congress of Information Processing, Paris, June 1959.

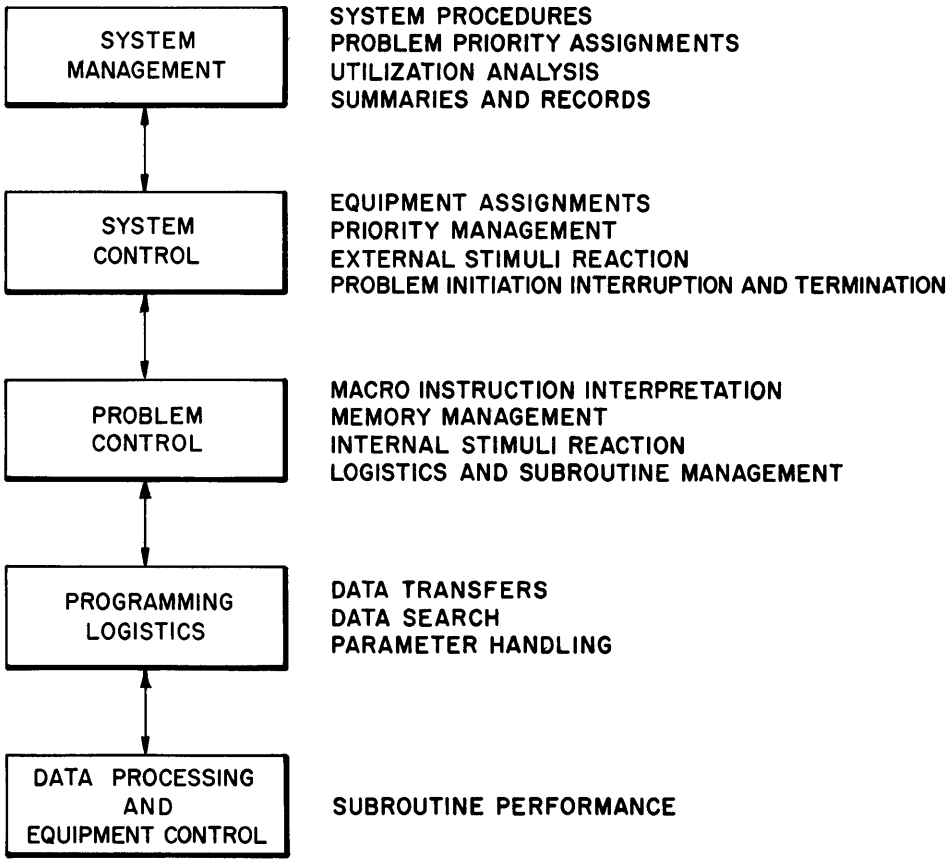


Figure 1. Computer Control Hierarchy

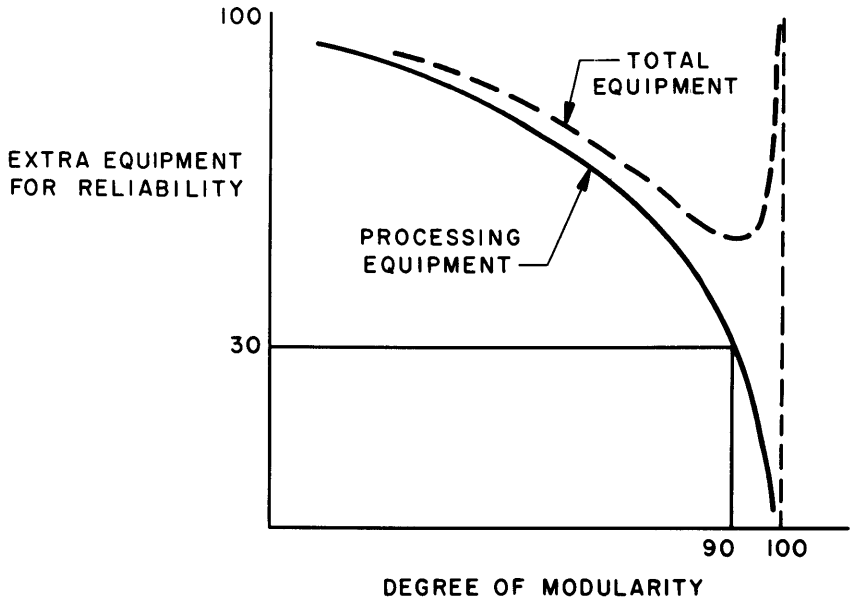


Figure 2. Reliability and Modularity

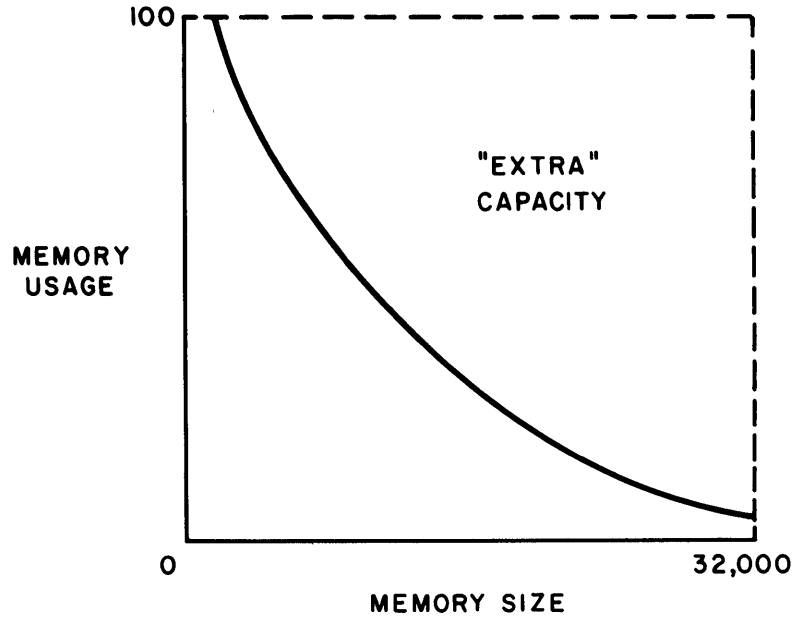


Figure 3. Memory Use

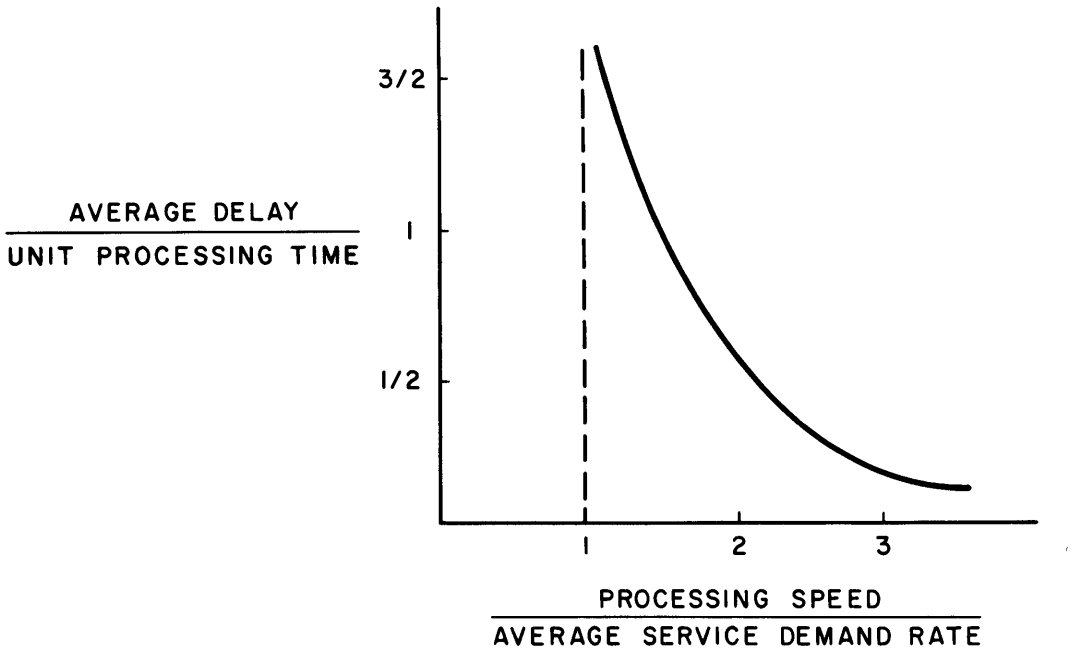


Figure 4. Queuing Theory Application

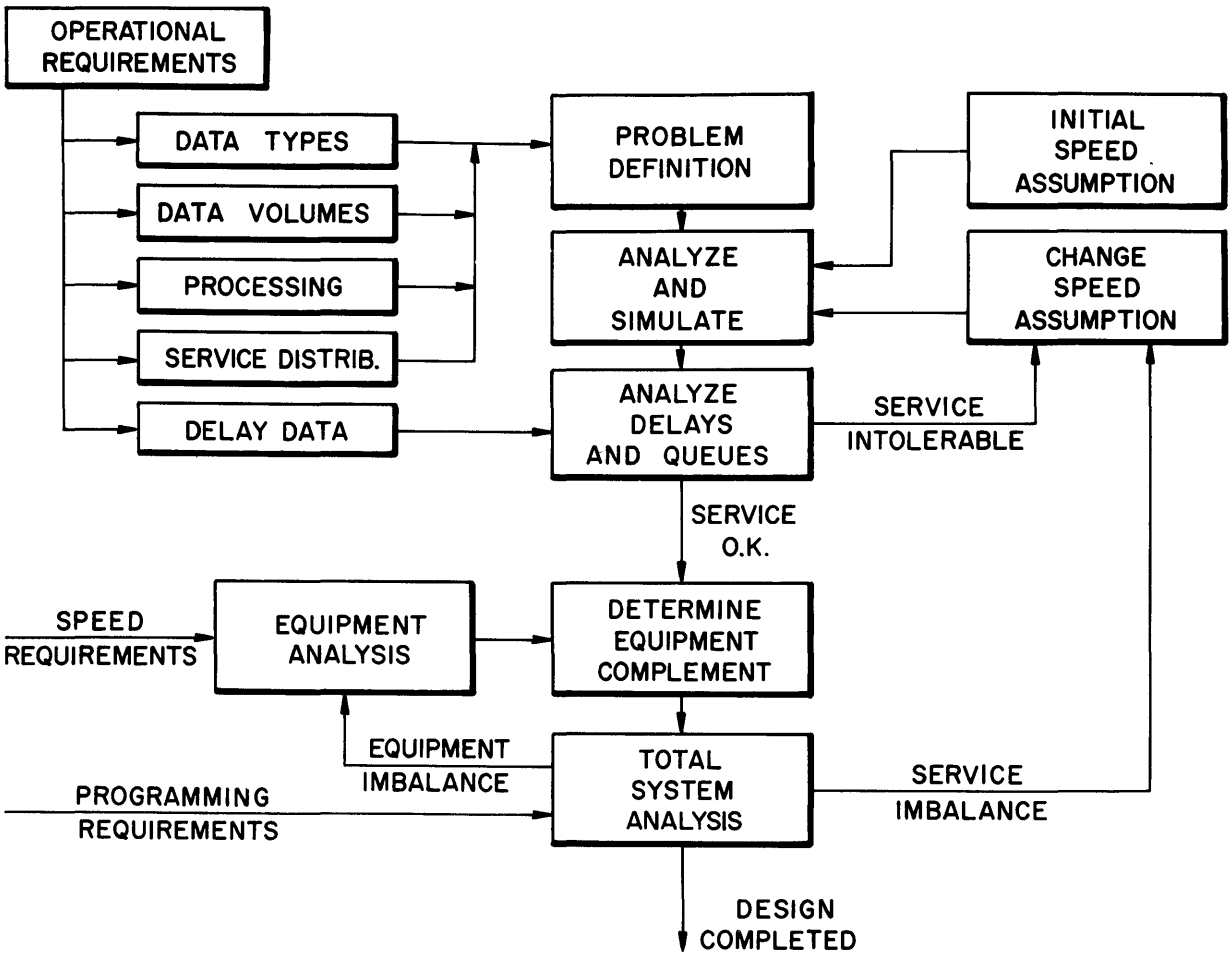


Figure 5. Information System Design Methodology

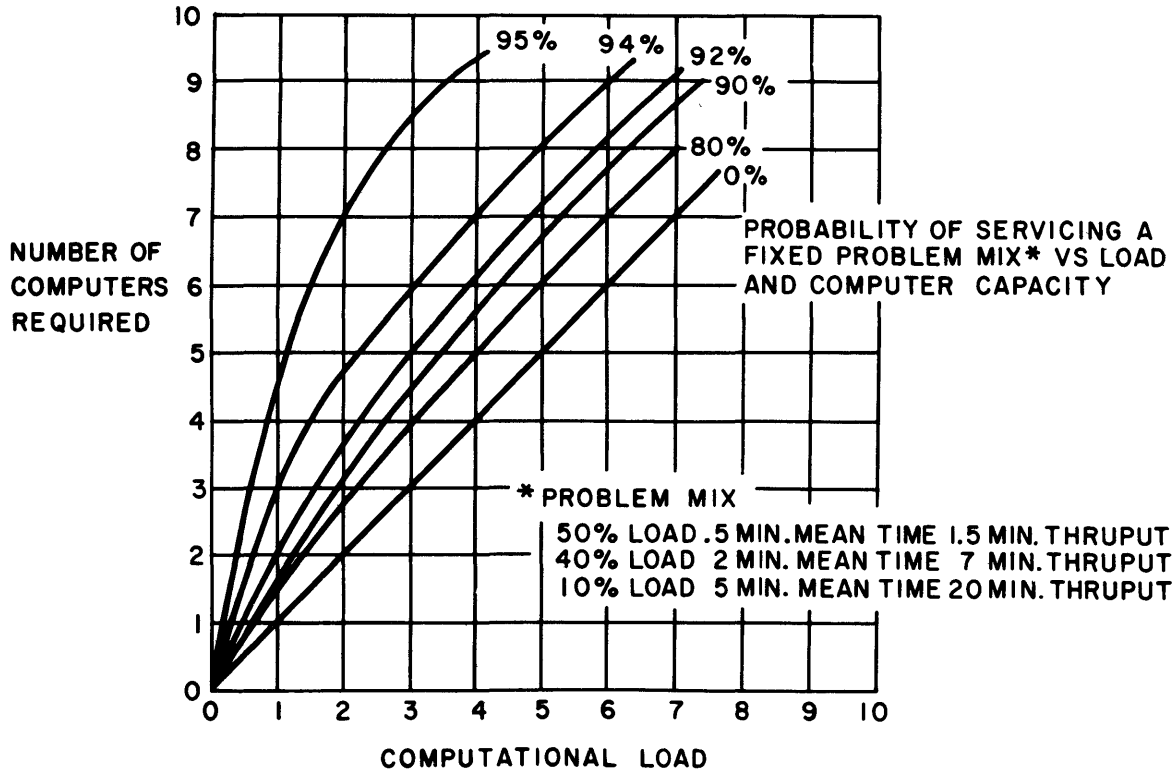


Figure 6. Queuing Analysis Example

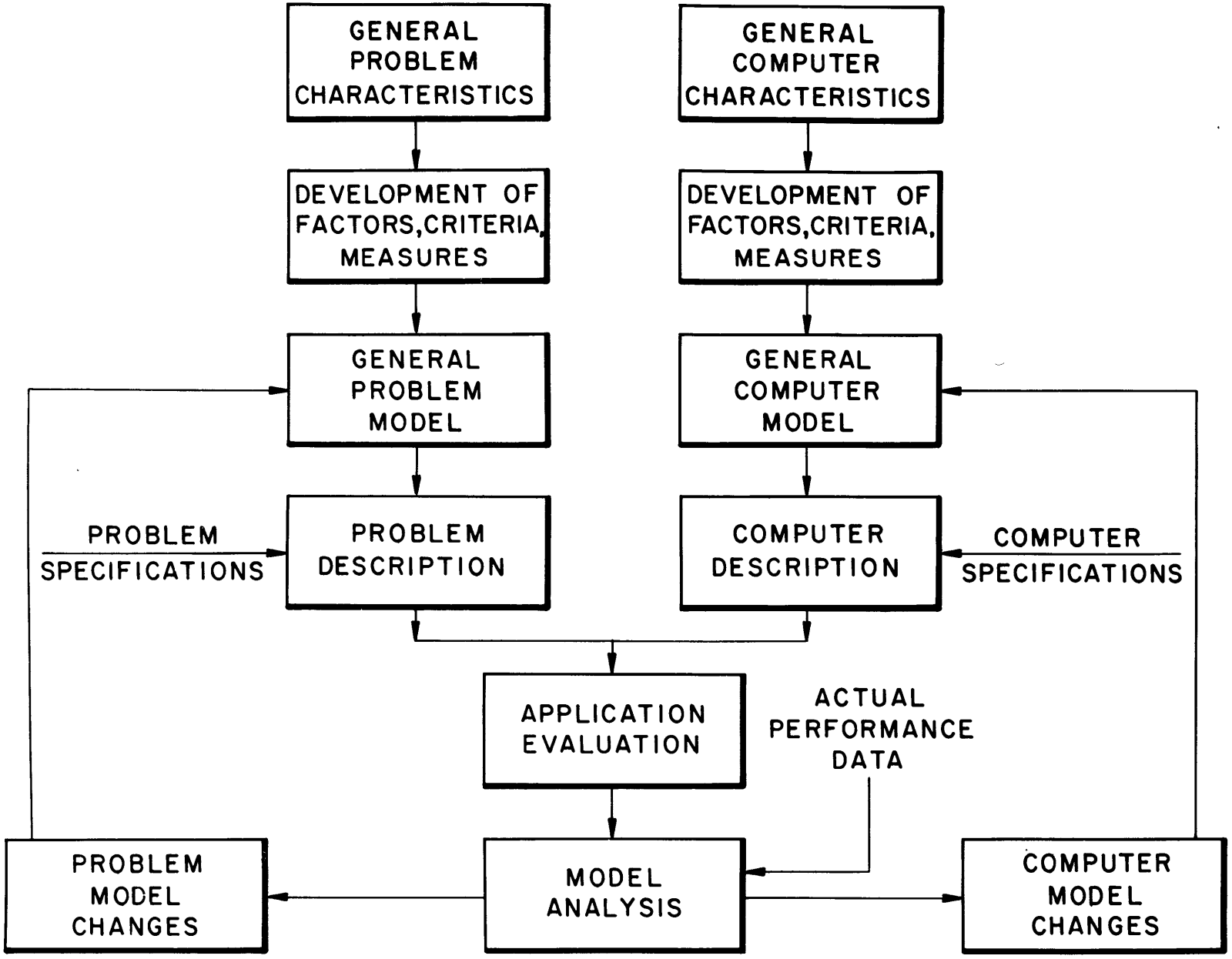


Figure 7. Application Evaluation Methodology

A MULTI-LEVEL FILE STRUCTURE FOR INFORMATION PROCESSING

L. Miller, J. Minker, W.G. Reed, W.E. Shindle

Astro-Electronic Products Division
Radio Corporation of America
Princeton, N.J.

Summary

The main objectives of this paper are to present a multi-level file structure for natural or textual language, and a computer algorithm which logically associates elements of the presented file structure. These objectives are met by showing how the formalized data is mapped into a vocabulary that is amenable to automatic computation. This vocabulary consists of a set of indices which retain the syntactual association of the inputs. These sets of indices constitute a file organization which employs an associative list-type structure of the type utilized by Newell, Simon and Shaw in their Information Processing Language.[1]

The problems of natural language input and how they are surmounted are described. The detailed multi-level file structure can be used with systems whose input includes textual language that may be structured or formalized into an "attribute-attribute value" format, whose accuracy and/or reliability is questionable, and which is received in disjointed elements that have to be synthesized into meaningful associations.

I. Introduction

This paper is concerned with an application of digital computers to the field of information processing. In particular, we shall restrict ourselves to the processing of highly formalized information specified in a natural language, exhibiting characteristics which prove it a source of data for automatic development. Although the automatic development of information is restricted to formalized data alone, information of a non-formalized nature which relates to the formalized data can be retrieved also by these techniques.

In the first portion of the paper, a multi-level file structure to accommodate the processing of the information is developed. An application utilizing this file structure is then made to illustrate its usage in the retrieval and association of data already existing in the system.

The work reported on in this paper was sponsored by OACSI, the Office of the Assistant Chief of Staff for Intelligence, Hq. Department of the Army. The work is part of a study nicknamed Project ACSI-MATIC, concerned with the potential uses of modern data-processing equipment and procedures in the activities of certain headquarters military intelligence operations of the U.S. Army.

The nature of the problem can best be characterized by describing it as one in which the processing of free language plays an extensive role. This is a consequence of the fact that the basic input to the system is in the form of reports and documents in which pertinent events and conditions have their statement largely in a narrative language. In the computer approach which has been proposed as a solution, this narrative characteristic of the input is formalized in order to take advantage of the computer's ability to process information.

A prerequisite for the utilization of this ability of the computer is the necessity to elicit formalized information from the input. The substance of the information in the input was subjected to a detailed investigation to determine whether or not a formalized structure could be imposed upon it. A set of formal categories was determined in the consideration of the body of information constituting input to the system. These categories are descriptive of the major elements which the system encompasses and have relationships among themselves which reflect similar relationships between the elements which they represent. The elements enter into relationships of synonymity and hierarchy which the system must establish and confirm.

The specific set of categories chosen are termed descriptors of the system. A particular instance in a category or descriptor class has been termed a descriptee. There is a set of descriptor classes which has the special feature that a single descriptee in any one of these descriptor classes uniquely identifies a major element of the system. Any descriptee from a descriptor class in the special set has this property. An associative relationship can be established for any set of descriptees in these special classes which denote the same element. The special descriptor classes having this feature are called biunique descriptors, and the descriptees included in them are called biunique descriptees.

With the descriptor classes explicitly formulated, the descriptees can be recognized and extracted from the input documents. An input specialist then assembles these descriptees into modules called multiplets which form the basic source of material for automatic processing in the system. These multiplets are stylized aggregates

of descriptees wherein the descriptor classes represented provides an index to the special assembly of subroutines in the internal system which the multiplet requires in order to provide for its full development. This is a consequence of the fact that relationships between the descriptees in the multiplet have been predetermined and are dependent upon those descriptor classes represented.

A typical multiplet of information to the system would be as follows: (D, x_1; C, x_2; L, x_3). It reads that the descriptee x_1 of the descriptor class D is related in a unique way to the descriptee x_2 of the descriptor class C and is located at x_3 (the location descriptor class is indicated by L).

When the multiplets of descriptees are extracted from an input document there usually remains a residue of information which has pertinence to the problem, which must enter into the system. These fragments, as they are called, do not participate in automatic processing within the system, but serve as a background against which the associated multiplets can be better developed. The associated multiplet provides a coordinated index to this fragment. A basic feature of the system is this relationship between multiplet and fragment and the manner in which it is preserved.

As an introduction to the file structure, the automatic development that takes place with respect to an input multiplet will be briefly discussed. There are two stages of development of the multiplet. The first stage develops and standardizes the descriptees within the multiplet, independent of the other descriptees. The second stage utilizes the specific combination of elements comprising a multiplet.

The first stage of development is best illustrated by the following simple example: The desired machine input when locations are mentioned might be coordinate, city, and state, if a city is mentioned. However, inputs to the system might contain a coordinate and city without mention of a state. It is therefore necessary to develop the descriptee and standardize it before further processing commences so that at a later date one can retrieve the information under the hierarchic location term "state". Unless the data is initially developed before decisions are made about it, both information processing and information retrieval might be difficult and time consuming, if not impossible. The adequacy of information retrieval is to a large degree determined in the input stage.

After initial development and standardization of the information contained within the multiplet, it is then possible to process the data against the existing information. The system logic, the content of the files, and the input information define the processing that takes place within the formalized files. In a sense each multiplet determines its own processing through the system. The objective of processing multiplets is to categorize the information with respect to all other information within the files. To categorize the information a substantial amount of retrieval and association of data is often required. Another portion of the system logic specifies whether or not attempts should be made to search for, retrieve, and determine associations between the input item and the file information. If it is so specified and associated information exists, the machine logic augments the input multiplet with the associated data and attempts to categorize this new multiplet (testing for consistency, completeness, etc.). Rules are specified whereby the machine can automatically resolve certain situations of a general nature. In clearly defined and precise situations the machine simulates man's intelligence.

Throughout the system man plays an important role in analysis and decision making required with respect to the data. Based upon the categorization of the information the man must specify what the ultimate conclusion of the machine processing should be. At the present stage of development the machine logic only determines the consistency consequences of decisions, but it is dependent upon the man for the final disposition of the results.

The ability of the system to logically associate information is important for two reasons. In the first place, by associating the information in the multiplet with other multiplets, it might be possible to relate terms that previously did not appear to be related giving man the capability to retrieve information that he would not have been able to obtain had the multiplet been disassociated. Secondly, forming associations is useful for systematically purging data from the file of information to automatically retire data. This is an effective way to keep a dynamic system from growing beyond the bounds of computer storage media. Retirement of outmoded data is the only long-range effective means of maintaining an efficient system.

II. The Multi-Level File Structure

The multi-leveled file structure accommodates information at various levels of formalization and authenticity. The more highly organized information is separated from the incomplete or unauthenticated data and is available as a guide to the other information. One system objective is to elevate information from the unauthenticated class to the authenticated class. The ability to form associations facilitates this function. This structure can be generalized to accommodate systems where information entering it has the following properties: firstly, it can be formalized in the sense described above; secondly, it is questionable in regard to its accuracy and reliability; and finally, it has the property of being received in disjoint elements which have to be synthesized into meaningful associations that will yield maximum information content. The following description, although oriented towards one particular problem,

represents a file structure applicable to a broad class of systems.

The multi-level file structure has these components:

1. Glossaries
2. Index Files
3. Buffer Files
4. Information Files
    a. Authenticated Files
    b. Intermediate File

The Glossaries are a series of files which serve first to standardize and develop the various alternatives which occur for descriptees associated with certain descriptor classes. Certain descriptor classes will have Glossaries associated with them. The incoming multiplets are initially processed into the system through these Glossaries and are rendered into a vocabulary amenable to the computing system. In addition to this function the Glossaries may serve as Index Files (Index Files are explained below). Glossaries are ordered in a natural sequence depending upon the class.

Each descriptor class has an Index File associated with it unless a Glossary is utilized in this capacity. These files are essentially lists of descriptees segregated into their respective descriptor classes. Index Files represent a second level of file organization and are utilized as directories to the Information Files, that is, to authenticated multiplets in the Authenticated Files, and to the unauthenticated multiplets and the non-formalized data in the Intermediate File. The latter is accomplished by associating with each descriptee a machine address wherein this information is located. This machine address constitutes part of the Buffer Files to be explained shortly. Each Index File is ordered in a sequence natural to it (alphabetic, numeric, etc.). The format of an entry in an Index File is (descriptee; reference to a Buffer Address). The subset of Index Files relating to descriptor classes containing only biunique descriptees is termed the Biunique Descriptor Index File and will be utilized later in the paper.

The third level of files are the Buffer Files. To comprehend this file it will be necessary to specify what happens to a message when it enters the system. With every such message is assigned a unique machine address location, termed an index. This index refers to the location where the multiplet, auxiliary data, and the non-formalized textual information (fragment) is retained; that is, the address of the item in the Intermediate File explained below. If as a result of processing, (either by man or machine), the information is authenticated, the machine location of the authenticated information is also noted. The index, or the machine address of the multiplet, is placed in the Buffer File portfolio of each descriptee occurring in the multiplet.

To be more explicit, the format of an entry in one of the Buffer Files is: (index: L_1, L_2). As has been stated, the index relates to an Information File address of the multiplet. This is actually an address in some storage medium; preferably random access. The symbols L_1 and L_2 represent links to the descriptee portfolio. This portfolio or list is maintained in order according to the index number. The link L_1, associated with the first index in the list, is the address of a previous index in the descriptee portfolio; the link L_2, associated with this index, is in turn the address of the next index in the portfolio. The list of indices is cyclic, i.e. the last index in the list is linked to the first index. Therefore it is possible to retrieve an entire portfolio of indices by entering at the first address as provided by the particular descriptee, or more important, by entering at any point on the portfolio chain. Furthermore, the chain need not be given sequentially; it may be scattered throughout the file. This last fact permits a most important and useful file structure, namely all indices within a descriptor class of the Buffer File can be sorted in ascending order with the index as a key. The actual sort may be accomplished by either listing these indices in sequential sorted order or by listing in random order using two links L_3 and L_4 appended to the index. This file structure obviates a great deal of search time since the relative position of an index may be determined within a small area of the Buffer File and only this area need be searched for a particular index.

It should be emphasized at this point that in contrast to data of variable length contained within the Index Files, the entries of the Buffer Files are of fixed length. This allows the Buffer File entries to be utilized easily when required in processing. As will be shown subsequently, considerable usage is made of the indices contained in this file during the processing and association of data relevant to incoming messages.

The Information Files are a set of files which contain modules of associated descriptees and related information wherein the associations satisfy certain specific system criteria. These files represent the main repositories of succinct and organized information about the descriptor classes of the system, and, as such, are the principle source of data for the compilation of special summaries and for replies to interrogations.

One system goal is to determine the authenticity of information. This has dictated a definition of the Information Files into two distinct files: The Authenticated File (AF) and the Intermediate File (IF). If a multiplet is established (by some means) to be authentic, it is placed in the AF; otherwise it is placed in the IF. The Authenticated File contains, in addition to the authenticated multiplets, certain auxiliary information such as dates.

The Intermediate File is the file into which is placed the entire input message, composed of multiplet, fragment, and any auxiliary

auxiliary information peculiar to the system (e.g., date of receipt of input, source of input, etc.) when the information has yet to be authenticated. The location where this input message is placed in the storage medium provides the index for use in the Buffer Index Files, as explained before.

III. Usage of the Multi-Level File Structure

The purpose of this section is to illustrate the usage of the multi-level file structure to determine associations that may exist between multiplets residing within the system. Suppose that it is desired to examine the textual material within the fragments in the Intermediate File relating to some descriptee or combination of descriptees which appear in the Index Files. Given each descriptee, it is possible to retrieve its portfolio of Buffer File indices and utilizing some combination of the Intermediate File indices contained in these portfolios (e.g., their union or intersection) it is possible to retrieve the desired textual information. The file structure, although oriented towards formalized data, can be utilized to retrieve non-formalized data. The technique for retrieving formalized data from the Information Files is identical.

The process of developing information to the extent that a more inclusive fact can be deduced from more than one inclusive fact has been termed "cycle and string formation". The process of determining cycles or strings depends upon whether or not biunique descriptees are contained in the multiplet. Furthermore, the method remains applicable if more biunique descriptors are added to the system other than originally contemplated.

Cycles and strings are defined as follows: Two multiplets are said to be logically associated if they have at least one biunique descriptee in common. Then, if the multiplets can be ordered such that the first is logically associated with the second which in turn is logically associated with a third, etc., a cycle of n constituent multiplets exist if the nth multiplet is logically associated with the first. In a cycle a particular multiplet is logically associated with at least two multiplets: the preceding and following multiplets. If n multiplets may be logically associated in a manner where a particular multiplet is logically associated with only one other multiplet, a string of n constituent multiplets exist.

An example of each is shown in the figure 1, where the ABCD multiplet is logically associated with the ECFG multiplet through the descriptee C. In like manner the ECFG multiplet is logically associated with the HFKL multiplet through the descriptee F. Likewise the HFKL multiplet is logically associated with the BHIJ multiplet, which in turn is associated with the ABCD multiplet through the descriptee B. Notice that the loop BCFH is indeed a cycle in the graph-theoretic sense--hence, the name. If the BHIJ multiplet were absent, the string BCF would be defined.

Whenever cycles and strings are determined, the constituent multiplets describe the same subject. It is therefore possible to integrate the multiplets into one multiplet. All of the descriptees in these constituent multiplets are not necessarily required for cycle and string formation. It is possible for biunique, as well as for non-biunique descriptees of one multiplet, to contradict the corresponding descriptees in another multiplet. Those descriptees of multiplets which are not required for cycle and string formation are termed the residual cycle or residual string.

Cycles and strings are to be formed using all multiplets relevant to a particular input multiplet. It is necessary that all multiplets relevant to the input multiplet be retrievable or reconstructable. In actuality, this is required only for the biunique descriptees of the multiplets. To retrieve these multiplets directly is time consuming, particularly since there may exist many levels of relevant multiplets. To reconstruct the biunique portions of these multiplets is the interesting approach. It will be shown that this can be accomplished exclusively via the Biunique Descriptee Index File and Buffer Files.

No matter how cycles and strings are determined, the complete technique is a three phase operation:

Phase 1. Retrieve all multiplets relevant to a particular multiplet
Phase 2. Form all possible cycles and strings
Phase 3. Validate these cycles and strings.

These phases shall be discussed in order after introducing the notation to be used.

Assume first that all multiplets relevant to the input message have been retrieved. Let there exist a total of n biunique descriptor classes in the retrieved multiplets. Then for the jth biunique descriptor class let t_j be the total number of biunique descriptees of the jth class contained in all these multiplets. Denote the ith descriptee in the jth descriptor class to be D_{ij}; $(1 \leq i \leq t_j;\ 1 \leq j \leq n)$. Now it is possible that a particular descriptee occurs in more than one multiplet. The list of indices to multiplets containing the descriptee D_{ij} is denoted L_{ij} $(1 \leq i \leq t_j;\ 1 \leq j \leq n)$. These lists of indices are the desired output of Phase 1 which is the input to Phase 2.

The technique of retrieving these lists, which will be used in reconstructing the biunique descriptee portions of all multiplets, is accomplished as follows:

1. Examine the Index File for those biunique descriptees contained in the input multiplet. For those that exist, retrieve the associated Buffer File portfolios or lists of indices. Each portfolio will be found in that section of

the buffer file corresponding to the descriptor class of the associated descriptee. Add the index of the input multiplet to these lists tagged with the appropriate L_{ij}.

2. Starting with the first index of the first portfolio, "search" the other sections of the Biunique Descriptee buffer file for this index. This required search utilizes the relative magnitude of the particular index and the sorted order of each section of indices; therefore, the "search" time is comparable to "function table lookup" time. When an index is found, retrieve that portfolio in which it is contained unless it was retrieved earlier. Tag these portfolios or lists with the appropriate L_{ij}. Since there can be only one descriptee for a particular descriptor in a given multiplet, the same index cannot be a member of more than one portfolio in any one descriptor class section of the buffer file. Therefore, there is no need to "search" the same section of the buffer file for indices contained in an associated L_{ij}.

Repeat this procedure for all portfolios determined.

3. When all portfolios have been processed as in step two (2), all lists L_{ij} of indices to relevant multiplets have been determined. Retain these lists as they are but merge all of the indices into increasing sequence, eliminating duplications to form a list, S of p indices. Order this list in increasing sequence.

The required merge utilizes the sorted order of all relevant portfolios.

4. For each descriptee list, L_{ij}; $(1 \le i \le t_j;\ 1 \le j \le n)$, form a corresponding binary number L'_{ij} of p bits whose qth bit is 1 if and only if the qth index in S is contained in L_{ij}. Now consider the binary numbers L'_{ij} $(1 \le i \le t_j;\ 1 \le j \le n)$. An examination of these

$$m = \sum_{j=1}^{n} t_j$$ binary numbers placed such that

their qth bits are in vertical alignment reveals a reconstruction of the biunique descriptee portion of that multiplet whose index is the qth element of S. Multiplets are reconstructed vertically and all multiplets containing the same descriptee are represented horizontally.

This completes the first phase of the cycle and string process; that is, all multiplets relevant to a particular multiplet have been retrieved; furthermore, the biunique descriptee portions of these multiplets have been reconstructed.

Phase 2, that is the formation of cycles and strings, will be accomplished by a logical product operation on the binary numbers L'_{ij} and the result used as a mask on S. The operations employed are now defined.

a. Logical product, denoted by $\wedge$, produces from two binary numbers a binary number whose qth bit is zero if and only if one or both of the qth bits of the two binary numbers are zero.

b. Masking, denoted by M, produces from a list of p elements, S, and a binary number of p bits, a list of elements from S which are in the same relative position as the non-zero bits in the binary number. Thus, for example, if we let the binary number L'_{ij} mask S we obtain the list of indices L_{ij}. Notationally this is written:

$$L'_{ij} \quad M\ S \longrightarrow L_{ij}$$

Then, the actual determination of cycles and strings is as follows:

5. Perform:

$$(L'_{ij} \wedge L'_{rs}) \quad M \quad S \longrightarrow S\ (i,j;r,s) \text{ for} \qquad (1)$$

$$(1 \le r \le t_s;\ j+1 \le s \le n;\ 1 \le i \le t_j;\ 1 \le j < n)$$

where $S(i,j;r,s)$ are lists of indices of all relevant multiplets containing both the ith descriptee of the jth descriptor class and the rth descriptee of the sth descriptor class.

6. If the results of the qth product-mask operation (1) are non-vacuous, $S(i,j;r,s)$ is stored and its address is placed in the qth position of a list H, capable of holding

$$\sum_{j=1}^{n} t_j \sum_{k=j+1}^{n} t_k \qquad (2)$$

memory addresses. If the results of a logical product is empty (no multiplets exist which contain both descriptees D_{ij} and D_{rs} indicated by (1)), store a word of zeros (or some symbol) indicating this fact in the list H. Let the first element of H be h.

Before continuing the algorithm, it is necessary to present the reasons for developing the list H and indicate generally how this list is used. Cycles and strings are determined by choosing a fixed descriptee for each descriptor class. If at least two of the chosen descriptees exist in one or more multiplets, these multiplets are eligible to be constituent multiplets of a cycle or string. From this class of eligible multiplets, a cycle or string is formed depending upon which multiplets can be logically associated. All cycles and strings are determined by repeating this process for all descriptee sets formed by choosing one and only one descriptee from each descriptor class. Since there are t_k descriptees for the kth descriptor, each one of these descriptors must be considered in combination with all of the other descriptees. Thus, there exists

$\prod_{j=1}^{n} t_j$ sets I_v, which must be examined for cycles and strings. Every set $I_v; (1 \le v \le \prod_{j=1}^{n} t_j)$ has at most $\frac{n(n-1)}{2}$ sets of non-vacuous relevant indices $S(i,j;r,s)$. In all probability, these subsets will not contain the same number of items (or indices) for all values of (i,j) and (r,s).

One function of the list H is to obviate the search for particular subsets of indices S(i,j;r,s) in cycle and string formation.

The existence of cycles and strings is determined in part by knowing if at least two biunique descriptees occur in at least one multiplet. It is unimportant what the multiplets are. In other words, cycle and string existence is determined in part by noting if the sets S(i,j;r,s) are non vacuous. The list H contains the addresses of the non-vacuous sets S(i,j;r,s), and a "void" symbol for the vacuous sets of relevant indices. Therefore this list of constant size items is used to determine cycle and string existence instead of the list of indices S(i,j;r,s).

It is noted in passing that this list of addresses could contain the binary numbers $L'_{ij} \wedge L'_{rs}$ instead of the addresses of the sets S(i,j;r,s) obtained after masking the list S of all relevant indices by the particular binary number. Then the "void" symbol is binary zero.

7. Now continuing the algorithm, the existence of cycles and strings is determined by examining the addresses of the vth set of relevant indices I_v. The elements of I_v depend upon which particular descriptees are chosen for each of the n descriptor classes. Then, for the jth descriptor, a particular descriptee shall be termed $D_{i_j,j}$; $(1 \le i_j \le t_j;\ 1 \le j \le n)$. The set of address of I_v $\left(1 \le v \le \prod_{j=1}^{n} t_j\right)$, contained in the list H, is determined in a systematic manner by computing for j=1:

$$\left[(h-1) + i_j \sum_{y=j+1}^{n} t_y - \sum_{z=s}^{n} t_z + i_s\right]$$

and for j>1:

$$\left[(h-1) + \sum_{w=1}^{j-1} t_w \sum_{x=w+1}^{n} t_x + i_j \sum_{y=j+1}^{n} t_y - \sum_{z=s}^{n} t_z + i_s\right] \qquad (3)$$

for the range of values: $(j+1 \le s \le n;\ 1 \le j < n)$ and $(1 \le i_n \le t_n;\ 1 \le i_{n-1} \le t_{n-1}\ ;\ldots;\ 1 \le i_1 \le t_1)$.

8. Then, for the addresses determined for each I_v $\left(1 \le v \le \prod_{j=1}^{n} t_j\right)$:

A cycle exists if there exists a subset of these addresses such that every corresponding (i,j) and (r,s) in I_v appears at least twice.

A string exists if the cycle criterion holds for all but two and only two of the (i,j) and (r,s).

The constituent multiplets of the cycle or string are in both cases those multiplets whose indices are contained in the sets S(i,j;r,s) with the restriction that only one (but any one) index be selected from each set. If there exists more than one element in a set S(i,j;r,s) there exists more than one set of constituent multiplets which form the same cycle or string. This completes the second phase of the cycle and string formation algorithm.

After a cycle or string has been determined, its authenticity is established in the phase 3 of the algorithm. This may be accomplished in part by examining the fragmentary information (dates, sources of information, etc.) associated with the constituent multiplets. Furthermore, the validity of a cycle is strengthened by the non-existence and/or the resolvability of residual descriptee contradictions. Other tests of authenticity may be devised on the basis of multiplet content or information contained. If a cycle or string and/or its associated integrated multiplet passes all validity tests, it must be processed through the system logic to determine Authenticated File contradictions. The final disposition of a cycle or string and its constituent multiplets is dependent upon the particular application.

It should be noted that all cycles and strings are determined by the given technique after all relevant multiplets have been identified and then reconstructed; thus, the first phase of the technique may be used as a device to retrieve all relevant multiplets. More important, notice that the only time the narrative information was used occurred in determining the existence of biunique descriptees in the input information and in the final authentication of an existing cycle or string. Otherwise, all of the presented processing of information in cycle formation is accomplished by use of only the indices in the Buffer Files. Thus, an important characteristic of the presented file structure is the ability to associate information, accomplish certain types of processing, etc., using the fixed format indices instead of the narrative form of the information.

IV. Conclusions

It has been shown how the elements of the proposed information-processing system have been formalized in order that they can be incorporated into a machine system. A file structure has been established to encompass these formalized elements to provide for their full development. The formalization of elements receives its ultimate expression in the establishment of the Buffer File indices which have a fixed-field format, and thus lend themselves readily to automatic manipulation. These Buffer indices with the associated descriptees provide a powerful tool not only for automatic manipulation of data, but also in its retrieval for manual manipulation.

The stylization of the input data, as was shown, is but the first step in the complete formalization of information within the system. The ultimate consequence is the emergence of the multi-level file structure which it is believed will facilitate the development of the desired information. This is accomplished in a file structure that accommodates a variety of

information in different stages of development and authenticity. Also, it is believed that incoming information has been efficiently factored into terms which provide a full key which has reference back to the incoming data. These terms (essentially descriptees) become entries into the Index Files. The mechanism of the Buffer Indices insures a facile method for the reconstitution of these terms into aggregates which may even be more organized and revealing than those found in the original input.

The manual aspects of the proposed system should not be overlooked. One of the manual features arises due to the complexity of the information involved; i.e. it may occur in one of the processing routines that an impasse arises because of unresolvable contradictions or lack of information, at which point, the machine signals for manual intervention. It may occur that the man is able to resolve the difficulty without further access to information within the machine. The full textual information in the form of fragments relevant to the situation within the Intermediate File is available for resolution if necessary, and is supplied to the man through the various interrogation mechanisms available. In addition, every automatic process has decision points at which the process may signal the alternative it has chosen, and if the man disagrees with it, he may inhibit this alternative and pursue his own. It is believed that the proposed system achieves a balance between a man and a machine, allocating to each the duties for which they have the best capability.

Reference

(1) Newell, A., Shaw, J.C., Simon, H.A., "Programming the Logic Theory Machine," Proceedings, Western Joint Computer Conference, IRE, February, 1957.

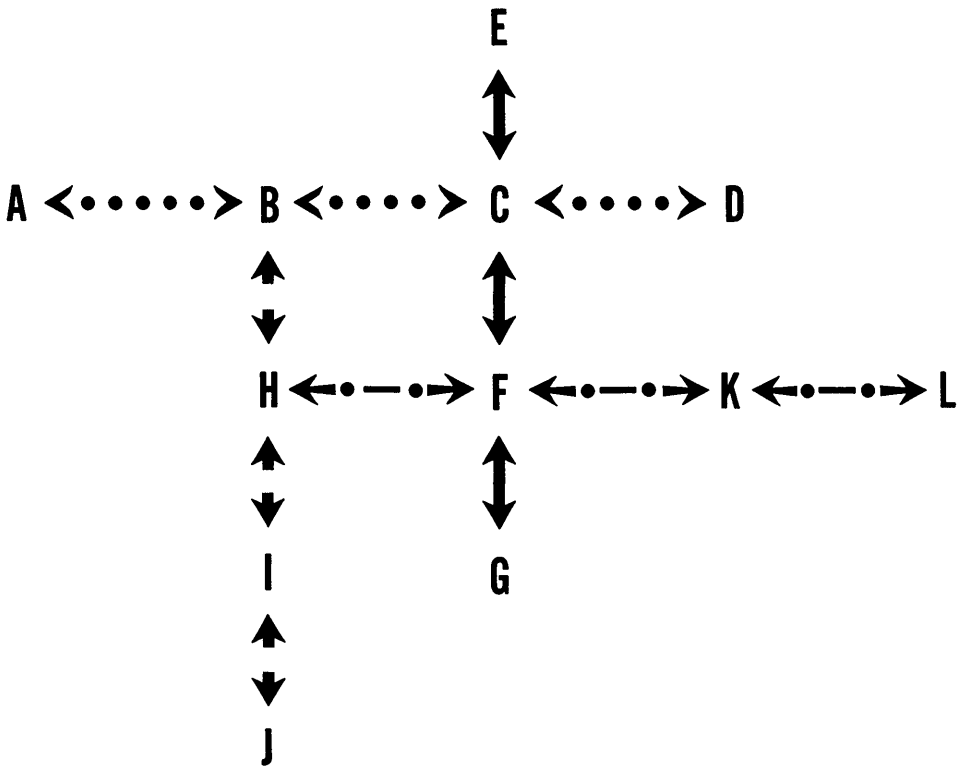


Figure 1. Representation of Cycles and Strings Formed by Multiplets

SYMBOLIC LOGIC IN LANGUAGE ENGINEERING

H. M. Semarne

Research Programming Coordinator
Douglas Aircraft Company, Inc.
Santa Monica, California

Introduction

Not long ago, information activities were still considered merely incidental to the technical operations of industrial concerns, while such activities are now fast becoming a tangible portion of company efforts. The problems of information storage and retrieval increase even while solution methods are being studied. There is an urgent need for new techniques which will enable clerical help with machines to carry out searches, thus liberating engineers and scientists for work on technical problems. These, in turn, include the problems of documentation research.

In order to organize the use of recorded knowledge, it is obviously desirable to:

1. Classify all information to be stored;
2. Correlate all facts to guide future decisions;
3. Select recorded information relative to a given problem.

The development of a large-scale retrieval system will largely depend on the results of work in the classification of concepts rather than of words. There is little sense in classifying words to organize information because the irrational, natural language of a document is hardly ever the best systematic language for use in a machine. This difficulty is a serious obstacle to the accurate correlation of recorded facts and, by extension, to phrase-by-phrase translation from one natural language into another. Since natural languages are rather inefficient tools for the symbolic expression of ideas, the methods of symbolic logic can be employed to analyze, formulate, and correlate these ideas before they are used, "couched in the capricious imagery of words", so to say.

Making use of the symbolic logic truth-matrix computer technique discussed in earlier papers 1, 2, and of other logic methods, it is possible to construct decision-devices useful in the classification, correlation, and selection of documents, as well as in the machine translation of languages. The basis of the symbolic logic approach is non-arithmetical. It features a very flexible matching ability. The approach, general with respect to machinery, does not suffer from the handicap of a searching program written expressly for an existing special-purpose machine. The latter may operate by quasi-mathematical processes, and entail long stretches of "housekeeping" operations.

Another important consideration in the planning of an information retrieval system is its adaptability to expansion. It is one of the purposes of the symbolic logic methods to provide for the eventual embedding of the present system in one of far greater scope, preferably in nesting-block fashion.

The approach taken is not to be confused with one limited to the use of the logic of collections of messages without regard for meaning. It is here intended to do more than merely catalog the frequencies of musical tones in hopes of having someone perceive a melody.

The Progressive Stages Of Logic Analysis

The techniques of mathematical or symbolic logic which underlie the proposed approach to documentation research will be discussed only insofar as they go beyond the rudiments of this discipline as given in textbooks on symbolic logic or, in part, in 1, 2. It is the object of the present paper to show how symbolic logic, as a tool, can be introduced gradually into the problems of information storage and machine searching and translation, just as the scope and the flexibility of the system developed are gradually increased.

1. Seven two-propositional functors and denial (Figure 1) are used to analyze the relationships of a body of information and to separate out various classes of factors. The relationships thus brought out might go unnoticed in the confusing form in which information, such as a large group of observational data, or a set of statements in a natural language, is usually available. For instance, a chemist may encounter this:

   A white crystalline solid contains nitrogen. Its melting point is 114° and the solubility division is W. The aqueous solution is neutral to litmus. When the solid is heated, it distils at 222°; water is evolved during this distillation. The distillate reacts with nitrous acid to form a water soluble acid that boils at 118°.

A logic statement corresponding to this description may be given as:
(see Figure 2 for proposition-key)

KKKNaKcNdCcPab
KNge
KKKijkl
Kqr
KPonm
Kuv
K∅z

Note that for propositions k,l,m, and z table-lookup is indicated

2. By means of the basic two-propositional functors, the truth-matrix technique of[1] and a method of encoding and matching through the characteristics (truth-values) of statements, a convenient and versatile basis is provided for searching, sorting and selecting operations. (see example below)

3. Group-theoretical relations[3] among the characteristics can be used to simplify considerably the making of decisions. Tedious operations commonly performed by Boolean algebra are thus paralleled by a binary matching technique which makes possible the ordering of otherwise very unwieldy clusters of statements. Through this technique, simplification and normalization (i.e. transcription into the form using only and, or, not) become automatic.

All the characteristics of a given number of propositions form a finite Abelian group with respect to the operation Λ. By comparing characteristics through the equivalence operation - 1 if a pair of bits is alike, 0 otherwise - it is possible to have the machine find group elements which have the simplest symbol equivalents.

e.g., the expression

VKVNPabcNVVNSabNSbcNSacKKabc
has characteristic 0 1 1 0 1 0 0 1
(Figure 3)

Testing the characteristic of a:

0 1 1 0 1 0 0 1	characteristic
0 0 0 0 1 1 1 1	a
1 0 0 1 1 0 0 1	by E-(equivalence) operation
0 1 1 0 0 1 1 0	inverse, (by N-operation) this indicates the presence of the A-operation
0 1 1 0 0 1 1 0	bΛc
1 1 1 1 1 1 1 1	tautology proves this

Hence, the expression simplifies to
aΛ(bΛc)
or AaAbc
or, in normal form, (since Axy is the same as VKNyxKNyx)
VKNVKNcbKNbcaKNaVKNcbKNbc

4. Probabilistic weighting factors can be admitted to provide for the statistical handling of information. For example, they could serve for machine-decisions concerning pertinency ratings of documents retrieved.

For example, a materials engineer may make the following request: What rubber material should be used in a certain O-ring packing exposed to organic phosphate ester hydraulic fluid on the inside surface and atmospheric oxygen on the outside surface, and able to withstand 225°F hydraulic system temperature and mechanical abrasion.

There are perhaps 25 families of rubber materials against which to match the resulting assertion pattern. Some of the required properties may be obtained more generously than others from any given rubber material. It may be necessary to establish a preference order among disclosures, since it would be desirable to obtain not only a material just fulfilling the requirements, but the best possible one for the purpose at hand. The task may be carried out as outlined in Figure 4. This example may serve to illustrate the high degree of specification attainable in this type of machine searching.

5. In order to achieve more freedom in translating diversified and intricate statements in a natural language into machine language, logic operators,[4] or quantifiers (Figure 5) could be called in.

These quantifying operators could be introduced as special notes by which to evaluate properly the results of a truth-matrix analysis. Alternatively, the quantifiers could be approximated, for purposes of computation, by prearranged threshold values which would then be used like probabilistic weighting factors (above).

The functional relationship of information retrieval systems to arithmetic compiler languages may be noted here. Documentation work essentially deals with patterns of thought and their relations to specific units of recorded information. While it may be undesirable to mold work in such a discipline on a technique of programming mainly arithmetical processes, it may nevertheless be profitable to uphold compilers as good examples,

philosophically speaking. Compiler languages, such as Fortran, Speed, etc., are representations at the operator, or function, level. The transcription of documents, instead of being done on an elementary symbol-by-symbol basis, without reference to context, should also graduate to the function level. The relationships among the clue-words of a document should be inherent in the record.

The Progressive Stages Of Information Handling

Before proceeding with the application of symbolic logic, one may review the need for it.

The point of departure for all documentation research can be said to be the traditional library index. The first step in this work is due to the need to go from the fixed and little expandable subject categories of the library index to flexible categories of information such as are provided by many special libraries today.

In multi-aspect indexing, documents are stored under groups of clue-words which characterize their contents, and retrieval is effected by requests made up of any combination of clue-words. This is a long step ahead and away from fixed-category filing.

It now appears that the machine runs into a snare on retrieval. If all the clue-words pertinent to a document are combined into one large group, there will be false drops due to lack of resolution. If, on the other hand, a document is indexed only under a few clue-words, it may be overlooked on many requests. That is to say, much of the advantage of the system over traditional indexing is then lost.

This is the central quandary of information handling. Many authors have attacked the problem in many ingenious ways, and some of them have successfully moved far beyond that stage, [5,6,7,8,9,10] to find an optimum solution.

One trend is thus toward more detailed indexing, and another toward simplification. Moreover, these trends are also tied in with machine speeds and machine memory capacities. Consequently, most of the new work in documentation research is centered at the intersection of these two trends. Among the varied results which seem to arise, not all, unfortunately, are the products of the essential double-criterion:

Is the system commensurate with the specific information needs to be dealt with, and is it extensible?

Three levels of document description can be defined:

1. identification of symbols
2. identification of their order
3. identification of their logic relations

In this last level, information concepts can find expression through logic relations among the underlying clue-words. (Figure 6)

Symbolic Logic Treatment Of Information

Machines are now linked with the storage, analysis, and retrieval of information. There is little point in attempting to discuss information handling without rapid electronic computers, but it must be realized that the greatest contribution of the machine to this field does not lie in speed, but in the right type of searching.

The latter depends on the design of the search program which depends, in part, on the method of storing the information. From the viewpoint of logic, the encoding of the information for storage and the encoding of search requests can be regarded as one and the same task. The mechanics of storage assignment for the former, and the mechanics of matching clues for the latter operation, are the only important points in which they are different.

To write a search request means to write a set of specifications which should be as precise as possible. The burden on the writer of the request can be immensely lightened and he can be enabled to write a great many such requests in the time formerly allotted to one request, by the simple expedient of having the machine compose the request for him. This can be done through symbolic logic, as shown above (Figures 2,4,6).

All the essential facts known about a document, such as the title, serial number, origin, year of publication, location in a file, descriptive terms, and its subject content, (materials involved, properties observed, processes performed, ambient conditions, applications, etc.) are merely listed, regardless of possible redundancies. Each one of the facts or propositions listed may be linked with some of the other propositions or with extraneous useful matter concerning the document sought. There now arise a number of statements, such as: a or else b, (Aab) a or b or c or all three, (VVabc), a implies c but not b unless d and e are present, (KCVNdNeCacCKde CaKcb), etc. The machine is directed to run through the Truth-Matrix Analysis as discussed earlier[1], and the result is a binary number of 2^n bits (for n propositions). This number, called the characteristic of the statement, is obtained one bit at a time; for every "yes" or 1 bit, a binary number of n bits (for n propositions) denoting the combination of propositional values corresponding to the "yes" result, is automatically recorded in memory. A set of these numbers, corresponding to a given document, constitutes a complete assertion pattern for that document (on encoding) or a complete description of the search request (on retrieval). A special but frequently occurring case is that of disjoint characteristics which is due to a statement in which the propositions can be grouped in independent, or disjoint, substatements.

For the example used above, the disjoint characteristics would be:

0010000000000000
0011001100000000
0000000000001111
1111111100000000
0000000011000000
0000000000001111
0000001100000011

The assertion pattern derived from the above characteristics would be:

0010	0000	1100
	0001	1101
0010	0010	1110
0011	0011	1111
0110	0100	
0111	0101	0110
	0110	0111
1100	0111	1110
1101		1111
1110	1000	
1111	1001	

This assertion pattern is compared with assertion patterns of all entries within a pertinent range. The pertinent range is defined by combining the original statements with a standard set of conclusions, e.g. implications like: water evolution on distillation of a solid implies a hydrate, reaction with nitrous acid implies the presence of a primary or secondary amino-group etc. The resulting combined characteristic clearly outlines the possible entries sought, and the search pinpoints it.

The basic logic "in-and-out" procedure may be summarized as follows:

a. Analyzing and Coding:

1. Collect facts or propositions
2. Compose propositions into logic statement
3. Set up proposition-key (dictionary), by defining the propositions
4. Analyze statement and obtain its identifying assertion pattern
5. Store characteristic and proposition-key addresses

b. Analyzing and Retrieval:

1. Collect facts or propositions
2. Compose request propositions into logic statement
3. Analyze statement and obtain its assertion pattern
4. Match assertion patterns
5. Check proposition-key for print-out of meanings.

The disclosure obtained in this manner is highly unambiguous, having been released upon matching a assertion pattern which denotes the same logic relationships as that of the request.

The dictionary entries would be kept out of the way of the characteristics. When part of a document record is devoted to textual material and part to a code, the effective search rate is severely cut due to the idling of the circuitry during the passage of the textual part of the record. The search operation should therefore be broken into two portions, the first being the matching of the logic form or assertion pattern of the document and of the proposition code, and the second being the special table look-up in the proposition-key (Figures 2,4,6). The latter operation results in a match. On the other hand, the proposition code and its dictionary could remain related for easy identification of the code for a document.

The amount of detail used in describing concepts stored or disclosed in the document file depends on the needs of the user and the nature of the application. These will also determine whether a single-shot search procedure or an iterative analysis procedure is to be followed. If the file for an area is appreciably less branched in its descriptive terms than is the type of request addressed to it, the iterative analysis approach may be more advisable. No special encoding method is necessary in the preparation of iterative analysis searches. The form of the request suffices to determine the depth of the search and the logic classes to be considered during that search, or that step of the search. The gradual branching of logic decisions is automatically controlled by changes in the request statement.

The iterative variant of symbolic logic searching can be employed to determine the degree of relevance of a disclosure as compared with that of another. It is necessary only to run several searches, each passing through a different logic network and to count the matches of certain individual clue-words on the records of various documents examined.

Again, it is the needs and the skill of the user, and the nature of the application, that determine the extent of automaticity desired in the composition of the request sentence.

A further study of the role to be played by logic request sequences in the guiding of the design of electronic equipment for searching and selecting should be made.

Sufficient Clues And Probability Aspects

Besides the need for retrieving information, there is also the necessity to analyze documents for various purposes. It may be desired, for instance, to examine a collection of data from various sources in order to deduce the answer to an apparently extraneous question. Or, on receiving or recalling various barely related facts, it may be desired to find a special pertinent document. Such situations will be referred to as "sufficient clues" problems. (The word, sufficient, is used on a hopeful note).

Suppose, as a simple example, that these items are collected:

1. There are three authors, x,y,z, each having written one of three documents, α, β, γ, not necessarily in this order.
2. There are three other documents, a,b,c, the respective authors of which are the same as those of the three above documents, respectively.
3. Document c deals with topic one
4. Author y writes on a second topic
5. Document b is not mathematical in nature
6. One of documents a,b,c, namely the one marked by the corresponding greek letter, (designating a paper written by y) deals with a third topic.
7. Author y has written on the topic of that one among documents a,b,c which has mathematical symbols in it.
8. Author z does not know document α

Upon a question such as: Which is the author of document α? It can be shown that the straightforward answer, "x is" can be rather suggestively obtained by means of the same symbolic logic truth-matrix analysis as that used above. It is necessary merely to make a proposition-dictionary, jot down all the sentences occurring and let the machine compose them into an overall statement. The characteristic will give the answer. (Figure 7)

Of course, the actual purpose of using "sufficient clues" is the correlation of hints and guesses in hopes of locating a document fitting a very narrow description.

Many of the decision-making procedures developed from symbolic logic principles can be readily adapted to the calculation of probability factors, instead of bare "yes" and "no" answers. In the present application to documentation, this is likewise the case. The algebraic equivalents through which the logic calculations are programmed for the computer permit the introduction of decimal fractions for probability factors, in lieu of 0 and 1 alone. Dependence must be set on the type of probability data obtained. If the relative probabilities were given in terms of the reverse relations, i.e. in terms of descriptors with respect to documents, rather than the other way round, then some appropriate transformation formula would be used.x) Alternatively, without much extra effort, weighted multiple retrieval could be effected, i.e. each of several documents could be retrieved with an attached order of desirability. It would then be possible to have the machine judge through a numerical criterion. The operator may, on the other hand, prefer to use his own judgement as to the comparative suitability of his results.

Justification for many of the manipulations of probability factors arising in this field is beyond the scope of this paper and may be found in the Luce-Raiffa form of utility reference theory formulated by von Neumann and Morgenstern.

x) e.g., if $p(a:b_i)$ is the probability of a with respect to b_i, and $p(b_i:a)$, mutatis mutandis, then

$$p(b_i:a)=p(a:b_i)p(b_i)\Big/\sum_j p(a:b_j)p(b_j)$$

Some Aspects Of Machine Translation

In view of the parallel relation between the symbolic logic approach to both document encoding and information retrieval, it is easy to see the further parallel relation between these and machine translation.

The subject of encoding per se has not been discussed in the present paper. Suffice it to say that, for the construction of a flexible, economical and inclusive proposition-dictionary, the Semantic Code discussed by John L. Melton[5] appears to be exceptionally promising. Other approaches to this part of the information problem are also being considered.

Machine translation, (MT) especially translation from one natural language to another, has received a great deal of attention and many results have been reported to-date from various parts of the world. This subject deserves more than a few remarks at the end of the present discussion. As a matter of fact, machine translation should be treated in a report devoted to the subject of the utilization of essential scientific data from foreign countries. A brief outline of the contribution to MT of techniques based on symbolic logic analysis will be given here.

The main tasks in MT are:

1. Construction of a formal system (including word lists) for describing natural languages.

2. Definition and evolution of algorithms for transferring (translating) from one system to another.

3. Development of the principles necessary for programming and coding these algorithms into the machine.

Symbolic logic as a tool enters into all three problems, especially the first one. The third problem, being the one most closely related to the similar problem in information searching generally, responds directly to the symbolic logic treatment discussed above. In other words, the truth-matrix analysis, the binary characteristic, the group-theoretical relations among characteristics, the logic pattern matching technique are all applicable to the handling of information consisting of linguistic algorithms.

The second problem is often merged and sometimes confused with the first. This happens largely because of the difficulty of divorcing semantics from the compilation of appropriate word-lists. The development of the necessary algorithms is basically a logic problem. For the source-target language pair, the proper formula structure can be set up by symbolic logic. Here, of course, quantifiers play an increasingly important role. The search for semantic structure is, in fact, the prime effort in today's move away from the primitive word-by-word look-up idea of MT. Multiple meanings of words as well as grammatical structure of sentences cannot be taken into account without this search.

Some successful attempts have been made to attack the multiple word meanings by the statistics of word-frequencies. The grammatical structure of sentences is being widely investigated by means of morphology (word order and inflection) applied to both source and target languages.[12,13] It is expected that symbolic logic characteristics can here be used advantageously to differentiate in a very simple way between structures.

By matching the constraints in structure of the source language with those of the target language, meanings rather than word-messages could be translated and furthermore, the storage requirements of MT can be much reduced in this way. The size of a well-organized glossary is, in fact, another leading question so far left unanswered partly because of the uncertainty as to the eventual results of structure studies. In the Russian language, for instance, 86% of the running words are inflected, and the total number of entries required for one noun average from 6 to 10, and one verb may require 59 entries[12]. The split-glossary technique of Dr. Milos Pacak, of Georgetown University, based on morphological analysis, may provide a much needed short cut. The classes and subclasses proposed by Dr. Pacak can be controlled by logic considerations.

A system of "Semantic Semaphores" is in preparation by the author. This, it is hoped, will aid in the establishment of word relationships and the preparation of word patterns relating to the compilation of thesauri. These, of course, also call for decisions based on familiarity with specific technical fields.

For input to the machine, a satisfactory approach may be that of W. R. Nugent[14] or one developed in the U. S. Patent Office[15]. Mr. Nugent uses pseudo-alphabets having mnemonic names, which can easily all be directed by one single language pattern, but permitting a symbol capacity of 500Q.

Conclusion

Lest one become so deeply involved in theoretical considerations of a retrieval system as to lose sight of its practical requirements, he should keep in full view the logical economy with which human beings, such as a good research librarian or a good oral interpreter, work.

The dexterity of the specialized human mind in "homing" rapidly toward the search goal must remain the ultimate aim of the documentation researcher, until that aim is reached and can be surpassed by a machine of machines.

Too often, an information system is exhorted because it has a remarkably large file or an astute way of responding to one type of request. For that matter, a dove picking out someone's fortune from a large pile of envelopes is also remarkable, if somewhat one-sided. It is perhaps not so much what a human being looks for in searching a library, as what he discards along the way that should tip off those who wish to direct machines to do as well as human beings.

A student having studied up on a large variety of topics for a comprehensive examination, seems to expel from his mind, as soon as he sees the questions on the examination, all those topics which do not pertain to these questions. A trained librarian or a trained interpreter similarly cuts down with bold strokes the search area through a few significant decisions which rapidly brings him to the eventual choice. Many other people do this in other connections.

What the machine must be told is how to perform this "preliminary" narrowing-down process which enables it to undertake the search "proper" only after the most time-consuming blind-man's buff through the logic network has been carried out and the irrelevant paths have been discarded. Machine searching, similarly to machine translation, may be best divided into two stages: the rough search (or rough translation), and the fine search (or fine translation). No exact theoretical dividing line can be given.

In a human being, this narrowing-down process works by means of two-valued logic, and therefore it must work likewise in the machine. The question now arises: Can present-day machines or machines now being designed along well-worn

lines and capable perhaps of expediting the "fine search", be expected to master the real executive-robot task of the "rough search"?

It is suggested that the answer is: No! The "rough search", the portion of the work that almost demands a chess-player's ubiquity and judgment coupled with far superior speed, is not likely to be performed adequately for some time to come. In machines, non-arithmetical design concepts with as yet unrealized searching skills will first have to be developed. It may be well to have a close look at one of the few truly progress-oriented machines now in existence, viz. the Searching Selector at Western Reserve University[16,5].

A system of "logic traps", as mentioned above, must be achieved which rapidly discards the classes of topics that are not wanted by answering yes or no to well-ordered questions. It will not do merely to make up for the lack of adequate binary-logic networks by larger and still larger files and by greater machine speeds.

Often, the aspiration to higher speed obscures the purpose of searching. Of course, the highest possible speeds are desired and, in fact, machine speed is in itself a result of the improvement of the logic search paths. Shortcomings in logic, however, cannot be covered up by fast "horsework", either in the machine or in the paper-and-pencil system which may serve as a program for the machine. It is as incongruous to expect good results from a hasty approach in this field, as it would be for some computer manufacturer to expect good results if he felt that not every one of the magnetic core elements in his computer's memory needs be built carefully just because there are so many of them.

As Dean Shera remarked, " The dim light of the electronic tube has led us ever faster along the wrong path. For it is not in speed, but in capabilities that the great promise of automation lies . . ."

One may note that transistors shed an even dimmer light.

References

1. R.G. Larkin and H.M.Semarne, "Symbolic Logic in Engineering", 14th Annual ACM Meeting, MIT, paper #77, September 3, 1959.

2. H.M. Semarne, "Symbolic Logic Decision Devices in Documentation" Proceedings of the International Conference for Standards on a Common Language for Machine Searching and Translation, 6-12 September 1959, Western Reserve University, Cleveland, Ohio.

3. D. B. Netherwood, "A Theory of Logical Groups", Thesis, Air Force Institute of Technology, March, 1957.

4. K. Dürr, "Lehrbuch der Logistik", Birkhäuser, Basel, 1954.

5. J.W. Perry and Allen Kent, "Tools for Machine Literature Searching" Vol. I of Library Science and Documentation. Interscience Publ., N. Y., 1958.

6. M.R. Hyslop, "Inventory of Methods and Devices for Analysis, Storage and Retrieval of Information", in "Documentation in Action" (ed. by J.H. Shera, A.Kent, and J.W. Perry).

7. H.R. Koller, E. Marden, H. Pfeffer, "The Haystaq System: Past, Present, and Future" in "The Design of New Systems-Mechanical Problems Reports, U.S. Patent Office, Washington, D.C., 1957.

8. M.E. Maron, J.L. Kuhns, L.C. Ray, "Probabilistic Indexing: A Statistical Approach to the Library Problem", paper #13 presented to the 14th National Meeting, ACM, MIT, September 1, 1959.

9. H.P. Luhn, "The Automatic Derivation of Information Retrieval Encodements from Machine Readable Texts" paper presented at International Conference for Standards on a Common Language for Machine Searching and Translation, Cleveland, Ohio, 6-12 September 1959.

10. M. Taube and Assoc., "The Mechanization of Data Retrieval", Documentation, Inc., 1959.

11. R.D. Luce, H. Raiffa, "Games and Decisions" John Wiley and Sons, Inc., N.Y.,N.Y., Chapter 11, and Appendix 1.

12. M. Pacak, "Morphology in Terms of Mechanical Translation" Georgetown University, paper presented at International Conference for Standards on a Common Language for Machine Searching and Translation, Cleveland, Ohio, 6-12 September 1959.

13. G. Herdan, "The Statistics of Structured Meaning", papers presented at the International Conference for Standards on a Common Language for Machine Searching and Translation, Cleveland, Ohio, 6-12 September 1959.

14. W.R. Nugent, "A Machine Language for Documentation and Information Retrieval", Itek Corp., paper #15 presented on 14th Annual ACM Meeting, MIT, 1-3 September 1959.

15. S.M. Newman, R.W. Swanson, K.Knowlton, "A Notation System for Transliterating Technical and Scientific Texts for Use in Data Processing Systems", Patent Office Research and Development Report #15, 22 May 1959.

16. Allen Kent, "Machine Literature Searching at the Center for Documentation and Communication Research", report presented at the International Conference for Standards on a Common Language for Machine Searching and Translation

FIGURE 1

TABLE OF TWO-PROPOSITIONAL FUNCTORS

Polish Prefix Notation	Tie-Symbol	Traditional Designation	Meaning	Characteristic Value
Na	$\bar{a}$	non	not a	10
Kab	a•b	et	a and b	0001
Aab	a∧b	aut	a or else b	0110
Vab	a∨b	vel	a and/or b	0111
Pab	a↓b	Peirce's arrow	neither a nor b	1000
Eab	a≡b	par	a like b	1001
Cab	a⊃b	prae	if a, then b	1101
Sab	a/b	Sheffer's stroke	not both a and b	1110

FIGURE 2

TYPICAL SIMPLIFIED PROPOSITION KEY
(kept down to four propositions for each of 7 disjoint statements)

Physical Description

a = it is a liquid substance
b = it is a gaseous substance
c = it is a crystalline substance
d = it is a colored substance

Composition

e = it contains N
f = it contains O
g = it contains only C and H
h = it contains S or P

Melting or Boiling Point

i = it has a melting point (m.p.)
j = it has a boiling point (b.p.)
k = its m.p. is exactly 114 degrees
l = its b.p. is exactly 222 degrees

Solubility

m = its solubility division is W
n = aqueous solution acid to litmus
o = aqueous solution basic to litmus
p = aqueous solution has pH δ

Distillation

q = it distils
r = it loses water on distillation
s = it sublimes
t = it tars

Reactivity of Substance

u = distillate reacts with nitrous acid
v = it produces an acid
w = it reacts with phenylhydrazine HCl
x = it produces an amide

Reactivity of Derivative I

y = derivative I has m.p. of α degrees
z = derivative I has b.p. of β degrees
a = derivative I is water-soluble
o = derivative I is in solubility division γ

FIGURE 3

EXAMPLE OF TRUTH-MATRIX

$\{[(\overline{a \downarrow b}) \vee c] \cdot (\overline{a/b} \vee \overline{b/c} \vee \overline{a/c})\} \vee (a \cdot b) \cdot c$

VKVNPabcNVVNSabNSbcNSacKKabc

a	0 0 0 0 1 1 1 1	
b	0 0 1 1 0 0 1 1	
c	0 1 0 1 0 1 0 1	
$a \downarrow b$	1 1 0 0 0 0 0 0	
$\overline{a \downarrow b}$	0 0 1 1 1 1 1 1	
$(\overline{a \downarrow b}) \vee c$	0 1 1 1 1 1 1 1	X
a/b	1 1 1 1 1 1 0 0	
$\overline{a/b}$	0 0 0 0 0 0 1 1	Y
b/c	1 1 1 0 1 1 1 0	
$\overline{b/c}$	0 0 0 1 0 0 0 1	W
a/c	1 1 1 1 1 0 1 0	
$\overline{a/c}$	0 0 0 0 0 1 0 1	Z
yvwvz	0 0 0 1 0 1 1 1	
yvwvz	1 1 1 0 1 0 0 0	U
$u \cdot x$	0 1 1 0 1 0 0 0	
$a \cdot b \cdot c$	0 0 0 0 0 0 0 1	
$(u \cdot x) \vee (a \cdot b \cdot c)$	0 1 1 0 1 0 0 1	characteristic

FIGURE 5

QUANTIFYING OPERATORS

	Prefix Symbol	Name	Meaning	e.g.
1.	Πx	ALL-operator	for every x	ΠxCxy
2.	Σx	EXISTENCE-operator	for some x (at least one)	ΣxCxy
3.	$\mathrm{O}x$	UNIQUENESS-operator	for exactly one x	OxCxy
4.	Mx	MAXIMUM-operator	for at most one x (none or one)	MxCxy
5.	Jx	CLASS-operator	the class of all x's such that	JxCxy

FIGURE 4

TREATMENT OF REQUEST WITH WEIGHTING FACTORS

1) Proposition-Key:

a = rubber material for use in O-ring packing

b = the packing to be exposed to hydraulic fluid on the inside

c = the packing to be exposed to atmospheric oxygen on the outside

d = the hydraulic fluid is an organic phosphate ester

e = the packing to withstand 225°F hydraulic system temperature

f = the packing to withstand mechanical abrasion

2) Weighting Factors:

the relative importance of b may be .8

the relative importance of c may be .6

the relative importance of e may be .5

the relative importance of f may be .3

These weighting factors may be re-interpreted as probabilities that the given property alone fulfil to requirements of the material.

The entries to be searched will, likewise, be set up with their respective weighting factors, thus making possible a preference order among the disclosures.

3) Symbolic Statement:

KKKKKabcdef (with b=.8,c=.6,e=.5,f=.3)

4) Assertion Pattern:

1,.8,.6,1,.5,.3

FIGURE 6

RELATIONS AMONG CLUE-WORDS

Request:

Article both in English and concerning aircraft or spacecraft, written neither before 1937 nor after 1957; should deal with laboratory tests leading to conclusions on an adhesive, used to bond metal to one of these: rubber or plastic; the adhesive must not become brittle with age, must not absorb plasticizer from the rubber adherent, and have a peel strength of 20 lbs./in; it must have at least one of these properties: no appreciable solution in fuel and no absorption of solvent

Clue-Words:

English, aircraft, spacecraft, 1937, 1957, laboratory, adhesive,metal, rubber, plastic, brittle, plasticizer, peel-strength, fuel, solvent

FIGURE 6
(Continued)

Proposition-Key:

a = it is an article in English

b = it is an article concerning aircraft

c = it is an article concerning spacecraft

d = it is an article written before 1937

e = it is an article written after 1957

f = laboratory tests were run

g = conclusions on an adhesive were obtained

The Adhesive:

h = the adhesive is used to bond metal to rubber

i = the adhesive is used to bond metal to plastic

j = the adhesive may become brittle with age

k = the adhesive may absorb plasticizer from the rubber adhesive

l = the adhesive may have a peel-strength of 20 lbs./in.

m = the adhesive may have appreciable solution in fuel

n = the adhesive may have absorption of solvent

Symbolic Statement:

KKaVbcPdeCfg
KAhiKKKNjNklSmn

FIGURE 7

A "SUFFICIENT CLUES" PROBLEM AND SOLUTION

x,y,z = documents written by authors x,y,z, respectively
α,β,γ, a,b,c = documents α,β,γ,a,b,c, respectively
p,q,r = the topic of the document is one, two, three, respectively
m = the topic of the document is mathematical

From statements 1. and 2. of the Problem, one gets 6 sentences of this form: AAxyz, all connected by conjunctions, and 18 sentences of this form: CCαxαKCβAyzCC$\beta\delta$Cγz, 9 for α,β,γ and 9 for a,b,c; all connected by conjunctions

From statements 3.,4.,5: Ccp,Cyq,CbNm,
From statement 6, one gets 3 statements of the form: CCCyαar
From statement 7: KKCCamayCCbmbyCCcmcy
From statement 8: CαNz

The total characteristic of 2^{13} bits simplifies to an assertion pattern of 13 bits which indicates the presence of the relationship VNαx or Cαx, i.e.$\alpha \supset$ x

THE FACT COMPILER: A SYSTEM FOR THE EXTRACTION, STORAGE, AND RETRIEVAL OF INFORMATION

Charles Kellogg
Intellectronics Laboratories, Ramo-Wooldridge
Canoga Park, California

Summary

The Fact Compiler is a system for the timely extraction of significant information from source data and for the storage of this information in an organized manner that permits rapid retrieval. In addition, the Fact Compiler can process or manipulate the stored data in a variety of ways, and it is adaptable for use with presend-day reporting techniques. The system is capable of orderly growth to meet the changing requirements of growing organizations.

Information is stored according to a logico-linguistic structure. This structure enables the system to: (1) directly interrogate personnel and request the reporting of specific information; (2) automatically present desired data at the appropriate time intervals; and (3) retrieve information according to subject, aspect, date, degree of specificity, and organizational unit.

The Fact Compiler

As society matures, production and use of information greatly increases. At the same time, individuals tend to narrow their fields of specialization in order to cope with this growth of information. Thus, the old adage about knowing more and more about less and less becomes more meaningful as time goes on. It becomes increasingly more difficult to assimilate and evaluate the mass of data contained in the paper work flooding many business, government, and university organizations. The techniques of automation have been applied in various ways to alleviate this problem. This paper will discuss an advanced system, the Fact Compiler, which uses new concepts to automate the storage and retrieval of information.

The Fact Compiler is a system for the timely extraction of significant information from source data and for the storage of this information in an organized manner that permits rapid retrieval. It has been designed to function as a centralized storehouse of important information. In addition to retrieving information, the Fact Compiler facilitates the processing or manipulation of the stored data in a variety of ways not readily possible with decentralized storage systems. The system is readily adaptable for use with present reporting techniques used in industry. At the same time, it can be used with fully automated input. The system is capable of orderly growth to meet the changing requirements of growing organizations. The Fact Compiler is an in-line processing system. That is, input data is immediately recorded in the memory and, therefore, is available for immediate use. The rapid response of the system permits the prompt discovery of trends in the stored data. It also encourages personnel to "browse" for information as an aid in finding all relevant data. The stored information, even if received from widely divergent sources, can be organized automatically and used in the generation of summary reports.

A key problem in designing the Fact Compiler was the development of a language capable of expressing the communication needs of personnel engaged in the input of information or in the formulation of a request for information. A large amount of linguistic and empirical study has resulted in a language structure that satisfactorily meets these objectives. Development of the language vocabulary required a precise formulation of the information needs of system users. These needs are expressed as a series of questions or interrogations which are indicative of the kind of information that should be stored within the system.

Interrogations are organized according to a logico-linguistic structure and are stored in the system memory. Input information consists of responses to the appropriate interrogations. The combination of an interrogation and its associated responses is called a factual statement. A fact is defined as an interrogation and one particular associated response.

General system features are shown in Figure 1. The system is composed of human and machine elements. In general, system personnel exercise judgment and perform decision-making functions, and the equipment elements perform the necessary routine processing. System personnel monitor and control operational functions, and aid in the input of data. The system uses a general-purpose digital computer and a high-capacity, rapid-access, magnetic memory. The greater portion of this memory is devoted to the file of facts. Other important files are the dictionary, or vocabulary of terms, and the interrogation vocabulary file. The input or extraction process is facilitated by the use of a communication console having an alphanumeric display tube and a special set of keys for data insertion. The output or retrieval process may use either the communication console or a high-speed printer.

Use of a Fact Compiler in a Business Firm

The concepts involved in the use of a Fact Compiler may be visualized in the application of the system to a hypothetical business firm. The organization structure of a typical business firm is illustrated in Figure 2. Some of the basic responsibilities of business management are the determination of the future course of the business by farsighted planning, the selection of qualified key personnel, development of a sound organization plan, and effective control through the delegation of responsibility. Management receives the data necessary for making decisions in these areas from a series of reports generated by line and staff departments. A common phenomena among

executive personnel is that of being "swamped" with reports. Development of procedures to filter information destined for management (i.e., selection of pertinent information and rejection of unnecessary data) is an important function usually performed by staff departments.

With a Fact Compiler, management and staff would, in effect, have an "extended memory" with which they could review particular aspects of the corporation's past history in various levels of detail. This type of review may permit better forecasts about future conditions such as costs, sale prices, raw material availability, and product demand. Also, the compiler could supply data indicative of the degree of control in effect within the firm. Correlations could be made between activities not previously compared and general economic conditions.

Interrogations for a business Fact Compiler System are organized into interrogation lists representing major subject categories of interest. Typical interrogation list categories might be "factory," "regional sales district," "subsidiary," "budget," "taxes," "legal," "patent," and "corporate assets." Information to be stored in the memory for a particular factory, regional sales district, or department is placed in an individual record for that factory, district, or department. This record is called a Unit Record. Unit Records may store information derived with the aid of one or more interrogation lists. The use of interrogation lists for data extraction and Unit Records for storage and retrieval is fundamental to the operation of the Compiler System.

Fact Compiler Language-Restricted English

A solution to the communication problem between humans and the computer files was of critical importance in the development of the Fact Compiler. It was evident that neither natural English nor machine language could be used. A compromise had to be made. After much experimentation, a language was developed which has proved satisfactory for both personnel and the computer; the language is called Restricted English.

Restricted English consists of a vocabulary of specially selected English words that are familiar and meaningful to personnel as well as being directly related to the types of interrogations and information requests that the system is designed to handle. Words are carefully selected with respect to meaning, since synonyms are not permitted. Restricted English terms may be "simplex" (single words) or "complex" (a word grouping or phrase). Complex terms are formed to represent unique, commonly occurring concepts. Rules have been developed to aid in the translation process from natural English word groupings to Restricted English simplex or complex terms.

The language consists of five parts of speech. Each of these different term categories serves a unique function. The parts of speech are:

Substantive. A noun or noun complex; a name for a thing or object.

Descriptor. Describes or limits definitions of substantives. They may be ordinary adjectives or past participles with adjectival functions.

Activity Connector. Relates several substantives in order to describe an action. They are usually present active or present passive participles.

Relational Connector. Prepositions, used for defining relationships between two substantives.

Interrogative Operator. Interrogative adverbial phrases used to determine the magnitude, quality, or position of things or objects of interest.

Restricted English does not use conjunctions or pronouns. Typical examples of business-oriented terms are shown in Table I. Most interrogations may be formulated by using four or five terms; however, a few may require as many as seven or eight.

The total set of terms defined for usage comprises the Fact Compiler dictionary. It is estimated that a dictionary of a few thousand terms should be sufficient to handle most business storage and retrieval applications. Each term in the dictionary is coded and assigned a four-digit tag for internal computer use. The tag identifies the unique term and the part of speech of the term.

Interrogation File Organization

Just as the dictionary defines a vocabulary of terms for the computer, the interrogation file defines a vocabulary of interrogations. As mentioned earlier, interrogations are filed by major areas of interest into interrogation lists. Each interrogation list is further divided into a five-level generic classification structure.

Levels of Generality

1.	Interrogation List	subject categories
2.	Capital Topic	
3.	Major	interrogation categories
4.	Minor	
5.	Sub	
6.	Sub Sub	

By assigning codes to each level, a "generic address" is formed which uniquely represents any interrogation used in the system. This address serves a very important purpose in identifying the subject category and level of generality of an interrogation. It also permits insertion of new interrogations at the end of a level without requiring a revision of the entire address structure as would be the case if an absolute address structure were used.

Interrogation file structure is outlined in Figure 3. Each interrogation is represented by its generic address, a series of four-digit term codes denoting interrogation content, and a series of criteria. Criteria are defined at the same time as the interrogations and permit the computer to perform the following operations:

1. Present the interrogation automatically to an input analyst on the basis of a predetermined elapsed time since the interrogation was last answered.
2. Present current factual data to personnel at desired periodic time intervals.
3. Select next interrogation on basis of response to present one.
4. Determine if an answer is in the proper form; if so, perform any processing that may be desirable before answer storage.

Extraction of Information—Input

The computer may take an active or passive part in the input of new information. This depends on the situation and system requirements. With little or no historical data in its memory, the computer's role would be basically passive, waiting for personnel to request interrogation displays and input answers. However, once initial data is stored, the Compiler System can compare data "age" with interrogation criteria and begin actively asking for new input at appropriate periodic intervals. This last feature can be quite useful in assuring the reporting of important facts and decreasing the possibility of reporting redundant facts.

Upon receipt of a new document, an input analyst must determine the Unit Record to which it applies and then select the major subject category involved. This information is conveyed to the system via push-buttons, and the input analyst is presented with a set of interrogation subject categories on the display scope. A process roughly analogous to the game of twenty questions ensues. The analyst, now aware of the categories of interest, reads the document until he finds reportable information. Depressing a key on his console, he generates an answer to an interrogation in the selected category. On the basis of the analyst's response, more detailed interrogations are presented as long as affirmative answers are supplied; negative answers cause the generation of higher-level interrogations, different subject categories, or result in terminating the interrogation generation.

The extraction of significant information thus proceeds, with the analyst alternately scanning the document and then reporting data in as much detail as possible. This extraction procedure, of course, does not depend on the existence of a source document. Source data that is in any form recognizable by humans may be used. Personnel could directly report facts from memory if desired.

Fact Storage Organization

The choice of an information storage file structure depends on many considerations such as the type of memory device used, estimated size of file, retrieval time requirements, and knowledge of the frequency distribution of various types of retrieval requests presented to the file.

The Fact Compiler memory is capable of storing one million factual statements with an average of ten answers per statement, for a total of ten million separate fact items. With this type of memory, it is desirable to store facts so that one or several entries to the memory will select the information records necessary to satisfy typical requests.

If the frequency distribution of requests were known in detail and did not change with time, an optimum file structure could be developed. However, the distribution estimated before the system is in operation is usually only a rough estimate, and the distribution will change with time as interest in various aspects of the stored information changes.

If it is necessary for retrieval time to be minimized, factual statements may be redundantly stored under several storage schemes, or storage schemes may be revised by the computer as the distribution changes with time.

For most purposes, such a high service rate would not be necessary, and one of the two following storage plans would be satisfactory. One or the other plan would be chosen on the basis of the predominant types of requests.

Fact Storage by Unit Record. A fact consists of an interrogation and the answer to that interrogation, derived from a particular source document. It is also associated with a particular staff department, factory, sales district, or other Unit Record. A Unit Record storage plan is shown in Figure 4. In the Unit Record shown, all information extracted from factory XYZ source documents is stored. Major columns are the statement generic address column and the response columns. A response consists of the answer extracted from a particular document (represented by its document number) and the date (date on document). Rows indicate specific generic addresses of interrogations used to extract information. An intersection of row and column will provide an answer to a particular interrogation referenced to the associated document number and date. In actual storage, each row is a separate variable-length field, and responses are scanned for selection purposes. Responses are stored in inverse time sequence, so the latest answer is nearest its generic address.

Fact Storage by Substantive Record. The second storage plan is based on the part of speech which stands for the name of the thing or object of interest—the substantive. Storage of factual statements by the names of things referred to in the statements is a powerful method because these names or substantives play a dominant role in most retrieval requests.

The layout of a Substantive Record storage plan is shown in Figure 5. Each Substantive Record contains the generic addresses of all statements which contain the particular substantive. The addresses are arranged in sequential order as in the Unit Record storage plan. However, as each address refers to one or more Unit Records, code numbers occur as sub-entries under each address. These represent the Unit Records for which each generic address applies. Therefore, generic addresses act as main entries and have Unit Record code numbers as sub-entries. Each sub-entry has stored with it a set of responses in the same manner as the Unit Record.

Retrieval

The concepts and procedures discussed up to this point pertain to the extraction of data from source documents and the storage of this data in an organized manner in a digital file unit. The stored data consists of factual statements—each statement consisting of an interrogation and a variable number of answers to the interrogation. Each answer is referenced to the number of the document from which it was obtained and the publication date of the document. Each statement is generically related to other statements in an interrogation list and is filed in accordance with the Unit Record to which it relates.

The retrieval system is capable of selecting the set of statements, the set of answers to these statements and, if required, the applicable document numbers that satisfy the conditions in a request for information.

Retrieval requests may take many forms. If, for instance, a top executive were interested in receiving the most up-to-date information on production schedules or sales forecasts, a console could be provided for him with a push-button for each category of information. Whenever the executive pushed one of the buttons, a signal would be sent to the computer which would select the latest answers to the relevant statements and present this information on a display device in the executive's office. For highly standardized requests, this method would be simple and effective. However, if all requests were of this nature, the Fact Compiler System would probably not be economically justifiable, as this type of limited, highly predictable information retrieval could probably be performed by humans at less cost.

As mentioned previously, the Fact Compiler is able to produce routine reports and present timely information on a daily basis automatically. It is felt, however, that the most important advantages of the system are its ability to retrieve and relate items of information not ordinarily compared or mentioned in normal business reports, and its capability of reviewing the past history of various aspects of business information which can be stored in some detail is desired. The ability to "probe in depth" into historical business data would seem especially advantageous for staff department personnel engaged in analysis of data for presentation to top management.

To provide this kind of capability, communication with the Fact Compiler must take place in a language powerful enough to express the many different and varied requests and yet simple enough so that humans can readily express what it is that they want from the file. An important part of the retrieval language is the vocabulary of terms used in defining the interrogations themselves. Terms are chosen from this vocabulary and are combined to express the factual information desired. A thesaurus will facilitate the correct choice of terms.

Restriction of the request to Interrogation List or Unit Record categories is made by including the names of the lists or records of interest. Periods of time for answer selection may be incorporated by using date operators such as: before-date, after-date, during-month (or year). Grouping within a request will be provided by the use of the logical operators AND-OR and appropriate use of punctuation. Certain arithmetical operations are allowable for selection and processing of answers. Examples are: Answers $\geq$ or $\leq$ a constant (for selection); sum, average (for processing a set of related answers). A retrieval request may be visualized as an algorithm which is capable of causing the Fact Compiler to select and present desired information to the requestor. The Fact Compiler retrieval language of English terms and operators permits appropriate connection of terms, restriction of request scope, and definition of processing operations. The combination of appropriate terms and operators will allow the ready generation of algorithms for the retrieval of information. The retrieval process is outlined in Figure 6.

As an example of a retrieval request, suppose it is necessary to have production and sales information on all dishwashers produced after June 1950. It is desired to relate this information to changes in the Gross National Product, Index of Industrial Production, and dishwasher development cost. The request may be formulated as follows: (Dishwasher, model number of, actual-forecast sales of monthly, actual-forecast production of monthly, development cost of; Gross National Product; Index of Industrial Production) after June 1950.

1. Parenthesis indicate all answers are to meet the date operator requirement. The date operator is "after June 1950."
2. Commas separate related terms.
3. Dashes separate modifiers.
4. Semicolons separate unrelated terms.

The decoding section of the retrieval program would break down this request, via the operators used and a knowledge of the allowable syntax patterns in statements, into the following phrases which may be partial or complete statements in the file.

1. Model number of dishwasher.
2. Actual monthly sales of dishwasher.
3. Forecast monthly sales of dishwasher.
4. Actual monthly production of dishwasher.
5. Forecast monthly production of dishwasher.
6. Development cost of dishwasher.
7. Gross National Product.
8. Index of Industrial Production.

The model number and production data may be stored in a Factory Unit Record. Development cost may be located in a Research and Development Unit Record. Sales information may be contained in several Regional Sales District Unit Records. Items 7 and 8 are reportable under a General Economic Parameter Unit Record.

In a file organized by substantives, all information is found under the three Substantive Records "Dishwasher," "Gross National Product," and "Industrial Production." However, in order to find the information, the generic addresses of the pertinent statements must be known. These generic addresses are found through the use of a fact

index. This index contains entries for all terms in the vocabulary and, for each entry, lists the generic addresses of each interrogation which contains that particular term.

Consider phrases 1 through 6 above. Each phrase consists of three separate terms. All addresses of facts that pertain to one of the phrases may be obtained by matching the addresses for each of the three entries and selecting those that are common to all three. When the appropriate sets of addresses are selected, the three Substantive Records may be directly addressed, the generic addresses located, and the date restrictions applied to date-answer combinations.

It may happen that retrieval requests that would be unanswerable may be formulated. This would most likely happen when too specific a request is made, so that the combination of terms used does not occur in any statement in the file. A request of this type would immediately be detected since conjunction of terms in the fact index would result in a zero set of generic addresses. This fact would be immediately presented to the requestor so he could modify his request.

Conversely, if a request were quite general, a tremendous amount of information would be presented. Before printing selected information, the computer would indicate the approximate volume of information that has been selected.

Monitor and Control

Figure 1 shows feedback to Monitor and Control from the input or extracting process, and the output or retrieval process. This feedback is an important part of the self-correcting procedure necessary to keep system performance at a high level of efficiency.

Feedback from the input process occurs when an input analyst finds that he cannot enter what he considers significant data into the machine memory because there are no appropriate interrogations in the interrogation vocabulary.

Feedback from the output process occurs when system users are not satisfied with the information contained in system-generated reports or answers to retrieval requests.

In either event, operator personnel must decide whether system capabilities should be changed to permit extraction, storage, and retrieval of new data. In making such decisions, they must be guided by firm management policy on the use of the system.

Systems capability may be modified by making changes in the interrogation vocabulary or criteria, the Restricted English vocabulary, or the Retrieval Operator vocabulary. Such changes are easily made at a special display console operated only by monitor personnel.

Additional functions of the group are to translate natural language requests into Restricted English, and to control and operate the system in accordance with the wishes of management.

Conclusion

The Fact Compiler System represents one solution to one category of information storage and retrieval problems. The predominant philosophy involved in its development has been that of an appropriate division of labor between humans and machines. Humans are most efficient in matters of judgment, machines in routine and precise processing. A machine will always need human-generated criteria in some form for determining what kind of information is significant or important since an absolute measure of the importance of information does not exist.

The value of a Fact Compiler System will be proportional to the amount of judgment and effort exerted in producing and maintaining a set of interrogations that adequately express the real requirements of system users. Producing such a set of interrogations is not an easy task but it is presently being done and is greatly facilitated by the use of a series of rules for translation from ordinary English to Restricted English Interrogation Lists.

A clearly defined unambiguous vocabulary in a specialized field of interest permits direct storage of factual information within a computer memory file. This file may be searched in several "dimensions" of index space. They are:

1. time,
2. subject category,
3. organizational unit (Unit Record),
4. aspect (substantives and their modifiers), and
5. degree of specificity (generic level).

In addition, the Fact File may be used as a powerful index to source documents. Requests may be formulated in a vocabulary which is easy to learn and to use. Generally-oriented or "browsing" requests may result in the discovery of relations between data never before apparent because of departmental report boundaries or because of the long time periods involved.

In many instances, management should be able to retrieve facts from the file quicker than they can find a report in a desk or file cabinet. Immediate access to this "extended memory" could help management in speeding up the decision-making process.

The Fact Compiler System equipment and computer program is capable of handling any interrogation vocabulary and set of facts that are generated in an appropriate form. For example, use by a large university might consist of allocating system time of several weeks each to medical, business, and engineering schools. Thus, the medical school might generate a Fact File of information about the diagnosis of various diseases, the business school might use the Fact Compiler to compare data from major industries, and the engineering school might use the system to store and retrieve data on use and development of digital computers. Each school could read in its own vocabulary and data, and then extract, store, and retrieve for the duration of its allocated time. At the end of the six-week cycle, the process could be repeated.

The organization of information within the Fact Compiler will permit the development of statistical procedures for automatic correlation of data within the file. This will be one of the objectives of future

system studies. It is expected that these studies will result in a capability for automatically finding cause and effect relationships between various types of information in the file.

Acknowledgement

The author wishes to acknowledge the valuable suggestions of S. Rothman, J. Dealy, and R. Penne of Ramo-Wooldridge, and the important work of Planning Research Corporation in the area of language studies.

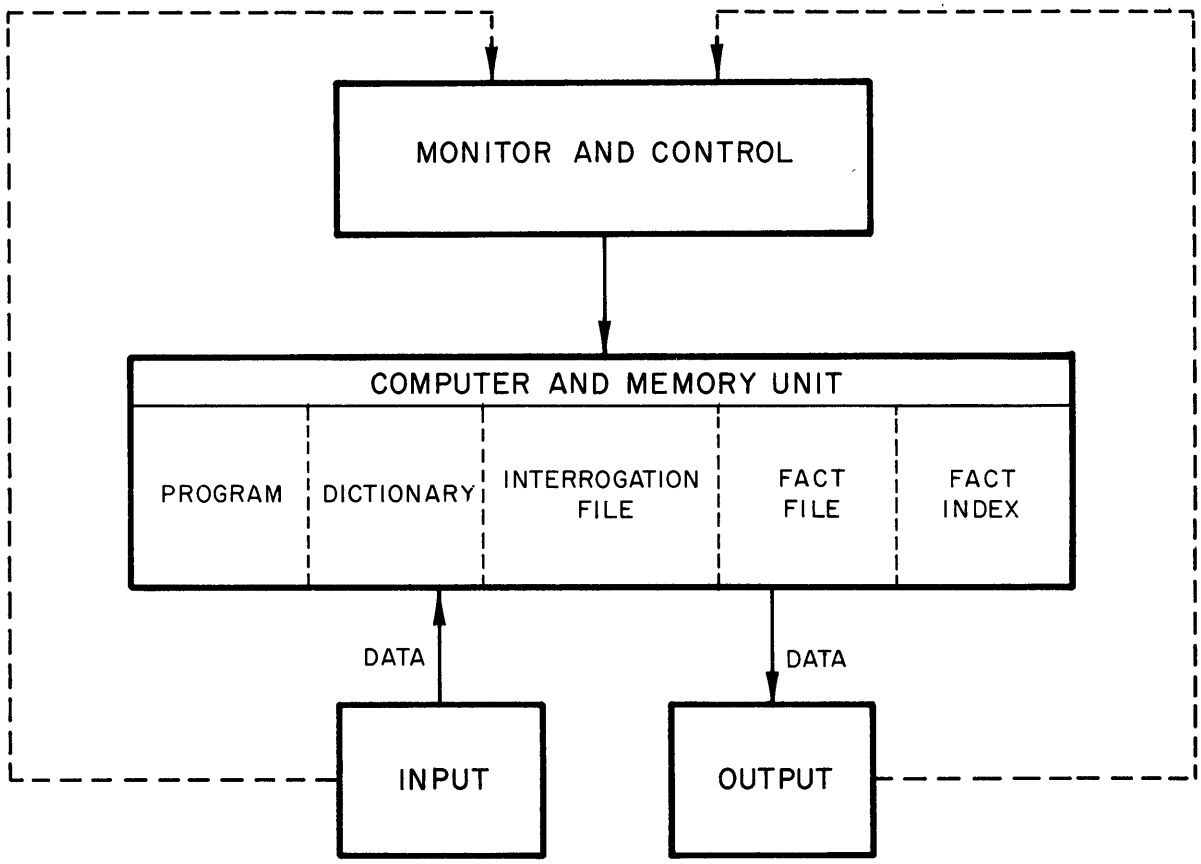


Figure 1 The Fact Compiler System

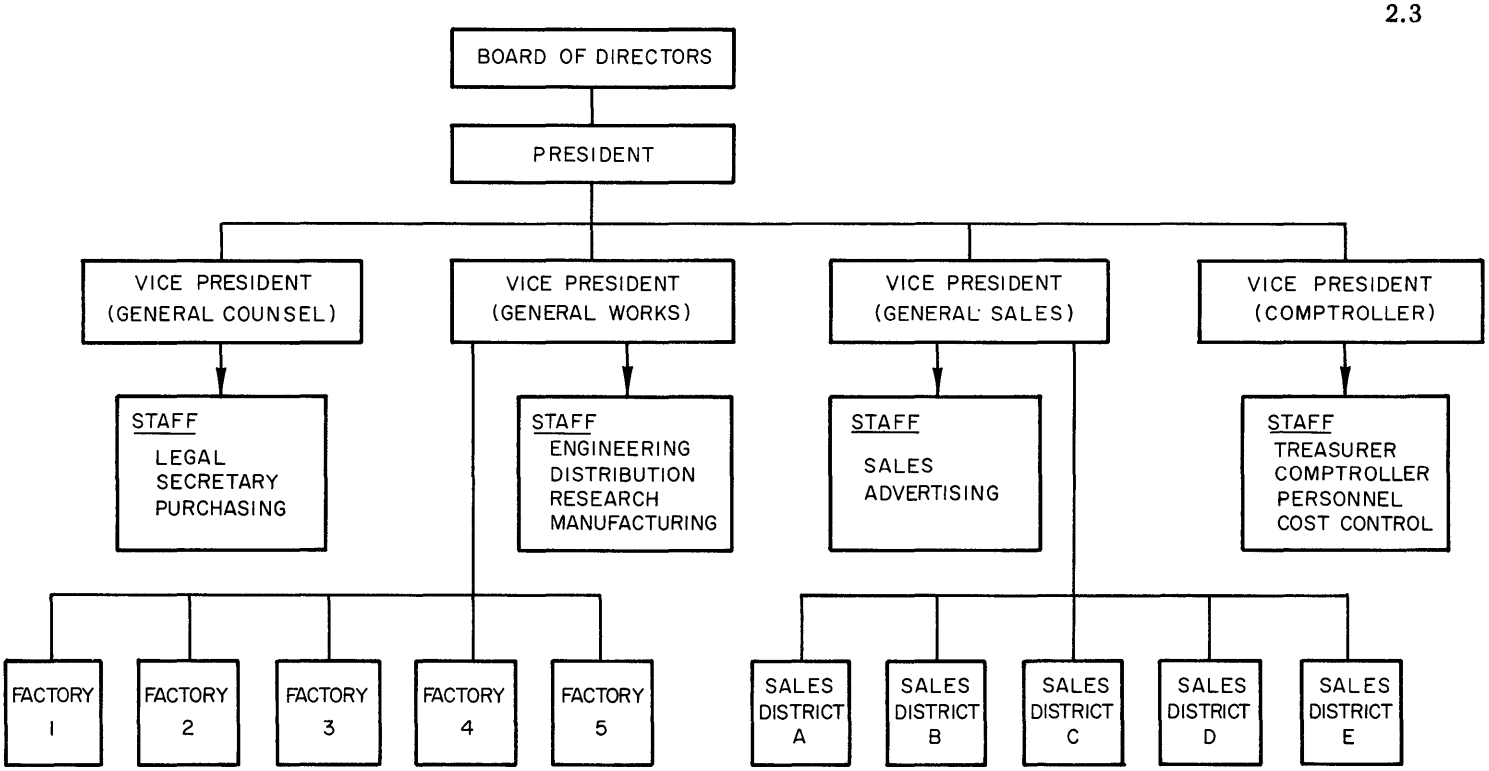


Figure 2 Business Organization Chart

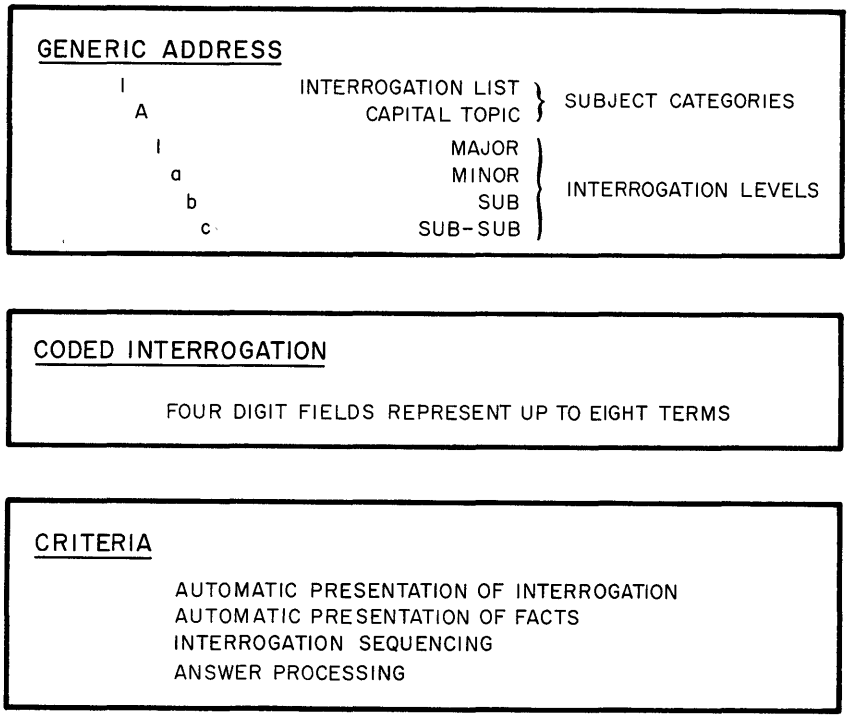


Figure 3 Interrogation File Organization

TITLE: FACTORY XYZ						
GENERIC ADDRESS	RESPONSE N			RESPONSE N-1		
	ANSWER	DOCUMENT NUMBER	DATE	ANSWER	DOCUMENT NUMBER	DATE
1A1aa						
1A1aaa						
1A1aab						
1A1aac						
1A1ab						
1A1aba						
1A1abb						

Figure 4 Unit Record Storage Plan

TITLE: OPERATING BUDGET							
GENERIC ADDRESS	UNIT RECORD CODE	RESPONSE N			RESPONSE N-1		
		ANSWER	DOCUMENT NUMBER	DATE	ANSWER	DOCUMENT NUMBER	DATE
1A1aa							
	03						
	05						
	09						
1A1aaa							
	03						
	05						
	09						
	10						
1A1aab							
	03						

Figure 5 Substantive Record Storage Plan

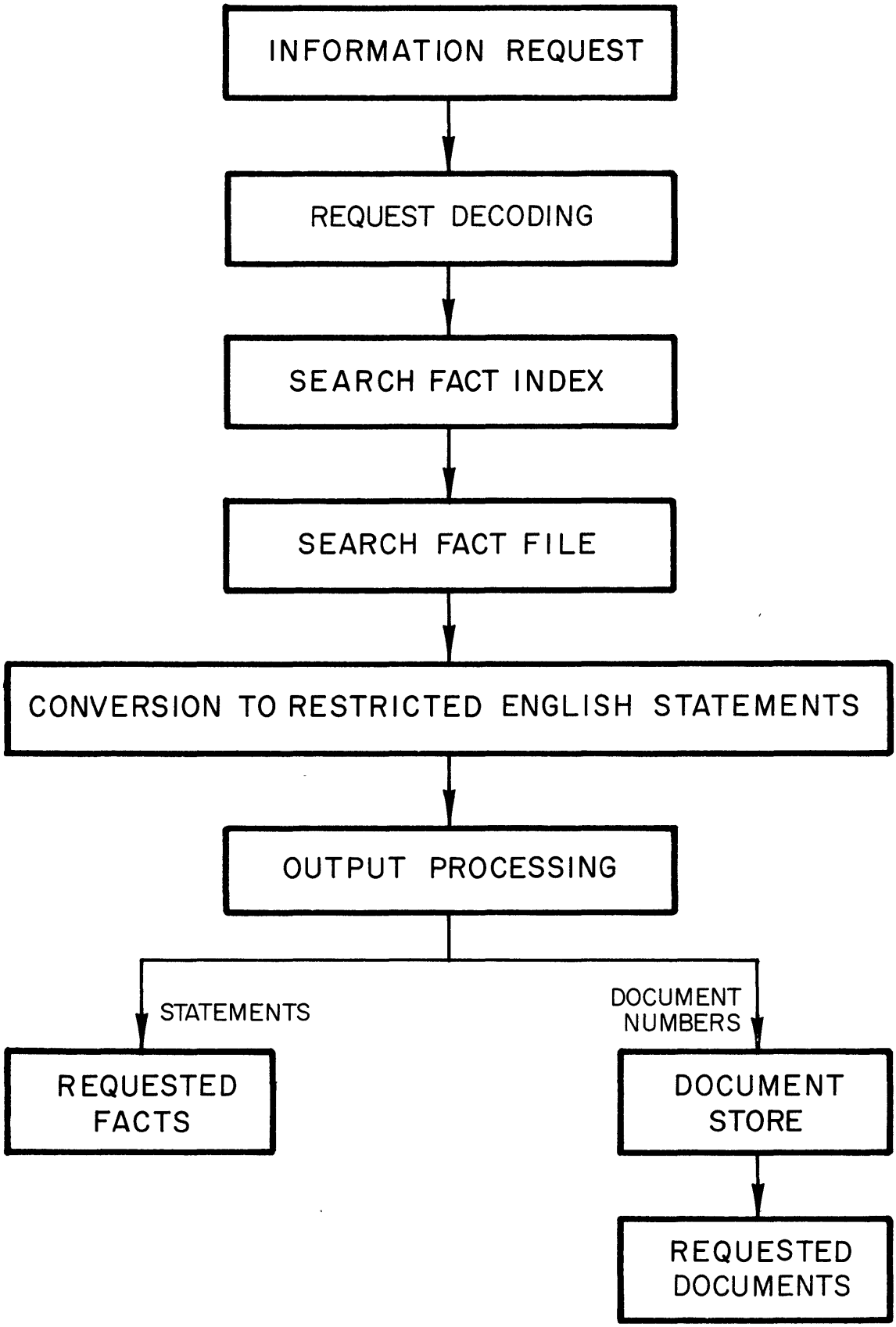


Figure 6 Retrieval

SUBSTANTIVE	DESCRIPTOR	ACTIVITY CONNECTOR	RELATIONAL CONNECTOR	INTERROGATIVE OPERATOR
GROSS NATIONAL PRODUCT	NEW	MOVING	IN	FORECAST VARIANCE OF
BUDGET	NAME	ARRIVING	AT	INVENTORY OF
CORPORATION	DECREASED	UNDERGOING	FROM	AMOUNT OF
KEY PERSONNEL	ESTIMATED	INCLUDING	FOR	PERCENTAGE COMPLETION OF
FOREIGN OPERATION	HEAT-TREATED	BEING MODIFIED	BY	NUMBER OF
SALARY	STATISTICALLY-CONTROLLED	BEING CONSIDERED	BEFORE	DOLLAR VOLUME OF

Table 1 Typical Restricted English Terms

A WORD-ORIENTED TRANSISTOR DRIVEN NON-DESTRUCTIVE READOUT MEMORY

by
T. C. Penn and D. G. Fischer
Central Research Laboratories
Texas Instruments Incorporated
Dallas, Texas

Summary

Using three-aperture ferrite memory cells, a 512-word non-destructive readout memory has been operated with transistor drivers. Writing and reading are both achieved with bipolar two-pulse sequences, which are, respectively, ERASE, WRITE and SET, READ. Due to the inherently non-destructive manner in which the cells are operated, no regeneration cycle is necessary to restore information to interrogated cells. The necessary bipolar writing and reading pulse sequences are conveniently provided by a novel magnetic switch matrix, in which all rows and columns are driven in coincidence except those intersecting at the desired address. The resultant driving current at the desired address is the sum of all the transistor driver currents, or 2(n-1) times the individual transistor driver currents for an "n" square matrix.

Introduction

Since the invention of the ferrite core coincident-current computer memory, there have been several systems advanced to utilize the best features of the coincident-current system while improving upon or circumventing its deficiencies. Many of the alternate methods utilized a multiaperture device fabricated from a square hysteresis-loop type of ferrite as the building block for the memory storage or logical operations performed by the system. Each scheme used the ferrite material's ability to store a flux level or previously ordered flux geometry without a constant power expenditure. The Transfluxor, of Rajchman and Lo[1, 2], offered proportional output instead of just ONE-ZERO operation, or, if desired, a random-access ferrite core memory with non-destructive readout. The Hunter and Bauer[3] coincidence flux schemes as further improved by Baldwin and Rogers[4] were able to circumvent the coincident-current memory's inherent maximum drive current limitations and achieve improved access and switching times for a memory system. The dc bias offset method as used by Lawrence[5] overcame the necessity of using square hysteresis-loop ferrite material. All of these systems were built upon multiaperture ferrite cells and required driving currents somewhat larger than the coincident-current system. Since the coincident-current system in turn required driving currents on the order of one-half ampere, it was not feasible until recently to utilize solid state devices for driving a mamoery system. Constantine[6] has shown a load sharing matrix switch which is suitable for some applications.

The memory system described here is also based upon a multiaperture ferrite cell which, superficially, differs little from previous such schemes, either in driving currents or core geometry. This system, however, embodying the multiaperture cell and the novel driving matrix developed for it, offers the prospect of a random-access transistor-driven computer memory with non-destructive readout and access times comparable to a coincident-current system.

The multiaperture memory cell operation will first be described in detail. The novel driving matrix and the memory planes for a system using these cells will then be explained and finally the entire memory system is desscribed.

Multiaperture Memory Cell

The basic memory cell is shown in Figure 1. Cell geometry is important and the dimensions of this cell were fixed largely by compromise between performance, drive requirements, and fabrication ease. Cell performance is a fairly sensitive function of relative hole sizes and spacings (geometry), while drive currents vary inversely to the absolute hole diameters. The smaller the cells, the more difficult they are to fabricate by pressing from unfired ferrite powder. The cell chosen was near the minimum size which could be manufactured in the laboratory without special production techniques.

Every memory cell is normally linked by three windings, one each for Drive, Inhibit and Sense functions. Except for the necessity of noise cancellation in a plane of these memory cells, the Inhibit and Sense operations could both be served by a common winding.

Writing and Reading are each accomplished with a bipolar, two pulse sequence applied to the Drive winding, utilizing driving core output in both drive and reset directions. This allows a maximum simplification of the means connecting driving core and memory cell.

The pulse sequences used are shown in Figure 2. ERASE, READ, WRITE and INHIBIT pulses are typically 0. 5 to 1. 5 microseconds wide, depending upon clock rate used. SET pulse is typically 1. 5 times the width of the READ pulse. READ and WRITE pulses for simplicity are of equal amplitude, typically 1. 5 to 2. 5 amperes. ERASE and INHIBIT pulses are typically 1. 5 to 3. 0 amperes. SET pulse magnitude is generally a third to half of the ERASE current. Output voltages are typically 1 to 2 volts, with 0. 1 to 0. 2 microsecond widths. ONE/ZERO ratios are generally 5/1 to 10/1 for and individual cell.

A ONE is written into the cell by applying to the Drive winding the pulse sequence shown in Figure 2a.

If the cell was previously unmagnetized or was storing a ONE, the cell's flux distribution following the ERASE and WRITE pulses are shown in Figures 3a and 3b respectively. A pulse of either polarity into the Drive winding results in a flux around the center hole only, regardless of magnitude. Although a current through the Drive winding applies a magnetizing force around both the small center and large left-hand holes, the maximum primary magnetic path length around the small hole is less than the minimum magnetic path length around the large hole. As the current pulse increases from zero, the area encircling the small hole is saturated before the mmf applied around the large hole becomes sufficient to overcome the threshold coercivity of the ferrite material. The flux encircling the large hole cannot increase significantly through this saturated region.

Considering the above, it is seen that the ERASE and WRITE pulses result respectively in clockwise and counterclockwise saturation around the small hole as shown in Figures 3a and 3b.

A ZERO is written into the cell by applying an INHIBIT pulse in time coincidence with the WRITE pulse. As before the preceding ERASE pulse would have destroyed any stored information. The flux distribution following the INHIBIT or WRITE ZERO sequence is shown in Figure 3c. Saturation around the outside holes occurs because no net current flows through the center hole and mmf is applied only around the outside holes. This mmf directly opposes the post ERASE saturated flux around the small hole in the leg common to the left-hand hole. The mmf resulting from WRITE and INHIBIT pulses in coincidence therefore destroys the flux encircling the small hole and allows this flux to be redirected around the outside holes.

Readout is accomplished with the bipolar pulse sequence of Figure 2c applied to the Drive winding. To avoid extraneous output signals, stored information is detected by observing the Sense winding output voltage only while the READ pulse is applied to the Drive winding. Changes in flux linking the Sense winding during ERASE, WRITE and SET operations may also induce voltages in the Sense winding. These are, however, ignored and only those Sense winding outputs occurring in synchronism with the READ pulse are relevant to information stored in the cell.

If the interrogated cell stores a ONE, the small SET pulse assures that the flux encircling the center hole will be in such a direction that the following READ pulse will reverse it and induce an output in the Sense winding. Successive READ sequences may repeat this dual reversal as often as desired, resulting in non-destructive readout of a ONE.

If the interrogated cell stores a ZERO, the small SET pulse is of insufficient magnitude to overcome the coercivity of the flux path surrounding the large left-hand hole. The SET pulse leaves the ZERO-storing cell arranged such that the following READ pulse causes no flux change about the center hole and therefore induces no output voltage(ZERO readout) in the Sense winding. Repeated READ sequences also fail to effect flux changes in the cell storing a ZERO, allowing unlimited non-destructive readout.

From the discussion of the READ sequence, the governing factors determining SET pulse parameters will now be apparent. To obtain as fast operation as possible, it is desired to make the SET pulse of high amplitude. As the SET and ERASE pulses are of the same polarity, there is obviously a SET pulse amplitude above which a stored ZERO is destroyed. By reduction of the SET pulse amplitude below the destructive readout threshold, its volt-time area and hence flux reversal capacity has been decreased. The SET pulse is therefore increased in width to switch the majority of flux encircling the small hole of a ONE storing cell.

The erase mechanism may be clarified by the following explanation. If the cell previously stored a ONE, the ERASE pulse has the same effect as a SET pulse, and the resulting flux configuration is as shown in Figure 3a. An ERASE pulse applied to a cell storing a ZERO first reverses the clockwise flux encircling the left-hand hole, since the resulting mmf opposes this flux. This mmf also opposes the flux in the outside leg of the right-hand hole. Once the opposing ERASE mmf is large enough to destroy the fluxes encircling the outside holes, the mmf around the center hole is very intense and the available flux in the center legs is reoriented around the center hole as shown in Figure 3a.

The memory cells are normally arranged in multicell planes or matrices in which all cells are threaded by common Sense and Inhibit windings. Inhibiting is done on a planar basis and the effects of a random INHIBIT pulse on a non-selected cell must be considered. If the cell stored a ONE, the random INHIBIT pulse simply reverses the flux surrounding the center hole, just as a SET pulse would have done, and the stored ONE is unharmed. A READ sequence then applied to a cell in this condition would produce the usual Sense winding ONE output in synchronism with the READ pulse.

A random INHIBIT pulse applied to a cell storing a ZERO produces an mmf in the direction of the saturated flux around the right-hand hole and causes no flux change. It is seen that though the random INHIBIT pulse may cause flux changes in the unselected cells, no adverse effects occur and stored information is not destroyed.

The flux distributions utilized to explain the multiaperture cell's operation have been verified experimentally by integration of the voltages induced in a winding threading one or more holes at a time. By this technique it was possible to observe the net flux about any single hole or through any of the various legs of the cell. Though the flux distributions thus obtained may undoubtedly be explained in other terms, such as those employing crescent-shaped flux patterns, the relatively simple explanation presented here, supported by ample experimental evidence, is felt to be perfectly valid.

Complement Switch Matrix

The bipolar pulses required for the pulse program of the previously described multiaperture memory cell are conveniently supplied by a switch matrix of tape wound cores. To achieve the drive currents desired for high speed operation and still use transistors as the pulse generators, a non-conventional circuit is used. No dc reset current is used, allowing all of the transistor current flowing to be used for switching. In the complement switch matrix to be described, the complement of the row and column address is excited, i.e. all rows and columns except those intersecting the desired core. For example, a 3x3 array is shown in Figure 4 with appropriate drive windings. For simplicity no reset or word drive windings are shown. All cores are initially mangetized clockwise. If it is desired to select core 9, all rows and columns other than row 3 and column 3 are excited at an amplitude I. The excitation received by cores 1, 2, 4 and 5 is equal to the algebraic sum of the magnetomotive forces due to the 4 turn row, 4 turn column and single turn cancel windings. For these cores that is 4I+4I-4I=+4I, clockwise magnetization assumed positive. Thus none of these cores are switched irreversibly. Cores 3, 6, 7 and 8 are excited either by a row or column winding (not both) and the single turn cancel winding. Their excitation is therefore 4I-4I=0. Selected core 9 is driven exclusively by the single turn cancel winding. Core 9 is thus excited by -4I which causes it to switch irreversibly. The effect of this wiring scheme is to permit transistor currents to parallel into a single turn through the selected core.

In a practical system a reset winding is wired to produce clockwise magnetization in every core. In the system to be described, two identical complement switch matrices are used. The cancel winding of one matrix is extended and threaded in reset fashion through the other matrix. The number of turns required for the row or column windings of an "n" square matrix is 2(n-1). Individual word drive lines are associated with each switch core on a core per word basis as shown in Figure 5.

The transistor circuitry associated with the complement switch matrix drivers is shown in Figure 6. Current routing, current mode switching and grounded base switching are used to achieve speed from economical transistors. When either T_3 or T_4 is driven by the excite pulse in the absence of a negate pulse on D, the emitter of T_2 is driven positive with respect to its base and conduction takes place. If a 16x16 complement switch is used, 29 transistors associated with the other rows and columns are driven in like manner. One row and colum will be negated for address purposes. On the application of an excitation pulse simultaneously with a negate pulse, diode D conducts turning off T_1 which drives the base of T_2 more positive than its emitter. T_2 therefore will not conduct these conditions. The voltage on the common bus is low due to the conduction of thirty transistors in parallel compared with the voltage developed across the switched core. Independent supply voltages E_1 and E_2 allow control of the pulse amplitudes and durations for two separate excite pulses later to be called ERASE and SET. Though this circuit appears deceptively simple, many subtleties caused other circuits to be rejected in favor of this simple one. Enumeration of some of the more important considerations will be discussed during system description.

Memory Cell Matrix

Figure 7 illustrates a 3x3 memory plane wiring scheme for non-destructive readout of multiaperture cells showing planar Inhibit and Sense windings. Noise cancellation is used as in conventional memory planes, necessitating a separate inhibit line.

Memory System

Figure 8 shows the block diagram of a test vehicle embodying previously described circuits. In operation one switch matrix is operated in a "selective mode", i.e. it selects the address of the word to be written or read. The other switch matrix is operated in a "non-selective" mode, to reset the selective mode switch matrix. Either of the switch matrices may be used for either mode, but for simplicity SM-1 will be described in the complement selective mode and SM-2 will be chosen for non-selective mode reset. Thus SM-2 will be driven in such a manner that no cores in SM-2 switch by driving all rows and columns; but SM-2's current will switch the previously selected core in SM-1 back to the reset position.

The program shown in Figure 2 will be used for a four pulse sequence of ERASE, WRITE, SET and READ. SM-1 driven selectively generates a large ERASE pulse which clears all memory cells on the addressed word line. The polarity of the ERASE pulse is opposite to the READ and WRITE pulses. This operation is followed by driving SM-2 non-selectively which resets the addressed core of SM-1. A WRITE pulse is generated by this operation which is of sufficient magnitude to write a ONE in all addressed cells. Those planes in which a ZERO rather than a ONE is desired are energized simultaneously with an INHIBIT pulse applied to the memory plane inhibit winding. Both SM-1 and SM-2 are now in the normal state, that is all cores switched to the same sense. Clearly this operation may be followed either by another Write sequence or, as will be shown, by a Read sequence.

The read operation consists of two phases; the SET and READ. SM-1 is driven selectively at an amplitude which will not clear cells in the ZERO state and for a duration sufficient to switch all the cell flux desired. Following the SET pulse, a READ pulse is provided by driving SM-2 non-selectively to reset SM-1. Simultaneously a strobe gate energizes the memory plane sense amplifiers. If a ONE is present in a plane, sufficient output voltage will be generated to excite the sense amplifier. If a ZERO is present, the noise voltage is of insufficient amplitude to energize the sense amplifier. Strobing is necessary to prevent spurious outputs during the other operations.

Governing factors pertaining to the previously described row and column switching circuit may now be considered. It had first been thought that adequate speed and current could be obtained by using a dc source and driving row and column gates in parallel to excite the switch matrix. The required independent control of duration and amplitude for the ERASE and SET pulses, however, make this method very difficult to implement. Another fault of such a scheme is the loading effect caused by the transistor drivers during reset operation. These loading effects may be due to one or all three of the following:

1. Minority carrier storage
2. Output capacitance of the driver circuits
3. Conduction of the driver due to polarity of the induced voltage (such as emitter followers).

In the configuration described, larger output currents result at speeds not possible in the dc collector supply type of operation. Further, the polarity of the induced voltage during reset aids turn off by negatively biasing the emitter of T_2 in Figure 3. The adjustment of SET pulse amplitude may be performed in a single stage by a simple change of dc level. Negation of a row and column is independent of the applied ERASE and SET pulse amplitudes and durations, greatly simplifying the system.

In the test vehicle shown, although random access is allowable, the test program used provides sequential addresses via a shift register ring for negating one row and column at a time. This greatly aided trouble shooting during "de-bugging".

The writers are indebted to W. A. Kluck whose basic ideas started this investigation and to W. T. McKay, Jr. for his circuit contributions.

References

[1]Rajchman, J. A. and Lo, A. W., "The Transfluxor - a Magnetic Gate With Stored Variable Setting", RCA Review, vol. 16, June, 1955.

[2]Rajchman, J. A. and Lo, A. W. "The Transfluxor", Proc. I.R.E., vol. 44, p321; March, 1956.

[3]Hunter, L. P. and Bauer, E. W., "High Speed Coincident - Flux Magnetic Storage Principles", J. Appl. Phys., vol. 27, p1257; November, 1956.

[4]Baldwin, J. A., Jr., and Rogers, J. L., "Inhibited Flux - A New Mode of Operation of the Three-Hole Memory Core", J. Appl. Phys., vol. 30, p58s; April, 1959.

[5]Lawrence, W. W., Jr., "Recent Developments in Very High Speed Magnetic Storage Techniques", Proc. Eastern Joint Computer Conf., 1957, p101.

[6]Constantine, G., Jr., "A Load-Sharing Matrix Switch", IBM Journal of Research, p204; July, 1958.

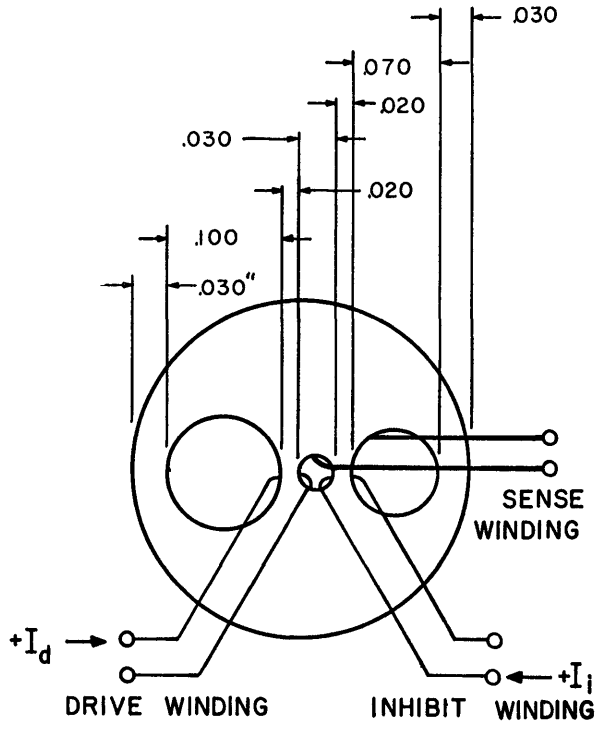


Figure 1. Dimensions and windings for a typical multiaperture memory cell. Typical thickness is about 0.045 inches.

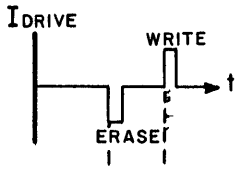


Figure 2a. Bipolar current pulse sequence applied to Drive winding for storing either a ONE or a ZERO.

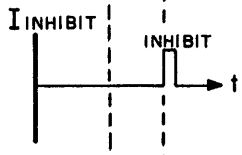


Figure 2b. Current pulse applied to Inhibit winding for inhibiting or writing a ZERO.

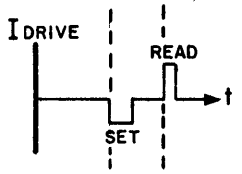


Figure 2c. Bipolar READ sequence for interrogating multiaperture cell.

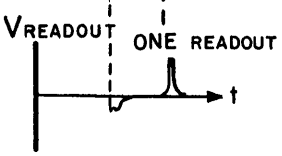


Figure 2d. Typical ONE readout voltage from interrogated cell, showing coincidence of readout and READ pulses.

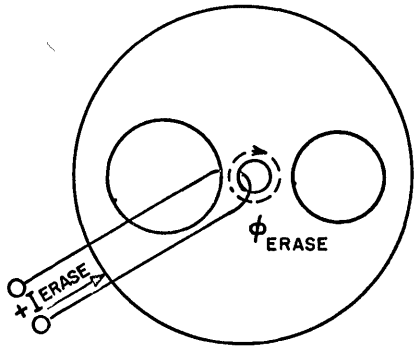


Figure 3a. Flux distribution resulting from application of ERASE pulse to a cell storing a ONE.

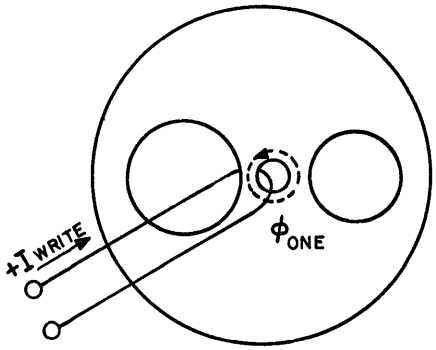


Figure 3b. Flux distribution resulting from WRITE pulse. The cell is now storing a ONE.

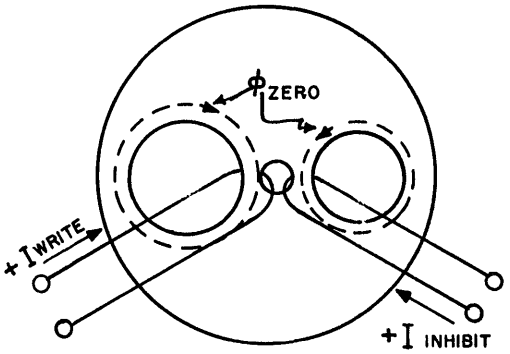


Figure 3c. Flux distribution resulting from INHIBIT or WRITE ONE sequences. The cell is now storing a ZERO.

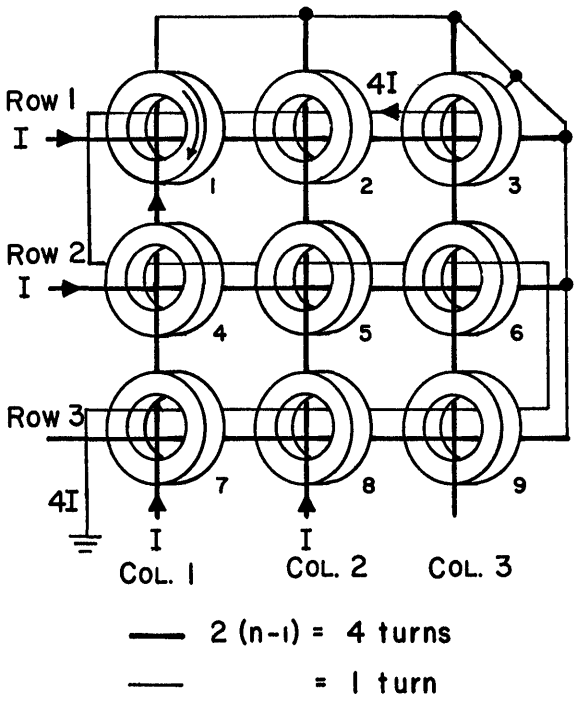


Figure 4a. Complement switch matrix.

+4I	+4I	0
+4I	+4I	0
0	0	-4I

\+ = Clockwise Initial Magnetization

\- = Counterclockwise Switching Direction

Figure 4b. MMF chart.

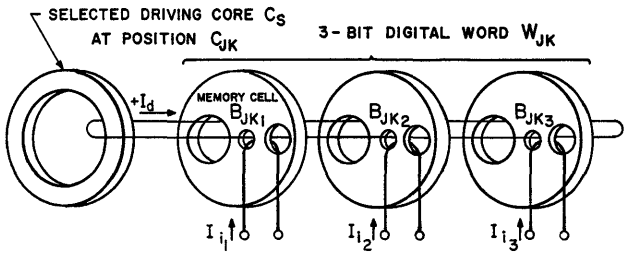


Figure 5. Driving core and word line scheme used to drive multiaperture memory cells.

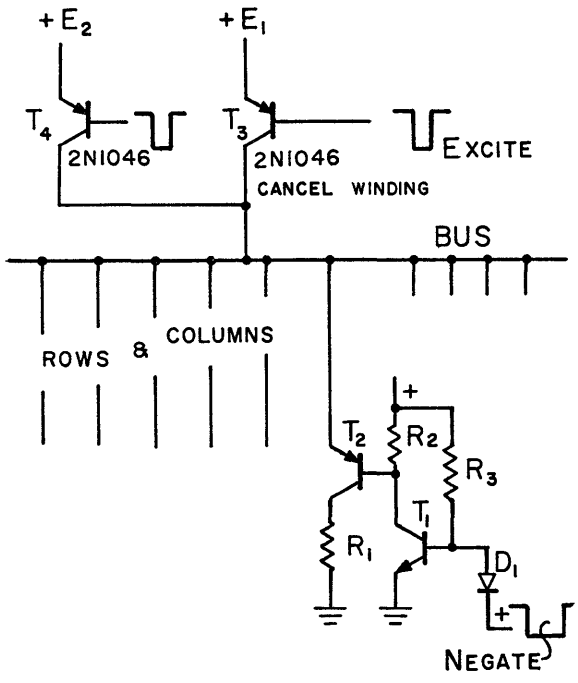


Figure 6. Transistor circuitry for driving switch matrix.

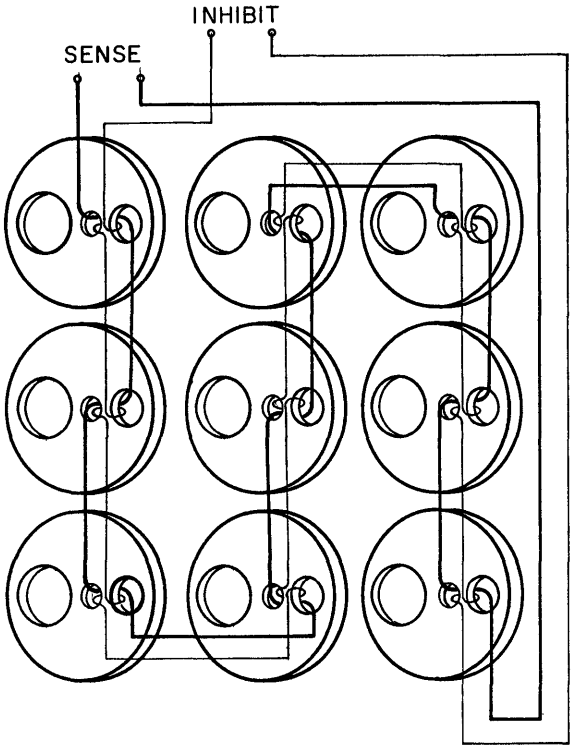


Figure 7. Typical memory cell matrix with planar Inhibit and Readout or Sense windings.

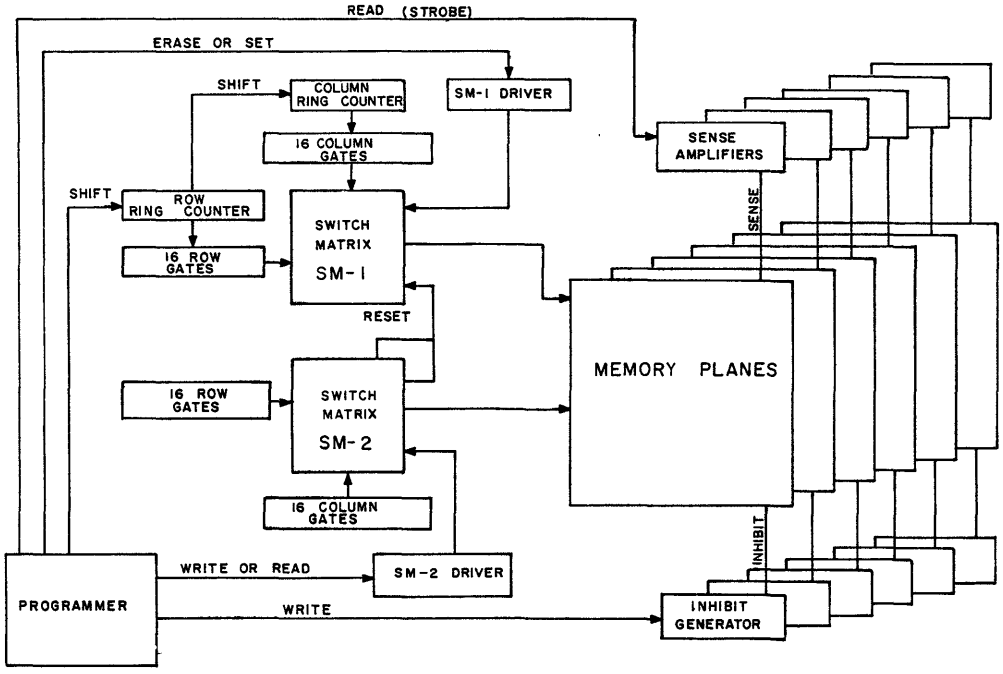


Figure 8. Test vehicle for system utilizing multiaperture memory cells.

UNIFLUXOR: A PERMANENT MEMORY ELEMENT

A. M. Renard & W. J. Neumann
Remington Rand Univac
St. Paul, Minn.

Introduction

The Unifluxor is a new binary permanent memory element which appears to have the advantages of high-speed operation, easy fabrication, and low cost. Unlike cores, twistors, capacitors, and other commonly used memory devices, the Unifluxor does not depend upon the hysteretic properites of some nonlinear material but instead uses the inductive characteristics of magnetically coupled wires.

A Unifluxor memory array consists of a printed-circuit board upon which are etched longitudinal drive lines and transverse sense lines. Each intersection of a drive and sense line represents one bit, with the bits of one word lying along the same drive line. Thus the array operates in the word-organized mode.

The particular state of each bit depends upon the presence ("one") or absence ("zero") of a copper slug. The copper slugs, properly spaced and oriented, are contained on a plastic film cemented in place over the printed-circuit board (Figure 1). In order to change the contents of the memory, it is necessary to substitute a new cover film with the desired pattern of copper slugs.

Theory of Operation

"Zero" State

Consider a wire in the plane of and perpendicular to a closed wire loop (Figure 2). When the switch is closed and current flows in the wire, a cylindrical magnetostatic field is created around the wire, the effective boundary of which is indicated in the diagram by dotted lines. The total flux within the loop then is $\emptyset_1 + \emptyset_2$.

At the time the switch is closed and the field is building up, an emf (E) is induced in the loop according to Faraday's equation:

$$E = -N d\emptyset_t/dt$$

$$= -N(d\emptyset_1/dt + d\emptyset_2/dt) \qquad (1)$$

Since $\emptyset$ is equal to the product of the flux density (B) and flux area (A), equation 1 may be rewritten as:

$$E = -N\left[d(B_1A_1)/dt + d(B_2A_2)/dt\right] \qquad (2)$$

For the case illustrated, the following equalities exist:

$$B_1 = -B_2$$

$$A_1 = A_2$$

Thus, the emf's generated in the two halves of the loop are equal in magnitude but opposite in sign, and the net output is zero. This in fact is the condition for a Unifluxor in the "zero" state where the loop is the sense line, the wire the drive line, and the switch the logic which selects the particular drive line.

"One" State

Consider a second loop-and-wire arrangement similar to the first (Figure 3). Here, however, a copper slug has been placed in the path of $\emptyset_2$. At the time the switch is closed and the field is building up, Eddy currents and other losses (depending upon the configuration) are induced in the slug. The effect of the losses is to reduce B_2 such that:

$$B'_2 = B_2 - \triangle B_2$$

If the new value B'_2 is substituted in equation 2, it becomes evident that a net emf proportional to $\triangle B_2$ will be induced in the loop and a signal will appear at the output transformer. A similar signal but opposite in sign will appear when the switch is opened and the field collapses. This is the condition for a Unifluxor in the "one" state.

Design Considerations

The amplitude of the "one" signal depends upon:

1. The rate of change of the current in the drive line;
2. The final amplitude of the current;
3. The spacing between the two legs of the sense line loop, and
4. The amount of flux imbalance introduced by the copper slug.

Each of these parameters influences the design of an operating memory. General considerations are discussed below, followed by values of the parameters chosen for a working model of the Unifluxor memory.

The need for a rapid build-up of a current limits the length of the drive line, since as the line is made longer its inductance becomes larger and hence objectionable. The desired cycle time also influences to a smaller degree the maximum allowable length.

The final amplitude of the current determines the effective boundary of the flux of the drive line, that is, the point at which $B(i_{max})$ is essentially zero. The drive lines adjacent to the selected line must fall outside this boundary; otherwise noise will occur whenever the drive line on one side of the selected line has a slug and the line on the other side does not. The amplitude of the current thus determines the minimum spacing between drive lines.

The position of the effective boundary of the drive line field is one factor in determining the flux area; the second factor is the spacing between the legs of the sense line loop. In order to obtain an area of sufficient size and consequently a "one" signal of reasonable strength, a compromise must be reached between current amplitude and the spacing of the loop. For a very small current, if the drive lines are close together, the sense line legs must be widely spaced, thereby keeping the over-all size of the array relatively constant.

The amount of flux imbalance caused by the slug is determined by its position, size, and shape. Theoretically, an open slug which completely cancelled the flux in area A_2 would cause a maximum $\triangle B_2$, the decrease resulting not only from Eddy current losses but also from the field of the counter-emf that would be induced in the loop. In practice, a solid rectangular slug is sufficient, yielding a strong "one" signal with excellent signal-to-noise ratio.

Working Model

Physical Arrangement

A working model of the Unifluxor memory has been constructed for experimental purposes. The memory has a capacity of 64 words of 50 bits each. The 64 words are contained on two arrays of 32 words each. The drive lines of the two arrays are shared in common; the sense lines are independent. The configuration of the memory element used in the model is shown in Figure 4, which is an enlarged diagram of several bits.

Note that the sense line crosses the drive line at an angle. The adjacent drive lines are crossed at the same angle but in the opposite direction. Skewing the sense lines in this way serves two purposes: It permits the slug to be so placed as to interrupt B_2 in two areas, thus achieving more imbalance, and in addition it tends to cancel any noise originating from nonselected drive lines.

To read any given word requires full current on the selected drive line and no current in the other drive lines. The selection must be near perfect, else each leakage current will cause small outputs from the "one" bits through which it passes. This noise, multiplied by the number of drive lines linked by the same sense line, can reduce the signal-to-noise ratio and mask the selected bit.

One way of obtaining perfect selection is to use a separate driver for each bit. The economic obstacle this method imposes, however, obviates its use in a large memory. For the laboratory model a switch-core selection system was chosen which involves four drivers, eight gates, and 32 switch cores. Each switch core selects two drive lines, one on each of the independent arrays. The outputs from the two corresponding sense lines appear at gates at the input to the sense amplifier. The gates permit the output from only one sense line to pass at any one time.

Two different sense amplifiers have been designed. One amplifier produces an output whenever any negative-going pulse appears at its input. No output is produced for positive-going inputs. The second amplifier produces an output only if the leading pulse is negative, with no output under other conditions. Both amplifiers have extremely fast recovery times.

A full discussion of the address logic is omitted from this paper, since the logic is similar to many other addressing schemes for word-organized memories. However, a novel method of selecting the drive cores has been devised, which reduces the noise level far below that usually experienced.

The principle is illustrated in Figure 5. In quiescent state, the address translator holds the driver and gates cut off. (The full schematic of only one gate is shown, the circuits of the others being identical.) No current flows through the core winding associated with gate 1 because of the blocking action of the diode. So also for the other two core windings: No current flows.

Assume that the driver and gate 1 are now simultaneously enabled, as indicated by the small waveforms on the diagram. A low-resistance path from ground is created through Q1, through the winding of the left-hand core, through Q2 to the positive side of the supply. The magnitude of the current through the winding is sufficient to select the core, and an output pulse appears on drive line 1. No current flows in the nonselected core windings because gates 2 and 3 remain closed.

Although it would appear possible theoretically to eliminate the drive core and let Q1 drive the drive lines directly, in practice this causes ringing on the drive line.

Electrical Characteristics

A 500-milliampere pulse 150 millimicroseconds wide is used as the drive current. The flux created by the current at its maximum has a density of approximately one gauss at a distance one millimeter from the drive wire, as calculated from the equation:

$$B_{(gauss)} = (\mu \; I \; 10^4/2\pi R)\theta$$

This equation considers the drive line as infinitely long, an assumption justified by the ratio of its actual length to the diameter of the of the effective field. The value of μ may be taken as that for free space ($4\pi 10^{-7}$). θ is a unit vector normal to the drive line.

The net emf induced in the sense line by a "one" bit has a range of 8 to 12 millivolts. The corresponding output from the sense amplifier is a rise from -4 volts to 0 volts. Figure 6 shows the sense amplifier outputs for both "one" and "zero". The signal-to-noise ratio is greater than 15 to 1.

At present, the memory is being operated at interrogation intervals of 420 millimicroseconds. An interval of 350 millimicroseconds appears possible with the present design. The maximum possible interrogation rate of the memory element itself is not known, the recovery time of the sense amplifiers and the speed of the address logic being the limiting factors in the present design. No heating or other adverse effects have been noted from repeated interrogation of the same drive line.

Fabrication

The drive lines are etched on an epoxy glass substrate and made flush with the surface by heating the substrate to 375°F and placing it in a 50-ton press. A Mylar film with a copper sheet bonded to it is cemented in place on the substrate, and the sense lines are etched from the copper sheet.

The copper slugs are etched on a second Mylar film, with a slug in each bit position. When the contents of the memory are determined, the slugs in the "zero" positions are punched out and the film cemented to the sense and drive line array.

(For customer use, thin stiff cards appear more desirable as a base for the slugs. A mechanical arrangement which holds the cards firmly in position against the control lines without cement has been developed, and the cards may easily be exchanged. A device to prepare cards with the desired slug arrangement semiautomatically is now being investigated. Using standard punched paper tape as input, the device activates a bank of lamps, exposing photo-resist material on the previously sensitized card. The card is ready for use after development and cleaning.)

None of the fabrication processes are critical or expensive, nor do any require highly skilled operators. In the model built for use in the laboratory, a density of approximately 200 bits per cubic inch was achieved with little effort. The array may further be scaled down by decreasing the distance between drive lines and the width and separation of the sense line loops. (In the laboratory model, these dimensions were made conservatively large.)

Conclusions

The Unifluxor is particularly suited for instruction storage in such fixed-program machines as process control and missile guidance computers. The speed of the element is sufficiently high as to adapt it for use in next-generation machines. Although life tests to prove its reliability have not yet been performed, the element itself does not appear to have any characteristic that would change either with time or operation, and hence the reliability of the memory control circuits would be the governing factor.

Another potential application now being investigated is dictionary storage for machine translation. The low cost of the Unifluxor may make possible random-access memories of a size hitherto unfeasible economically. If the memory element itself was the only aspect of this problem, the Unifluxor would solve it; the cost of the control circuits, however, remains and must first be reduced.

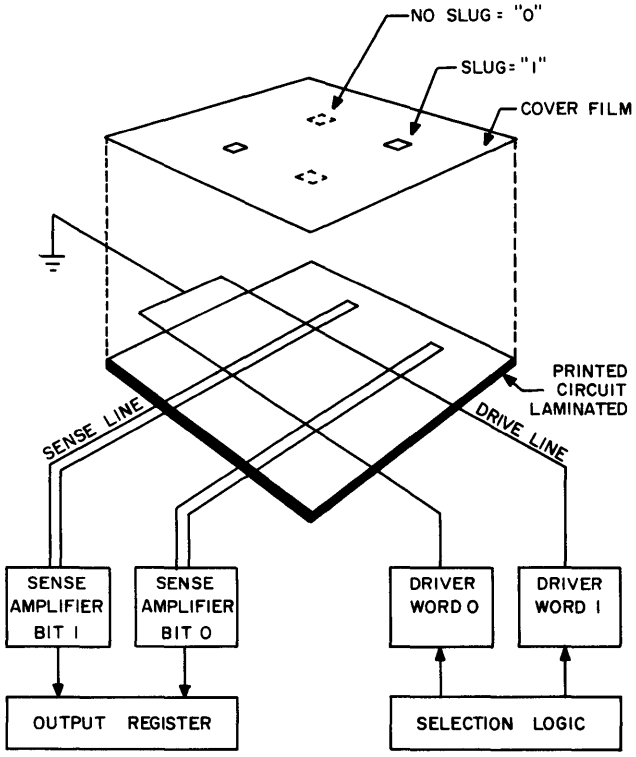


Figure 1. Simplified Diagram of Unifluxor Memory

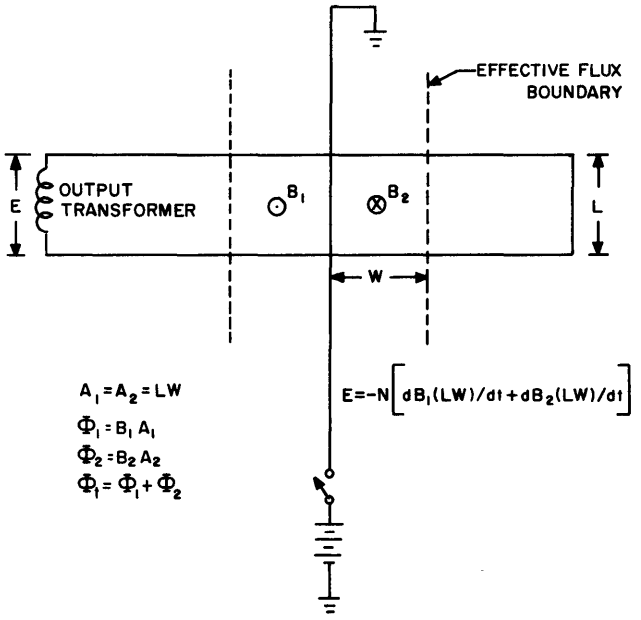


Figure 2. "Zero" Configuration

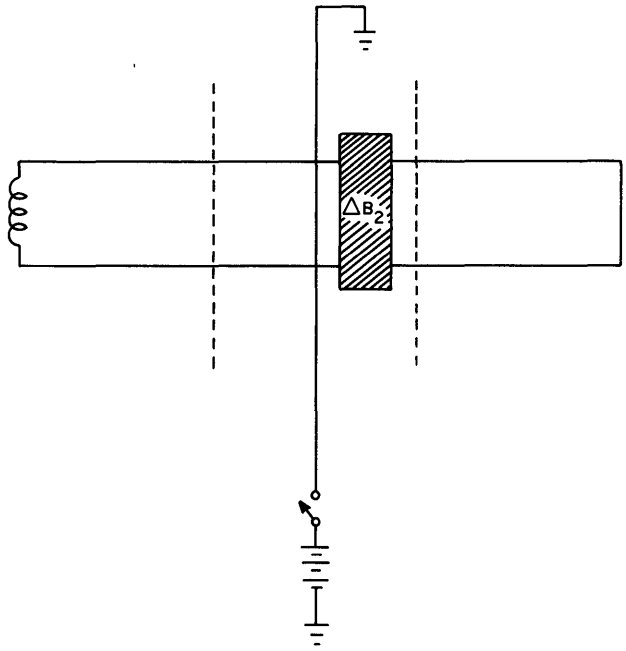


Figure 3. "One" Configuration

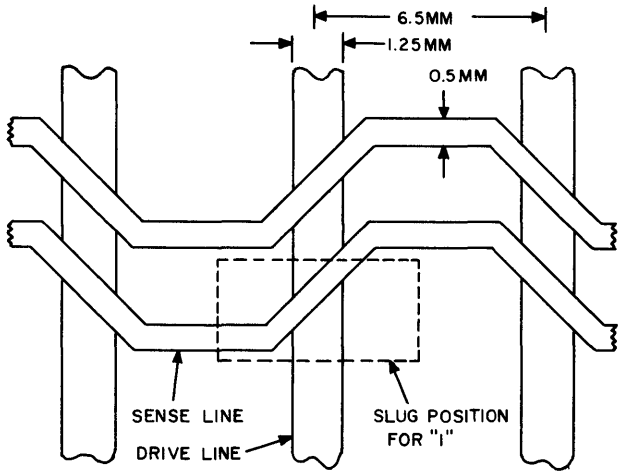


Figure 4. Enlarged Section of Unifluxor Array

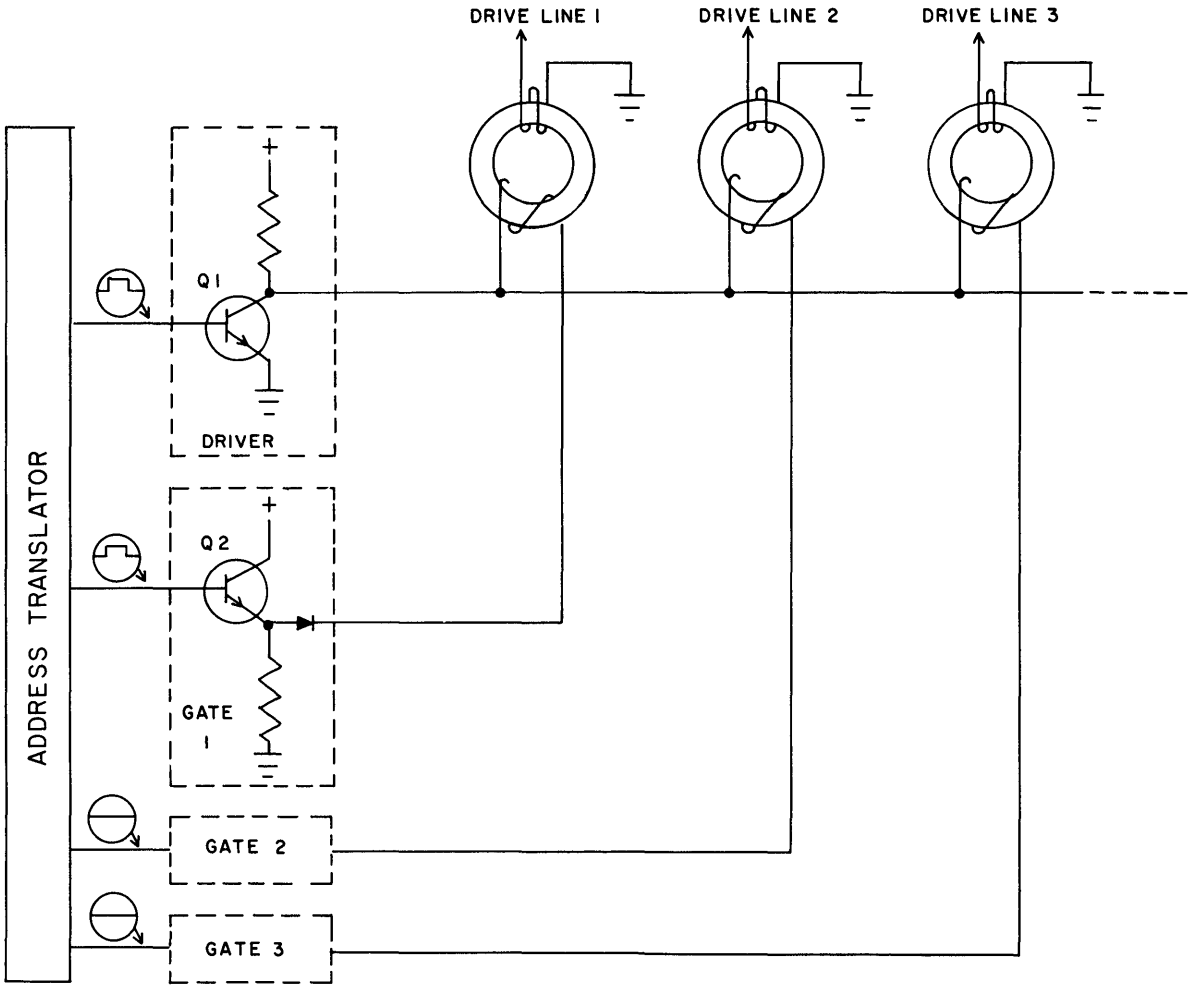
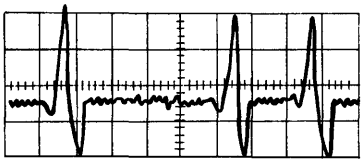


Figure 5. Core Selection



PATTERN 1011 IS THE RESULT OF INTERROGATING FOUR SEPARATE DRIVE LINES IN SUCCESSION.

Figure 6. Sense Amplifier Output

CHARACTERISTICS OF A MULTIPLE MAGNETIC PLANE THIN FILM MEMORY DEVICE

By

Kent D. Broadbent
Hughes Research Laboratories
Culver City, California

Sei Shohara and George Wolfe, Jr.
Hughes Ground Systems Group
Fullerton, California

Summary

In a previous paper,[1] a ferromagnetic film memory was described having a complex structure of evaporated magnetic, dielectric and conducting materials. The multiple plane geometry was designed to enhance the characteristics which are desirable in practical digital computer memories without requiring extreme uniformity and control of the magnetic properties.

This paper describes the electrical characteristics of prototype memory elements, and discusses their application in computer memory systems. A short review of the basic magnetic structure and mode of operation is given first to provide a common reference for discussion.

Magnetic Structure and Mode of Operation

The structure to accomplish the selection and storage functions within the matrix is comprised of four ferromagnetic films appropriately interlaced with evaporated drive and read-out conductors, and insulators. All nineteen layers (Figure 1) are vacuum deposited. Figure 2 represents schematically the reference (binary 1) state of the four magnetic layers. The dimensional ratios are shown highly exaggerated for clarity.

The four-layer magnetic complex is in a high or low magnetostatic energy state depending on the sense of M vectors within the films. The low energy, statically stable states are those in which two M vectors point in a direction opposite to the remaining two. Any other configuration results in an unstable high energy state which quickly decays to a stable condition. The difference in film widths causes this decay to proceed by a path involving least expenditure of energy. Due to the difference in the volumes of magnetic material, this minimum-energy transition occurs by reversal of the uppermost film available for reversal. This is illustrated in Figure 8. In going from the "1_r" state to either of the "1_d" states, the uppermost film available for reversal is the third layer. The fourth (read-out) film thus remains in the original direction and no voltage is induced in the sense conductors. It is seen that any subsequent pulses applied to either X or Y merely affects the first two layers; only coincident drives on X and Y can establish either the "0_r" or "1_r" states.

Electrical Characteristics

Of primary interest in memory applications are the switching characteristics and the selection ratio of the basic memory cell. Also, in the case of evaporated memories, RLC parameters are of added importance because of the extremely small dimensions attainable. These are now given for the prototype elements.

Resistances of the conductors of prototype elements are centered around two ohms per element. Inductances are in the millimicrohenry range, and capacitances between conductors range from 80 up to 200 micromicrofarads per element. The magnitudes of resistance and capacitance pose some problems in designing an optimum practical array, as discussed later.

To obtain the switching characteristics, pulse sources of variable amplitude and polarity are applied to the X and Y drive conductors connected in series, as shown in Figure 3. The peak output voltage and switching time (measured between 10 percent points) is plotted against current pulse amplitude in Figure 4. The "disturbed 0" outputs normally plotted on graphs of this type are not shown here since it could not be measured with the existing equipment. This is discussed further under selection ratio tests. The rise time of the driving pulse is 20 mμs throughout the entire range of currents. The leveling-off effect of switching time at high currents can be ascribed partly to the finite current rise time. This was checked by using a mercury relay pulser with 1 mμs rise time and a traveling wave scope. At a maximum current of 3 amperes, the limit of the pulser, the switching time observed was 10 mμs shorter than shown in the plot of Figure 4. The switching times below 1.2 amperes are identical with the values of Figure 4, showing that switching time is amplitude limited below 1.2 amperes for 20 mμs current rise time. The relay pulser was not used to plot Figure 4 for the reason that a high-current transistor driver capable of rise times below 20 mμs do not appear practical at this time. On the other hand, rise times significantly greater than 20 mμs would begin to hinder the inherent speed capacity of the device. The figure of 20 mμs was thus chosen as the best compromise.

In the low current region, switching can be

detected at currents as low as 160 ma. if sufficient pulse width (0.6 μs) is used.

To determine the selection ratio, the circuit of Figure 3 is slightly modified so that the X and Y windings can be driven separately by adding another pair of pulse generators. The X winding is driven by 3 amperes (limit of equipment) rising in 20 mμs in the READ direction and 500 ma. in the WRITE (reset) direction. The Y winding is driven with 500 ma. in both directions. The photograph in Figure 5 shows superimposed traces of the output voltages, all during READ time, under three different conditions:

1. Only X is driven in the READ direction followed by a coincident WRITE drive.
2. WRITE drives are removed.
3. Both X and Y are driven coincidently in the READ direction followed by coincident WRITE drives.

The output voltage resulting from the third condition represents a full select "1" and is shown for reference. The second output is the "zero" signal reference since no writing occurs prior to READ, and the output for condition 1 shows the effect of a 3-ampere half-select pulse. The difference in output voltages between conditions 1 and 2 is due to the presence of Y drive during condition 2 but not during condition 1. The ringing is due to the combined effect of element capacities and the access leads from the prototype measuring jig setup. It is seen that the memory element is not disturbed by half-select currents which are greater than an order of magnitude times the current amplitude sufficient to initiate switching (160 ma.) when applied coincidently. Exactly the same results are obtained when the Y winding is half-selected although its associated magnetic plane occupies a position closer to the output magnetic film.

Besides the switching characteristics and selection ratio, another parameter of interest, especially in fast memory systems, is the delay of the output signal in propagating over the sense conductors. The problem is accentuated in the thin film element because the extremely small dimensions result in larger values of capacity (as well as resistance) than are encountered in ordinary core systems. The effect of these were measured as shown in Figure 6 by introducing a 5 millimicrosecond pulse in a sense winding passing through several elements. Two traces are photographically superimposed on a common baseline, one with only helix A connected and the other with only helix B connected. The time displacement between the two waveforms constitute the total delay. Any possibility of inequality in electrical line lengths is cancelled out by repeating the measurements with A and B helices interchanged. The average delay per element was observed to be about 0.3 mμs. This is somewhat larger than is desirable for a memory of large size, so that the elements currently under development have reduced capacitance, with some sacrifice in output voltage amplitude, resulting in reduction of delay by an order of magnitude.

Memory Organization

The individual memory cells and interconnecting conductors of prototype matrices are evaporated in a manner such that the operation of the memory is similar to the conventional coincident current memory plane. That is, the coincidence of a X and Y drive current select a memory cell. One important difference is that the X and Y drive current amplitude can exceed the switching threshold by a factor of 20 or more without affecting the information state of a memory cell. This is because of the virtual magnetic decoupling between the X drive layer or Y drive layer from the sense layer when either the X or the Y drive line are selected individually. The switching speed of the selected memory cell is then a function of the drive current. Switching time in the order of 40 to 80 millimicroseconds are obtained with relatively low amplitude drive currents. We now consider some of the factors affecting the design of arrays.

Figure 7 illustrates a possible arrangement of memory elements and planes. Sense lines are not shown, for clarity. Assume that suitable bipolar drivers are tied to the X, Y access lines, providing READ and WRITE current pulses. It can now be shown, with the aid of Figure 8, that any sequence of non-coincident drive currents cannot alter the stored information although the M-vector configuration can enter different statically-stable magnetic states. In Figure 8, the 1_r and 0_r states are taken as the initial states, and all possible configurations arising from subsequent non-coincident drives are shown for both cases. It is first noted that it is not possible to go from the 1_r state to the 0_r state or vice versa with any sequence of disturb pulses. Second, once a 1_d or 0_d state is entered, it is not possible to return to the reference states. In all cases the fourth layer is magnetically decoupled from the driven layers. Further, once the disturbed states are entered, the two bottom layers are not affected by further disturb pulses. This "decoupling" feature implies a somewhat more subtle writing technique than core memories using inhibit currents for digit control.

Memory organizations based on biasing a winding or adding (or modifying) a conductor (to obtain economy in the driver count) are presently under evaluation. One important consideration in this evaluation is developing an optimum method of interconnecting the elements to obtain cancellation of capacitively coupled noise, analogous to the use of diagonal sense windings in core arrays to cancel inductively induced noise. Another consideration is the effect of the relatively high resistivity of the conductors. If IR drops are traced in Figure 7, it can be seen that care must be exercised to insure that the voltage drops in

access lines do not cause excessive voltage gradients across the extremely thin insulating layers.

In applications involving large numbers of cells a reduced cell length and reduced magnetic film thickness is used. One of the virtues of the cell is that, where desirable, thicker films and longer lengths may be employed to achieve substantial outputs, such as with the .200" cell in obtaining all the previous data. However, in large systems the problems of power dissipation per cell, back e.m.f. per cell, output signal delay per cell, and maximum storage density per surface area dictate the use of a smaller cell. Smaller size is feasible because of the vacuum techniques employed, and the use of a reacting system such as this, where flux may be switched at any location, becomes practical due to the minute amounts of magnetic material that can be employed using vacuum, vapor-phase processes.

Conclusion

The multi-layer memory elements offer considerable promise in application to high speed memory systems. A number of problems remain to be solved in designing peripheral circuitry compatible with the speed capabilities of the device, and in improving certain characteristics of the device itself. These are currently under investigation.

Acknowledgement

The authors wish to acknowledge the contributions of C. T. Alderete who obtained much of the data used in the paper.

Reference

1. "A Vacuum Evaporated Random Access Memory", K. D. Broadbent, Proceedings 1959 Special Technical Conference on Nonlinear Magnetics and Magnetic Amplifiers, September 1959.

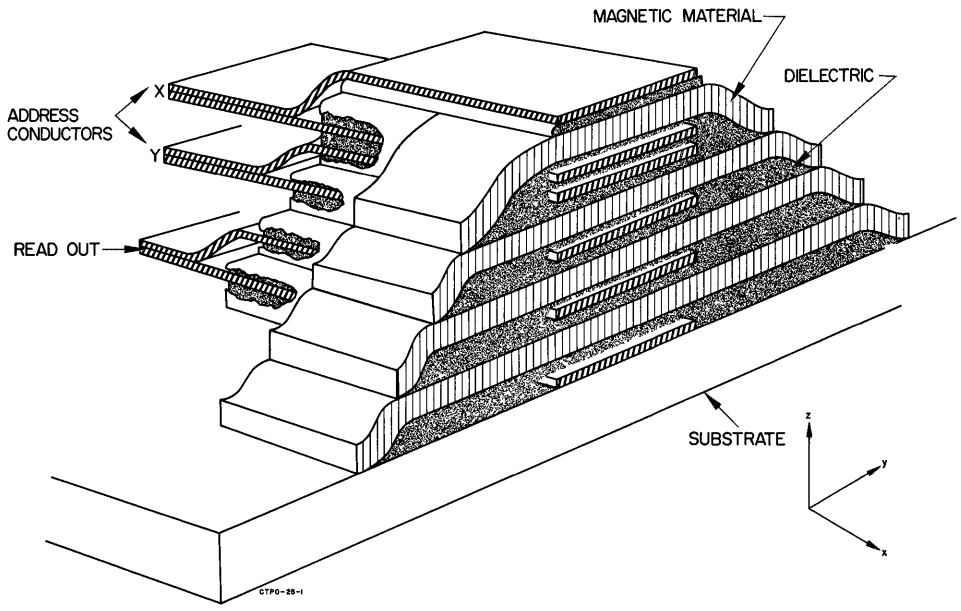


Figure 1. Schematic section of an evaporated memory cell.

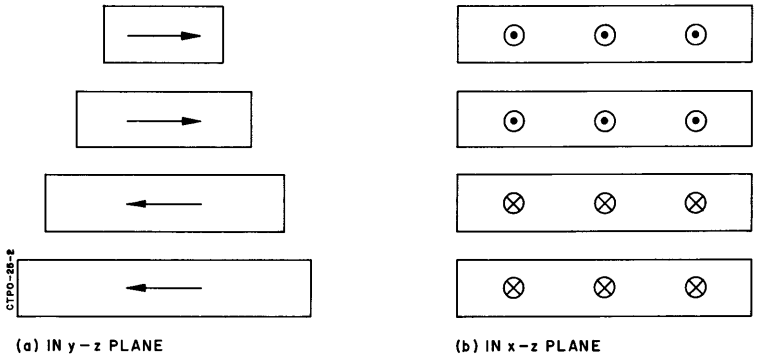


Figure 2. Schematic representation of the four-plane magnetic system.

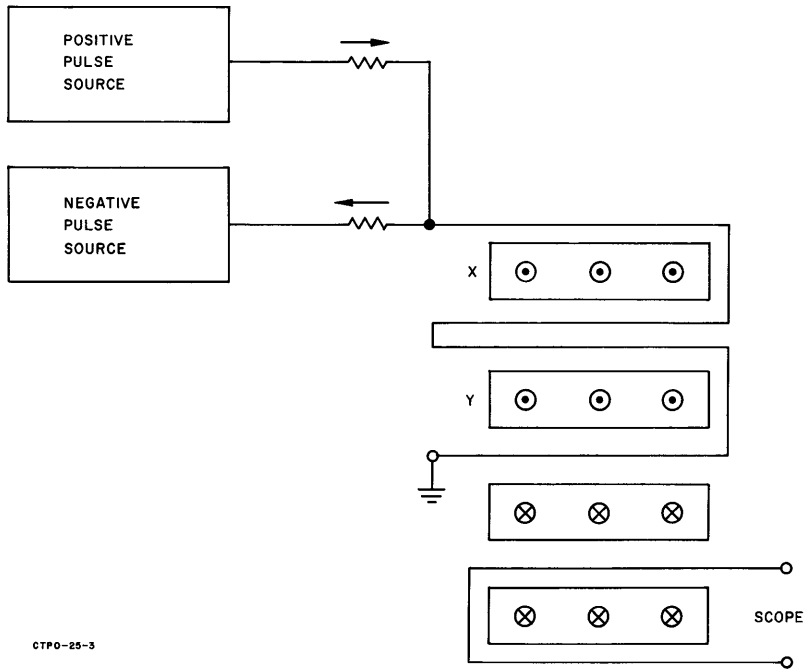


Figure 3. Basic test set-up for obtaining switching characteristics.

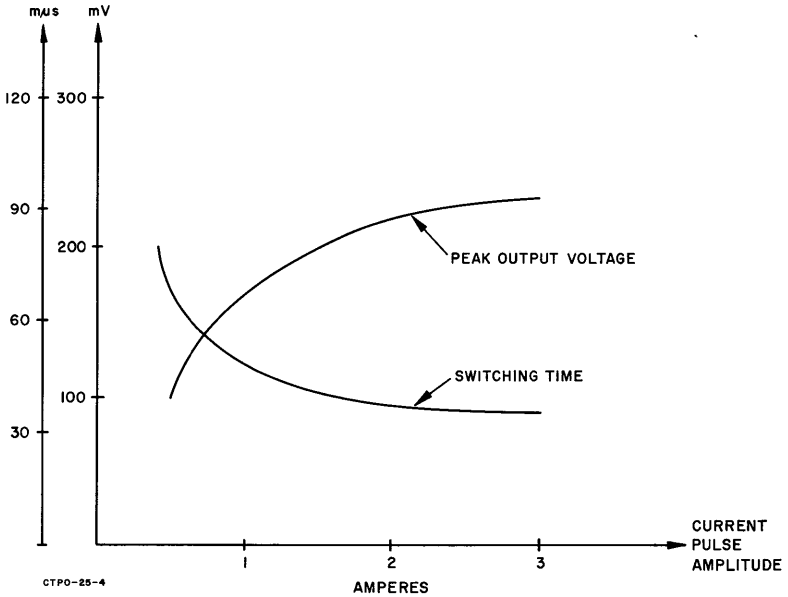


Figure 4. Switching characteristics.

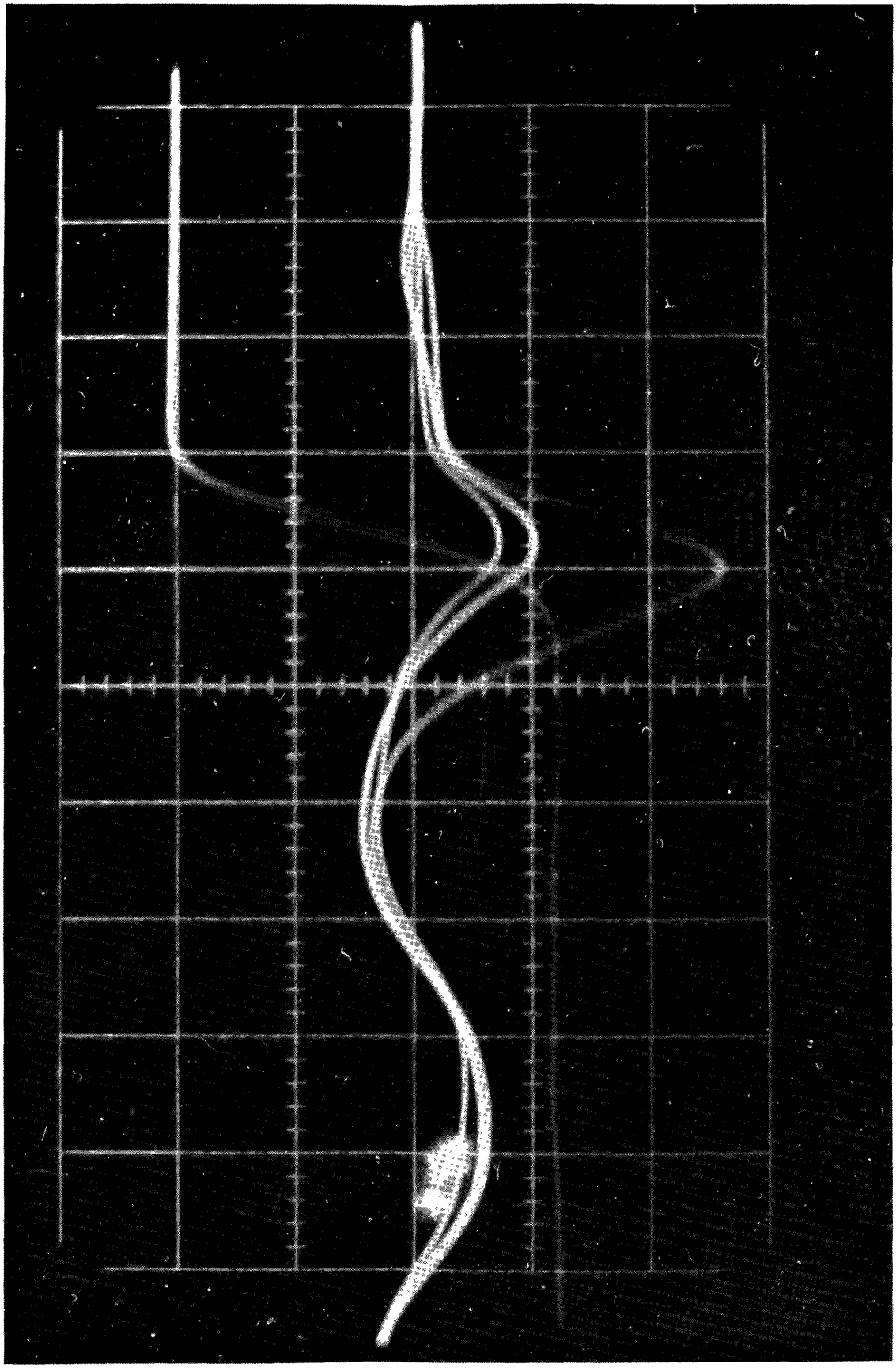


Figure 5. Half-select waveforms (20 mμs/cm right to left, current waveform 1 amp/cm).

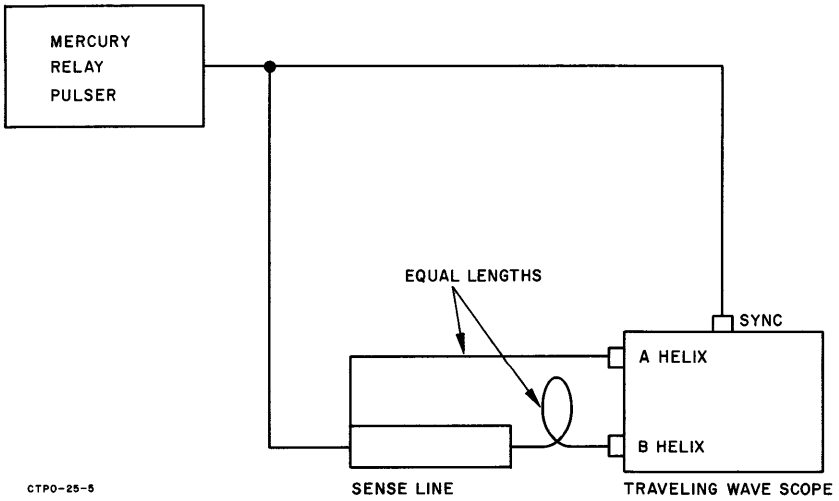


Figure 6. Delay measurements.

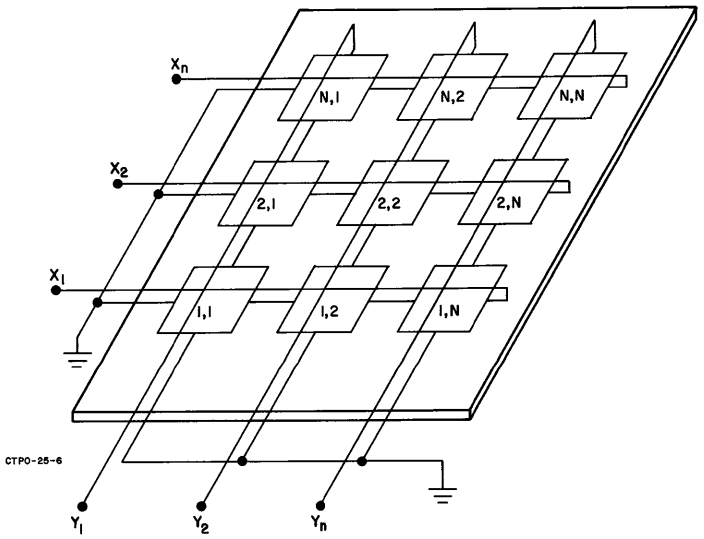


Figure 7.
Simplified representation of a possible memory organization.

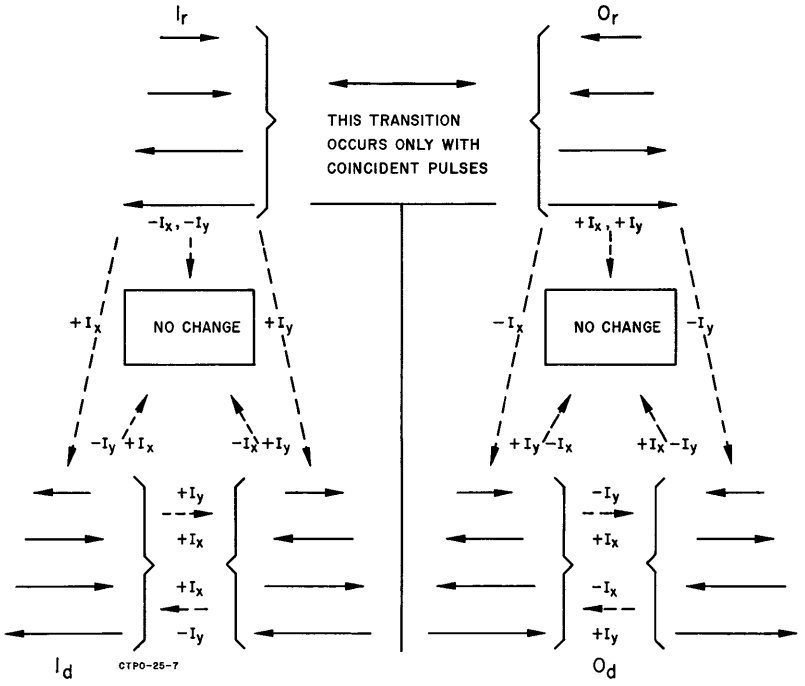


Figure 8. Possible M-vector configurations.

ANALOG TIME DELAY SYSTEM

Charles D. Hofmann
Electronics Engineer
and
Harold L. Pike
Senior Electronics Engineer

Convair (Astronautics) Division
General Dynamics Corporation
San Diego, California

Introduction

The Convair-Astronautics Time Delay System is being developed to make possible the delay of analog functions over a greater range than is possible with more conventional means. The system is designed to handle ten channels of analog data, delay each for the same time interval, and reproduce the functions within 0.1% of the original composition for all frequencies up to 100 cycles per second. The delay is required to be variable from 0.01 to 10 seconds, ±5 milliseconds. This paper is concerned with the system design and will not attempt to discuss applications.

In order to obtain the wide delay range required, a special magnetic tape system is employed. The delay is achieved by recording information on the tape with a recording head, and reading the information back at the prescribed time with a separate head. The tape travels at a uniform rate in a continuous loop. The delay is therefore directly related to the length of tape between the write and read heads. This is controlled by a servo system which continuously monitors this "delay" loop and compares it with a value preset directly in milliseconds. The high degree of accuracy is obtained by converting the analog input voltages to digital form and recording it. A single analog-to digital converter is used for this purpose with the various analog inputs being commutated to the converter. The read head retrieves the digital information from the tape after the desired delay. The information is then re-commutated to ten digital-to-analog converters. The use of separate digital-to-analog converters for each input-output channel eliminates the need for holding amplifiers and commutation of the analog output voltages. Correlation between input and output commutation and delay control is achieved through a central timing system. The overall flow of information is shown in Figure 1.

Tape Transport

The tape transport is designed to use a 100 foot continuous loop, one inch magnetic tape. Information is recorded non-return to zero on the tape by a 14 channel recording head designed to handle a forty kilocycle repetition rate as the tape moves at 100 inches per second. At the prescribed time later, this information is retrieved by the read head as the tape continues in its loop path. Since a delay of from 0.01 to 10 seconds is required, the amount of tape between the recording and the reading heads must be variable from one to one thousand inches. This tape is loosely piled in the tape delay storage compartment. The tape is pushed over the recording head where air pressure assures contact with the head. This is necessary since, to achieve the minimum one inch loop, it is impractical to place the drive capstan between the recording and reading heads. If the delay loop exceeds four inches, the tape is picked up below the recording head by a vacuum capstan which helps maintain tension in the tape across the recording head. With a delay loop of less than four inches, the tape pulls free of the vacuum capstan and its own centrifugal force supplies the required tension. The tape, after passing the read head, moves through a compartment similiar to the delay storage compartment where all tape not used in the delay loop is stored. Vacuum is used to maintain tension in the tape wherever it is necessary. The principle features of the tape unit are shown in Figure 2.

Timing

As shown in Figure 3, the heart of the timing system is a 960 kc crystal clock which generates the fundamental frequency used by the digitizer for bit selection. This frequency is divided by twelve to obtain an 80 kc rate used by the delay servo counter. Dividing again by two gives the fundamental 40 kc sampling rate. A ten bit shift register uses this rate for input and output commutation and channel identification.

Delay Servo

The tape transport uses two three-phase synchronous motors, one to drive the write and vacuum capstans and the other to drive the read capstan. The first motor is driven directly from the 60 cycle three phase line while the read motor is driven by a variable frequency (three-phase) oscillator. The frequency of this oscillator is controlled by a voltage level corresponding to the output of a comparator. The recording head records the 40 kc clock on the tape and, at the same time, adds it into a counter. The read head subtracts this 40 kc signal out of the same counter so that the balance in the counter represents the difference between the counts recorded and the counts read. This balance is proportional to the delay in the system and and is continuously compared against a preset count representing the delay required, set directly in milliseconds. The output of the comparator then regulates the speed of the read motor as discussed. This system is shown in Figure 4.

Analog-To-Digital Conversion

The analog-to-digital converter samples the input analog voltage and converts it to digital form using ten bits for amplitude representation plus one bit to indicate polarity (sign). To obtain the required accuracy, each channel of input information is sampled at a repetition rate of four thousand samples per second. Since a single digitizer is used for all ten channels of analog inputs, the overall sampling rate of the digitizer is forty thousand samples per second. A ten bit shift register driven by the forty kilocycle clock is used to activate diode bridge switches for sequential input commutation. The digitizer consists of eleven flip-flop operated bit switches which contribute current through calibrated resistors to a summing junction to which the input signal is also connected. The polarity of this summing junction is the normal carrier signal input to a "ring modulator". The signal input of the "ring modulator" is the 960 kc clock frequency. The output is a 960 kc signal which leads or lags the clock by 90 degrees depending on the polarity of the summing junction. By comparing the phase of the clock with the modulator output, the summing junction's polarity is determined.

The common "ripple-down" method of digitizing is used. Starting with the most significant bit, the bit switches are turned on sequentially by an eleven bit shift register driven by the 960 kilocycle clock. Each bit switch activates a zener diode regulated, negative voltage source which is summed through a calibrated resister to the summing junction. The current contributed by each bit is proportional to the significance of the bit. If the summing junction goes negative, the bit reset gates are activated so that when the next bit switch is turned on, the preceeding switch is turned off. In the event the input analog voltage is negative, the sign bit is activated which contributes a positive current to the summing junction of a magnitude equal to that contributed by a maximum input voltage. The effect of the negative input voltage and the sign bit is a current equal to a positive input which is the complement of the absolute value of the input signal. As a result, the digitizer will always handle a positive input voltage. When all ten bits have been sampled a "transfer" pulse is generated by the eleventh bit of the shift register. This pulse is "and" gated with each bit switch through eleven transfer gates so that each transfer gate corresponding to each "on" bit switch will pass the pulse. The output of each transfer gate is coupled to a symmetrically driven flip-flop which serves as a write head driver amplifier. The eleven channels of digital information are thus recorded in parallel on eleven of the fourteen channels of the tape. A change in direction of saturation of the tape corresponds to a "one" being recorded, while no change represents a "zero". Figure 5 is a block diagram of this system. Figure 6 is a time versus sequential operation chart of the system.

Digital-To-Analog Conversion

This section, shown in Figure 7, includes fourteen read amplifiers, eleven of which receive the digital information from the tape; ten groups of eleven output commutation gates; a ten bit shift register; and ten digital-to-analog converters each consisting of eleven bit switches, gates, and a summing amplifier with a low pass filter. The outputs from

the eleven read amplifiers (pulse for a "one", no pulse for a "zero") are gated to the proper output switches by the 10 bit shift register. This shift register is started in the output channel No. 1 position by a pulse from the tape which indicates that input channel No. 1 is being read, and is then advanced by the recorded forty kilocycle clock as the other input channels are read. The output bit switches are flip-flops which control diode bridge output gates. These gates apply a standard reference voltage through a calibrated resistance to the summing junction of the output amplifier. The summation of all the contributions from the on gates reproduces the analog equivalent of the digital information received from the tape.

Additional Features

The limits on the delay of 0.01 to 10 seconds have effectively been expanded by use of a "recirculate" provision, making it possible to transfer information from any output channel back to any input channel without cumulative errors. This is done by making available at a patch panel all bit control flip-flop outputs which may be gated directly back to the write head driver amplifiers so that they are re-recorded without any conversions in place of the usual input information. This enables a single channel of continuous information to be delayed up to 100 seconds by recirculating the information through all ten channels successively. Many variations are possible which make the recirculate feature a versatile instrument. This system is completely transistorized. Printed circuit boards are mounted in slide-out racks making it possible to check any circuit during operation. Major system check points have been brought out to a test jack panel enabling checkout with a minimum of effort.

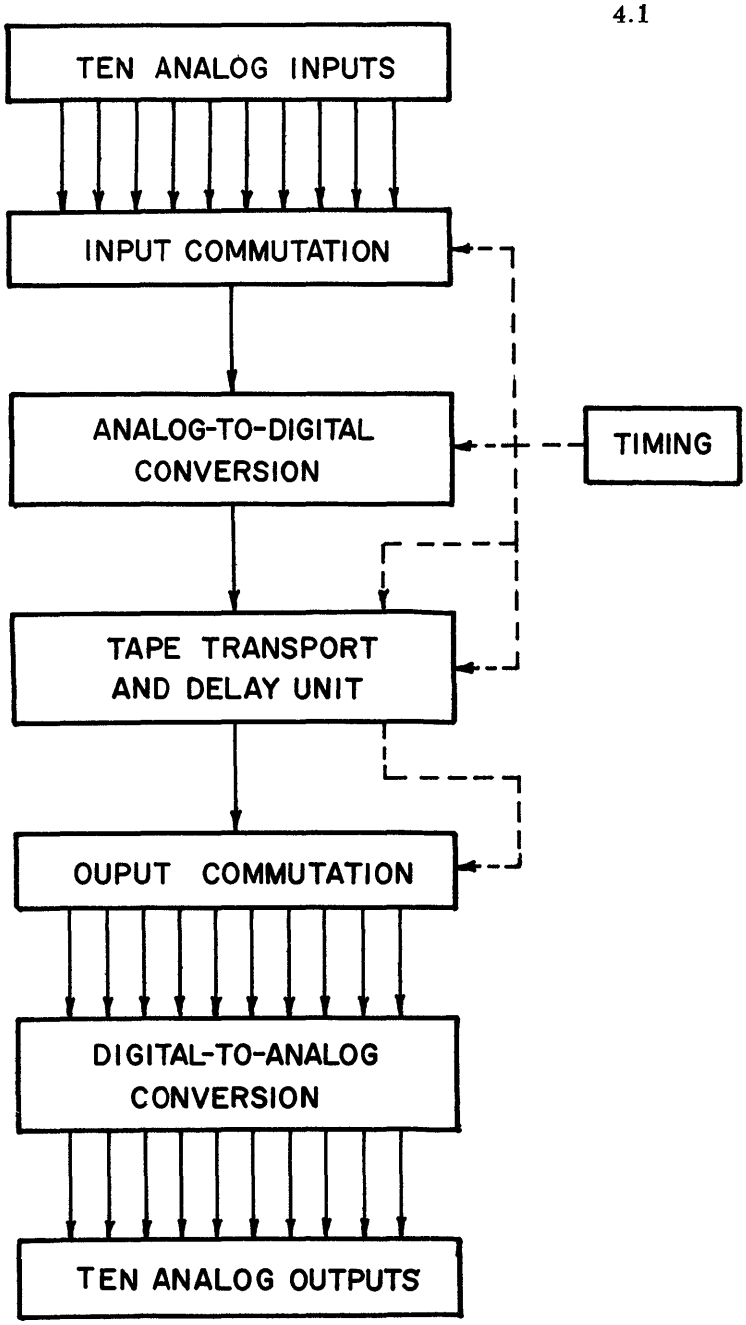


FIGURE I
INFORMATION FLOW DIAGRAM

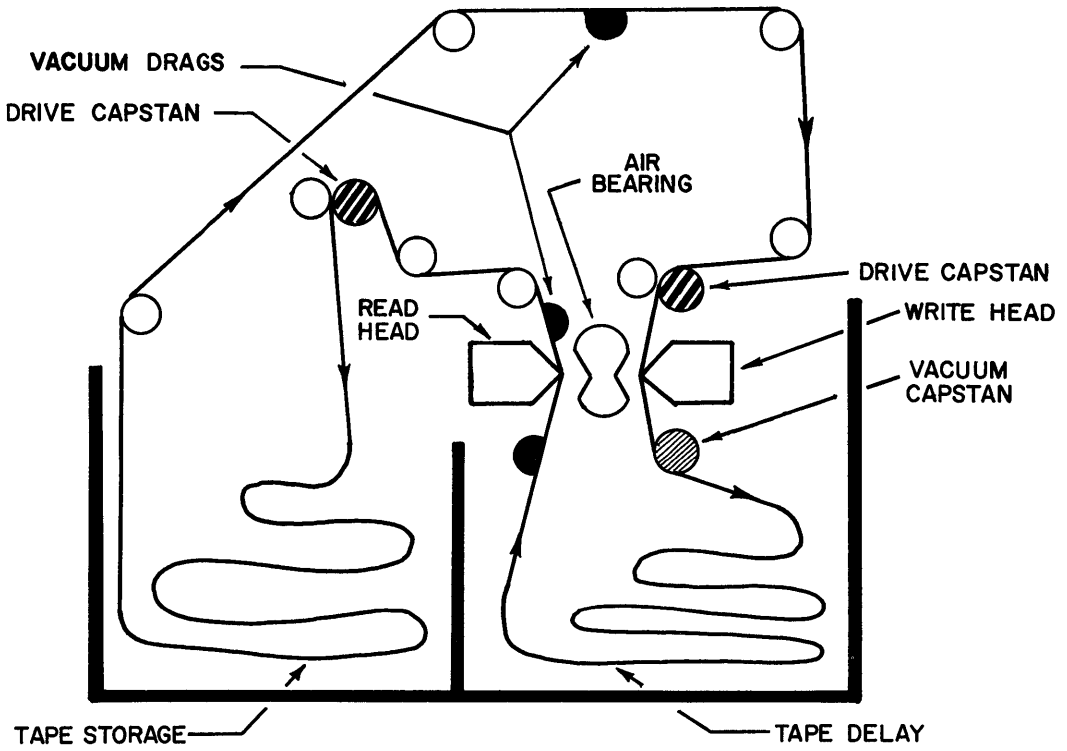


FIGURE 2
TAPE TRANSPORT

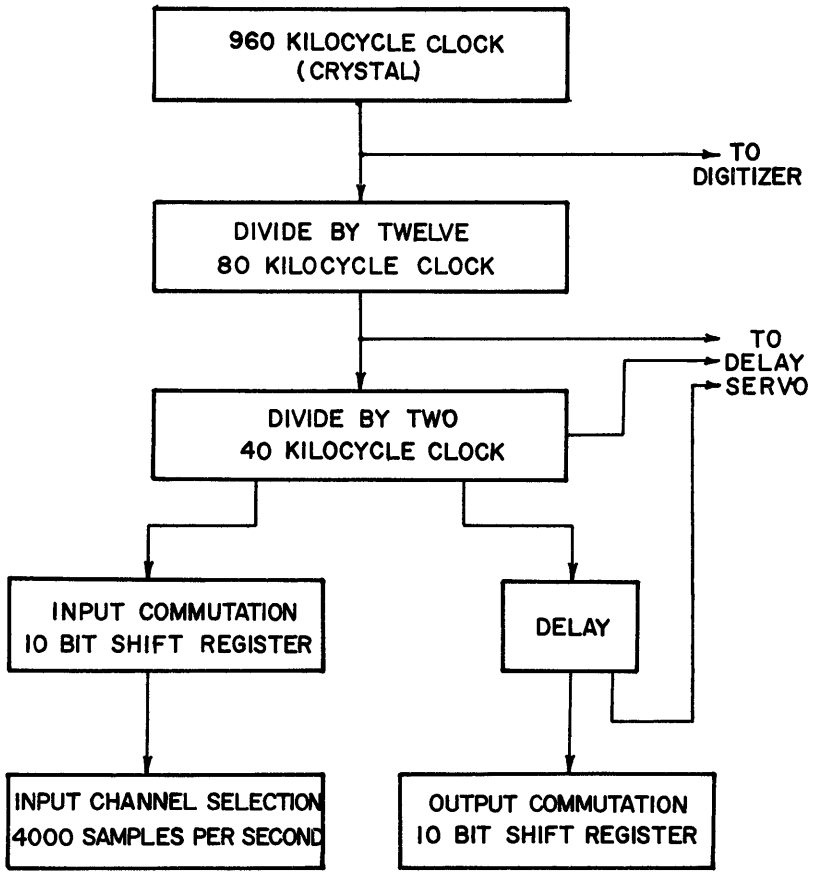


FIGURE 3
CENTRAL TIMING SYSTEM

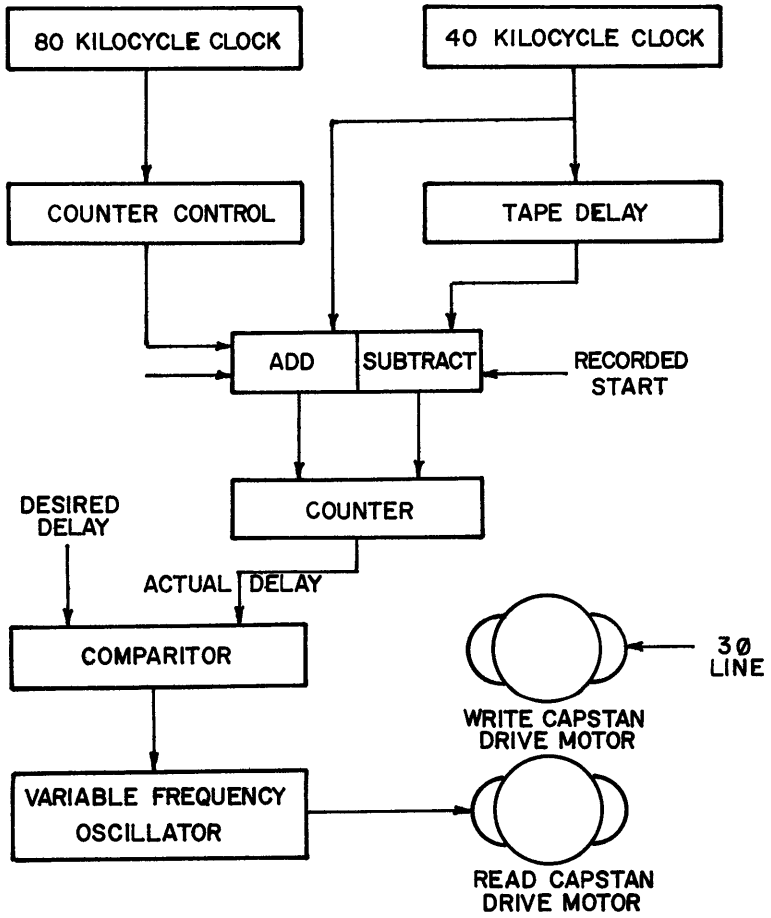


Figure 4
DELAY SERVO

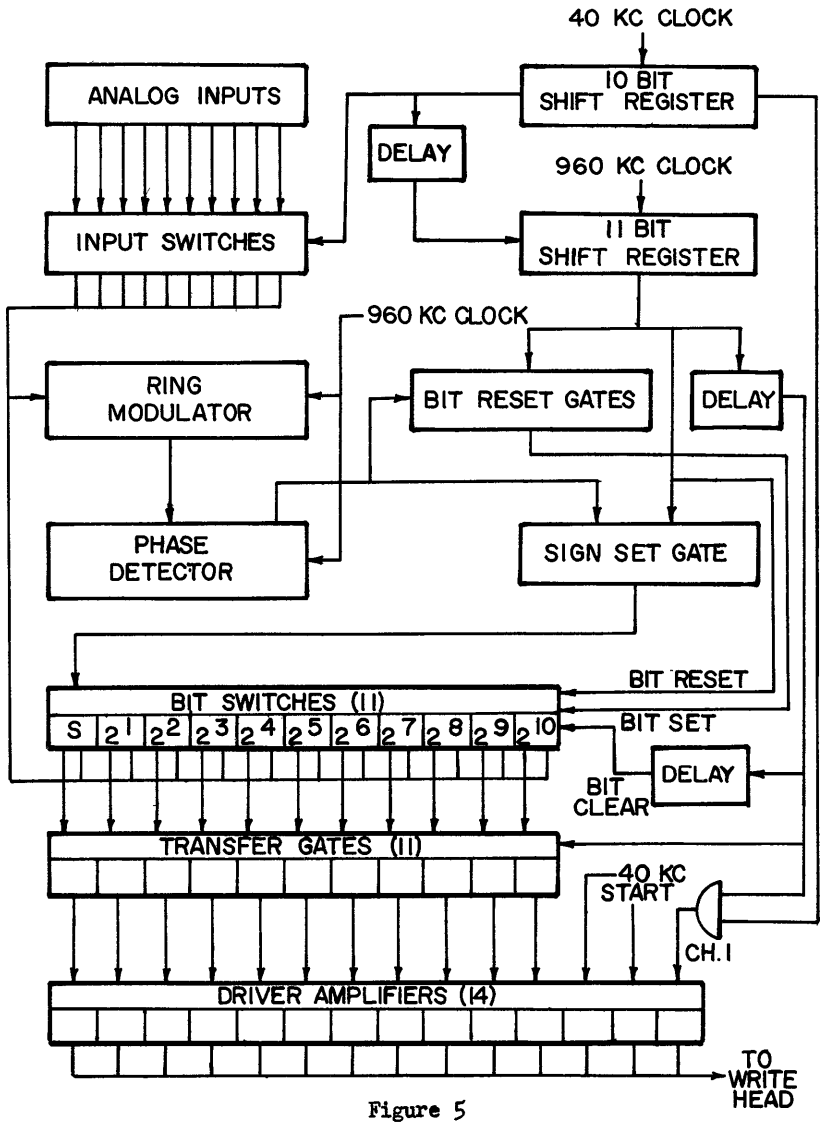


Figure 5
ANALOG TO DIGITAL CONVERSION

CONTROL	TIME μSEC	OPERATION
INPUT COMMUTATION SHIFT REGISTER STEPS ON FORTY KILOCYCLE CLOCK	0	TURN OFF INPUT SWITCH FOR CHANNEL 10 TURN ON INPUT SWITCH FOR CHANNEL 1
	2	SWITCHING TIME FOR INPUT COMMUTATION SWITCH.
10 MICROSECOND DELAY OF FORTY KILOCYCLE CLOCK.	4	
	6	
	8	
DELAYED FORTY KILOCYCLE CLOCK STARTS ELEVEN BIT SHIFT REGISTER- POSITION 1	10	IF SUMMING JUNCTION NEG. TURN SIGN BIT ON. TURN ON BIT 1 IF SUM. JUNC. NEG. TURN OFF
2	12	2 BIT 1
3		3 2
4	14	4 3
5		5 4
6	16	6 5
7		7 6
8	18	8 7
9		9 8
10	20	10 9
11		10
DELAYED ELEVENTH PULSE	22	OPEN TRANSFER GATES TURN OFF THE ON BITS TRANSFERRING
	24	ALL ONES TO DRIVER AMPLIFIERS.
	25	

FIGURE 6
TIME-OPERATIONAL SEQUENCE CHART FOR
DIGITIZING CHANNEL 1

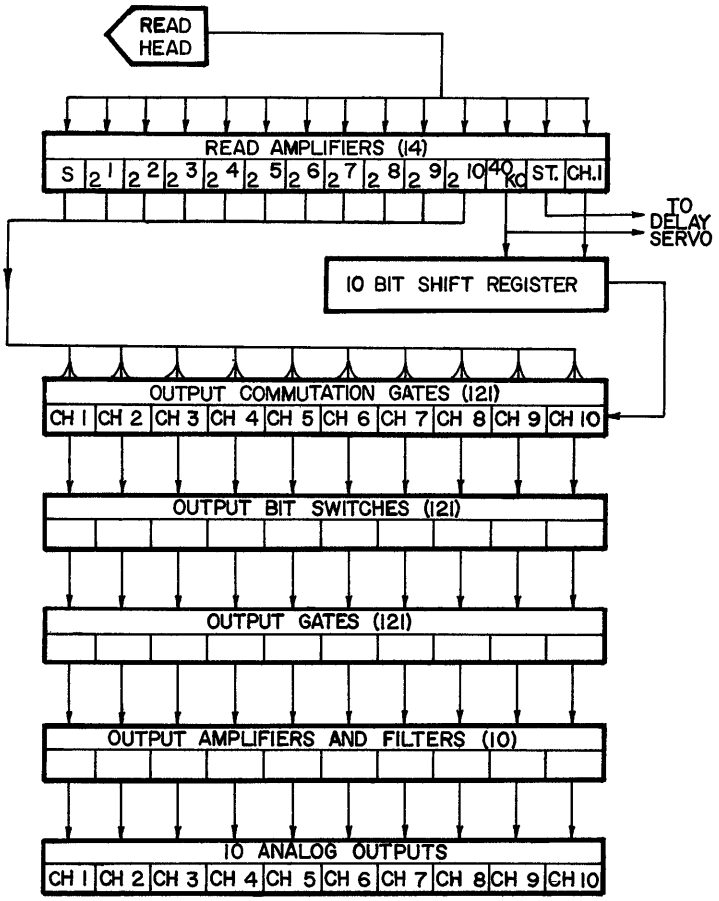


FIGURE 7
DIGITAL TO ANALOG CONVERTERS

DAFT--A DIGITAL/ANALOG FUNCTION TABLE

by

R. M. Beck and J. M. Mitchell

Packard Bell Computer Corporation
Los Angeles, California

Abstract

The design of a digital function generator based on incremental digital computer techniques is described which is capable of generating both mathematical functions and arbitrary functions. The mathematical functions are generated using parallel Digital Differential Analyzer methods. The arbitrary functions are obtained by generating interpolating curves through programmed numerical points. The mathematical functions can be generated with a high degree of accuracy and repeatability while the arbitrary functions can be generated to a high degree of repeatability but an accuracy limited by second order interpolation methods.

The interpolating equation generates a curve made up of a chain of parabolic segments passing through thirty-two numerical points which may be unequally spaced along the X axis. Both second order and first order interpolation may be performed.

Fourteen bit incremental Analog-to-Digital and Digital-to-Analog converters are included which enables the use of the DAFT system in conjunction with analog computer facilities. A 3 megacycle clock rate is used, enabling the transfer of information at 100,000 increments per second allowing for very high speed computation.

Introduction

This paper describes the design for a set of incremental computer blocks. The basic blocks of this system are shown in Figure 1. The incremental inputs and outputs of a number of these blocks may be interconnected to solve a large class of research and development problems. The rate of transfer of information between such blocks is 100,000 increments per second which allows for very high speed computation. The operation of the blocks is numerical to allow for very high computational accuracy. This form of computer system should prove particularly useful for solving those complex problems in which present analog computers exhibit prohibitive total errors.

The Integrator, Variable Multiplier, Constant Multiplier and Servo Units are of the Digital Differential Analyzer type. The DAFT Unit is a Digital/Analog Function Table. The Analog-to-Digital and Digital-to-Analog Converter Units operate incrementally to convert between voltage information and digital information. These converters provide communication links between the incremental computer and either an analog computer or a process control system. The operation of these interconnected blocks is controlled by a central Control Unit. This Control Unit serves to enter problem parameters, monitor the operation of the system, and synchronize the system operation. This equipment can be used to generate many mathematical functions such as shown in Figure 2.

When a function cannot be expressed satisfactorily in an analytic form, the DAFT unit may be used to generate it as an arbitrary function as shown in Figure 3. The mathematical functions can be generated with a high degree of accuracy while the arbitrary functions can be generated to a high degree of repeatability but an accuracy limited by second order interpolation methods.

DAFT Function Generator

General

One particular interconnection of incremental units (see Figure 3) will be given special attention in this report. This system of units accepts an input voltage, X, generates a function, f(X), from a numerical function storage and generates Z = f(X) as an output voltage. In this way an analog function generator is obtained with a very high degree of repeatability.

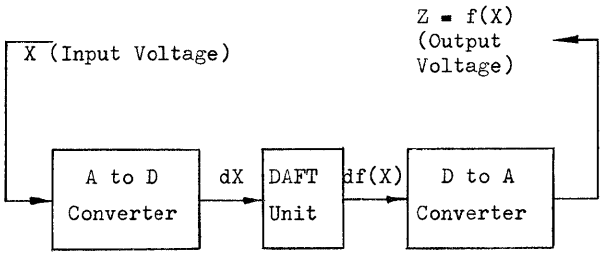


Fig. 3 Arbitrary Function Generator

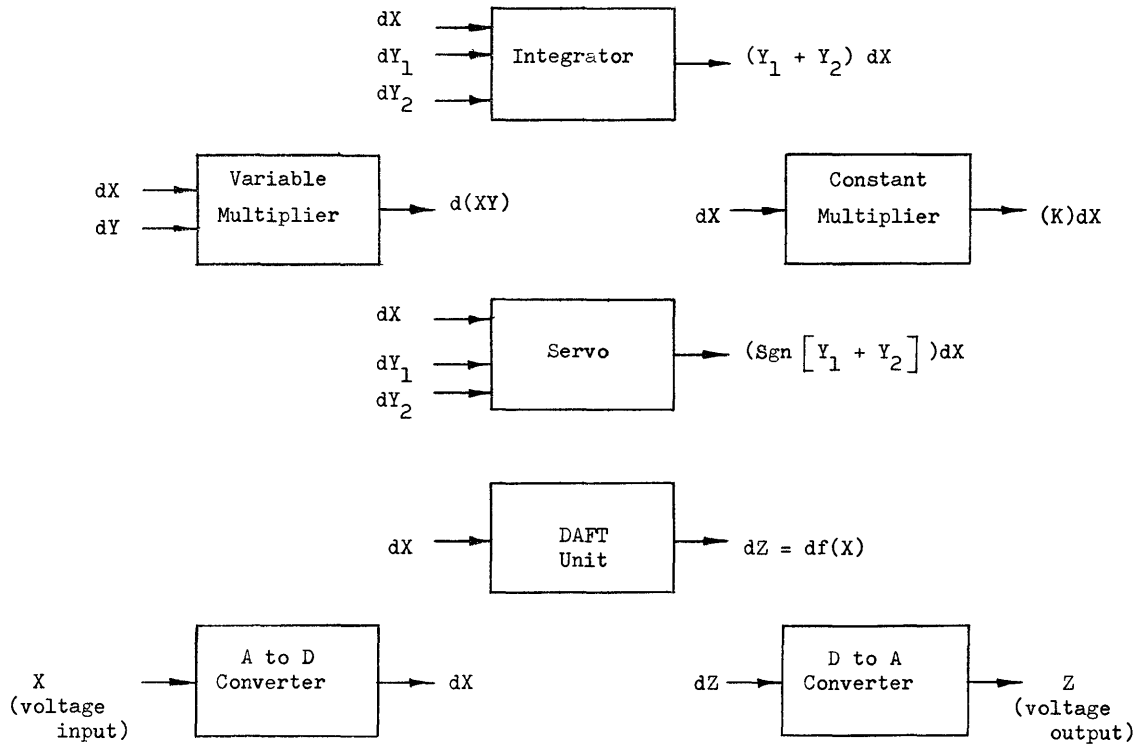


Fig. 1 Incremental Computer Blocks

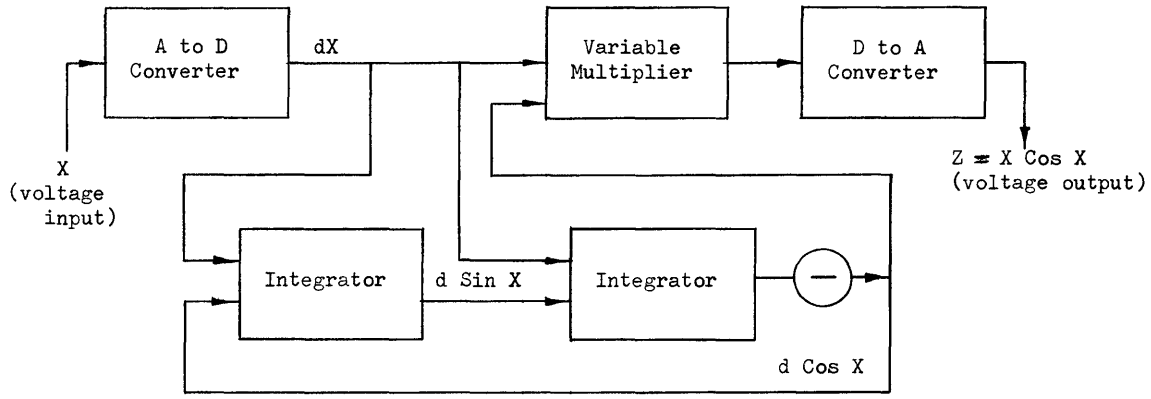


Fig. 2 Mathematical Function Generator

Incremental Technique

It is characteristic of incremental computers to generate fewer dZ output increments than dX increments. If the X scale of the graph, $Z = f(X)$, is divided into 16,384 dX increments and the Z scale is divided into 16,384 dZ increments, the Z function cannot generally extend over this full scale. Figure 4 indicates the range of Z for several types of functions.

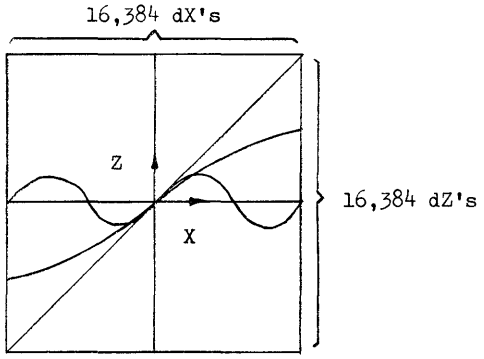


Fig. 4 Incrementally Generated Curves

When the curve is Z = X, the curve can extend over 16,384 dZ's. The maximum excursions for other curves are given below:

½ cycle sine curve ----10,420 dZ's

1 cycle sine curve ---- 5,210 dZ's

2 cycle sine curve ---- 2,605 dZ's

4 cycle sine curve ---- 1,302 dZ's

8 cycle sine curve ---- 651 dZ's

This does not mean that the output amplitude necessarily becomes reduced with higher frequency curves (the output amplitude can be scaled up at either the D to A converter or by an output amplifier), but that the resolution between output values becomes coarser.

The scaling of X at 16,384 dX's is consistent with the resolution of a high precision converter and is chosen to give a fine resolution for the output generation.

Numerical Definition of Z = f(X) Curve

The curve to be defined is to be divided into 32 ΔX increments along the X scale. Each ΔX_i will have $(128)(N_i)$ of dX increments, where N_i is an integer from 1 to 16. The nominal value for N_i is 4 so that the nominal value of ΔX_i is 512 dX's. Individual ΔX_i's may be one-fourth the nominal or four times the nominal, but the sum of the 32 N_i's should be 128 so as to span the full scale of 16,384 dX's. The X scale may be visualized as divided by 129 grid lines with a spacing of 128 dX's between lines. The values of Z at any 31 of these grid lines may be used to define the curve provided that no more than 15 grid lines are skipped as a group. It is assumed that the first and last grid lines are always used to give 33 values of Z or 32 ΔX's and 32 ΔZ's.

The Z values are measured in units of dZ increments. The values used for Z and X must be chosen subject to some restrictions.

The first restriction is that $\Delta Z_i/\Delta X_i = n_i/128$, where n_i is an integer from -128 to +127. This states that slope, $\Delta Z_i/\Delta X_i$, is restricted to one of 256 different values (since $\Delta X_i = 128\ N_i$, $n_i = \Delta Z_i/N_i$).

The second restriction is that one of the X-Z points that is chosen be the origin. This requires that curves which do not pass through the origin be displaced vertically be the constant, f(0). This constant can be readily added back to the generated curve in the analog computer as well as in the incremental computer.

The third restriction is that the change of slope between ΔX increments be such that $n_{i+1} - n_i = m_i$ where m_i is an integer from -128 to +127.

The fourth restriction is that $|n_i - \frac{1}{2} m_i| < 128$ which may also be expressed as $|3/2\ n_i - \frac{1}{2} n_{i+1}| < 128$. This is necessary to limit the slope used in parabolic interpolation and will be mentioned later. Note that if the other restrictions are satisfied this restriction will be satisfied whenever $|n_i| < 64$.

Interpolation Method.

The interpolation formula to be used in the region X_i, is the following:

$$dZ = \left[\frac{\Delta Z_i}{\Delta X_i} + \left(\frac{\Delta Z_{i+1}}{\Delta X_{i+1}} - \frac{\Delta Z_i}{\Delta X_i}\right)\left(\frac{X - X_{i-1}}{\Delta X_i}\right)\right] dX - \frac{1}{2}\left(\frac{\Delta Z_{i+1}}{\Delta X_{i+1}} - \frac{\Delta Z_i}{\Delta X_i}\right) dX$$

or

$$dZ = \left[\frac{n_i}{128} + \left(\frac{m_i}{128}\right)\left(\frac{X - X_{i-1}}{128 N_i}\right)\right] dX - \frac{1}{2}\left(\frac{m_i}{128}\right) dX.$$

Figure 5 shows the interpolation curves for Z in the intervals, ΔX_1 and ΔX_2

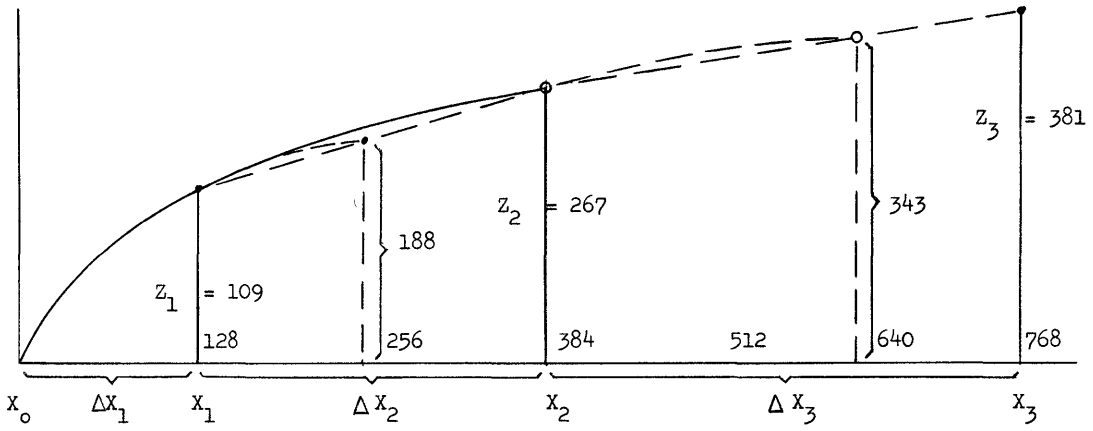


Fig. 5 Interpolation Curves

In general, the interpolation formula yields a curve that is a chain of parabolic sections. The parabola used in the increment ΔX_i passes through three points:

1) X_{i-1}, Z_{i-1}

2) X_i, Z_i

3) $X_i + \Delta X_i$, $Z_i + \frac{\Delta Z_{i+1}}{\Delta X_{i+1}} \Delta X_i$

This linearly interpolated third point is much easier to use than (X_{i+1}, Z_{i+1}) for the third point of the parabola.

The interpolation system is indicated by the block diagram in Figure 6. The dX increments, of which there are 128 N_i as X varies through ΔX_i, are scaled down in number by $1/N_i$ to give 128 increments per ΔX_i. The increments are scaled down further in number by 1/128 to give one increment per ΔX_i. These i increments are summed in the i counter to control the selection of m_i and N_i from the storage.

As X varies through ΔX_i, m_i is added into the A register 128 times. The A register has 64 in it at the beginning of ΔX_i (this is 1000000 in binary). After m_i has been added into the A register 128 times, the A register will have overflowed m_i times and again have 64 in it. The overflows of the A register are summed in the B register. At the beginning of each ΔX_i increment, the B register contains the value, n_i, which is increased by m_i to $n_i + 1$ at the end of the ΔX_i increment.

While X varies through the ΔX_i increment, m_i is added into the C register 128 N_i times. The C register operates much like the A register, beginning and ending with 64 in it, but overflowing $(N_i)(m_i)$ times. These overflows are scaled down by -½ and summed with the overflows from the D register to obtain the dZ output increments.

While X varies through the ΔX_i increment, the arying value in the B Register is added into the D register $128N_i$ times. The value added from the B register into the D register corresponds to $n_i + m_i (X-X_{i-1}/128N_i)$ dX from the interpolation formula. The output from the -½ scaler added into the D register corresponds to $-\frac{1}{2}(m_i)$ dX from the interpolation formula. By taking the overflows for the sum of these terms from the 7-bit D register, we obtain a 1/128 factor or

$$dZ = \frac{1}{128} \; n_i + m_i \frac{X-X_{i-1}}{128N_i} \; dX - \frac{1}{128} \; \frac{1}{2} \; m_i \; dX$$

The DAFT Unit operates as a numerically drift-free interpolator which traces out a smooth curve through the Z_i points regardless of the number of cycles of the X input voltage. This curve will have a point by point uniformity for all cycles of X.

When linear interpolation is desired, the overflow signals from the C register may be used as the d Z output and the A, B and D registers not be used. The curve traced in this case will be a chain of straight line segments joining the Z_i points. The values of n_i will be taken from

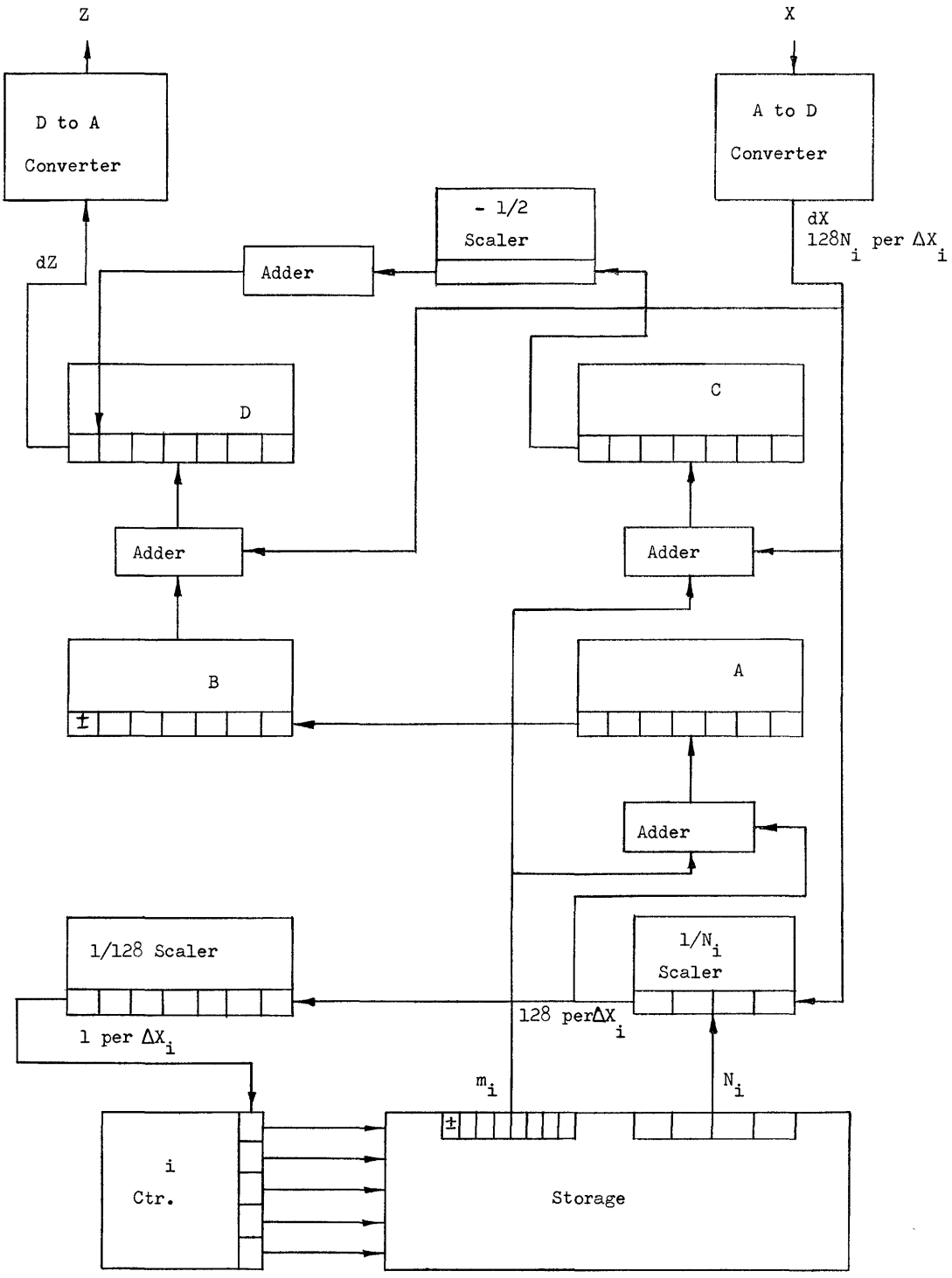


Fig. 6 Interpolator Block Diagram

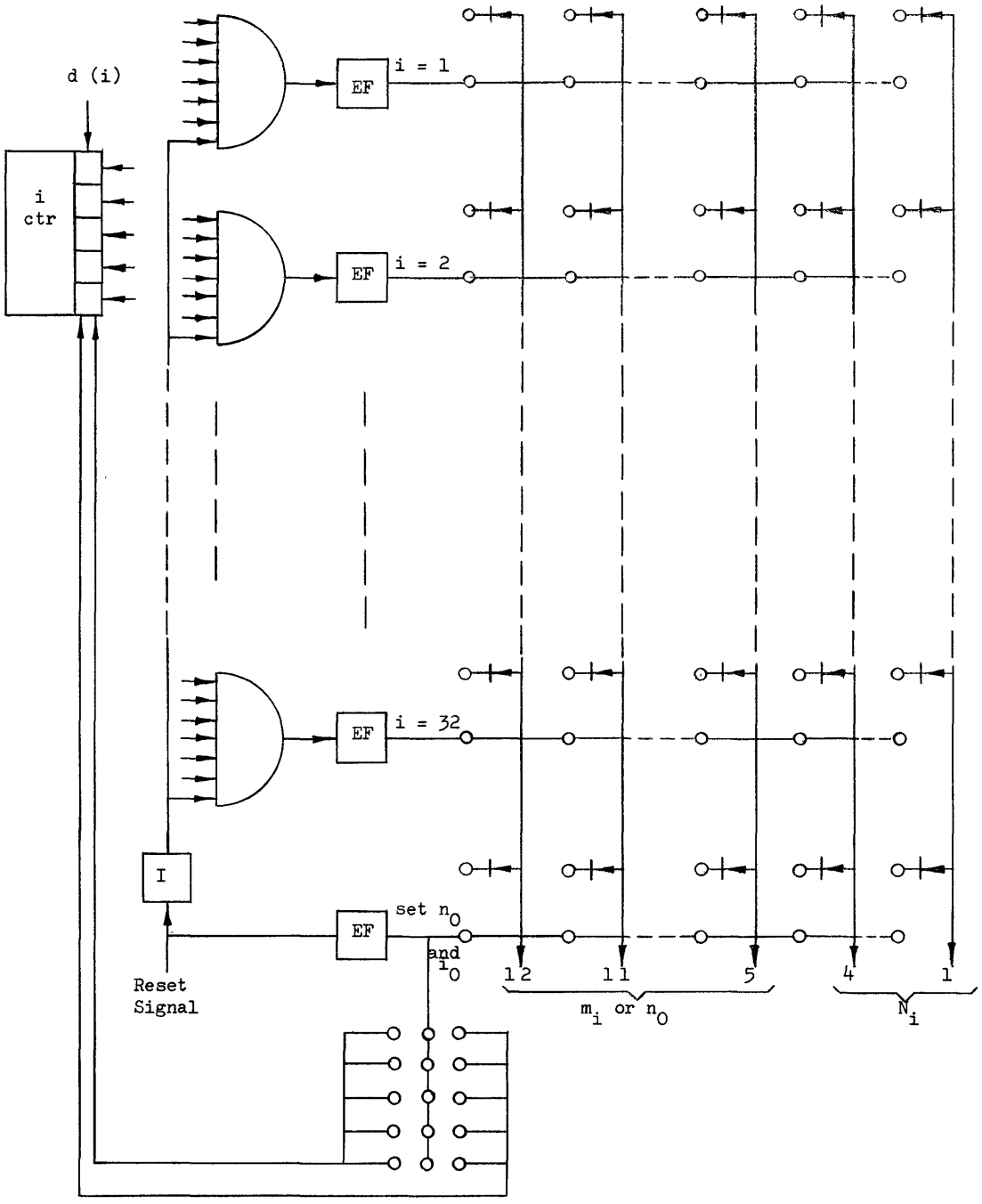


Fig. 7 Patchboard Storage Circuit

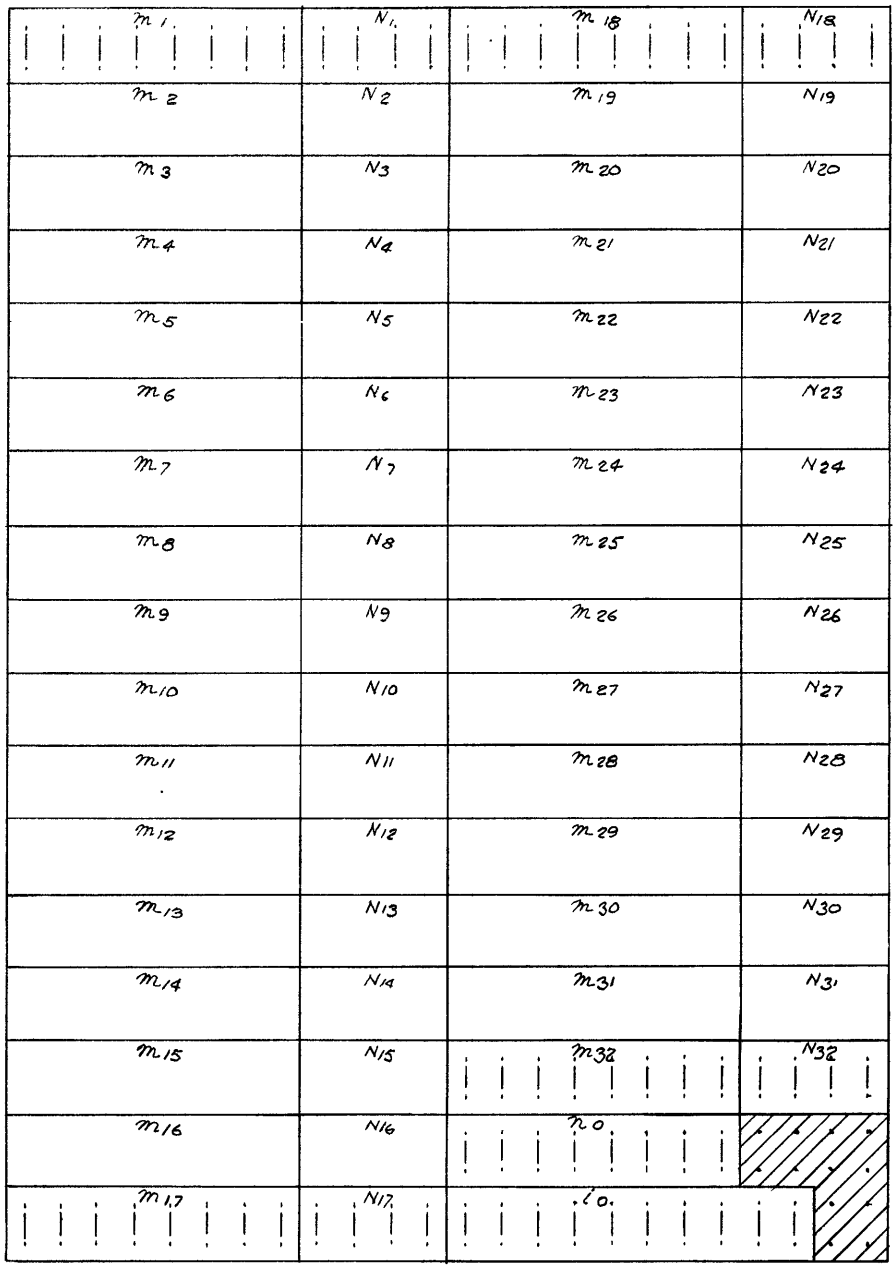


Fig. 7a Function Patchboard Layout

storage rather than m_i. Only the first two restrictions on the X-Z points will apply in this case.

Method of Operation

The design of the system is arranged for ease of operation and testing. To set the function generator system into operation, the function patchboard is inserted into the DAFT Unit and the Control Unit Reset button pressed. Pressing this button does the following:

1) Clears the X and Z counters in the converters to zero.

2) Transfers the origin values of n and i from the patchboard into the B register and i counter of the interpolator.

3) Sets 64 into the A, C and D registers and clears the Scalers of the interpolator.

When the Control Unit is switched into computation, the function generator system proceeds to generate the curve Z = f(X) from the origin to the current value of the X input voltage in 0.082 seconds or less. The function generator will then track on the variations in X until the Control Unit is switched out of computation. When the system is switched out of computation, the values of X and Z are locked on the last computed point of the curve. When computation is restarted, the generator will slew at the rate of 100,000 dX's per second to the current value of X.

To perform a quick test for errors in the operation of the generator, the push button to set the X input to zero may be pressed. The X and Z indicators should both slew to a zero reading when this button is pressed and back to the current (X,Z) point when this button is released.

To verify the operation of a new function patchboard, the manually controlled X input voltage may be used to obtain the (X_i, Z_i) readings on the indicator lamps that are prescribed for the function. When the readings are incorrect, the patchboard can be corrected and the revised data verified directly.

Circuits

All circuits are designed using germanium transistors and diodes. No tubes or mechanical choppers are used. All components are used at a conservative power and tolerance rating. The design goal is to achieve very reliable operation at low power consumption.

The converter units use the circuits developed for the Packard Bell Computer Corporation MULTIVERTER equipment.

The circuits used for the DAFT interpolator are the circuits used in the Packard Bell Computer Corporation TRICE computer. These circuits consist of transistor flip flops, diode gates, and distributed delay line storage elements.

Figure 7 shows the gating circuit designed for the Patchboard Storage Unit. This circuit has the moderate speed requirement of obtaining a new set of values on its 12 output lines within five microseconds after the i counter changes. One of 32 diode-and-gates transmits a DC signal depending on the state of the i counter. This signal is sent through an emitter follower at an 8-volt level to a 12-contact common of the patchboard. The signal is connected or disconnected with patchboard jumper to twelve diode-Or-gates. The twelve Or-gate lines present the interpolator with the m_i and n_i values in parallel.

When the Reset Signal comes from the Control Unit, it is used to block the reading of m_i and n_i values and introduce the value for n at the origin on the Or-gate lines. This signal is also used to transfer the value for i at the origin to the i counter flip flops.

X Frequency Response and Accuracy

The DAFT Unit has a maximum operation rate of 100,000 dX increments per second. This rate is chosen to be compatible with the other units of the system. When the X scale is divided into 16,384 dX increments, the value of X for a sinusoidal input of frequency, f, is given by

$$X = (16,384/2) \sin 2\pi ft$$

and the maximum rate of change of X is

$$(dX/dt)_{max} = 16,384\pi f.$$

Therefore, the system will track if the maximum full scale input frequency is

$$f = 100,000/16,384\pi = 1.95 \text{ cps.}$$

When the tracking frequency of 1.95 cps is exceeded by a signal of, say, 3 cps, the Analog-to-Digital Converter fails to follow closely the steep portions of the input curve. Figure 11 indicates the discrepancy that will occur in this case. The result is approximately a triangular wave with 5% amplitude reduction, a maximum error of 21% of full scale and a phase shift of 20°. The function generator continues to function properly but for the distorted input curve.

The described system units can be used at

higher frequencies by reducing the number of dX's per full scale of X. For example, the X input resistor of the Analog-to-Digital Converter can be changed to give 8,192 dX's per full scale of X and the sum of the N_i values reduces to 64. The maximum full scale input frequency can then be 3.9 cps. The number of dX's for full scale can be further reduced to 4096 and all 32 N_i values restricted to 1's. This will accomodate a full scale input frequency of 7.8 cps.

In general, the following relationship exists between frequency and the number of dX's:

(number of dX's/full scale of X) = $100{,}000/\pi f$

When the curve, Z = f(X), is characterized as a sine curve of C cycles,

(number of dZ's/full scale of Z)
= $(1/\pi C)$ (number of dX's/full scale of X)

Therefore,

(number of dZ's/full scale of Z)

= $100{,}000/\pi^2 Cf = 10{,}132/Cf$

Where the output is expressed as a binary number with p binary stages used, the formula may be written as

$$2^p = 10{,}132/Cf$$

This last equation essentially defines the frequency-accuracy product of the DAFT function generator system.

As an example, if the input frequency is 3 cps and the Z = f(X) curve resembles 2 cycles of a sine curve, the generator can be set up to give an output curve that is smooth to one part in 2^{10}.

No. of dX's = $100{,}000/\pi(3) \cong 10{,}496 = (82)(128)$

$\sum N_i = 82$

No. of dZ's = $10{,}496/\pi(2) \cong 2^{10}$

for C = 2 cycles and f = 3 cps

For another example, if the input frequency is 30 cps and the Z = f(X) curve resembles one cycle of a sine curve, the generator can be set up to give an output curve that is smooth to one part in 2^8. Only eight ΔX increments may be used for this input frequency.

No. of dX's = $100{,}000/\pi(30) \cong 1024 = 8(128)$

$\sum N_i = 8$

No. of dZ's = $1024/\pi(1) \cong 2^8$

for C, = 1 cycle and f = 30 cps

In order to obtain more ΔX_i increments for high frequency inputs, the DAFT Unit will have one other scaling arrangement:

1) The 1/128 Scaler will be changed to a 1/16 Scaler

2) The B register will be shortened to 4 bits plus sign.

3) The number of dX's/ΔX_i = $(N_i)(16)$

4) n_i will be an integer from -16 to +15

5) m_i will be an integer from -16 to +15

6) $|n_i - \frac{1}{16}m_i| < 16$

The last example may then have all 32 X increments used

No. of dX's = $100{,}000/\pi(30) \cong 1024 = 64(16)$

$\sum N_i = 64$

No. of dZ's = $1024/\pi(1) \cong 2^8$

for C = 1 cycle and f = 30 cps

Summary

The set of units described provide a flexible, high performance system with which arbitrary functions and mathematical functions can be obtained at high speed and high accuracy. The arbitrary functions are set up in numerical form on a patchboard and generated with first or second order interpolation. The mathematical functions are generated in their analytic form with interconnected Digital Differential Analyzer units. The performance of the system in generating both types of functions is approximated by the formula

$$2^p = 10{,}132/Cf$$

where

p = number of binary bits precision (including sign) in the output,

C = The complexity of the function measured in terms of the number of sine curve cycles it contains,

f = the full amplitude input frequency of the independent variable input.

The independent variable may be followed to a precision of one part in 16,384 and the

output variable may be generated up to a precision of one part in 10,000. The system operates at the rate of 100,000 cps so that in the worst case, the delay between a variation in the input and a corresponding variation in the output will be much less than 50 microseconds.

Another type of function not treated explicitly in the body of this report is that of generating an analytic function of time such as

$$Z = \sin 2\pi ft$$

This function can be generated with a performance formula of

$$2^p = 100{,}000/\ f,$$

where p is the number of bits precision (including sign) in the output. The value of p is limited to 14 or less by the output converter. For f of 64 cps the sine can be generated to 8 bits plus sign and for f of 200 cps the sine can be generated to 6 bits plus sign. These curves can be generated for an unlimited number of cycles without amplitude drift.

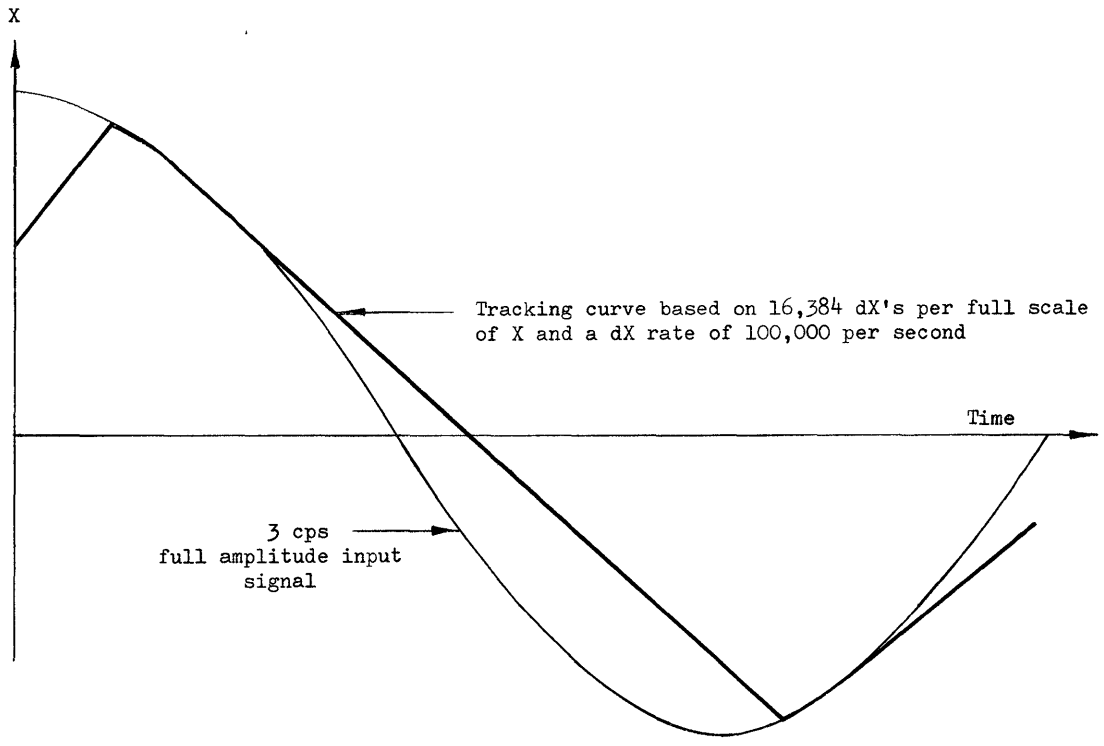


Fig. 8 Slope - Limited Tracking Curve

MATHEMATICAL APPLICATIONS OF THE DYNAMIC STORAGE ANALOG COMPUTER

Jack M. Andrews
Computer Systems, Inc.
611 Broadway
New York 12, N. Y.

Many recent developments in analog computers have substantially increased their versatility and speed. Probably the two most important of these innovations is the automatic repetitive operation at frequencies of 60 cps and the development of high speed memory with a tracking error of less than 10 microseconds. With repetitive operation and the simultaneous solution of large matrices of ordinary differential equations at speeds of 60 cps, the analog is much faster in operation than the digital. However, an inherent disadvantage is the size of computer required for a complex problem. In general, the more complex or larger a problem, the more conventional analog computer equipment that has been required for its solution. The digital computer, on the other hand, has, from its earliest inception, employed memory to allow time sharing of components. In this manner, a large complex problem could be solved by time sharing the digital computer equipment and using the same equipment repeatedly to solve the larger scale problems. Therefore, large problems required longer solution times rather than larger computers. The disadvantage of larger sizes in the analog computer has been overcome by the development of high speed memory which is employed in a manner similar to that utilized in the digital computer for time sharing to increase the time of solution rather than increasing the size of the computer for the more complex problems. However, the order of magnitude increase in solution time is much less severe with the analog computer employing memory than it is with the conventional digital computer, since the analog simultaneously solves systems of equations, while the digital computer must use reiterative procedures for such a system.

SUMMARY

The development of the high-speed analog memory has made possible sequential calculations through time sharing of the computer components. One of the most striking comparisons is found in the process industry, where distillation is a critical test of the computing capabilities of both the analog and digital machines. Prior to the development of high-speed repetitive operations and dynamic storage analog computers, a four component distillation problem with overhead, sidestream, and bottoms products would require in excess of 1100 amplifiers for the simultaneous solution of a 30 plate column, or conversely, a solution time of several hours. With the modern DYSTAC Analog Computer in repetitive operation, such a solution can be obtained with only 43 amplifiers in approximately 1 minute of operating time. This is a very favorable comparison with the more expensive digital computers.

Memory can also be employed through high speed sequential calculations to obtain definite integrals for automated, self-solution of various problems. One of the most important of these in industry is the optimization of a definite integral with respect to one or more independent variables. Probably the most common example is the realization of an economic optimization for an operating process. However, other criteria can be employed for optimization such as the least mean squares fit of data in correlation schemes or the minimization of transients with respect to design parameters in an instrument control system.

Furthermore, many partials can be reduced to multiple definite integrations. Under these conditions a combination of repetitive operations and non-repetitive operations (real-time) is employed together with the continuous or point memories to provide a simple, accurate, and rapid solution for multiple integration.

The variables are not generally separated in partial differential equations. Where they can be separated mathematically, a multiple integration technique, similar to that just discussed, is readily employed to

obtain a solution to the proboem. Where the partials cannot be separated mathematically, the difficulty generally lies in a resolution of the initial conditions or boundary conditions. In the latter circumstance, a forced balance solution which is readily incorporated into the analog computer may be of assistance in arriving at a solution of the problem.

The development of memory for the analog computer has made it possible to incorporate many digital techniques into the current analog procedures. Continuous memory for the analog computer is an adaptation of digital techniques employing Newton's Divided Difference Interpolation Formula. The employment of continuous memory immediately provides solutions for various types of transient operations and difference-differential equations problems. The ordinary solution time of even the most complex problems in this category is about 20 seconds.

Other applications which have stemmed directly from the repetitive operations and memory include an all electronic transport delay which is derived from standard computer components and differentiation without the noise that has heretofore been associated with such an operation on analog computers.

DISCUSSION

DYNAMIC MEMORY

In order to develop the theory of dynamic memory, the performance of a standard integrator in both non-repetitive operation or real time will be discussed first. The circuitry of such an integrator is generally similar to that presented in Figure 1.

In Figure 1, the high gain amplifier is shown with the feedback capacitor (C) and input resistor (R_1) which constitute a standard non-repetitive operating integrator. In order to reset the integrator to the applied initial condition voltage, both relays A and B are required. During the reset period of non-repetitive operation, relay A is open and relay B is closed. During this period, the capacitor is biased to produce at the output of the integrator a value of E_0, which is the negative value of the initial condition imposed on the initial condition resistor network. When the operate cycle is initiated, relay B is opened and relay A closes, so that the value of the voltage input E_1 can be integrated. The time-integrated value of E_1 from the appropriate initial condition is then available as the output voltage E_0.

If the integrated value of E_1 is a function of an independent variable, x, then E_0 may be expressed as F(x). Figure 2 shows the variation in the output voltage E_0 for a decaying function during the operate and reset cycles.

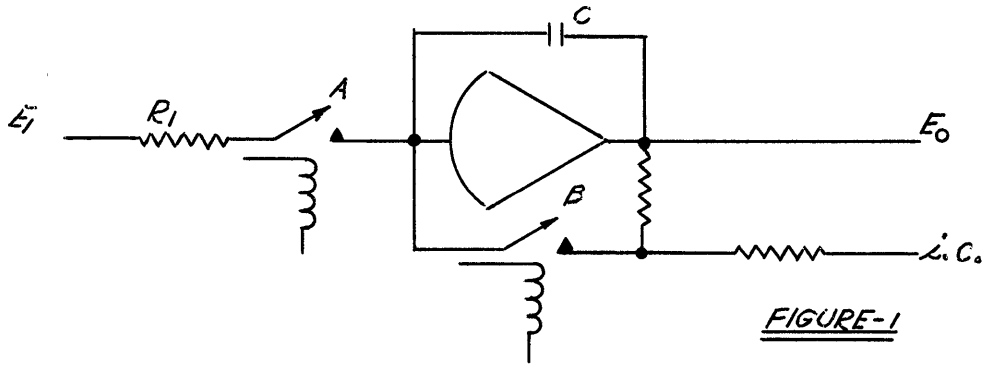
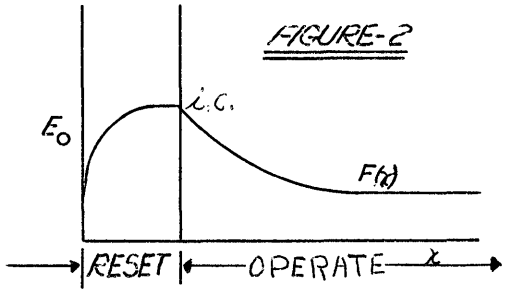
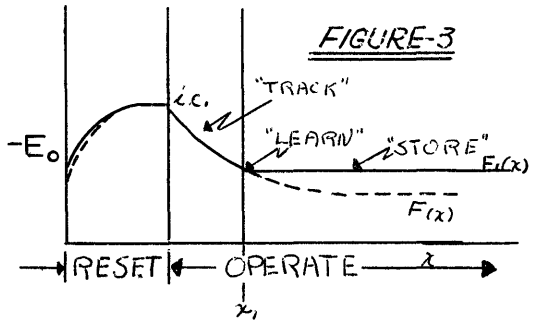
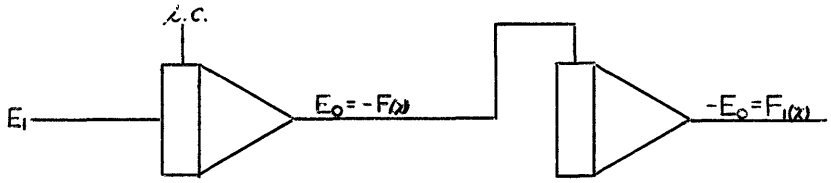


FIGURE-1







At the beginning of the operate cycle, the value of E_0 is equal to the initial condition. During this subsequent integration, the value of E_0 follows the time integral of x which corresponds to F(x). At the termination of the operate cycle, the computer is returned to the reset period where the value E_0 changes to correspond to the initial conditions.

If another integrator is in reset (relay B closed) and has no input other than F(x) through the initial condition channel, it will track F(x) and have for an output voltage the value -F(x) as shown in Figure 3. If the B relay is caused to open at any value (x_1) of the variable x, then the output will subsequently remain constant at $F_1(x)$ for all values of x greater than x_1. During the reset period, if x is less than x_1, the memory will track the resetting of F(x).

In this manner, by eliminating the normal input voltages and using only the initial condition input, memory may be incorporated into a standard integrator. While this procedure might be followed with any high-precision analog computer, the long RC time constant of the reset circuit and the slow operating speed of the B relay (one millisecond or greater) would limit its application to non-repetitive solutions with long time constants. It will be shown that practical applications of this technique require repetitive operation at frequencies of from one to two hundred cycles per second, preferably 50 cycles per second or greater. Utilization of this procedure in repetitive operation is possible only when the reset relay B is replaced by an electronic equalizing gate or switch which operates at a speed of approximately 5 microseconds and resets the amplifier independent of the RC time constant of the initial condition circuit.

Four types of memory mode are ordinarily employed in the circuits necessary for complex problem solutions. Shorthand symbols for these memory circuits are presented in Figure 4.

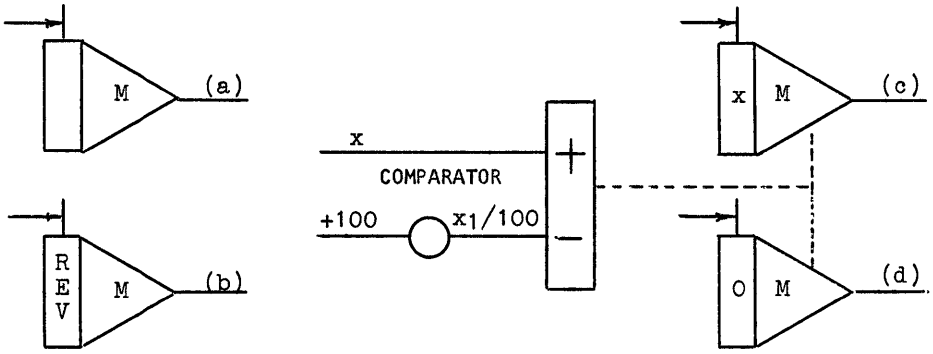


FIGURE 4
DYSTAC MEMORY SYMBOLS

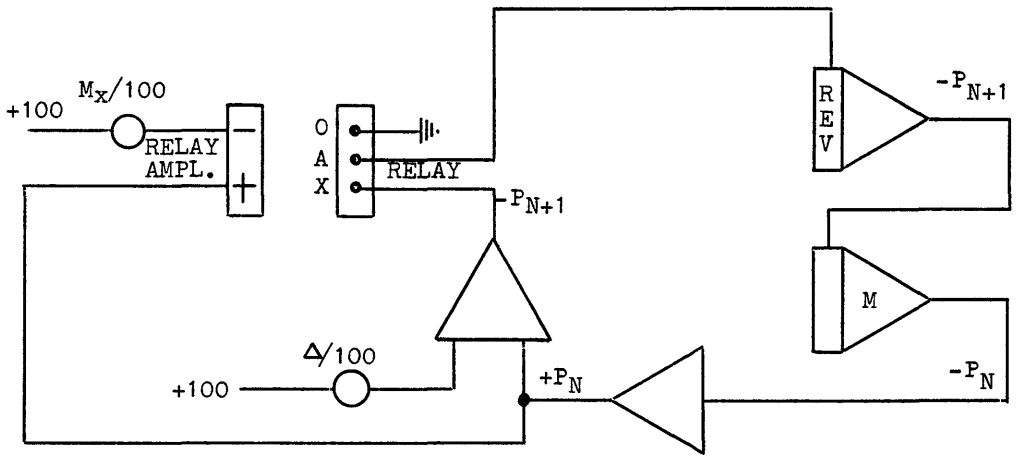
The letter M is utilized to indicate that each of the four amplifiers is employed in the memory application and is a reminder that the normal input channels to these amplifiers should not be used; the only inputs should be through the initial condition channel. The standard integrator, termed a forward memory and shown in Figure 4.(a), "tracks" the function present at the initial condition input during the computer reset period, "learns" it at the start of the operate period and "stores" or holds it during the operate cycle.

In Figure 4. (b) a reverse memory is shown. In this application the polarity of the control signal is reversed so that the electronic reset switch operation is reversed, and the output voltage tracks the i.c. voltage during the operate period. During the reset period, the output voltage is stored at the value last available through the initial condition input at the termination of the operate cycle. Both of the memories in Figures 4. (a) and 4. (b) are controlled by the repetitive, automatic reset pulses generated by the computer. In the last two Figures, 4. (c) and 4. (d), the electronic reset switches or equalizer gates are actuated by external voltage comparators rather than by the repetitive, automatic reset pulses. Figure 4. (c) shows an amplifier whose electronic switch is normally closed until the external comparator is actuated by an independent or dependent variable whose value exceeds that of an arbitrary voltage x_1. Figure 4. (d) shows an integrator memory whose reset electronic switch is normally open until the comparator is actuated. In the case of Figure 4. (c) the output voltage tracks the value available at the initial condition input until the comparator is actuated, at which time the output voltage is locked or stored at the instantaneous initial condition value. In Figure 4. (d) the output voltage is stored and constant until the comparator is actuated. At this time the output voltage tracks the initial condition input.

SEQUENTIAL CALCULATIONS

In order to perform high-speed sequential operations which are automatically controlled by the computer, it is necessary to have an accounting circuit which records the number of steps that have been performed and whose output voltage can serve as an indication for relay amplifiers and comparative circuits through which the type and complexity of the sequential calculations can be regulated. An accounting circuit

which provides these features is presented in Figure 5. This circuitry is composed of reverse and forward memory together with an inverter, summer, relay, and relay amplifier. If, for the purposes of discussion, it is assumed that M_x has a value of 30 and Δ has a value of 1, a constant value of one volt will be added to P_N through a summer.

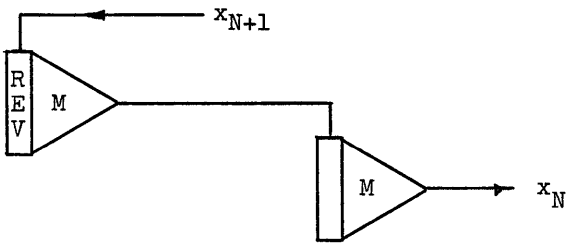


ACCOUNTING CIRCUIT
FIGURE 5

The relay amplifier will compare $+P_N$ with a value of 30 and operate the relay, driving it from the normally closed (x) to the normally open (0) position when P_N has attained an equivalent of 30 volts. Let us assume that P_N has reached a value of 30 volts. This value will be maintained at the output of the forward memory during the remainder of the current operate cycle since the forward memory will be in "store". With the arm of the relay at the normally open position, a ground will be imposed through the initial condition channel of the reverse memory which will track this value and "learn" it at the end of the current cycle. During this reset period the reverse memory will store the output value of zero and the forward memory will track and learn it since its electronic equalizer gate is closed during the reset cycle. In the next operating cycle the forward memory will have learned and stored a value of zero which will appear as its output. During this cycle when P_N has a value of zero, the relay will return to its normally closed position and the value of P_N plus one volt will appear at the initial condition input of the reverse memory. Therefore, during the next operate cycle the value P_{N+1} appearing at the output of the reverse memory will be one volt. This value is in turn stored at the output during the reset cycle, and the forward memory will learn a value of 1 volt which it will maintain during the following operate cycle. In this manner, P_N will have voltage values varying in incremental steps of 1 volt from 0 to the maximum value of M_x. During the corresponding cycle the value appearing as P_{N+1} at the output of the reverse memory will always be 1 volt greater until P_N is equal to the maximum voltage M_x at which time P_{N+1} will be zero. Either P_N or P_{N+1} may be used as control voltages to actuate either relay amplifiers or electronic switches to control the various sequential calculations which are incorporated in the cycle from $P_N = 0$ to $P_N = M_x$. The cycle thus effected by this control will be

reiterative and will cycle through the various sequential calculations until the repetition is halted by the operator. This feature provides a very efficient means for conducting sequential algebraic calculations such as are encountered in distillation, difference differential equations, partial differential equations and in control problems where the control variables are cycled through a series of operating conditions or boundary conditions.

The ratchet circuit shown in Figure 6 is another example of a highly effective memory circuit which finds its most general application in the memorization of values from one cycle to be used in the succeeding cycle.



RATCHET CIRCUIT
FIGURE 6

The forward memory will produce a value of x_N which will be utilized throughout an entire cycle to calculate the value of x_{N+1} that is necessary for the following cycle. This value, x_{N+1}, is derived from the associated problem circuitry, and will be memorized during the reset period to provide x_{N+1} for the calculation of x_{N+2}, etc.

Distillation is one of the most severe tests of the capabilities of either a digital or analog computer to solve sequential calculations. Before the development of DYSTAC, the typical distillation problem was either too expensive or too tedious to solve on the analog computer, and the digital computer was employed almost entirely in the distillation calculations of the process industries.

However, as discussed in a technical data sheet by Computer Systems, Inc. entitled "Application of DYSTAC to a Multi-Component Distillation" this new analog development provides a rapid and economic solution of multi-component distillation problems by employing both the ratchet and accounting circuits for time sharing of computer components. The size of the computer is independent of the number of plates and is dependent only on the number of components to be distilled and the number of product withdrawal points. Since the procedure is perfectly general, any desired specifications can be selected for a given case within the prescribed degrees of freedom. The solution is based on a trial and error procedure with a continuous solution frequency of 60 plates per second. A solution with an accuracy of 0.1% can be obtained for a given case in about 1 minute after the basic data has been set in the computer.

Only 43 operational amplifiers are required for a four component distillation with overhead, sidestream, and bottoms withdrawal products. This contrasts with a requirement in excess of 1100 amplifiers for conventional analog computer equipment wherein the entire column is solved simultaneously. If more than 30 plates are required, the amplifier requirement is proportionately increased. In the current application, the number of amplifiers is independent of the number of plates because the various electronic components are time hhared so that from cycle to cycle each component is used repeatedly in successive solutions for the various plates.

THE DEFINITE INTEGRAL

In many problems it is desirable to evaluate in repetitive operations at definite and sometimes changing limits the integral of various quantities. Typical examples include economic optimization, the minimization of transients through design parameters in a control system, and the evaluation on a least mean squares basis of the proper correlation coefficients.

The evaluation of a definite integral by the employment of X and 0 memories in repetitive operation is illustrated in Figure 7.

indefinite integral as it is produced in repetitive operation mode until x is equal to a. At this point, the X memory locks and the 0 memory begins to track with the result that the 0 memory has as its output the previous evaluation of the definite integral from zero to a from each preceding cycle. It is quite feasible to vary the limit of integration a from one cycle to another. In this case it would be provided by some other portion of the circuitry in accordance with the desired problem solution.

When it is desired to obtain the economic optimum operating conditions for a process it is generally necessary to evaluate dffinite integrations. A general out-

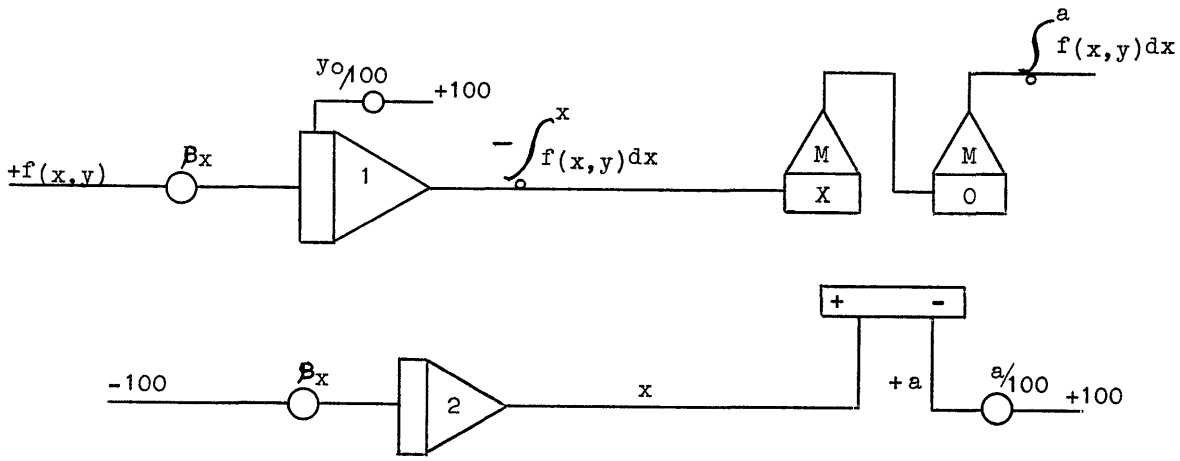


FIGURE 7

In this figure, it is desired to determine the definite integral of f(x,y) with respect to the variable x. The desired function is the drive for integrator 1, wherein the proper initial condition y_o is also available, and the output of integrator 1 is the indefinite integral with respect to x. In this application, integrator 1 will be operated in the repetitive mode. Integrator 2 will also be operated in repetitive mode, and its purpose is to generate a ramp function for the evaluation of the independent variable, x. x is compared with a value a from reference voltage to establish the upper limit of the desired definite integral. The X memory will track the

line of the procedure for optimization follows.

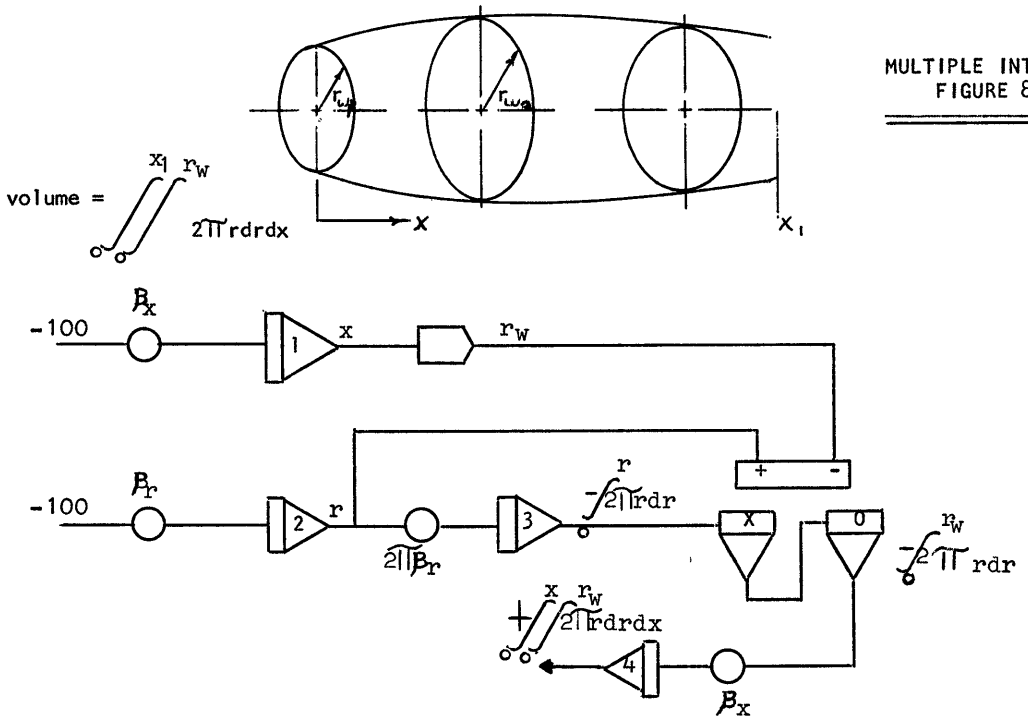
First, the equation for economic evaluation of the products and cost of operating variables under investigation is derived. This will probably contain definite integrals as in the case of the products from a fixed bed catalytic reactor, wherein the amount of catalyst is fixed, although the operating conditions such as temperature, pressure, and feed may be varied for optimization purposes. Second, the economic equation is differentiated with respect to the variables under investigation. Each derivation will provide a new and

independent optimization equation which will contain derivatives of the original definite integrals. The desired optimum is realized when the value of each derived equation in the matrix has a value of 0. The derivatives containing definite integrals from the second step are to be evaluated in repetitive operation by the use of X and 0 memories as indicated in the analysis of Figure 7. Once the successive derivatives have been evaluated by repetitive operation, they are employed with the necessary algebraic signs derived from the maximum or minimum nature of the calculations to drive NRO integrators whose outputs are arbitrarily equated to the respective operating conditions. The operating variables as produced from the NRO integrators are employed in feed-back loops for both the REPOP calculation and the simultaneous evaluation of the optimization derivatives. When the desired optima are reached, the NRO integrator drives will be zero, and the operating conditions will remain constant at their respective values. It is also quite feasible to place restrictions on the operating conditions by the employment of X and 0 memories. If any of the variables become restricted before their respective optima are attained, the remaining variables will still be optimized employing the restricted values for the pertinent variables.

A similar technique can be employed with repetitive operation evaluation of definite integrals and feed-back loops from non-repetitive operation to evaluate the design variables which minimize the transients in an instrumentation problem. Another example is the evaluation of the correlation coefficients which provide the best least mean squares fit of observed data.

MULTIPLE INTEGRATION

There are many instances where multiple integration requiring successive evaluations of a definite integral is a very complex procedure that necessitates analog computer techniques more advanced than those heretofore available. The fixed bed converter with axial and radial temperature gradients, as proposed by Wilhelm, is a classic example of a problem which can be solved by the multiple integration technique. Although this problem is too complex to detail in the current presentation, the philosophy involved in such a calculation can be summarized with a trivial case. The multiple integration to be described is presented in Figure 8, where it is desired to determine the volume of a cylinder with varying radius. This volume is the double integration of the



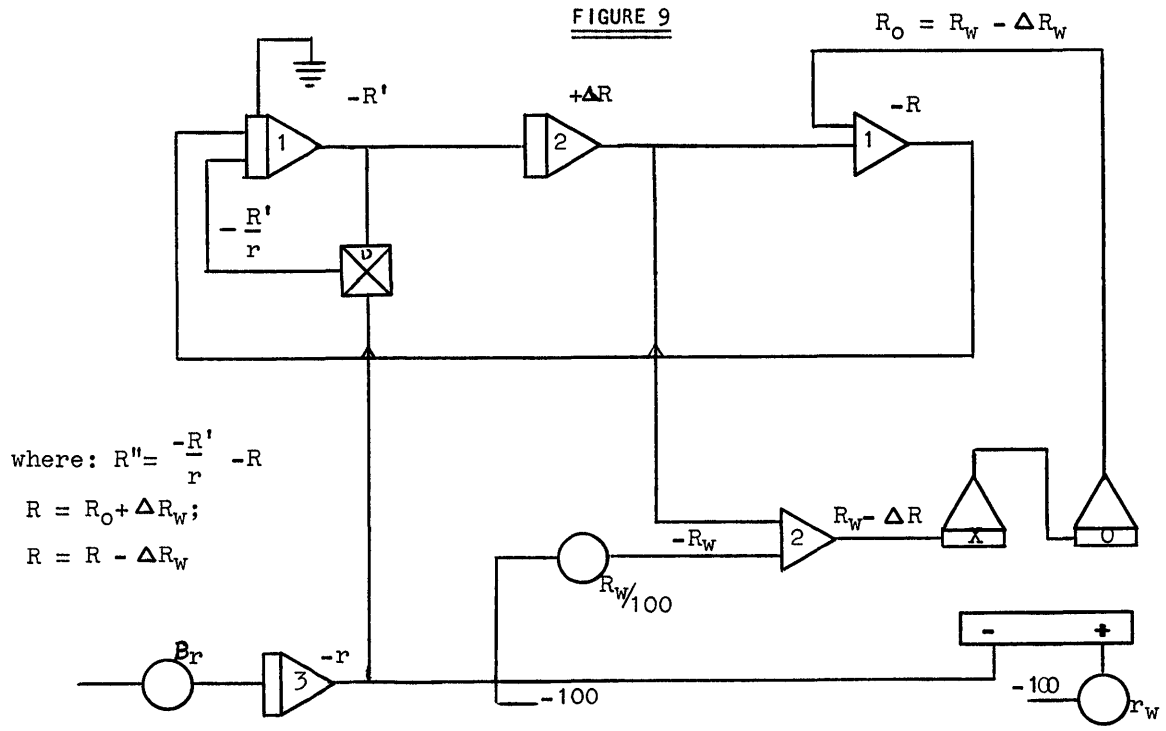
MULTIPLE INTEGRATION
FIGURE 8

radius and the length. The radius at the wall for any particular point along the axis is obtained from a diode function generator. The radial integrators operate in repetitive mode while the axial integrators operate in non-repetitive mode. The value of x is obtained from integrator 1 (non-repetitive) and fed to the DFG from which the instantaneous value of the radius at the wall, r_w, is obtained. r is obtained from the repetitive operating integrator 2 and is fed through a pot set of $2\pi\beta r$ to repetitive integrator 3 which gives the cross-sectional area at any point in a cycle. The upper limit of this integration is fixed by X & 0 memories. The X memory tracks until r is equal to the radius of the wall, at which point it locks. The 0 memory starts tracking when the comparator shows that the wall has been reached, and its output is always the previous value of the cross sectional area at the wall. This, in turn, is directed to integrator 4 operating non-repetitively. The output of this last integrator is the volume of the cylinder at any axial location, x.

While the previously described multiple integration is trivial, the same sequence is followed in developing the temperature gradients in a fixed bed chemical reactor operating under steady state conditions. Vaporized reactants are introduced at the inlet to the reactor, and the total reaction in the first increment of the reactor is integrated radially. The total heat released in the first increment is also calculated so that both the concentration of the reactants and their temperature can be calculated for the input to the second increment. As in the previous illustration, the radial increments of the fixed bed reactor are calculated in repetitive mode while the axial variation is again summed with non-repetitive integrators. The radial variation of both temperature and concentration of the individual reactants and products is presented continuously in the oscilloscope and can be plotted for any desired radial point as a function of x, or axial length, on an X-Y plotter. The increment in reactor length can be reduced for each cycle until further reductions, do not change the problem solution. Under these conditions the maximum computer accuracy has been achieved and the minimum solution time can be determined. In this case, a reactor solution time of 20 seconds with a frequency of 60 cps will provide 1200 increments, or slices, through the reactor bed and should be sufficient.

The solution procedure outlined cannot be obtained from conventional analog computers because the known radial boundary conditions are split and the radial function R must be solved automatically by the computer in repetitive operations. In this particular problem, the value of r (a function of radius, r) is known at the wall while R' is known at the center of the tube where the radius, r, is zero. This solution is readily accomplished in repetitive operations with the DYSTAC memories as shown in Figure 9. The solution technique reduces the split boundary value problem in R to an initial condition problem.

FIGURE 9



In Figure 9, integrators 1, 2 and 3 are all in the repetitive operations mode. Integrator 3 produces r, the independent variable which is divided into R' to furnish one of the feedback loops. The other feedback loop is provided by -R from summer 1. If R_0 were known the solution in the repetitive operating mode would be complete. An automated solution is obtained by observing that the value for R_w at the wall is equal to the initial condition value plus the integrated change in R at the wall, ΔR_w. In order to determine R_0, ΔR is continuously added to the appropriate value of R_w in summer 2. The output of this summer is not the required value of R_0, because there is no subscript w on the ΔR increment. This subscript is realized by the employment of X and 0 memory. The X memory tracks until the wall of the cylinder is reached, as determined by a comparison of R_w on a pot with the independent variable r. When this comparison is realized the X memory stops tracking, and is locked on the value which is then a trial value for R_0. The 0 memory is locked until a comparison is realized, at which time it tracks or learns the value in the X memory. In this manner, the previously determined trial value for R_0 is always available as the output of the 0 memory. In the trial and error solution sequence, some value for R_0 will be maintained as the initial condition throughout a cycle. From this value ΔR_w can be calculated at the wall, and a new value for R_0 will be assumed. The sequential assumptions will be followed until there is no further change with the value for R_w being precisely equal to the required value. The solution sequence is very rapid.

In the previously discussed application the independent variables in the problem have been separated by the assumptions on which the problem was hypothesized. In the more general case, the variables must be separated mathematically and frequently they cannot be separated at all. However, there are procedures where both of the latter cases can be generally handled satisfactorily with the DYSTAC Computer. In the first case, the variables are separated mathematically, and the multiple integration technique can be employed. In the other case a form of forced balance can be used for continuous self-solution of the problem's boundary conditions.

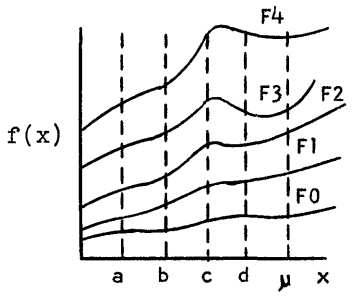
Variables can frequently be separated by a conventional substitution for the dependent variable of a product of two new dependent variables which are respectively functions only of the individual independent variables. Another means of seperation is the employment of Laplace transformations when one of the variables has been transformed, an ordinary differential in terms of the transformed variable is obtained. A solution of this ordinary equation will in turn yield an explicit evaluation of the transformed variable in terms of the transform s and the remaining independent variable. The final explicit equation can be solved using standard instrumentation techniques to evaluate the transformation operator in repetitive operation while employing values of the remaining independent variable generated from non-repetitive components.

CONTINUOUS MEMORY

Point memory and the ratchet circuits will solve many problems involving ordinary and partial differential equations; however, there is frequently a requirement for continuous memory of a varying dependent or independent function. Although there are several variations of the proposed continuous memory circuit, a simple illustration will serve to demonstrate its potential applications. The memory circuitry to be investigated is shown in Figure 10.

The most common requirement of continuous memory is illustrated by the variation of f(x) with the solution curves 1, 2, and 3. In this case the function F1 for f(x) is required as input to the computer circuit to predict F2. Solution F1 will be required for one full cycle of the repetitive operation during which the values of F2 for the subsequent operating cycle will be produced. In the subsequent operating cycle the values from F2 will be required to calculate another curve F3. The memory circuit is based upon Newton's Forward Interpolation Formula. This relationship states that if for equal increments of the dependent or independent variable x the values of f(x) are known, these values may be used to express f(x) as a function of x as shown in the polynomial equation. The values of the coefficients, A, B, C and D, for this equation are derived from the differences as shown in the example of a difference tabulation. Thus, if the values of $f(x_a)$, $f(x_b)$, and $f(x_c)$, and $f(x_d)$ are known, they may be directed to a summing network from which the coefficients, A, B, C and D, can be derived. Simultaneously, the value of x may be sent to a function network which produces the required products of x, x-1, and x-2, that are necessary for Newton's Formula.

One simple scheme whereby the required values of f(x) for curve 1 may be retained in memory while new values for curve 2 are learned for the subsequent cycle is shown in the four pairs of X and 0 memories. During the first cycle the values of $f(x_a)$, $f(x_b)$, $f(x_c)$ and $f(x_d)$ for curve F0 are present at the outputs of the respective 0 memories; these values are then fed to the required summing network that produces the coefficients, A, B, C and D, which are used in the development of the curve F1 as x varies. At the same time the respective X memories have relay comparators which open the first memory when x is equal to a, the second memory when x is equal to b, the third memory when x is equal to c, and the fourth memory when x is equal to d. Since the X memories are fed by the computer with the newly calculated durve F1, at the end of the cycle the values of $f(x_a)$, $f(x_b)$, $f(x_c)$, and $f(x_d)$ for curve F1 are locked in the respective X memories. It is necessary to translate these values into the 0 memories for the subsequent cycle, and this is done at the end of the current cycle. This is accomplished by one common comparator circuit that switches all four 0 memories when the value of x is equal to μ which is a value greater than x_d. This comparator operates the



$$f(x) = A + B(x) + \frac{C(x)(x-1)}{2!} + \frac{D}{3!}(x)(x-1)(x-2)$$

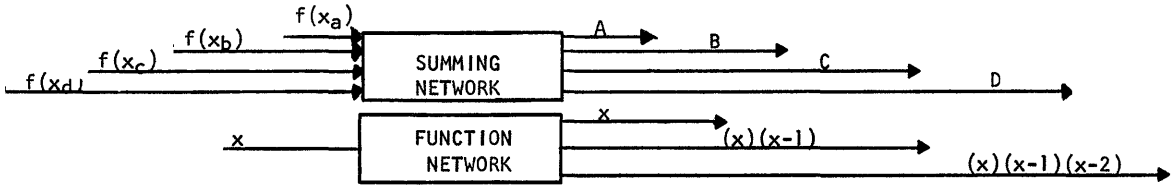
x	f(x)	Δ	Δ^2	Δ^3
a	2	10	3	-1
b	12	13	2	
c	25	15		
d	40			

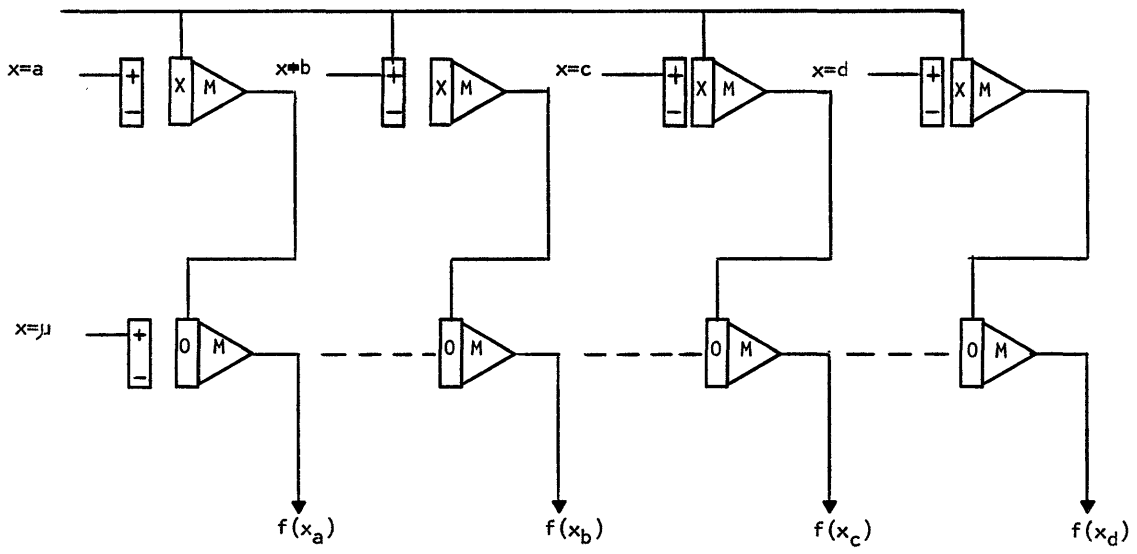
where: $f(a) = 2 = A$

$\Delta a = 10 = B$

$\Delta^2 a = 3 = C$

$\Delta^3 a = 1 = D$





CONTINUOUS MEMORY CIRCUITRY
FIGURE 10

electronic reset switches and allows each one of the 0 memories to learn the new value of the corresponding X memory. Then the values of the desired coefficients for the F1 curve are available to produce curve F1 through Newton's Formula during the subsequent cycle from which the solution F2 can be obtained. In a similar manner, solution F3 is obtained from F2, etc.

In general, where three increments or four points are utilized in the Difference Formula, a third degree approximation of the desired function can be derived. If more accurate approximations are required, fourth, fifth, sixth, and higher degree polynomials can be evaluated by the inclusion of more memories and higher order differences.

When continuous memory as previously discussed is incorporated in a calculation, the parameter for the curves F_0, F_1, F_2, F_3 and F_4 may be a time increment which leads directly to transient problem solution. The increment might also be a theoretical stage in a stagewise operation and in this manner difference-differential equations can be solved. Many other applications can be visualized for continuous memory.

OTHER APPLICATIONS

Although the details will not be discussed at the present time, an all electronic transport delay can be developed from the DYSTAC memories. This delay uses memories in a bucket brigade circuit with any desired accuracy of interpolation based on Newton's Divided Difference Formula or any other interpolation technique that is desired. Any degree of accuracy may be obtained with this transport delay by balancing the number of sample points and the interpolation. Its accuracy is far in excess of that available from the Pade approximation.

The infinitesimal delays which can be incorporated into X and 0 memories can also be employed to differentiate and these differentiations are relatively noise free. In this manner, many problems which have not been practical heretofore because they required differentiation can now be successfully analyzed with an analog computer.

It may be concluded that the DYSTAC Computer has provided rapid economic solutions to problems which have heretofore been so tedious or expensive to resolve that either gross approximations have been incorporated into the solution or else the problem has been neglected.

(022460 s)

RECOGNITION OF SLOPPY, HAND-PRINTED CHARACTERS*

Worthie Doyle

Staff Member
Lincoln Laboratory
Massachusetts Institute of Technology
Lexington, Massachusetts

Summary

This report describes a pattern recognition scheme particularly intended to handle noisy and highly distorted data. The sample is subjected to a set of tests at the conclusion of which a single decision is made. The decision is made on the basis of experience obtained from prior processing of labelled samples. The method has been applied to hand-printed English capitals but is evidently general. Results are given for some trials made on the IBM 709.

1. Introduction

In any recognition scheme the raw data are transformed or tested and from the results decisions are made. Usually[1, 2] the process consists of a sequence of tests together with rules for branching after each test, the final branch furnishing the decision. Any recognition method employing such a sequence of decisions to reach an ultimate verdict is evidently limited by the weakness of its worst tests. Tests which one might off hand think infallible are easily beaten by malevolent nature. For example, in the course of devising tests for character recognition a good datum to consider seemed to be the maximum number of intersections of the character by a horizontal straight line. Most M's were expected to have four; of the 46 samples at hand 11 did. Our noisy data also included two each of R and T and one each of A and N having four. Unless the tests are good indeed or the samples to be recognized quite free of noise and distortion a parallel processing technique seems preferable.

With parallel processing all tests are made before any decision is undertaken. Though the individual tests may be poor a reliable decision can be available provided there are enough tests, each contributing some different fractional bit of information. Such a set of tests may be considered a code with redundancy, an appropriate counter to noise and distortion. The method here used to combine the test results resembles correlation detection of electrical signals and is an elementary realization of Pandemonium[3].

Another point sometimes overlooked in applying tests to data from nature, as opposed to ideal representatives, is that the results, while still significant, are not what consideration of the ideal representatives might have led one to expect. Thus the action to be taken following a particular test should be determined by observing the results of testing real known samples from the population of characters one eventually hopes to be able to recognize rather than by applying preconceived rules. Such a "learning" process seems to be present in only two schemes so far reported[4, 5]. Some preconception is unavoidable but is here present only in the choice of the data reduction mechanism and not in the decision making process itself.

The scheme to be described incorporates both these notions -- decision reserved until all test results are in, and discrimination based on "learning" from real data.

2. General Description of Character Recognition Scheme

This section describes what has been programmed and tried. Some reasons for these choices and against certain others are given in section 5, where some planned improvements are also noted.

The present organization is depicted in figure 1. The box "Process" first does some minor filling and deleting on the original pattern (details in Appendix A), hopefully a noise-reduction measure, and then produces a sequence of symbols representing the outcomes of various tests* applied

* The work reported in this paper was performed by Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology with the joint support of the U.S. Army, Navy and Air Force.

* The tests used for trials on hand-printed English are listed in Appendix B.

to each sample. For example, one already mentioned test on characters produces as outcome the maximum number of intersections of the character by a horizontal straight line. Thus for most O's this test produces the symbol 2. The results of this data reduction are collected in a table, the first census, which contains a tally of the number of occurrences of each test outcome for each pattern. A section of this census corresponding to application of the test just mentioned to ten characters looks like figure 2. (This census is for about half the population). In the learning phase such data are collected for each test and are organized in the hierarchy - Test, Character, Outcome, Number. After all the labelled samples have been processed the census is re-organized in the order - Test, Outcome, Character, Number; finally the observed incidences of outcomes are combined with assumed a priori probabilities of occurrence for each character to calculate the inverse probabilities for each character given test and outcome. Effectively these probabilities form the block labelled "Census".

In the working phase the block "Process" is identical to the corresponding one in the learning phase and does exactly the same data reduction job. "Compare with experience" amounts to using the observed outcomes of the tests on unknowns to look up and extract pertinent portions of the second census for use by the next block.

The block "Decide" forms weighted averages of the inverse probabilities corresponding to the observed outcomes from the separate tests and indicates which character corresponds to the highest of these averages of a posteriori probabilities. This is our guess.

3. Hand-Printed English (Problem)

The particular restrictions which distinguish the example tried were chiefly determined by cost and convenience. Herb Sherman, for work reported in (1), had already collected several hundred samples by having visitors to Lincoln Lab print their names and addresses in a series of quarter inch squares. Some of these samples had been quantized in a 32 x 32 array of bits and these characters, totalling 329 samples, were kindly supplied on IBM cards by Mrs. Betty Howell. Later a further 600 were made available on flexowriter tape prepared on TX-2 by Phil Peterson.* Since the trials were to be made on the IBM 709 this 32 x 32 array was augmented by a double row of zeros all around, it being convenient to store the raw bit patterns as 36 consecutive machine words. This choice of format means that fine detail will be lost, which in turn imposes a restriction on the size of characters to be processed -- they should not be so small that quantizing into a 32 x 32 array destroys essential information. Aside from this, distortion and some tilt are allowed as well as noise in the form of stray bits and holes. The characters may be anywhere in the field. To appreciate the nature of the data used consult figures 3 and 4 which reproduce most of the A's and R's involved in our trials.

Finally because of the general paucity of data only the letters A E I L M N O R S T were used for the trials, those letters accounting for about two thirds of the total samples. This circumstance makes recognition easier, of course, so that fewer and cruder tests could be used in the trials. Very little thought was given to the exact nature of the tests except that they were required to be easily programmed and fast running, a supposed advantage of Pandemonium being its ability to get along with relatively poor tests.

4. Hand-Printed English (Results)

For a trial of the scheme on the 10 characters the stock was shuffled and split into two parts, each of which contained about half of the samples for each character. One half of the samples were fed to tne learning phase and the other half used as unknowns. The roles of the two halves were then interchanged for a second trial, so that each half was recognized using the experience derived from testing the other half. The results of such a pair of trials appear in figure 5. The diagonal terms are observed frequencies of correct recognition, while an off-diagonal term is the frequency with which a character labelled as at left was incorrectly guessed to be the character at top. The "fraction correct" is the average of the diagonals.

In addition, several sets of weights were tried for the tests used, including omission of sets of similar tests. With some omissions the error rate increased as much as 5%, though generally the process seems insensitive to the adjustment of test weights, indicating a fair degree of information overlap.

5. Discussion

Most of the visible defects of the process just described are due to the manner of laying up the experience derived during the learning phase and of referring to it when identifying an unknown. Ideally, when testing a sample the results of all the separate tests should be retained in combination and the experience stored as if there were a single grand test whose outcome is a vector composed of the outcomes of the several elementary tests.* However, such a procedure would require a considerably greater amount of storage space for experience if any reasonably large sample of a

---

* The messy job of providing these last with identifying labels was cheerfully done by Martha Evens and Connie McElwain, while Barbara Lucy progrmmed the transformation from tape format to 36 word bit-matrices.

* Unpublished note of Les Wilson.

population were presented for learning. This consideration was responsible for the original decision to throw away the information provided by the joint occurrence of test results. If it were possible to impose a measure of sameness on the space of test results then the storage capacity required for experience would be vastly reduced. Without some means of deciding degree of likeness between literally different test outcomes it is necessary to store a complete test outcome vector for each different combination of outcomes that occurs. The procedure described in section 2 stores the separate outcomes in isolation, so that storage requirements grow proportionately to the number of outcomes for each of t tests rather than as the t-th power of the number of outcomes.

Some intelligent compromises should permit retaining most of the information inherent in the joint outcomes without undue increase in storage requirements and two such compromises are going to be tested.

First of all, it is possible simply to combine a small number of the original tests into a single larger test whose outcome is the sequence of symbols resulting from mere stringing together, in preassigned positions, of the outcome symbols from the separate tests. If the groups thus combined consist of blocks of tests which, as blocks, are roughly independent of each other then most of the information of a single grand test will still be retained. This approach is almost ready for trial with one reasonable grouping of our present 28 tests into 10 combinations and will be reported on later.

Second, certain of the tests, for example those for cavities, may be so designed as to have a natural measure of distance between results. For these the experience may be stored in a more compact form and a mixed look-up procedure used in the block " compare with experience " . This would produce not only savings in data storage, but also an ability to make decisions even when test results on unknowns differ from all the results obtained on the labelled samples during the learning phase.

Appendix A

The Filtered Image in Storage

The bit matrix originally presented is slightly modified to fill small holes and eliminate isolated bits and pieces. Both the filling and elimination rules used are rudimentary and depend upon examination of the 3 x 3 array whose center bit is under consideration. Rules are given in figure 6, the deletion rule being strictly from Unger[2]. The fill rule has been applied first since our data appeared to suffer more from gaps than extra spots.

The modified image is then supplemented by three rotated replicas, each of 36 machine words, which represent views from the other three sides. Thus most of the basic tests yield four separate pieces of information (sometimes two). It is these modified images which are actually tested. They are referred to as the North, West, South, and East views in appendix B, where the tests used are listed.

Appendix B

Tests Used in Trial

Each test was given a six character name and the tests are here listed under these names. The outcome of each test is a string of at most 12 digits from the range 0 to 7 inclusive. There is no necessity for these choices; they happen to be convenient when trials are to be made on a general purpose binary calculating machine like the IBM 709. The description of each test states the rule for determining this sequence of symbols, starting from the filled and deleted bit pattern. To help this description a typical character is reproduced in figure 7. The result given under each test is for this A.

RHCSEQ 1212
The sequence composed of the numbers of intersections of the character by a horizontal line, scanning from top to bottom, after adjacent repetitions have been eliminated.

RVCSEQ 1212321
As above, but going cross-wise.

VERUNS 1212
The sequence composed of the relative lengths of the runs of each section of RHCSEQ. Let H, W be height and width of the character and [n] denote greatest integer not exceeding n. Then each digit is given by [4 x RUN/H] + 1.

HORUNS 1111122
As above, but for the runs leading to RVCSEQ.

HOMXSC 2
The biggest digit in RHCSEQ.

VEMXSC 3
The biggest digit in RVCSEQ.

VERSTR 2 HORSTR 2
Take the number of blocks of bits in each horizontal (vertical) slice of the array. Add these numbers and normalize by height (width).

The next block of tests contains four of each type, corresponding to views from north, east, south, and west. The results from the sample are given in that order, following the test' s root name.

BOTSG 1 2 2 1
That number of intersections which first occurs for a significant (now 1/5 H or W) number of slices as one moves in from the side in question.

BEN 11 10 11 21
Work in from a side until you encounter a slice with more than one block of ones. Starting with this slice, OR together with it each succeeding slice. After each new slice is OR-ed in count the number of blocks of ones. When the count decreases stop. Output of the test is a pair of digits A, B. The first, A, is the last count of blocks. The second, B, is 4 times the number of steps before stopping divided by the height or width, whichever is appropriate.

EDGE 1 3 1 3
Moving in from an edge, the first three nonempty slices are OR-ed together and the number

of bits in the rightmost block of ones of this word is compared with the height or width. Output is [4 x BITS/H (or W) + .5].

BOT 121 220 220 121

Work in from a side and at each slice 1) count the number of blocks of ones in this slice, 2) OR this slice into a word consisting of the union of all previously processed slices and 3) keep count of the number of slices processed. When the result of 1) exceeds 1 (or you run out of slices) stop. Output of this test is three digits A, B, C:

A. count of the number of blocks of ones in the union of slices
B. number of blocks of ones in the slice where you stopped
C. the count in 3) normalized to height or width:
[4 x COUNT/H (or W) + .5].

CAV 1 1 2 1

Work in from a side and form union of first non-empty slice and its successor. Then alternately adjoin to this union successive slices and count the blocks of ones in the union. Stop when this block count decreases or you are halfway through the character. Output is the penultimate block count.

The following two tests form useful combinations of the EDGE test results, recovering some previously discarded information.

NO/SOU 3

Form the ratio of the outputs of NEDGE and SEDGE. Consult a list of thresholds to derive a digit from 1 to 5 as test output. 5 means "very big", 3 "about the same" and so on.

EA/WES 3

Same as above, but for EEDGE and WEDGE.

Acknowledgement

The author wishes to thank Ben Gold and Oliver Selfridge for a lot of stimulating conversation and many useful suggestions.

References

(1) H. Sherman, "A Quasi-Topological Method for Machine Recognition of Line Patterns", to appear in Proc. of International Conference on Information Processing about Jan. 1960.

(2) S.H. Unger, "Pattern Detection and Recognition", Proc. of IRE, Oct. 1959, p. 1737.

(3) O.G. Selfridge, "Pandemonium: A Paradigm for Learning", paper presented at Symposium on Mechanization of Thought Processes, National Physical Laboratory, Teddington, Middlesex, England, Nov. 27, 1958.

(4) Grimsdale et al., "A System for the Automatic Recognition of Patterns", Proc. I.E.E., paper No. 2792M, Dec. 1958, p. 210.

(5) W.W. Bledsoe and I. Browning, "Pattern Recognition and Reading by Machine", Eastern Joint Computer Conference, Dec. 1959.

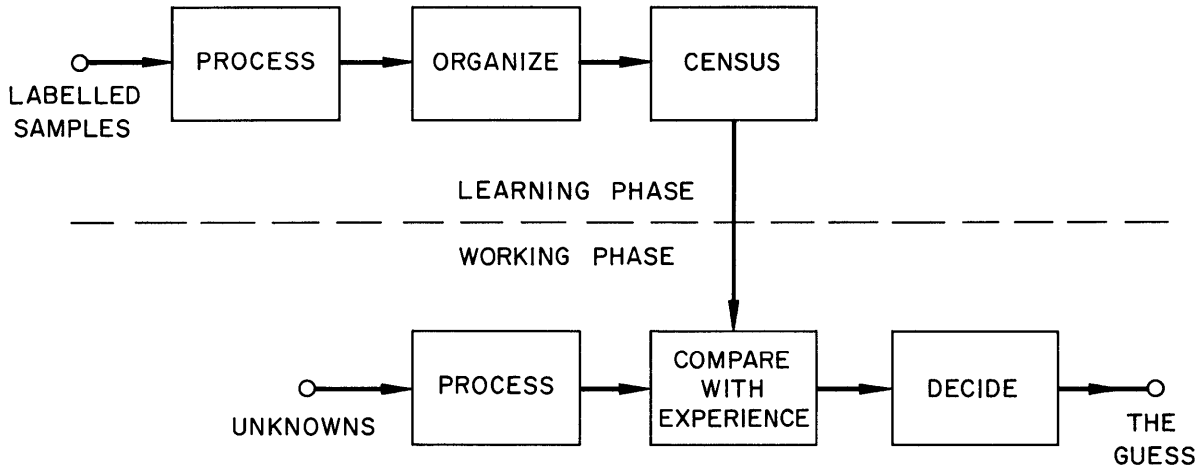


Figure 1
ORGANIZATION OF PATTERN RECOGNITION SCHEME

Outcome	A	E	I	L	M	N	O	R	S	T
1		3	24	8					11	
2	29	34	1	14	1	2	28	21	26	24
3	9	9		1	12	26	6	10	1	8
4	1				10	1		2		2
5					1					

Figure 2
PORTION OF FIRST CENSUS

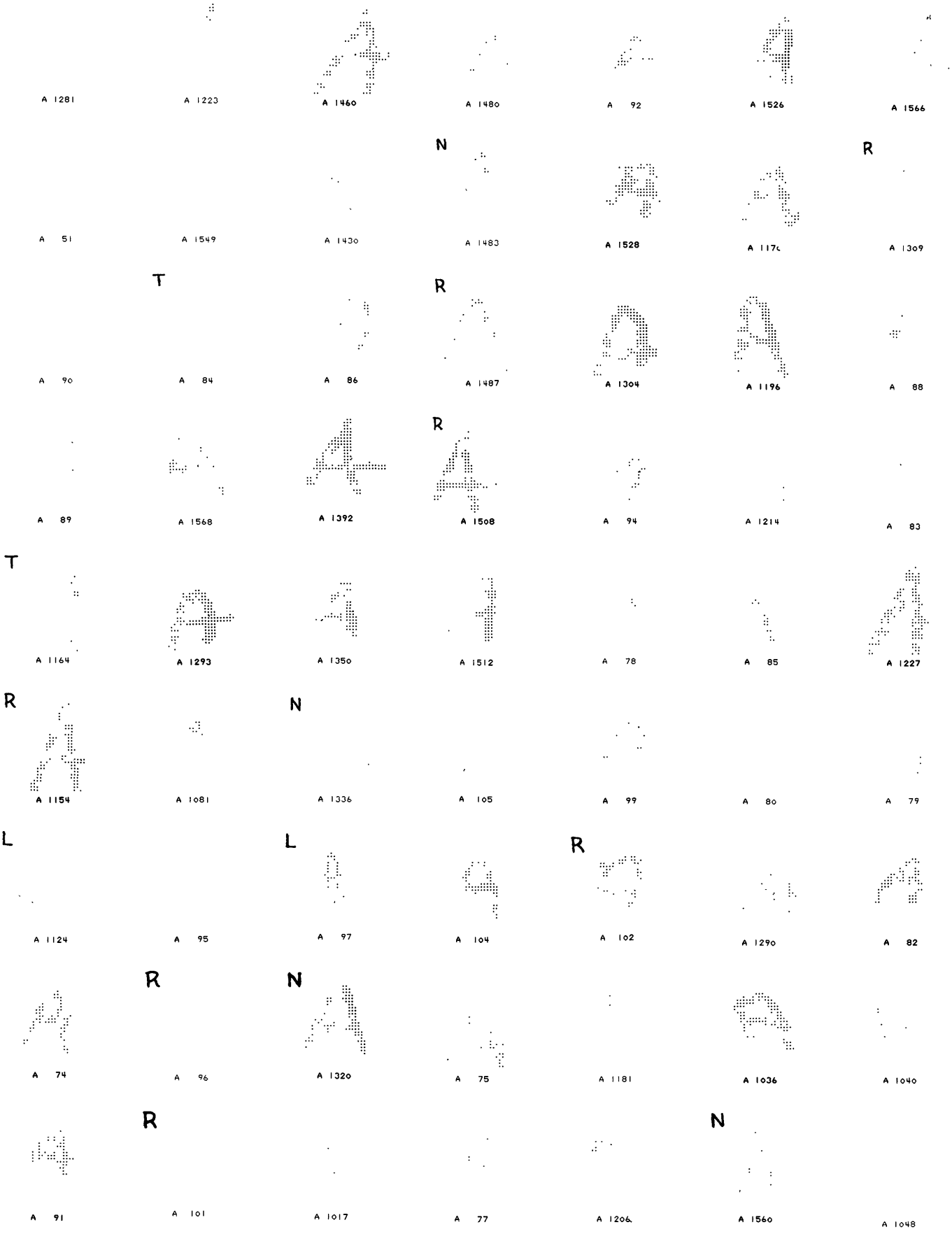


Figure 3
SOME A's
LABELS AT UPPER LEFT ARE THE ERRONEOUS IDENTIFICATIONS

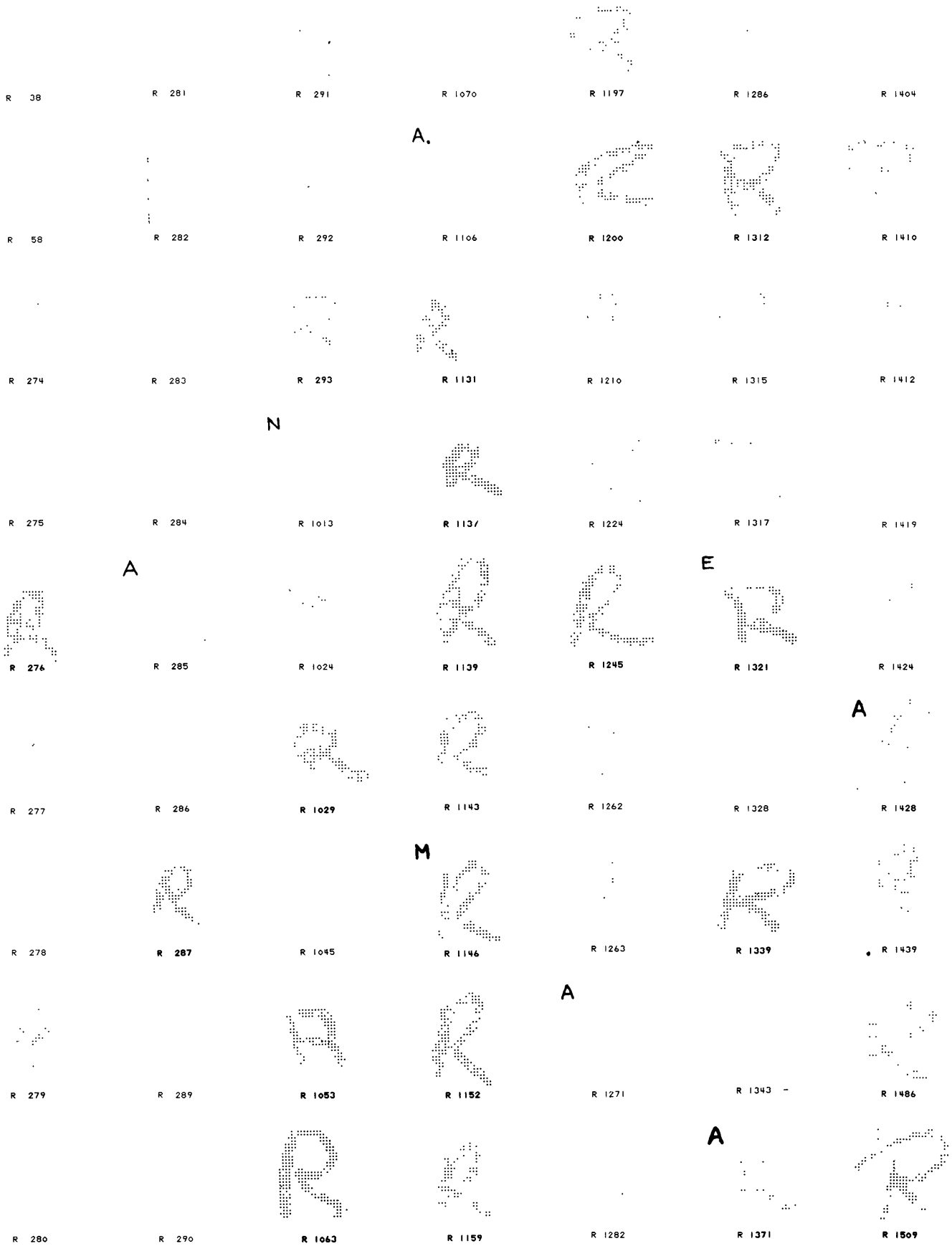


Figure 4
SOME R's
LABELS AT UPPER LEFT ARE THE ERRONEOUS IDENTIFICATIONS

	A	E	I	L	M	N	O	R	S	T
A	.72		.03		.06	.03		.17		
E		.95							.02	.02
I			.96						.05	
L			.18	.82						
M					.96	.05				
N					.08	.89				
O					.03	.07	.87	.03		
R	.03				.03	.03		.90		
S		.03	.20						.74	.03
T			.05							.95

about .875 right

	A	E	I	L	M	N	O	R	S	T
A	.74			.03	.03	.13		.08		
E		.94	.02					.02	.02	
I			.92	.04					.04	
L			.13	.79						.08
M			.04		.83	.13				
N					.18	.79	.04			
O							1.00			
R	.09	.06			.03			.79		.03
S		.03							.95	.03
T			.03							.97

about .872 right

Figure 5
CONFUSION MATRICES (ABOUT ".875 RIGHT" AND ABOUT ".872 RIGHT")

a b c

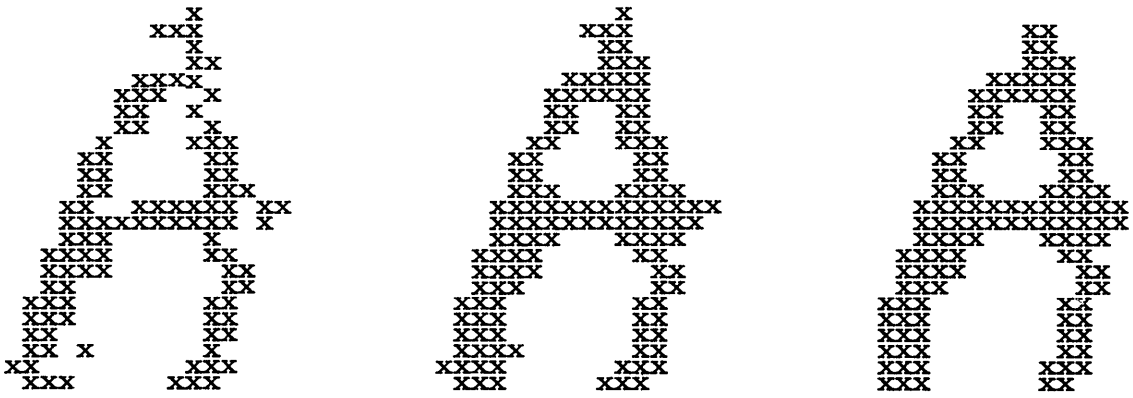
d e f

g h i

Fill: $e' = e + a \cdot i + b \cdot h + c \cdot g + d \cdot f$

Delete: $e' = e \cdot (b + c + f) \cdot (d + g + h) + e \cdot (d + a + b) \cdot (f + i + h)$

Figure 6
RULES USED FOR IMAGE FILTERING



Original | After Filling | After Deleting

NO/SOU	3	HORSTR	2	NORBOT	121
HOMXSC	2	WBOTSG	1	EASBOT	220
VERUNS	1212	EBOTSG	2	S EDGE	1
VERSTR	2	SOUBEN	11	W EDGE	3
SBOTSG	2	WESBEN	21	N EDGE	1
NBOTSG	1	NORBEN	11	E EDGE	3
EA/WES	3	EASBEN	10	SO CAV	2
VEMXSC	3	SOUBOT	220	WE CAV	1
HORUNS	1111122	WESBOT	121	NO CAV	1
RHCSEQ	1212	RVCSEQ	1212321	EA CAV	1

Figure 7
SAMPLE CHARACTER WITH TEST OUTCOMES

EMPIRICAL EXPLORATIONS OF THE GEOMETRY THEOREM MACHINE *

H. Gelernter, J. R. Hansen, and D. W. Loveland
IBM Research Center
Yorktown Heights, New York

Introduction

In early spring, 1959, an IBM 704 computer, with the assistance of a program comprising some 20 000 individual instructions, proved its first theorem in elementary Euclidean plane geometry [1]. Since that time, the geometry theorem-proving machine (a particular state configuration of the IBM 704 specified by the afore-mentioned machine code) has found solutions to a large number of problems[2] taken from high school textbooks and final examinations in plane geometry. Some of these problems would be considered quite difficult by the average high school student. In fact, it is doubtful whether any but the brightest students could have produced a solution for any of the latter group when granted the same amount of prior "training" afforded the geometry machine (i.e., the same vocabulary of geometric concepts and the same stock of previously proved theorems).

The research project which had as its consequence the geometry theorem-proving machine was motivated by the desire to learn ways to use modern high-speed digital computers for the solution of a new and difficult class of problems; a class heretofore considered to be beyond the capabilities of a finite state automaton. In particular, we wished to make our computer perform tasks which are generally considered to require the intervention of human intelligence and ingenuity for their successful completion. The reasons behind our choice of theorem-proving in geometry as a representative task are set forth in detail in an earlier paper[3]. We only remark here that problem-solving in geometry satisfies our definition of an intellectual activity, while being at the same time especially well suited to the approach we wished to explore. The fact that geometry is decideable is irrelevant for the purpose of our investigation. The methods employed by the machine are suitable as well for the proof of theorems in systems for which no decision algorithm can exist.

*This report is a summary of the full paper to be submitted for publication in an appropriate journal.

We shall not labor the question as to whether our machine is indeed behaving intelligently in performing a task for which humans are credited with intelligence. The psychologists offer us neither aid nor comfort here; they have yet to satisfactorily characterize such behavior in humans, and have rarely considered the abstract concept of intelligence independent of its agent. In the final analysis, people are occasionally observed to do things that may best be described as intelligent, however vague the connotations of the word. These are, in general, tasks involving highly complex decision processes in a potentially infinite and uncontrollable environment. We should be most happy to have our machine duplicate this kind of behavior, whatever label is affixed to it.

Heuristic Programming and the Geometry Machine

The geometry machine is able to discover proofs for a significant number of interesting theorems within the domain of its ad hoc formal system (comprising theorems on parallel lines, congruence, and equality and inequality of segments and angles) without resorting to a decision algorithm or exhaustive enumeration of possible proof sequences. Instead, the theorem-proving program relies upon heuristic methods to restrain it from generating proof sequences that do not have a high a priori probability of leading to a proof for the theorem in question.

The general problem of heuristic programming has been discussed by Minsky[4] and Newell, Shaw, and Simon[5]. The particular approach pursued by the authors has been described at length in the papers to which we have already referred [1, 3]. We shall therefore defer to the presentation of the machine's detailed results in the full paper summarized here for a description of how these results were achieved. It should be recorded here, however, that the geometry machine operates principally in the analytic mode (reasoning backwards). At each stage of the search for a proof, a goal exists which must be "connected" with the premises for the problem by a bridge of axioms and previously established theorems or lemmas. If the connection cannot be made directly, then a set of "subgoals" is generated and the process is repeated

for one of the subgoals. Heuristic rules are used to reject subgoals that are not likely to prove useful, to select one from those remaining to work on, and to choose particular axioms and theorems to use in generating new subgoals. The machine does depart from this procedure in a number of circumstances (in setting up an indirect proof, for example), but these cases account for only a small fraction of the total search time.

The computer program itself was written within the framework of the so-called Newell-Shaw-Simon list memory[6]. In order to ease the task of writing so massive and complex a machine code, a convenient special-purpose list-processing language was designed to be compiled by the already available FORTRAN system for the IBM 704 computer[7]. The authors feel that had they not made free use of an intermediate programming language, it is likely that the geometry program could not have been completed.

Summary of Results

Since its initial solo performance, the geometry machine has existed in several different configurations. In its earliest and most primitive form, the system was equipped with a single major semantic heuristic[8]. That first system was, however, able to prove a large number of interesting, though admittedly simple theorems in elementary plane geometry[9]. The heuristic rule in question, which is independent of the particular formal system under consideration, may be described in the following way. All subgoal formulae that are generated at a given stage of the proof-search are interpreted in a model of the formal system; in our case, the model is a diagram, a formal semantic interpretation. If the interpreted subgoal is valid in the diagram, it is accepted as a possible step in the proof, provided that it is non-circular[10]. Otherwise, it is rejected.

As an experiment, a number of attempts were made to prove extremely simple theorems with the latter heuristic "disconnected" from the system (i.e., all non-circular subgoals generated were accepted). In each case, the computer's entire stock of available storage space was quickly exhausted by the initial several hundreds of first level subgoals generated, and, in fact, the machine never finished generating a complete set of first level subgoals. We estimate conservatively that on the average, a number of the order of 1 000 subgoals are generated per stage by the decoupled system. If one compares the latter figure with the average of 5 subgoals per stage accepted when the diagram is consulted by the machine, it is easy to see that the use of a diagram is crucial for our system. (Note that the total number of subgoals appearing on the problem-solving graph grows exponentially with the number accepted per stage.)

Since the procedure described above is a heuristic one, errors are occasionally made in the selection or rejection of formulae as subgoals. The diagram is made available to the machine in coordinate representation to finite precision. Formulae are interpreted by transforming them into an appropriate calculation on the numerical coordinates representing the point variables. For example, to check the validity of a statement concerning the equality of two segments, the length of each segment in the figure is calculated, and they are then compared to a certain preassigned number of decimal places. If, instead, the statement concerned parallel segments, the slopes would be calculated and compared. In a small number of cases, roundoff error has propagated beyond the allowed value, so that valid subgoals were rejected, or invalid ones accepted. It is important to point out, however, that in no case could this effect result in a false proof. Where valid subgoals were rejected, the machine found alternate paths to the solution. Where invalid ones were accepted, the machine failed, of course, to establish them within the formal system. In the worse possible case, the interpretation error could prevent the computer from finding any solution at all, but never could it lead to an invalid proof.

It should be clear at this point that the diagram is used only to guide the search for a proof by supplying yes or no answers to questions of the form: "Is segment AB equal to segment CD in the figure?", or "Is angle ABC a right angle in the figure?". There is no direct link between the diagram and the formal system in the geometry machine. The behavior of the machine would not be changed if the coordinate representation were replaced by a device that could draw figures on paper and scan them.

In the basic theorem-proving system described above, after a set of subgoals has been generated, each member of the set is explored in order. The next subgoal in line is not examined until the one preceding it has been followed down to a dead end. Too, in generating the next level for a given subgoal, every applicable theorem available is pressed into service.

This system was soon extended by the introduction of selection heuristics for both subgoals and subgoal-generating theorems. The subgoal selection heuristic assigns a "distance" between each subgoal string and the set of

premises in a vaguely-defined ad hoc formula space. At each stage, the next subgoal selected is that which is "closest" to the premises in formula space. The generator selection routine recognizes certain classes of subgoals that are usually established in one step. For such "urgent" subgoals, the appropriate generator is withdrawn immediately, and an attempt is made for a one-step proof (of that particular subgoal) before generating the full set for that formula.

The extended system is able to prove a number of somewhat more difficult theorems that are beyond the capacity of the basic machine[11]. For those problems within the range of both systems, the former is, on the average, about three times faster, and generates about two-thirds the total number of subgoals in half as many subgoal generation cycles as required by the basic system. The average depth of the problem-solving graph for the refined system, about seven to nine levels, is two-thirds the average depth for the basic system.

By the addition of a simple construction routine, the theorem-proving power of the machine is expanded to include an entirely new class of problem, hitherto logically unattainable. The routine, called upon only when all other attempts have failed, allows the machine to join two previously unconnected points in the diagram, and extends the newly-created segment to its intersections with all other segments in the figure. The new segment, when it intersects previously given ones, introduces new points into the problem which are named by the machine and become part of the problem system.

At this stage in its development, the geometry machine was capable of producing proofs that were quite impressive (Appendix I).* Its performance, however, fell off rapidly as the number of points in the diagram increased. This effect was due largely to the fact that unlike humans, who generally identify angles visually by their vertices and rays, the computer specifies an angle by a predicate on three variables, the vertex and a point on each ray. Consequently, the equality of angles 1 and 2 in figure 1 below may be represented in thirty-six different ways, since each angle has six different names. Formal rigor demands, too, that the equality of angles ADH and EDG, for example, be proved rather than taken for granted. It should be clear that where the condition above exists, the search for a proof quickly bogs down in a mass of uninteresting detail.

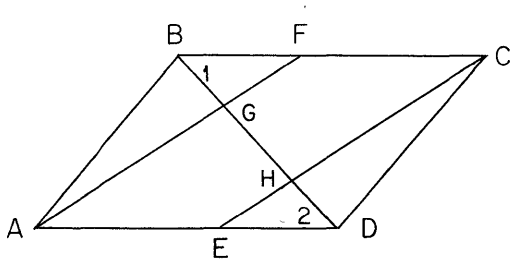


Figure 1

In the current system, the angle problem is solved by allowing the machine to use the diagram to identify a given angle with its full set of names, and to assume the equality relationship between different names for the same angle, as does its human counterpart. The geometry machine in its present configuration is able to find proofs for theorems of the order of difficulty represented by the following:

> THEOREM: If the segment joining the midpoints of the diagonals of a trapezoid is extended to intersect a side of the trapezoid, it bisects that side (Appendix II).

Limitations of the System

It will be immediately evident to those familiar with the properties of formal logistic systems that unless a construction which generates a new point is introduced by the machine, all problems are solved within the framework

---

*In the proofs appended to this paper, the non-obvious predicates have the following interpretations:

OPP-SIDE XYUV	Points X and Y are on opposite sides of the line through points U and V.
SAME-SIDE XYUV	Points X and Y are on the same side of the line through points U and V.
PRECEDES XYZ	Points, X, Y, and Z are collinear in that order
COLLINEAR XYZ	Points X, Y, and Z are collinear.

of a propositional calculus, however complex its structure. Although the machine's present construction routine can and does generate new points, we could not expect our results to be of great interest to logicians until a full set of possible constructions (corresponding to a complete set of existentially quantified axioms) is made available to the system to abet its search for a proof.

An equally serious limitation on the formal generality of the theorem-proving machine is imposed by our method for determining the well-formedness of strings within the logical system. In order to attain the necessary speed and efficiency in processing, well-formed formulae are defined by schema rather than recursively. The kind of statement that can be made in the system is then determined by the schema available to the machine. The practical effect of this loss in generality is to restrict rather severely the freedom with which algebraic statements in geometry may be manipulated.

In addition to the above, there are a number of non-essential bounds on the theorem-proving ability of the machine. These are a consequence of the limited speed and memory capacity of the computer for problems of such highly combinatorial character. Improvements in either of the above will be immediately effective in extending the class of machine-solvable problems in both quantity and difficulty.

Conclusion

The initial goal of our research program in machine intelligence has been attained. If the interrogator were to restrict his probing to the area of theorem-proving in elementary Euclidean plane geometry, our machine could be expected to give an excellent account of itself in competition with a human in Turing's well-known "imitation game"[12]. Of course there are many other problem areas (solving arithmetic problems, for example) where computers have always been able to compete successfully with humans. The significant point is that a knowledgeable interrogator would certainly avoid such areas in his questioning, while he might well (until now, at any rate) introduce a plane geometry problem in a calculated attempt to separate the men from the machines[13]. Although the stage is now set for the argument that any distinct area of human intellectual activity will in the same way succumb to the inexorable logic of electrons, switches, and gates, we defer to our philosopher colleagues for debate on the implications of that contention, at least until the time that computers have been programmed to consider such issues.

There are a number of consequences of our work that are, fortunately, more concrete than that alluded to above. Perhaps the most important are those relating to inferential analysis, a new branch of applied logic first characterized by Wang[14]. Inferential analysis "treats proofs as numerical analysis does calculations", and is expected to "lead to mechanical checks of new mathematical results" and, more important, "lead to proofs of difficult new theorems by machine". It is expected that our techniques for the manipulation and efficient search of problem solving trees and our results concerning syntactic symmetry[10] will prove to be useful tools in pursuing the goals of inferential analysis.

Contributions have been made, too, in the area of techniques for computer implementation of complex information processes. Results pertaining to the design and use of intermediate languages for the specification of list manipulation processes have been reported elsewhere[7]. The latter work indicates clearly the requirements of a digital computer system designed for optimum execution of such list processes. In brief, a list-processing computer should possess hardware facilities for:

1) Generalized indirect addressing; specified in the indirectly addressed instruction to arbitrary depth and in arbitrary order from either the left or the right field of a two-address register,

2) Effective address recovery; making available the terminal content of the address register (the final address in a long and complex indirect address chain, for example) as the address field for a subsequent operation,

3) Field logic; a greatly expanded set of interfield operations within a full register sectioned according to some previously established convention, and

4) List search operations; a list equivalent of the conventional table look-up instruction.

The bulk storage input-output requirements for a list-processing computer are severe, and are not included in the enumeration above. The system design of a digital computer for the manipulation of list structures will be described in detail in a subsequent paper.

Finally, we consider the implications of our work for the basic problem of machine intelligence. The geometry machine, we feel, offers convincing evidence of the power and fruitfulness of heuristic programming for the solution of problems of a certain class by computer. In our experience, the theorem-proving power of the machine has often been extended by the addition of a single heuristic to a degree equivalent to a three to fivefold increase in the speed or storage capacity of the computer.

Our program has proved to be disappointing as a tool for the study of the more elementary trial-and-error types of machine learning, largely because of the rather low rate at which it accumulates experience. It is reasonable to expect, however, that the geometry machine might yet be pressed into service in an investigation of the higher, conceptual types of machine learning, providing that one will someday know how to formulate the problem.

If nothing else, our work offers some qualitative indication of the order of magnitude of difficulty for problems that could be expected to yield to contemporary computer technology. Three years ago, the dominant opinion was that the geometry machine would not exist today. And today, hardly an expert will contest the assertion that machines will be proving interesting theorems in number theory three years hence.

Footnotes and References

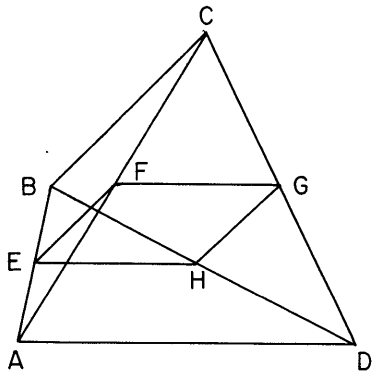
1. H. Gelernter, "Realization of a Geometry Theorem-Proving Machine", Proc. of the International Conference on Information Processing, Paris, 1959
2. More than fifty proofs are on file at the present time.
3. H. Gelernter and N. Rochester, "Intelligent Behavior in Problem-Solving Machines", IBM Journal of Research and Development 2 (1958): 336-345
4. M. L. Minsky, "Some Methods of Artificial Intelligence and Heuristic Programming", Symposium on the Mechanization of Thought Processes, Teddington, 1958
5. A. Newell, J. C. Shaw, and H. A. Simon, "Report on a General Problem-Solving Program", Proc. of the International Conference on Information Processing, Paris, 1959
6. A. Newell and J. C. Shaw, "Programming the Logic Theory Machine", Proc. of the Western Joint Computer Conference, (1957): 230-240
7. H. Gelernter, J. R. Hansen, and C. L. Gerberich, "A FORTRAN-Compiled List Processing Language", Journal of the Association for Computing Machinery 7, April 1960
8. A semantic heuristic is one based on an interpretation of the formal system rather than on the structure of the strings within that system.
9. A number of these proofs are reproduced in reference 1 .
10. H. Gelernter, "A Note on Syntactic Symmetry and the Manipulation of Formal Systems by Machine", Information and Control 2 (1959): 80-89
11. See, for example, Appendix A of reference 1.
12. A. M. Turing, "Computing Machinery and Intelligence", Mind 59, (1950): 433
13. It may be argued (and undoubtedly, it will be argued) that the truly knowledgeable interrogator, cognizant of the decideability of geometry, would certainly avoid this area as well, perhaps preferring the manifestly undecideable parts of the predicate calculus or number theory to effect the distinction between man and machine. We recall here that our methods are independent of the decideability of the formal system, and, in fact, Wang[14] and Gilmore[15] have developed techniques that have produced proofs for theorems in the undecideable area of the predicate calculus.
14. H. Wang, "Toward Mechanical Mathematics", IBM Journal of Research and Development 4, January 1960 : 2-22
15. P. C. Gilmore, "A Proof Method for Quantification Theory", IBM Journal of Research and Development 4, January 1960: 28-35

Appendix I

```
   PREMISES
**************
QUAD-LATERAL ABCD
POINT E MIDPOINT SEGMENT AB
POINT F MIDPOINT SEGMENT AC
POINT G MIDPOINT SEGMENT CD
POINT H MIDPOINT SEGMENT BD

   TO PROVE
**************
PARALELOGRAM EFGH

   SYNTACTIC SYMMETRIES
**************************
BA, AB, DC, CD, EE, HF, GG, FH,
CA, DB, AC, BD, GE, FF, EG, HH,
DA, CB, BC, AD, GE, HF, EG, FH,
```

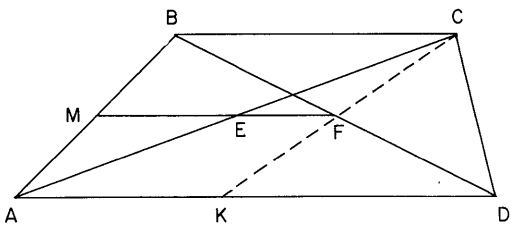



```
   PROOF
***********
SEGMENT DG EQUALS SEGMENT GC
      DEFINITION OF MIDPOINT
SEGMENT CF EQUALS SEGMENT FA
      DEFINITION OF MIDPOINT
TRIANGLE DCA
      ASSUMPTION BASED ON DIAGRAM
PRECEDES DGC
      DEFINITION OF MIDPOINT
PRECEDES CFA
      DEFINITION OF MIDPOINT
SEGMENT GF PARALLEL SEGMENT AD
      SEGMENT JOINING MIDPOINTS OF SIDES OF TRIANGLE IS PARALLEL TO BASE
SEGMENT HE PARALLEL SEGMENT AD
      SYNTACTIC CONJUGATE
SEGMENT GF PARALLEL SEGMENT EH
      SEGMENTS PARALLEL TO THE SAME SEGMENT ARE PARALLEL
SEGMENT HG PARALLEL SEGMENT FE
      SYNTACTIC CONJUGATE
QUAD-LATERAL HGFE
      ASSUMPTION BASED ON DIAGRAM
PARALELOGRAM EFGH
      QUADRILATERAL WITH OPPOSITE SIDES PARALLEL IS A PARALLELOGRAM


   TOTAL ELAPSED TIME   1.03 MINUTES
```

Appendix II

PREMISES
****
QUAD-LATERAL ABCD
SEGMENT BC PARALLEL SEGMENT AD
POINT E MIDPOINT SEGMENT AC
POINT F MIDPOINT SEGMENT BD
PRECEDES MEF
PRECEDES AMB



TO PROVE
****
SEGMENT MB EQUALS SEGMENT MA

NO SYNTACTIC SYMMETRIES
****

I AM STUCK, ELAPSED TIME 8.12 MINUTES
****
CONSTRUCT SEGMENT CF
EXTEND SEGMENT CF TO INTERSECT SEGMENT AD IN POINT K

ADD TO PREMISES THE FOLLOWING STATEMENTS
****
PRECEDES CFK
COLLINEAR AKD

PROOF
****
SEGMENT BC PARALLEL SEGMENT AD
PREMISE
COLLINEAR AKD
PREMISE
SEGMENT KD PARALLEL SEGMENT BC
SEGMENTS COLLINEAR WITH PARALLEL SEGMENTS ARE PARALLEL
OPP-SIDE KCDB
ASSUMPTION BASED ON DIAGRAM
SEGMENT DB
ASSUMPTION BASED ON DIAGRAM
ANGLE KDB EQUALS ANGLE CBD
ALTERNATE-INTERIOR ANGLES OF PARALLEL LINES ARE EQUAL
PRECEDES CFK
PREMISE
PRECEDES DFB
DEFINITION OF MIDPOINT
ANGLE KFD EQUALS ANGLE CFB
VERTICAL ANGLES ARE EQUAL
SEGMENT DF EQUALS SEGMENT FB
DEFINITION OF MIDPOINT
TRIANGLE FDK
ASSUMPTION BASED ON DIAGRAM
TRIANGLE FBC
ASSUMPTION BASED ON DIAGRAM
TRIANGLE FDK CONGRUENT TRIANGLE FBC
TWO TRIANGLES ARE CONGRUENT IF ANGLE-SIDE-ANGLE EQUALS ANGLE-SIDE-ANGLE
SEGMENT KF EQUALS SEGMENT CF
CORRESPONDING SEGMENTS OF CONGRUENT TRIANGLES ARE EQUAL
SEGMENT CE EQUALS SEGMENT EA
DEFINITION OF MIDPOINT
TRIANGLE AKC
ASSUMPTION BASED ON DIAGRAM
PRECEDES CEA
DEFINITION OF MIDPOINT
SEGMENT EF PARALLEL SEGMENT AK
SEGMENT JOINING MIDPOINTS OF SIDES OF TRIANGLE IS PARALLEL TO BASE
SEGMENT EF PARALLEL SEGMENT KD
SEGMENTS COLLINEAR WITH PARALLEL SEGMENTS ARE PARALLEL
SEGMENT FE PARALLEL SEGMENT BC
SEGMENTS PARALLEL TO THE SAME SEGMENT ARE PARALLEL
PRECEDES MEF
PREMISE
COLLINEAR MEF
ORDERED COLLINEAR POINTS ARE COLLINEAR
SEGMENT FM PARALLEL SEGMENT BC
SEGMENTS COLLINEAR WITH PARALLEL SEGMENTS ARE PARALLEL
SEGMENT FM PARALLEL SEGMENT DA
SEGMENTS PARALLEL TO THE SAME SEGMENT ARE PARALLEL
TRIANGLE DBA
ASSUMPTION BASED ON DIAGRAM
PRECEDES AMB
PREMISE
SEGMENT MB EQUALS SEGMENT MA
LINE PARALLEL TO BASE OF TRIANGLE BISECTING ONE SIDE BISECTS OTHER SIDE

TOTAL ELAPSED TIME 30.68 MINUTES

A SUGGESTED MODEL FOR INFORMATION REPRESENTATION IN A COMPUTER THAT PERCEIVES, LEARNS, AND REASONS*

Peter H. Greene
Committee on Mathematical Biology
The University of Chicago
Chicago, Illinois

Summary. This paper investigates how computers might represent enough of the structure of the percepts and concepts they handle so that they may sensibly be said to deal with the meaning of these things, rather than just to sort and recombine mere labels for the operator's percepts and concepts. One of the main requirements is that each element of information contain partial representations of many other elements and schemata for their interconnection. Some of these requirements may be met if it proves feasible to represent information in the form of vectors (such as modes of oscillation of a complicated network or resonator) which may be resolved into components in various coordinate systems. These systems represent various points of view from which the information may be regarded, and some of the information in each system is elicited by a probabilistic mechanism for use by a conventional computer.

This paper forms part of a program[1] of investigating mechanisms whereby computers can represent enough of the structure of the percepts and concepts they handle so that they, rather than the human operator, can be said to deal with the meaning of these things. The earlier paper stated some of the main goals; this paper proposes a model which may achieve some of these goals. We saw[1] that present-day computers deal primarily with external relations among concepts which are given in a form that does not represent their inner structure. Thus the concepts may be sorted and combined, but their meaning resides in the mind of the operator. We saw that an important task which precedes comparison and abstraction is the formation of impressions into representations adequate to sustain abstraction. Moreover, this primary challenge of forming impressions into logical elements is inseparably connected with the formation of a rule which gives the ordering and interdependence of the logical elements. Each element must contain partial representations of many other elements and schemata for their interconnection. We explained that if we regard the concepts of the computer as copies of a definite world of facts, we are doing the computer's job of carving out significant units. The particular units constructed by the computer would depend upon the generative principles of connection we have mentioned, and we were led to expect clusters of related thought precursors formed around individual generative principles built into the computer. We spoke of the need for analyses of meaningful units richer than their description in terms of conjunctions of atomic parts or their negations. Finally, we discussed problems involved in the important task of becoming acquainted with causal relations and the potentialities of things and actions—what would happen in situations not actually existing. The bibliography cited important studies of psychological behavior exemplifying all the preceding considerations. These studies, together with the philosophical studies there cited and with comments on the relevance of these ideas to computers, are reviewed in a previous paper.[2]

This paper proposes a new model for the representation of information in a computer which, if it proves feasible to realize, would lead to all the features of behavior that we have called for in the preceding summary, including certain structural properties of behavior resembling many of the psychological properties of perception, learning, and higher mental functions. A problem which will remain to be solved before such a model can become useful is to specify the precise way in which the properties will appear and get those situations to be just the ones we are interested in for practical reasons. Some of the motivation for part of the model was presented in an earlier paper by the author.[3] The present paper explains the model itself in a more systematic way and introduces features with numerous consequences not then known. Table 1 summarizes the various parts of the argument and indicates their interdependence.

Initially the model was intended to simulate important features of Gestalt perception, with the understanding that a more adequate theory could not arbitrarily separate perception and thinking. It was therefore very gratifying to discover that upon the introduction of a missing part of the model required for the understanding of Gestalt perception, the model without further extension predicted a type of behavior resembling theoretically inferred mechanisms of learning, thought processes, and certain integrated action patterns performed by animals.

In a crude pattern recognizer capable of recognizing a class of patterns such as the front door lock of an apartment building, the discriminating features, or "perceptual units," are rigidly built into the objects to be discriminated. But we know that more sophisticated recognizers cannot be provided with such ready-made units, and it is our job to understand how the task preceding discrimination, namely the

---

*This research was supported by the Office of Naval Research under Contract No. Nonr 2121(17) NR 0/9-148. Reproduction in whole or in part is permitted for any purpose of the United States Government.

formation of stable units to be recognized and discriminated, may be accomplished. A machine (or person) lacking this process would combine all sorts of stimuli into meaningless groups. The model presented here is intended for that part of a machine which produces the stable, meaningful units, and is thus intended not to replace but to supplement other types of perceiving or reasoning devices.

Now let us state the kind of mechanisms which will be used, although only the subsequent discussion will reveal what connection they have with the problems that we have outlined. Information is to be coded as patterns of oscillation of complicated resonators or networks of oscillators. In particular, these patterns may be resolved into superpositions of normal modes of various kinds, and these normal modes serve as the symbols for certain stable percepts or concepts—e.g., for "good" Gestalten. The same pattern may be resolved into superpositions of any one of a number of complete sets of normal modes. Such transformations are carried out reversibly, and the modes of any such set will be thought of as representing possible outcomes of the elicitation of information regarding one particular aspect of the total pattern. The above processes are intended to simulate human processes which are normally never at the level of awareness. Elicitation of this information for responses and simulated perception at the level of awareness will require the following mechanisms (for reasons which cannot be guessed before the explanations which follow): First, the total pattern must be split into one of the possible sets of normal modes. Second, each such mode must be multiplied and averaged with the same shot noise, and the mode thus yielding the largest result selected to produce the percept or response and then returned to the population of total patterns. Subject to slight elaboration later, these are all the mechanisms we shall need: coding as modes of oscillation, together with a mechanism for elicitation involving random selection and recombination. For concreteness, I like to think of microwave modes in some sort of complicated resonator, but I do not know whether such a realization might be feasible. Even though no realization is known, it still seems worthwhile to derive unexpected psychological properties of mechanical processes.

The argument will be presented by indicating the new perceptual features introduced as the properties of the model are incorporated, one by one. At each stage of the argument appropriate behavioral desiderata, chosen from the topics of the first paragraph, will be shown to motivate the incorporation of the next mathematical property of the model. Thus the specific mathematical model just outlined is derived from such topics, which will form the bulk of this paper, and must not be regarded merely as philosophical background. Once these arguments are understood, and not before, the simulated psychological aspects of the behavior of the model may be pointed out rather simply.

Since this paper will propose a new means of information representation in computers, it is important first to have for a guide an intuitive idea of the kind of thing which is being attempted. We wish to build a machine that performs the necessary steps prior to discrimination and abstraction, and we are particularly interested in the versatility displayed in the processes of pattern formation and stabilization. Perception automatically adjusts itself to variations of the stimuli in a multitude of integrated ways, and it continually leads to the formation of new meaningful perceptual elements. Thus we seek a model which offers the possibility for such flexibility. Heuristically speaking, we shall not seek numerous models in each of which some class of response patterns is "built in," but rather a model which may be conceived of intuitively as a sort of arena in which complicated patterns can spring up, interact, and evolve "by themselves." We cannot hope to specify perceptual processes in all their complexity, but must allow them to arise in an evolutionary process of elaboration in which what is given consists of certain primordial elements, certain laws of transformation, and the opportunity to carry out these transformations. This is in accord with many mechanisms which are believed to operate in human perception and thought.

In the previous presentation of motivation for part of the model,[3] reasons were explained for wishing to represent "good" Gestalten by normal modes which had mathematical properties similar to those of eigenstates and transformation amplitudes in quantum mechanics (although the physical realization has nothing whatever to do with quantum mechanics). This is because these mathematical structures lead to theorems specifying behavior depending upon relational aspects of complex configurations and resembling perceptual phenomena. The first part of this paper will add further psychological features to the model and will be self-contained with no essential reference to quantum mechanics.

In accord with the program of studying the initial stages of perception which provide stable perceptual units for discrimination, we assume the existence of a device that can recognize perceptual units which we have transformed into sufficiently stable and standard form. Thus we start with the existing level of technology of pattern recognition. What might we add by developing the idea of coding as modes of oscillation of complicated and changeable networks or resonators?

First of all, we have the obvious fact that our perceptual units will be very complicated. They might be sorted and combined in the same ways as conventional symbols, but in addition they possess elaborate internal structure which may be used as a vehicle for the expression of relationships with other symbols and an agent or object of transformations induced by interactions with other symbols. Next, we have the equally obvious fact that complicated modes of oscillation "spring up by themselves" in the sense of our intuitive requirement, specified perhaps only by some frequencies or simple symmetries of the impressed energy. These patterns are more complicated than any we could build directly, and this phenomenon of a complicated pattern

springing up from a much simpler one might allow for richness of perception and thought without making unreasonable demands upon the complexity of the coding and memory. Evidence for the storage of ideas in the form of thought precursors is cited in a previous bibliography[1] and reviewed,[2] and such a method might provide a logical extension of the procedure of Shaw, Newell, Simon, and Ellis[4] in constructing their Information Processing Language. In this language the data are not inert and structureless but are provided in the form of data programs, and the data are obtained by executing these programs. A list of data may be specified by a list of processes that determine the data. These authors explain that their approach leads to a computer that contains at any given moment a large number of parallel active programs frozen in the midst of operation and waiting until called upon to produce the next operation or piece of data. Development of our model might provide feasible ways of "freezing" extremely complicated programs.

Two topics which immediately come to mind in connection with modes of oscillation are the superposition of patterns into complicated structures and the resolution of complicated patterns into linear combinations of basic oscillations. Although there is no more information in a linear superposition than in the components, we shall deliberately be working with a language in which, as befits perception, some information will be in the focus of attention while other information will temporarily be relegated to the background. Otherwise, only chaos would be perceived. We shall see later how our symbols will contain the latter kind of information in a latent form and how this information can be brought into the awareness of the recognizing part of the machine by a process which consigns other information to latent status. Thus the available information in superposition may be different from the available information in the components.

The preceding paragraph suggests that we are interested in observing the same pattern from different points of view in order to bring out different pieces of information. It seems natural to investigate the simplest possibility consistent with our concern for the superposition and resolution of modes, namely that the various points of view are represented by the components of the complicated symbol in various coordinate systems. That is to say, we resolve our symbol into combinations of various kinds of modes. What new perceptual features would this introduce? Since the components of the symbol in various coordinate representations are related by linear transformations, we shall have, once the observation procedure is explained, a schema for the connection and interdependence of observations called for in the first paragraph. This rule of connection will not be as rigid as it may seem at first because flexibility and variety will be introduced by the "latent" information in each representation which shows up only in other representations, so that from the point of view of the recognizing system, one representation does not tell all about the others. Second, the process of utilization of information will itself introduce probabilities and possibilities for recombination and variation of symbols, and this process can lead to the evolution of more complex forms.

In accord with the first paragraph, we see that any perceptual element in one coordinate system will contain partial representations of many other elements in that system or other systems. Moreover, linear transformations between representations do not coordinate particular parts of patterns in one representation with particular parts in another but are of a holistic nature, in accord with the first paragraph and with elementary facts of Gestalt perception.

Let us introduce more structure by considering the mathematical properties of the coordinate systems. The natural kind of coordinate system is provided by normal modes of oscillation, or more precisely, eigenfunctions of linear operators which characterize the system, that is, functions which are merely multiplied by a constant when the operators are applied to them. Later on we shall explain how these normal modes will provide precisely defined patterns which we shall identify with "good" Gestalten. What new perceptual characteristics are thus introduced? First of all, we have the general feature that stable perceptual units would have to appear in certain discrete, reproducible, integrated forms, which may change discontinuously from one to another with nothing in between. This is such a characteristic feature of human perception that we often tend to forget those instances in which we do see indefinite and merging forms. Evidence that stable percepts rest upon a basis of merging, streaming, scintillating, and reduplicating forms is adduced by Schilder.[5] This is the kind of thing our model tries to do; that is, to base stable perceptual units upon wave-like functions which permit superposition, decomposition, and transformation.

Specifying the operators of our system determines these normal modes which will constitute the stable perceptual units. Thus the system actively "carves out" units, in the terminology of the first paragraph, by means of operators built into the system or attained through a process of learning. When we discuss the process of elicitation and utilization of these perceptual elements, we shall see the interrelations among clusters of elements belonging to the same operators, and we shall derive behavior related to the clusters of thought precursors formed around individual generative principles mentioned in the first paragraph. We shall also derive a certain interchangeability in the early stages of perception among the modes belonging to a single operator.

Our last observation at this stage pertains to the linearity of the system. Characteristics of the system will be represented by operators, and perceptual units by combinations of their eigenfunctions. As a consequence, we have the fact that although the operators may be related by complicated nonlinear relations, the perceptual units will be linearly related. This results from the general mathematical fact that the eigenfunctions of any of the operators we shall consider can be linearly superposed to produce any function we shall require.

Our observations so far have been very general. Nonetheless, they are essential both to the derivation of the model and to the comprehension of the purpose of the model once derived. Our method so far has been to assume that a useful model can be constructed and to examine what the various parts will do. Only when we know this can we know how to put them together. Our conclusions so far, supplemented by the equally general comments upon selective awareness and group properties of perceptual transformations, will, as indicated in Table 1, lead to conclusions with a great deal of content.

Our next stage will be to see what the operators and eigenfunctions mean. We shall frequently be interested in the results of a number of different types of observation of the perceptual unit forming system by the recognizing system. Let us suppose that an observation of some type, which we shall label A, might reveal any one of the states a_1, a_2, etc., an observation of type B one of the states b_1, b_2, etc., and similarly for observation C, and so on. In describing events at a perceptual or mental level of organization, we customarily use a form of language in which we are able to formulate laws governing the influence of past events upon the present without considering the details of trace structures which exist at all intermediate times. We therefore seek a formalism which connects behavior at a single earlier observation with that at a single later observation, rather than providing a continuous description of observations. In particular, we shall often be interested in the conditional probabilities $P(a_i|a_j)$, $P(a_i|b_j)$, etc. that an observation of A will reveal the state a_i, given that a previous observation has revealed a particular state. In situations where an A observation must yield some a state, we have the equality $\sum_i P(a_i|b_j) = 1$ for an arbitrary state b_j.

We characteristically do not see all aspects of an object simultaneously but typically find one aspect at the focus of attention, while other aspects remain indefinite until some effort is made to perceive them clearly. If the perceptual experience depends in some statistical fashion upon a set of "observations," i.e., elicitations of coded information by the recognizing part of the machine, we might expect that observations of the first type all yield the same result, while results for the second type of observation will be less coherent. If we interpret our P's above to apply to two observations in this observed set, we might then have a situation in which $P(a_k|a_i) = \delta_{ki}$ (i.e., 1 if k = i and 0 otherwise), while $P(b_j|a_i)$ might take on various values. This statement is intended to cover a number of situations. For instance, we may suppose that two observations are performed in immediate succession, the second being applied to the information elicited as the result of the first. Then the requirement states that if an observation of type A applied to the original information has revealed the information a_i, then further observations of type A applied to the elicited information a_i can only continue to reveal a_i, while an observation of a different type B applied to a_i might reveal any of the b_j. According to a second interpretation, elicited information functions slightly differently. Suppose that a particular perception or action of type A is not correlated with a single element a_i, but rather with a particular distribution of weights among all the elements of the set $\{a_1, a_2, \ldots.\}$. A natural way of assigning weights is to perform a number of independent A observations upon the same information, thus obtaining a population of a's, and to let the weight of a_i be the proportion of a_i's in that population. Then the probability $P(a_k|a_i)$ may be interpreted as the probability that a randomly chosen member of the population is a_k, given that a previously randomly chosen member was a_i. The condition $P(a_k|a_i) = \delta_{ki}$ then states that all the members of the population are the same, so that the percept or act has only one nonvanishing component, and thus the condition allows for the existence of such "elementary" acts. A weaker requirement might allow for the existence of several, but not all components. This requirement, general as it may be in isolation, will in connection with our other general requirements shortly turn out to impose very specific requirements on the mathematical structure of the model.

The next stage in the development of the model is to try to combine this statistical formalism with the system of linear transformations which we have previously discussed. In order to simulate important features of human perception, we wish to find ways of relating the conditional probabilities between pairs of sets so that transformations may be compounded, or so that they may be decomposed into transformations involving intermediate sets, as for example by expressing $P(c|a)$ in terms of $P(c|b)$ and $P(b|a)$. These transformations must preserve the sums of probabilities, so that they add up to one as before. Our first thought is to let the previously discussed linear transformation scheme be identical with the well-known matrix multiplication of probabilities, $P(c|a) = \sum_j P(c|b_j)P(b_j|a)$. However, if we now refer back to the discussion of selective awareness in the preceding paragraph, we see that we shall in general be unable to obtain the situation there described, in which $P(a_k|a_i) = 0$ if $k \neq i$. The reason is that if the same matrix combination is to hold for all kinds of observations, we find by replacing c by a_k and a by a_i in the multiplication rule that $P(a_k|a_i)$ will be the sum of several positive terms, and this contradicts the requirement that it equal zero. That is to say, we can get from a_i to a_k via some of the b's, contrary to our desire, unless we can define a transformation according to which many of these paths cancel each other.

Landé,[6,7,8] in works which gave me my first notions of what to do about my conviction that something useful might come of analogies between quantum mechanics and perception, points out that there is just one known system of transformations which will do. If we restrict ourselves to the case of symmetrical probabilities, $P(b|a) = P(a|b)$, (a restriction for which certain heuristic justifications may be offered), then the transformations may be defined by complex valued matrices

$\|\Psi_{kj}\| = \|\Psi(a_k|b_j)\|$, such that the squared modulus $|\Psi_{kj}|^2$ of the element Ψ_{kj} equals the corresponding probability $P(a_k|b_j)$. Then it can immediately be seen that the requirements that the probabilities add up to one and that they satisfy the condition for selective awareness are equivalent to the condition that the matrices just defined be unitary matrices (the complex analogue of orthogonal matrices). We shall call the Ψ's by the name which analogous quantities have in physics, probability amplitudes, to distinguish them from the probabilities P.

Of course, we are not denying the validity of the well-known law of matrix multiplication of probabilities in favor of the matrix multiplication of probability amplitudes. The two rules apply to different situations, the former determining, as it must, the probability that c is elicited given that b and a have been elicited, and the latter determining the probability that c is elicited if only a has been elicited, expressing this probability in terms of two conditional probabilities that refer to observations not actually performed. This difference of course requires that we introduce a suitable precisely defined way in which the elicitation of b modifies the information.

Now we may connect this discussion of probability amplitudes with the previous groundwork concerning transformations between coordinate systems by remembering that the unitary matrices are precisely the matrices which transform between coordinate systems defined by sets of orthogonal and normalized vectors. And now we may turn the argument around and say: given a coding of information as complex oscillations, with components in various coordinate systems, we wish to define a process which will elicit the information corresponding to a particular component with a probability proportional to the squared modulus of the amplitude of that component. As emphasized in Table 1, we are led to this result only by combining the above desiderata at the behavioral level, and with this result we begin to collect the promised fruit of all the discussion that has gone before. Once we define a suitable mechanism for elicitation of information, we shall be able to use the discussions of qualitative properties of mathematical structures to show how the model will simulate certain important structural characteristics of thought. Then, when we add a suitable mechanism for the change of the Ψ in time (or with respect to some other parameter), we shall be able to derive some typical properties of Gestalt perception.

We shall shortly define a mechanism for the elicitation of information with a probability equal to the squared modulus of its amplitude. Accepting for the moment the assertion that this can be done, we may immediately derive operator and eigenfunction equations and thereby specify the eigenfunctions which will be the coordinate vectors. We may expect the observable properties of the oscillatory states to depend upon numerical functions which take different values in different states, because the input of the recognizing part of the machine can always be expressed in such a form. These values will be precisely defined without any statistical spread (we shall say "sharp") in the set of states for which some perceptual aspect is definite. In such a set of observations of type A, let us denote the precisely defined numerical value in state a_k by the same symbol a_k. Then for the mean value of a over all states corresponding to observation A, we clearly have $\bar{a} = \sum_k a_k P(a_k|s) = \sum_k \Psi^*(a_k|s)\, a_k \Psi(a_k|s)$, by definition of the relation between Ψ and P, where s denotes the state of the system in an arbitrary coordinate system. What will be the mean value of a in terms of the B coordinate system? If in the expression for $\bar{a}$ we perform the linear transformation to the B representation, we find that $\bar{a} = \sum_m \Psi^*(b_m|s)\mathbf{A}\Psi(b_m\, s)$, where $\mathbf{A}$ is the linear operator represented in the B system by the matrix having elements $A_{mn} = \sum_k \Psi^*(a_k|b_m) a_k \Psi(a_k|b_n)$. According to this formalism, the operator $\mathbf{A}$ is represented in the A system by the diagonal matrix $A_{kk'} = \delta_{kk'} a_k$. This standard kind of computation shows the origin of the operators mentioned in our discussion of stable configurations.

Now all one has to do to obtain eigenfunction equations defining the coordinate systems is to multiply both sides of the defining equation for A_{mn} by $\Psi(b_n|a_i)$ and sum over n, making use of the unitary nature of the Ψ's. The result is the eigenfunction equation $\mathbf{A}\Psi(b_m|a_i) = a_i \cdot (b_m|a_i)$. Given the matrix representing $\mathbf{A}$, one may solve this equation to obtain the various a_j and corresponding $\Psi(b_m|a_i)$. This equation states that the transformation amplitude matrix from any set of states to the set in which a particular type of observation is sharp must be an eigenfunction of the operator corresponding to that observation in the former set of states, and the numerical value a_i identifying the state must be the associated eigenvalue. We began by merely requiring a_i to be some number associated with the state. The present conclusion follows entirely from the requirement that probabilities be the squared magnitudes of the Ψ's, and we were led to this requirement by considering general behavioral desiderata.

Examination of the mathematical meaning of the Ψ's reveals that the eigenfunction $\Psi(b_m|a_i)$ is simply the set of coordinate values representing an a vector in the B system, so that we may alternatively consider basic coordinate vectors to be eigenvectors of operators, or the transformation amplitudes to be eigenfunctions of operators. It is easy to show[6,7,8] that our assumptions have implied that the operators must be Hermitian, and their eigenvectors will thus be orthogonal and provide suitable coordinate systems.

In a previous paper[3] I discussed a mechanism for the successive transformations of a set of inputs to a network, which produced a group of transformations depending upon a parameter τ. This parameter was the analogue of time in quantum mechanics, although in the case of the network it denoted spatial distance across a network

from an input mosaic to an output mosaic. It was briefly noted that if we had a network containing a large number of alternate paths for the signals, each path being specified by distributed phase lags, we would obtain transformations of coordinate systems described by equations analogous to the Schrödinger equation, namely $\partial\Psi/\partial\tau = jH\Psi$, where H is a Hermitian operator defined in terms of the phase lags. Such a formalism provides one interpretation for the earlier statement that normal modes will be stable, for the τ dependence of eigenfunctions of H will be entirely exponential with an imaginary exponent and will cancel out of the observation probabilities when we take the squared magnitudes in the elicitation process. We must ultimately be more specific about the detailed nature of H and other operators which determine the behavior of the system, because we will not be satisfied merely with quasi-perceptual behavior in unspecified situations. However, we shall be able to draw a large number of conclusions which follow from the formalism alone and which will thus be true independently of the precise nature of the operators. These sometimes enable us to work backwards from behavioral desiderata and thereby eliminate whole classes of possible operators. However, since this paper will be confined to conclusions which follow from the general formalism alone, we shall not discuss the problems of further specification of operators. We shall explicitly refer to the transformation equation of this paragraph only in connection with certain so-called "Gestalt constancies."

The last mechanism we shall have to introduce is the one which selects normal modes with a probability proportional to their amplitudes. The problem is as follows: Suppose the oscillatory pattern representing, for instance, all possible ways of looking at a complicated stimulus is a function Ψ of a number of variables, which we shall not specify in detail. For illustrative purposes we may consider the excitation at some point to be a function of time, $\psi(t)$. We have supposed all along that Ψ may be resolved into superpositions of various kinds of normal modes, or eigenfunctions. Thus, alternatively, $\psi(t) = \sum_n a_n\phi_n(t)$ or $\psi(t) = \sum_n b_n\xi_n(t)$, etc., depending upon the point of view from which the total configuration is being observed. Suppose the point of view is such that Ψ is split into the ϕ_n. The coefficients a_n will be complex numbers. We want to specify a mechanism which will pick out one of the normal modes and discard the rest, in such a way that the probability of picking the kth mode ϕ_k is equal to $|a_k|^2$, or actually to $|a_k|^2/\sum_n |a_n|^2$ in case Ψ is not normalized. How does one pick something with a probability proportional to the square of a complex number? Wiener[9,10] has considered this question in connection with the identical situation in quantum mechanics, and his conclusion is that he can think of only one non-quantum mechanical process which could do that. Therefore I decided to examine the consequences of incorporating such a process in the present model, and it was at this point that the model turned out automatically to possess many interesting structural properties of thought, together with the ability to learn by being "rewarded" in a simple way.

This selection process is quite simple and will now be described. Wiener gives two variants, of which only one will be considered here. Let $Y(t)$ be the integrated complex-valued output of a shot noise generator. Then dY, if it existed, would be the instantaneous output. Actually, this is a physically unrealizable idealization of a Gaussian random process. What one actually measures is the increment ΔY for a small increment of time. But the intuitive notation can be given a precise sense. It turns out that the integrals $\int \phi_n(t)dY(t)$ are all random variables with Gaussian distributions, and in case the ϕ_n are all normalized the distributions will all be the same. Moreover, if the ϕ_n are orthogonal to each other, that is, $\int \phi_n^*(t)\phi_m(t)dt = 0$ for $n \neq m$, then the Gaussian distributions will be mutually independent. Our general considerations led us to consider Hermitian operators, which have orthogonal eigenfunctions, so that this condition is satisfied. Now let us calculate the time averages of the normal components of Ψ multiplied by the shot noise: $A_n = \int a_n\phi_n(t)dY(t)$. This is accomplished by a physical device which finds the statistical correlation of two inputs. Finally, let these numbers A_n control a gate which lets through the mode having the largest A_n. Since the A's are random variables, this process picks the modes statistically, and the probability of picking the kth mode turns out, as shown by Wiener, to be just what we want, namely $|a_k|^2/\sum_n |a_n|^2$. We shall not go into detailed consideration of the suitability and implications of different ways of comparing the A's,[10] except to say that if comparisons are made in pairs, the order in which the pairs are picked makes little difference, and that it is also possible to compare pairs of superpositions of several modes and then split these up further. This latter point will have a useful interpretation. A possible way of performing the process is indicated in Figure 1. This general type of process turns out to have numerous interesting consequences, which we are finally ready to understand, all the groundwork having been prepared.

Restrictions on the length of this paper preclude all but the briefest discussion of these consequences. Further information, especially in connection with an attempt to understand the representation of information in the brain, will be published elsewhere. However, making use of the background of information which we have developed about the properties of the mathematical structures, we may derive most of the interesting features in a few sentences. Here is a list of them.

We begin with the ways in which information coded as modes of oscillation is modified in the process of elicitation by the recognizing part of the machine. The information exists in a population of "wave functions" Ψ, which are split up in various ways into superpositions of components ϕ_n or ξ_n, etc. The process of elicitation operates upon one of these resolutions and selects

one or more components, ignoring the rest. The selected components will then be identified or produce some action or be subjected to further transformations, and they may or may not subsequently be returned to the population of wave functions, depending upon the purposes to be served. Observing the same information from a different point of view is interpreted as splitting the wave function into a different set of components and then selecting some of these components. The various components in any particular resolution correspond to various ways the pattern could appear when looked at from the corresponding point of view, or to various activities in a particular mode of behavior.

If there is just one Ψ (instead of a population of them), then an observation from one point of view might elicit any one of the ϕ's, but subsequent observations from the same point of view will continue to elicit the same component. This is one way in which an ambiguous "percept" becomes clarified in the process of bringing it to the "awareness" of the recognizer. Another, and possibly more significant, way will be explained when we come to the effect of the shot noise on the latent information.

If the information selected by the elicitation process is returned to the population of wave functions, an interesting phenomenon results. There will be a repeated process of splitting the wave function into components of some type, discarding some of the components, returning the remaining randomly selected components to the population, splitting them into other types of components, and so on. This process brings about the reappearance of components which have previously been discarded. For instance, if only the ϕ_1 component is retained, and it is split into a superposition of several ξ components, and only one of the ξ's is retained, this ξ will in general when resolved in the ϕ system contain many of the ϕ's, and thus contain ϕ's which had previously been discarded. This is a perfectly elementary fact in any vector space, which has two immediate consequences for us. The first is that if an input excites one mode, then the result of the resolution and selection process is to excite other modes in the same coordinate system. In other words, the model exhibits a form of <u>association</u>. It is a standard problem in probabilities to calculate the association strength as a function of such things as the number of iterations of the resolving-selecting process, and the problem is being investigated. The second consequence will be explained in connection with learning.

The probability of finding a particular value for the numerical function which identifies the possible results of some type of observation is determined by the squared magnitude of the corresponding amplitude and is thus independent of any factor of the form $e^{j\alpha}$. This means that an arbitrarily large amount of information may be included in the phases of complex exponential factors of the amplitudes of modes belonging to some type of observation without making any difference in the results of that observation. However, this information will affect the results of a different kind of observation. For example, suppose that $\Psi = a_1 e^{j\alpha_1}\phi_1 + a_2 e^{j\alpha_2}\phi_2$, and that $\phi_1 = \Psi(1|i)\xi_1 + \Psi(2|i)\xi_2$. Then in an observation which reveals ϕ's, the probability of finding ϕ_1 and its associated numerical value will be $|a_1|^2$ independent of α_1 and α_2. However, in the ξ system $\Psi = [\Psi(1|1)a_1 e^{j\alpha_1} + \Psi(1|2)a_2 e^{j\alpha_2}]\xi_1 + [\Psi(2|1)a_1 e^{j\alpha_1} + \Psi(2|2)a_2 e^{j\alpha_2}]\xi_2$, so the probability of observing ξ_1 will be $|\Psi(1|1)|^2 \cdot |a_1|^2 + |\Psi(1|2)|^2|a_2|^2 + \Psi^*(1|1)\Psi(1|2)a_1^* a_2 \cdot e^{j(\alpha_2-\alpha_1)} + \Psi(1|1)\Psi^*(1|2)a_1 a_2^* e^{j(\alpha_1-\alpha_2)}$, which depends very much upon α_1 and α_2. This is what was meant by the statement that in any representation there would be latent information which did not show up in that representation but only in others. We first introduced the idea of selective awareness by requiring that the states in the focus of attention be clearly distinguishable, while the others are uncertain. The present consequence shows that each point of view will reveal information which will not be revealed by other types of observation. We may look at this fact in two ways. One way is to say that a perceiver must at any time disregard many things in the stimulus pattern in order to have any intelligible perception. The converse way is to say that knowledge of one aspect or even several will not include all the information needed to understand other aspects and their interrelations. The additional information required is the rule for the ordering and interconnection of impressions[1] mentioned in the first paragraph. This and several other interpretations of the meaning of the mathematical formalism are well known in the analogous situations in quantum mechanics, and credit has previously been given[3] to the sources of some of these formulations.

The next three consequences concern Gestalt perception. If a clear perception or an integrated action depends upon the coherent behavior elicited from a population of wave functions Ψ, then we might have a situation in which the clarity depends upon all the Ψ's being the same. This is called a <u>pure case</u> of an assembly in quantum mechanics; otherwise the assembly is a <u>mixture</u>. In connection with the preceding consequence, we may note that an observation from one point of view will turn a pure case into a misture with the same response probabilities <u>in that point of view</u>, but will disrupt the coherence of responses from other points of view. It is a simple mathematical fact that a composite system may be pure while its subsystems are not. Moreover, an observation of one subsystem may produce a change in the pureness of another subsystem. This fact has been discussed[3] in relation to the well-known facts of Gestalt perception that (a) a stable perceptual unit, or "good Gestalt," may become unstable and hard to see if embedded in a larger pattern, and (b) what was originally a single Gestalt may be broken up in

various ways by the addition of new lines in the visual stimulus pattern, so that the new Gestalten will not coincide completely with any of the old ones. Illustrations were given in that paper. This is of great importance in a perceptual machine, because we might wish to scan a pattern to find a meaningful pattern hidden it it, while at the same time we do not wish every chance grouping of lines in a stimulus to be perceived as a meaningful pattern.

The next Gestalt property rests upon the previously considered assumption that the transformation given by an equation like $\partial\Psi/\partial\tau = j\mathbf{H}\Psi$, the solution of which is some function $\Psi = \sum_n a_n\Phi_n$. In this case the following situation is well known in quantum mechanics. Suppose that conditions are changed in such a way that $\mathbf{H}$ becomes a function of a parameter μ. Then the solution of the equation will likewise become a function of μ which we may write $\Psi(\mu) = \sum_n a_n(\mu)\Phi_n$, where $a_n(0) = a_n$. The new response probabilities will be $|a_n(\mu)|^2$. Now one might expect the change in a_n to be roughly of the same order as the change in μ. It turns out instead that the change in a_n is of the same order as the rate of change of μ, that is, $d\mu/d\tau$. Thus, provided that μ changes very gradually, it can become quite large without inducing a significant change in the response probabilities. Examples were given[3] of very important analogous phenomena in perception, the Gestalt constancies. In order to perceive objects as meaningful units, one must ignore certain inhomogeneities, and it is well known, for example, that large spatial variations of luminance are not perceived, provided that they are gradual; but they become prominent when any discontinuity is introduced. An illustration has been explained in detail.[3]

The third Gestalt property likewise requires a wave equation. There is some reason to believe that while the constancy phenomena would occur with a wide class of wave equations, these equations must be first order in τ, as written above. In this case the time dependence of a wave function which is an eigenfunction of $\mathbf{H}$ will be exponential with an imaginary exponent, and thus will cancel out of the response probability. Probabilities arising from a superposition of such steady state solutions will, however, be time dependent, so that one source of change is derived from the interference of such steady state solutions for different states associated with $\mathbf{H}$. It is possible to make the interpretation that certain perceptual changes, inexplicable when only one point of view is considered, come from the information, not available to awareness in that point of view, about the range of possibilities of other aspects. Moreover, one can calculate the frequencies at which the perceptual transitions will occur. Because of the determinate form and possible discreteness of the normal modes, a change might be as abrupt as in the perception of the well-known picture of a staircase which can look as though it is being viewed either from above or from below. In general, the resolutions of Ψ into ϕ's and into ξ's given in connection with latent information may be used to predict a possible interference effect of a second stimulus introduced into the field. The situation is analogous to the interference effects observed when an electron has the opportunity to go through two holes.

Next we shall consider a general feature of the model which resembles the mode of organization of certain highly integrated action patterns in animal behavior. Reference to Figure 1 reveals that the information is elicited by a process which has two stages: first, splitting the wave function into normal modes, and second, selecting a mode by a mechanism utilizing shot noise. The first stage requires either very specific filters which separate the modes, or else (as shown) precise copies of the modes which can be used to obtain the various components in the same way one finds terms in a Fourier expansion. The second stage requires that the noise level be high enough so that the results of the integrations may exceed any thresholds which exist in the component which compares them. There is evidence from the study of instinctive behavior in animals (previously cited[1] and reviewed[2]) that complicated action patterns are broken down into a sequence of acts, each one released under the two conditions that a very specific stimulus be present and that a drive level be high enough. People are trying to build machines that do remarkable things, and it seems wise to see how those things actually are accomplished in nature. Even if the detailed mechanisms of instinctive behavior are not the same as those of the present model, it may be of interest that they can be described by the same flow charts at some level of abstraction.

Next we shall examine some ways in which the behavior of the model resembles structural characteristics of thinking. We have mentioned the importance of the fact that the wave function is split into many modes and some are selected randomly. The random selection, aided by systematic biases, of which we shall later give one example in connection with learning, can lead to the evolution of complex forms. It seems to be typical of thought and action processes that many alternatives are proposed out of which some are selected. In perception we have, for instance, the previously mentioned evidence that stable percepts emerge from a conglomeration of wavelike images. In instinctive behavior there is evidence that the animal behaves in a searching or random way until the two previously mentioned conditions release the block to action and allow selection of a particular action pattern. In thought pathology there is some evidence that a large variety of symbolically related ideas can emerge as the final thought product, which would be suppressed precursors of normal thought processes. In the development of thought in an infant, according to hypotheses of Freud, various rudimentary images become associated with drives. In the absence of the thing which will satisfy the drive and reduce its level, these images become activated in some way which raises them to hallucinatory vividness (like the images in dreams). If the activation level is below threshold, the necessary concentration may be

achieved by utilizing energy from another image, or by fusing several images related to the same drive so as to pool their activation energy. Any such image may stand for the drive in this form of thought, termed the primary process by Freud. Ideas may stand for their opposites, because at this stage there is no mechanism to distinguish between evoking an image to affirm it or to deny it. Thus ideas are associated by their relevance to the same drive, not because of formal logical relationships. Moreover, mutually contradictory ideas can exist side by side, for there is no mechanism for comparing them and rejecting one or both. We can find familiar examples of such thinking in the substitutions and fusions of images in dreams and in slips of the tongue, but it is the aim of the arguments[1] cited in the first paragraph of the present paper to make clear the fact that any organized and independent thought processes, however logical their outcome, must employ such a process. While such thinking is indispensable, it must be supplemented by a secondary process which enables the infant to get along in the world. Under the influence of learning by checking his ideas with the real environment, he must voluntarily delay the drive induced discharge of images so they are in accord with reality, and he must make connections, not between ideas which are related merely to the same wish, but between ideas which are related by virtue of having the same relation to actual experience. The absence of this unlimited fusion of ideas results in less pooling of the activation energy, and the representations become less vivid, taking the form of thoughts instead of images of a hallucinatory nature. These thoughts are now manipulated logically so that they may lead to actions producing gratifying changes in the environment rather than merely the hallucinations of gratification. In logic itself it appears,[1] that not only on the lowest levels of the synthesis of meaning but at the highest propositional levels, too, reasonable statements may be produced only by a process of rejection of a multitude of lawfully produced but unreasonable statements.

If these hypotheses are true, thought, while remarkable, does not appear inconceivable, and it seems worthwhile to see whether such processes may be built into machines. An example of a machine program which has some features of this generation of clusters of ideas around individual goals followed by selection of some of them by means of an evaluative process is described by Newell, Shaw, and Simon[11] in a discussion of chess playing machines. Their program contains a subprogram built around a set of about a dozen goals, each corresponding to some feature of the chess situation. Each goal has associated with it a move generator and an analysis and evaluation procedure. The move generator associated with a goal proposes alternative moves associated with that goal, regardless of suitability, on the basis of any connection with that goal. The evaluation and analysis procedures determine the value of the move from the point of view of that goal alone. The analysis procedure is concerned only with the acceptability of a move once it has been generated by the move generator. The consequences of the moves generated by all the goals are explored by evaluating possible game continuations generated by a different generator from the one which proposes moves. An executive routine makes the final choice of an acceptable move from the fifty or so proposed moves.

The present approach is to try to get things like this to happen by themselves without having to specify precise programs in advance, in order to allow the spontaneous elaboration of symbolic complexity. Let us examine the way information is elicited in the present model. There are two stages in this process. In the first stage the wave function ψ, from which all points of view may potentially be extracted, and which may even represent a large number of ideas in some integrated portion of the machine's "mind," is split into components in one or another coordinate system. Two possible resolutions of the wave function are represented by the hexagons and squares in Figure 1B. As we have proved mathematically, each of the components contains some latent information, represented by the shading, which is not available to the "awareness" of the recognizing part of the machine, but which becomes transformed in other resolutions into information which can be elicited. This latent information is thus necessary to the reversible transformations between different points of view. At this stage of merging and splitting all information is present only in the form of potentialities. Suppose that ψ is resolved into ϕ components and the shot noise generator is turned on. One of the ϕ's will then be elicited, provided that the noise level is high enough so that the threshold of the comparing component is exceeded. The expansion coefficients give the probabilities for elicitation of the various ϕ's, but there is no way of knowing precisely which one will be elicited in a given case. Thus the noise level acts in a way similar to the drive level which elicits images. Now suppose that the noise begins at a very low level, too low for the elicitation of one of the ϕ's. The threshold may be exceeded, however, by the result of the combination of the noise with a superposition of two or more of the ϕ's, since this result will be $\int(a_k\phi_k + a_1\phi_1 + \ldots.)dY$ instead of merely $\int a_k\phi_k dY$, and the corresponding elicitation probability will be proportional to $|a_k|^2 + |a_1|^2 + \ldots$ instead of merely $|a_k|^2$. Thus superpositions of images will be elicited, in analogy to the condensations of dream images. In general, at this stage there is no way to refer to a mode except by exciting it, and the association between modes comes from belonging to the same resolution of the wave function, i.e., in our interpretation, belonging to the same point of view.

Now let us examine the second stage, that is, the transformations which the symbols may undergo after having been elicited by the random process. Let us suppose that the elicited output does not come directly from the resolving apparatus (dotted line in Figure 1A) but comes instead via the multiplication component which receives its input from the resolving apparatus (solid line).

Let $\Delta Y(t)$ be the instantaneous output of the noise generator, which would have no well-defined meaning if the noise were ideal shot noise, but in practice will be $Y(t + \Delta t) - Y(t)$ over some short interval of time Δt. Then $\Delta Y(t)$ will be a random variable with a Gaussian distribution which is independent of the distribution of $\Delta Y(t')$ if $|t - t'| > \Delta t$. Under these assumptions, the output will no longer be $a_k \phi_k(t)$, but will be $a_k \phi_k(t) \Delta Y(t)$. Let us suppose that ϕ_1 and ϕ_2 have been elicited with their respective probabilities, so that we now have a new population of wave functions $\phi_1(t) \Delta Y(t - t_1)$ and $\phi_2(t) \Delta Y(t - t_2)$. The ΔY's will be at different times because the modes in the population are separated out at different times. The two modes will be present in the relative proportions $|a_1|^2 : |a_2|^2$. This population, as we shall see in a moment, may be described by the wave function $a_1 \phi_1(t) \Delta Y(t - t_1) + a_2 \phi_2(t) \Delta Y(t - t_2)$, which gives the right probabilities of elicitation, provided that the comparing element averages a population of inputs, because the average of $|\Delta Y(t)|^2$ is always the same constant for all t. Now let us return to our previous discussion of the meaning of the latent information, which represents potentialities in other modes of observation. In the expressions for Ψ in the ϕ and ξ systems, let us replace the exponentials $e^{j\alpha 1}$ and $e^{j\alpha 2}$ by $\Delta Y(t + t_1)$ and $\Delta Y(t + t_2)$, respectively. Averaging over the population and making use of the independence of $\Delta Y(t + t_1)$ and $\Delta Y(t + t_2)$, we find that only the first two terms of the expression in the ξ system remain, the interference terms having disappeared. The elicitation probability for ξ_1 will thus be $|\Psi(1|1)|^2 |a_1|^2 + |\Psi(1|2)|^2 |a_2|^2$, which is the same probability we would have predicted by considering the actual population of wave functions $\phi_1 \Delta Y(t + t_1)$ and $\phi_2 \Delta Y(t + t_2)$ in the relative proportion $|a_1|^2 : |a_2|^2$ rather than the combined wave function $\phi_1 \Delta Y_1 + \phi_2 \Delta Y_2$. This justifies our use of the combined wave function. However, before the wave function has interacted with the eliciting apparatus, $\phi_1 + \phi_2$ will not give the result as a population of ϕ_1's and ϕ_2's because of the interaction terms.

It may also be seen that the probability which has just been given for ξ_1 is precisely that which would have been obtained by using the matrix combination of probabilities instead of the matrix multiplication of probability amplitudes. The preceding argument has thus fulfilled the promise made in the beginning of this paper that an elicitation mechanism would be designed which would make the group properties of transformations between coordinate systems consistent with the necessarily valid rule of composition of response probabilities. This argument is, of course, familiar in quantum mechanics, the only difference being that in quantum mechanics the random multipliers are slightly different (α is a random variable in the factor $e^{j\alpha}$), and that we are using an ordinary mechanical device to do the multiplication. What is new about the argument is that the resulting model is being proposed for a concrete device which has many features of thinking, so we shall now examine the behavioral implications of the above formal mathematical property.

Prior to the interaction with the eliciting device, the modes of any resolution contain all the information needed for their recombination and subsequent resolution in another coordinate system. The original wave function could have been subjected to any kind of observation and thus could have exhibited any one of a number of properties. These properties may be mutually incompatible, so that no individual elicited wave function could exhibit more than one, although they might all be found in a population of responses. An individual wave function elicited by one of the kinds of observation may no longer contain the information which tells about the other kinds of observation, so that awareness of one aspect may preclude simultaneous awareness of other aspects. But all these mutually incompatible properties may simultaneously exist in the form of incompletely developed potentialities in the original wave function. This is another respect in which the first stage of the present model resembles primary process thinking.

We may look at this another way. The original wave function was potentially describable by a wide variety of predicates applicable to the ideas represented by the elicited modes. But interaction with the eliciting apparatus has revealed some potentialities and at the same time has in a precise mathematical way destroyed the information needed to put the modes together again and resolve them in a new way which can reveal other potentialities. Thus the wave function after such interaction can reveal a smaller set of potentialities. It may not yet have been recognized by the recognizing part of the machine, so that from the standpoint of the "awareness" of the machine it may still be unknown, but it will now be an unknown ϕ or else an unknown ξ, rather than an unknown something which could reveal ϕ aspects or ξ aspects, or any number of other potential aspects. Even if the properties coded as ϕ_1, ϕ_2, etc. exhaust all logical possibilities with respect to that aspect of the total pattern, so that in one sense saying that it is a ϕ_1 or ϕ_2, etc. is a tautology, this information nonetheless has important behavioral consequences for the machine even before it is recognized by the machine. We saw mathematically that this will make a difference in any subsequent elicitation of information from another point of view. In addition, knowing which aspect is relevant has important behavioral consequences because it entails using the appropriate rule of ordering and interdependence[1,2] mentioned in the first paragraph of this paper. This is a necessary stage of concept formation which precedes abstraction, and it is important because it tells the machine, in a sense, along what dimension to think. We may summarize this paragraph by saying that the elicitation process, even before its results are "noticed," has partially crystallized the information by limiting the set of potential predicates. If, in our example, ϕ_1 and ϕ_2 each represented a superposition of many modes, this crystallization would have been only partial, for while the ϕ_1 and ϕ_2 have been multiplied by independent ΔY's so that the interaction terms between them have been destroyed, all the modes

within ϕ_1 or ϕ_2 have been multiplied by the same ΔY, which makes no difference within the subspace of their superpositions (since $|\Delta Y|^2$ = constant). Thus within these subspaces potentialities exist which may be destroyed by further specification. All the above corresponds both to theoretical interpretations of underlying thought mechanisms and to common sense observations about gradual articulation of concepts which are not clear to begin with.

Our final observation about the second stage of information elicitation concerns logic. In distinction to the mode of representation of information in the first stage, which permits the same wave function to contain mutually incompatible potentialities, and which permits merging and resplitting, the information is now in a form in which only one of the potentialities has been realized. If the information in this form is utilized by the recognizing part of the machine, it will be displayed in a form which follows the usual rules of probability, as we have seen, and which may be manipulated by the usual logical procedures employed in computers. In fact, it is well known in quantum mechanics that the mathematical formalism used to describe the analogous features of observation--namely the projections of wave functions on particular coordinate axes--can be partially described in terms of the Boolean algebra of propositions.[12] Thus we may summarize this discussion by saying that information is changed in the process of elicitation into a form more amenable to logic, but at the expense of richness of interconnections. Thus the machine may use this modified form to explore the logical consequences of some concept, but in order to do something more creative it may have to return to the original form and start again.

So far we have seen how the model behaves in ways resembling some aspects of perception and thought. Now I shall very briefly indicate one of the ways in which it can learn. We shall assume that in some part of the machine information is elicited from some pool of wave functions, used for some purpose, and then returned to the pool of wave functions. This process is shown in Figure 2, which in addition summarizes some of the preceding discussion. Suppose that the contents of the pool consists of N copies of the wave function Ψ, and that at the moment they are split by the ϕ filters, so that $\Psi = \sum_i a_i \phi_i$, where the ϕ's are symbols for certain responses. Suppose that the shot noise generator is operating so that one of the ϕ's will be elicited, and let us suppose that we are interested in teaching the machine to perform one of the responses, say ϕ_1, rather than any of the other ϕ's. For the first elicitation the probability of obtaining ϕ_1 is $|a_1|^2$. In case this occurs, the operator (or some other part of the machine) "rewards" the machine by turning off the noise generator. Meanwhile, the ϕ_1 has returned to the pool, which now contains N - 1 of the ϕ's and one ϕ_1. The probability of eliciting ϕ_1 has now become $|a_1|^2(N - 1)/N + 1/N$, which is larger than before. The wave functions will be recombined and resolved again in a number of ways, but since the noise generator has been turned off, no information will travel around the cycle, so that no information will be discarded, and there will be no further change in the probability of finding ϕ_1. Now suppose instead that the first trial resulted in the response represented by one of the other ϕ's, say ϕ_2. This time the operator leaves the noise generator on for a while. The ϕ_2 returns to the pool, so that now there are N - 1 of the ϕ's and one ϕ_2. These wave functions become resolved in other coordinate systems, just as in the preceding case, but this time, since the noise generator is operating, information will go around the cycle and some of the components will be lost. Now let us split the resulting wave function into ϕ components for the next trial, and let us see what the new response probabilities will be. Referring back to the discussion of association, we observe that in the process of splitting and recombination the discarded component ϕ_1 will have returned to some extent. Thus the probability for ϕ_2 on the next trial will have increased somewhat, but not as much as that of ϕ_1, had the latter occurred on the first trial. The probability of any one of the ϕ's, say ϕ_j, will become $(1/N)\{1 + (N - 1)\cdot \sum_{n,m,\ldots,k,j} |\Psi(j|n)|^2 |\Psi(n|m)|^2 \ldots |\Psi(k|j)|^2 |a_j|^2\}$, in which the subscripts n, m,...k stand for the various states in the intermediate coordinate systems. The probabilities are not so easy to compute on subsequent trials and depend upon details of how wave functions in the pool become accessible to elicitation, so that further discussion will be postponed until a time when it can be done in a thorough and systematic way. However, it appears that some reinforcement learning is possible, reinforcement acting directly only upon the noise generator and not directly upon the wave functions themselves. The general idea is that the desired function remains, while the others are chopped up. The fact that the probability of elicitation of one of the ϕ's depends upon both the amplitude of the ϕ and the level of the noise, with reinforcement acting only upon the noise, resembles parts of certain theories of learning in animals, but this topic too will be reserved for more systematic discussion when that is possible.

This paper will conclude with a few brief remarks of a more general nature. First of all, the mathematical language employed is well suited to describe both the fluidity of the initial stages of perception and thought, in which ideas have the potentiality of being examined on the basis of a large number of cross-cutting systems, and the final stages in which definite and stable relations may be perceived. The language is also suited to describe a system in which structures are not always defined in advance. As in the illustrative example of microwave modes, a source of excitation which is only moderately complex can interact with a structure which is only moderately complex to produce modes which are tremendously complex, and which could not feasibly be constructed in any other way. In addition, these structures themselves are given the opportunity to become still more highly developed by

an evolutionary process of selection and recombination. In this sense the model resembles the desired arena in which patterns spring up and develop by themselves. In such a system the information is stored in the form of incompletely defined potentialities, which are realized only in interaction with the eliciting and recognizing parts of the system. The percepts, for instance, are in the combination of stages, not in either alone; for in the first stage by itself they contain all points of view at once and none alone, while in the second stage by itself they are in the form of the conventional computer representation of information, which is no more perceptual and subjective than the image on a television screen. The full computer might have to contain a conventional component for reception and useful transformations of the input, followed by the present model, followed by a more or less conventional component to discriminate among the stabilized images or ideas and produce an output. Of course if we were trying to design a machine which could have a knack for skilled actions, the same problems of fluidity of transformations would recur, as we may be convinced by noting that our signature comes out the same even if we hold the pen in the mouth or between the toes.

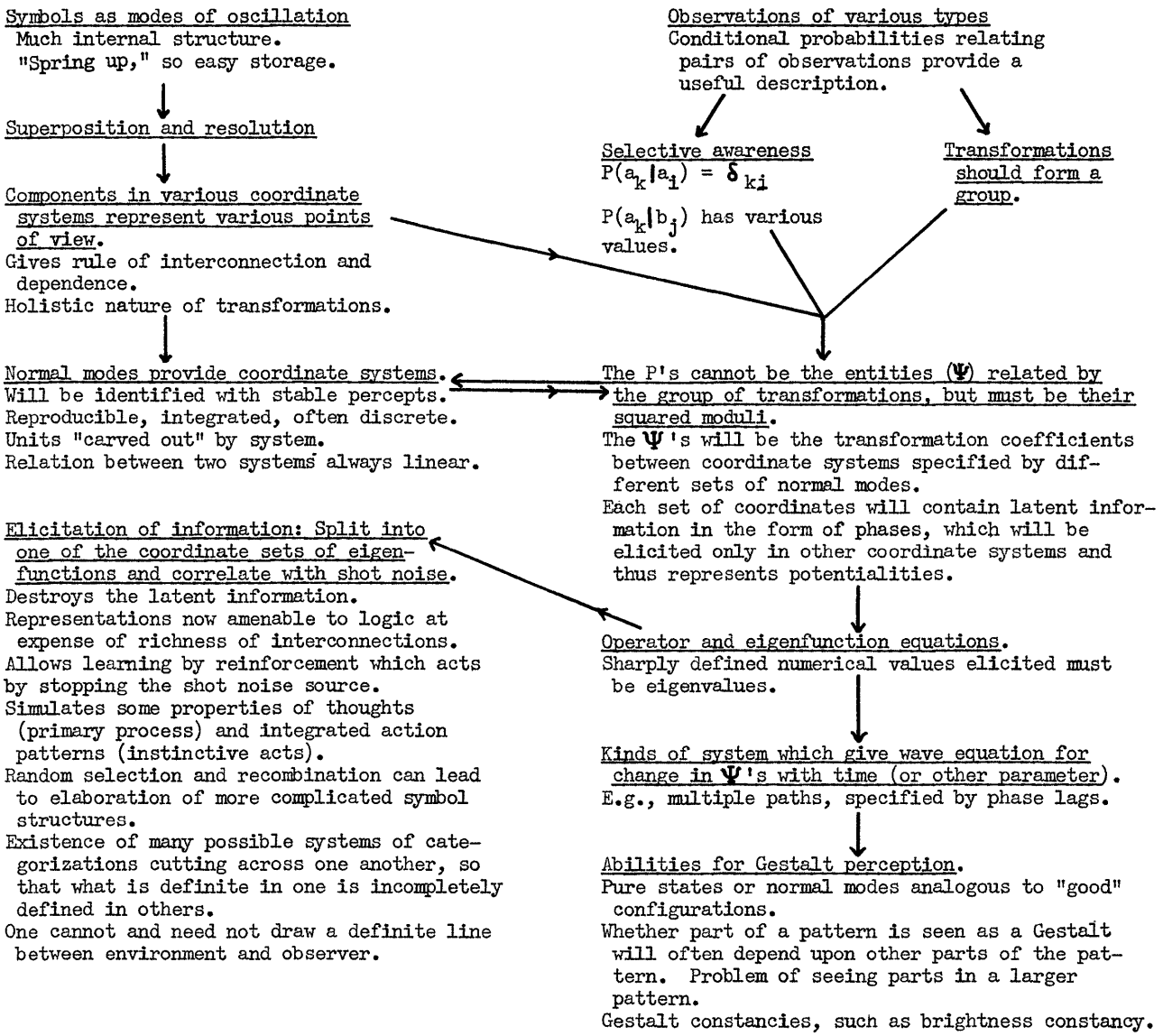
Since the information exists only in the combined input, wave function system, and recognizer, one might wonder where the boundaries are between the observed system, the observing mechanism, and the observer. Von Neumann has shown[12] in the precisely similar mathematical situation in quantum mechanics, that even though the transformations are different in the three subsystems, the combined result is independent of where the boundaries are drawn. Thus although in this system, as well as in the workings of the brain, it is impossible to draw sharp boundaries, perhaps that is not necessary for understanding the processes involved.

Finally, we have seen that this model makes a distinction between items of information which can merge and then exert interacting effects in new coordinate systems, and items which may be sorted and combined but remain separate and have independent effects in any new coordinate systems. It is conceivable that the evolutionary process of selection and recombination could lead to the elaboration of a complicated wave function which contained a large body of information which had become integrated in a way which is necessary for complex behavior. Whether or not such a situation could occur depends upon how clever we are in designing the proper rules of evolution, but the point to be made here is that the mathematical language we are using is (as is known in quantum mechanics) a good language in which to describe such a situation should it occur, or for the purpose of trying to make it occur.

REFERENCES

1. P. H. Greene, "An Approach to Computers That Perceive, Learn, and Reason," Proc. Western Joint Computer Conference; 1959. pp.181-186.
2. P. H. Greene, "The Fundamental Units of Perception: An Outline," Air Force Office of Scientific Research Technical Report, to appear in 1960. Revision of unpublished outline of series of lectures given in the Dept. of Psychology, Northwestern Univ., Spring, 1959.
3. P. H. Greene, "Networks for Pattern Perception," Proc. National Electronics Conference, Vol. 15; 1959.
4. J. C. Shaw, A. Newell, H. A. Simon, and T. O. Ellis, "A Command Structure for Complex Information Processing," Proc. Western Joint Computer Conference; 1958, and Rand Corp. P-1277.
5. P. Schilder, "Mind: Perception and Thought in Their Constructive Aspects," Columbia University Press, New York, N.Y.; 1942.
6. A. Landé, "Quantum Mechanics," Sir Isaac Pitman & Sons, Ltd., London, Eng.; 1951.
7. A. Landé, "Foundations of Quantum Theory: A Study in Continuity and Symmetry," Yale University Press, New Haven, Conn.; 1955.
8. A. Landé, "Quantum Mechanics, From Duality to Unity," Amer. Scientist, 47:341; 1959.
9. N. Wiener, "Nonlinear Problems in Random Theory," The Technology Press of The Massachusetts Institute of Technology and John Wiley & Sons, Inc., New York, N.Y.; 1958.
10. N. Wiener and A. Siegel, "A New Form for the Statistical Postulate of Quantum Mechanics," Phys. Rev., 91:1551-1560; 1953.
11. A. Newell, J. C. Shaw, and H. A. Simon, "Chess Playing Programs and the Problem of Complexity," IBM Journal of Res. and Devt., 2:320-335; 1958.
12. J. von Neumann, "Mathematical Foundations of Quantum Mechanics," translation by R. T. Beyer, Princeton Univ. Press, Princeton, N.J.; 1951.

Table 1. Interdependence of the Arguments



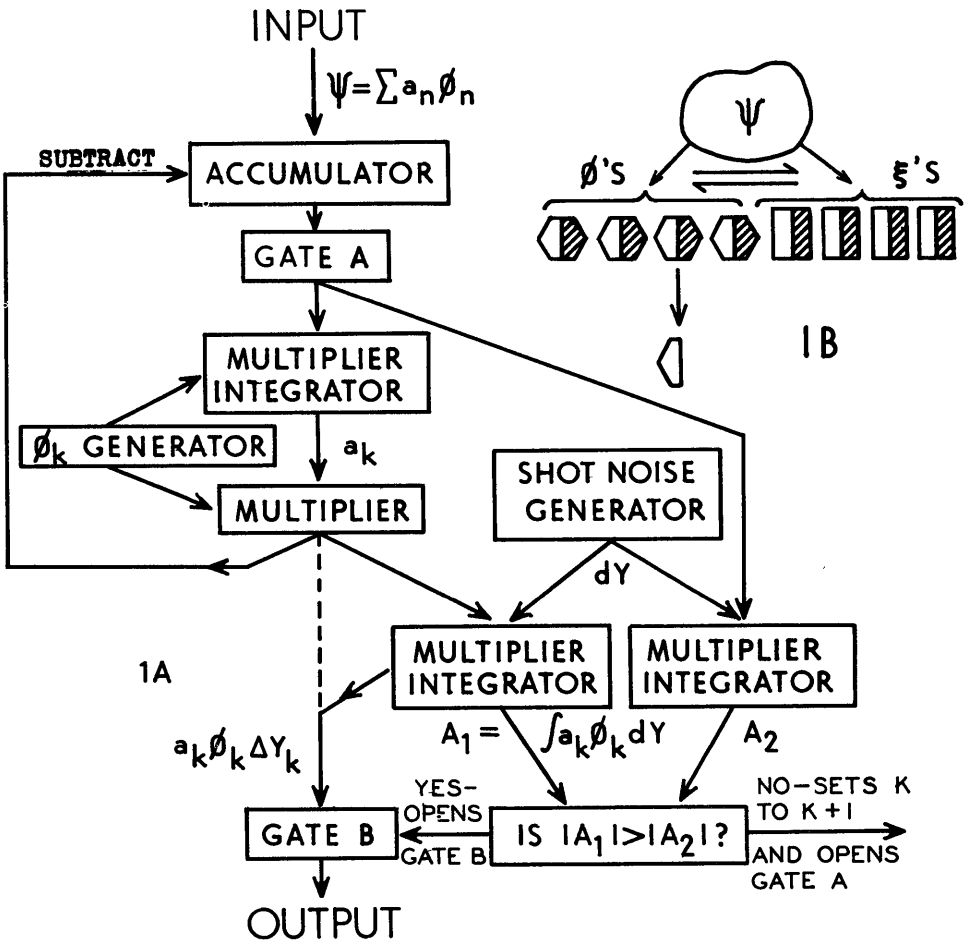


Figure 1 Flow Chart of the Model

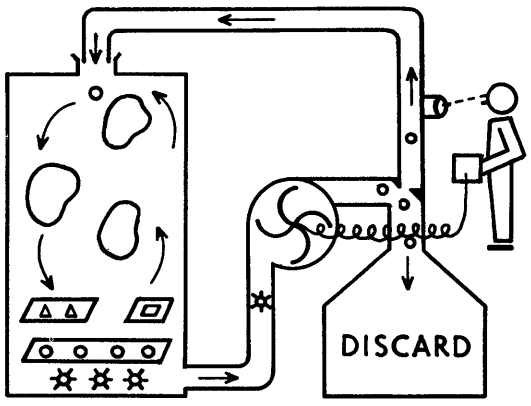


Figure 2
Pictorial representation of the functions of the various parts of the model, illustrating reinforcement learning.

A population of wave functions in the pool (depicted by the large objects circulating in the tank at the left) are split into components of some type (the small circular objects) by the appropriate separating mechanism (the grate with round holes), although they could have been separated into different sorts of components (by the other grates). The components contain information which enables them to combine again (the fringe on the circular objects). The noise which acts as the eliciting mechanism is depicted by the pump at a height representing the threshold. There is only room for one component at a time to pass into the pipe leading back to the information pool, so that all but the first component to reach that pipe will drop down and be discarded. In passing through the pump the latent information allowing recombination has been destroyed (the circles are now smooth). The man represents the operator of the machine or another part of the machine which can observe the action represented by the component which has been elicited. To reinforce that response the man turns off the pump; if another component has come through instead, he merely leaves the pump on for a while. He acts only on the pump; the elicited component returns to the pool in either case.

ANALOG COMPUTER TECHNIQUES FOR PLOTTING BODE AND NYQUIST DIAGRAMS

by

G. A. Bekey
L. W. Neustadt

Members of the Technical Staff
Space Technology Laboratories, Inc.
Los Angeles, California

Summary

Two methods of plotting Bode and Nyquist diagrams with an analog computer are presented. Both methods have the advantage of being able to give the open loop response from a closed loop simulation on the computer, which makes them applicable to systems with open loop poles in the right hand plane. Neither method requires any adjustment by the operator, other than to change the frequency of the input signal.

The first method makes use of some trigonometric identities, which yield functions which identically equal the real and imaginary parts of points in the Nyquist diagram. It makes rather extensive use of multiplying equipment, but no other non-linear equipment is required.

The second method makes use of diode and relay switching techniques to measure phase, and peak value readers to measure amplitude. One divider and no multipliers are used, but diodes and relays are necessary.

Nyquist and Bode plots are presented, as computed by both methods.

Introduction

In the study of linear feedback control systems by means of analog computers it is frequently important to examine the frequency response of the simulated system. Commercially available equipment can be used to obtain amplitude ratio and phase as a function of frequency for an open loop system. Unfortunately, many practical control systems are either unstable open loop, or they contain a number of integrations which result in considerable drift unless the loop is closed. The techniques presented in this paper resulted from an investigation of methods for obtaining the open loop phase and gain characteristics of the control system from its closed loop simulation on the analog computer. The information can be plotted either in Nyquist or Bode form.

Two alternative methods are presented in the paper. One of these makes use of trigonometric relations and obtains the coordinates of a point on the Nyquist diagram directly. The other uses filtering and switching schemes to obtain the coordinates of a point on a Bode diagram. Both of these methods are capable of obtaining the information from a closed loop simulation and make use of standard analog computer components.

Statement of the Problem

I. Open Loop Measurement

Consider the simple control system illustrated in Figure 1, where the forward transfer function is indicated by $G(s)$. If the system is stable open loop, and has no poles at the origin, it is possible to apply signal of known phase, amplitude, and frequency to the system represented by the transfer function $G(s)$. Thus, if we let

$$e(t) = E \sin \omega t \tag{1}$$

we obtain

$$c(t) = C \sin (\omega t + \phi) \tag{2}$$

and the open loop gain K is given by

$$K(\omega) = \frac{C(\omega)}{E} = \left|G(j\omega)\right| \tag{3}$$

while the phase is given by the angle ϕ, where

$$\phi(\omega) = \arg G(j\omega) \tag{4}$$

The Bode diagram consists of the quantities $K(\omega)$ and $\phi(\omega)$ plotted vs. $\log \omega$. The amplitude ratio or gain is generally expressed in db. The Nyquist diagram is the complex plot of the function

$$z(\omega) = K(\omega) \cos \phi(\omega) + j\, K(\omega) \sin \phi(\omega) \tag{5}$$

Most commercially available "control system analyzers" (Fuchs[1]) make use of these relations, and thus assume that the block $G(s)$ is available and that it can be supplied with a controllable input signal. Various analog techniques are possible for measuring the phase-gain characteristics in this case. Some typical techniques are the following:

(a) The in-phase and out-of-phase components can be evaluated by nulling the in-phase and out-of-phase components of the output with manually attentuated sine and cosine voltages. This technique is described by Larrowe[2].

(b) The in-phase and out-of-phase components of the output can be obtained from calculation of the respective Fourier coefficients of the output.

(c) A variable phase shifter and a filtering technique can be used, as described by Cowley[3]. This method relies on careful control of phase and amplitude of the input signal to the network in question.

(d) Perhaps the most commonly used, and crud-

[1]Numbered superscripts refer to references at the end of the paper

est method possible, is the comparison of input and output waves on adjacent traces of a strip chart recorder. Clearly, the measurements obtained by this method are approximate at best. The phase measurement can be refined through the use of Lissajous figures with either an oscilloscope or an x-y plotter.

In the literature, these methods are described for open loop operation only, and some require extensive modification for closed loop operation.

II. Closed Loop Measurement

Consider the block diagram of Figure 1 once again. It is desired to obtain the phase and gain characteristics of the transfer function G(s) from a closed loop simulation on the analog computer. The requirements for a satisfactory frequency response measuring scheme are taken to be the following:

(a) It must be capable of obtaining the open loop phase and gain characteristics of the system without breaking the loop.

(b) It must be capable of mechanization with standard computer components. If

$$r(t) = R \sin \omega t \tag{6}$$

then we can write the following steady-state relations:

$$e(t) = E \sin (\omega t - \theta) \tag{7}$$

and

$$c(t) = C \sin (\omega t - \theta + \phi) \tag{8}$$

The desired amplitude ratio is given by

$$K(\omega) = \frac{C(\omega)}{E(\omega)} = \left| G(j\omega) \right| \tag{9}$$

and the phase shift by the angle $\phi(\omega) = \arg G(j\omega)$.

There are several possible approaches to this problem. Two approaches will be described in this paper while a third one is described in Reference 4.

Trigonometric Method

The method described below yields a Nyquist plot by making use of some simple trigonometric relations.

I. Open Loop Computation

Let the input to a network be sin ωt, and let its steady-state output be $y(t) = K \sin (\omega t + \phi)$. Let us assume that all the poles of G(s) have negative real parts. Then in the steady state

$$y(t) = A \sin \omega t + B \cos \omega t \tag{10}$$

$$\frac{\dot{y}(t)}{\omega} = A \cos \omega t - B \sin \omega t \tag{11}$$

where

$$A = K \cos \phi \tag{12}$$

and

$$B = K \sin \phi \tag{13}$$

From (10) and (11), we have

$$A \equiv y(t) \sin \omega t + \frac{\dot{y}(t)}{\omega} \cos \omega t \tag{14}$$

$$B \equiv y(t) \cos \omega t - \frac{\dot{y}(t)}{\omega} \sin \omega t \tag{15}$$

The Nyquist plot is given by $z(\omega) = A(\omega) + j\, B(\omega)$.

Figure 2 illustrates the set-up which was used. A network with transfer function

$$G(s) = \frac{\omega_o^2}{s^2 + 2\xi\omega_o s + \omega_o^2}, \quad (\text{with } \omega_o = 1, \xi = 0.2)$$

was simulated. The output was plotted on an x-y plotter for discrete values of ω, and these points are shown on Figure 3, with the theoretical curve given for comparison. For each frequency the value of ω was set into the servo to position the three pots, and after the transient died out the values of A and B were recorded. Although electronic multipliers were used, servo multipliers could have been used just as well, if A and B had been evaluated in the "hold" mode (Relations (14) and (15) are identities).

Note that $\dot{y}(t)$ is easily available whenever the denominator of G(s) is of higher order than the numerator. If numerator and denominator have the same order, $\frac{\dot{y}(t)}{\omega}$ may be obtained from a "90° phase shifter", e.g. a network whose transfer function is $\frac{s-\omega}{s+\omega}$.

II. Closed Loop Computation

If G(s), the open loop transfer function, has poles on the imaginary axis or in the right hand plane, there is no steady state and the open loop response must be computed with the loop closed.

We have the following closed loop relations (assuming the input is sin ωt, the output is c(t), and the transient is zero):

$$e(t) = \sin \omega t - c(t) = E \sin (\omega t - \theta) \tag{16}$$

$$\frac{\dot{e}(t)}{\omega} = \cos \omega t - \frac{\dot{c}(t)}{\omega} = E \cos (\omega t - \theta) \tag{17}$$

$$c(t) = KE \sin (\omega t - \theta + \phi) \tag{18}$$

$$= AE \sin (\omega t - \theta) + BE \cos (\omega t - \theta)$$

where

$$A = K \cos \phi \tag{19}$$

$$B = K \sin \phi \tag{20}$$

$$\frac{\dot{c}(t)}{\omega} = AE \cos (\omega t - \theta) - BE \sin (\omega t - \theta) \tag{21}$$

By combining the above relations we obtain

$$AE^2 = c(t)\, e(t) + \frac{\dot{c}(t)}{\omega} \frac{\dot{e}(t)}{\omega} \tag{22}$$

$$BE^2 = c(t) \frac{\dot{e}(t)}{\omega} - \frac{\dot{c}(t)}{\omega} e(t) \tag{23}$$

$$E^2 = [e(t)]^2 + \left[\frac{\dot{e}(t)}{\omega}\right]^2 \tag{24}$$

Again the Nyquist plot is given by $z(\omega) = A(\omega) + j\, B(\omega)$. A computer diagram for solving equations (16), (17), (22), (23), and (24) is given by Figure 4. The solution for discrete values of ω is shown in Figure 5, together with the theoretical Nyquist plot. The transfer function G(s) was chosen as $(1 + \tau s)/s^2$ with $\tau = 0.55$.

The same comments relative to electronic multipliers vs. servos, and computation of $\dot{c}(t)$ explicitly apply to the closed loop and open loop simulations.

Although this method appeared suited for drawing continuous Nyquist plots, by letting ω vary continuously in the sine wave generator, attempts at this were not successful.

The advantages of this method are the following: Standard elementary analog circuits are used. No "peak value" readers, sampling circuits, or relay logic circuits are required, since the values A and B are computed for all values of t. With further investigation this method may therefore be adaptable to a continuous plot. No resolver is necessary since instead of amplitude and phase, the real and imaginary parts are computed. Furthermore, no adjustment by the operator, other than variation of frequency, is required. The main disadvantage lies in the amount of multiplying equipment required - particularly in the closed loop calculation.

Switching and Sampling Method

The method described below yields phase and gain information directly, for plotting a Bode diagram. The computation of gain and phase are done separately. The method is described only for closed loop computation. It can be applied to open loop calculations with obvious simplifications.

I. Amplitude Ratio Computation

The gain is obtained by measuring the amplitude of the input and output signals and dividing. The measurement can be done in a number of ways, two of which are shown in Figures 6a and 6b, respectively:

(a) The two signals are full-wave rectified and filtered. The resulting dc voltages are divided. (See Figure 6a)

(b) "Peak reading" circuits can be used to read the positive or negative peaks of the two signals. The gain is then given by the ratio of the peak values. (See Figure 6b)

For Bode diagram purposes, the resulting signal can be expressed in db, as shown in Figure 6 with a function generator.

II. Phase Measurement

The phase shift is measured by using diode and relay switching techniques. Consider the waveforms shown in Figure 7. Typical waveforms for e(t) and c(t) for a lagging phase shift are shown in 7a and 7b. If these two sine waves are converted into equal amplitude square waves, we obtain Figures 7c and 7d. The phase shift can be measured by evaluating the shaded area indicated in 7d, as a fraction of the total area for one-half period. The area measurement is done with an integrator, as shown schematically in Figure 8. The following comments refer to Figure 8:

(a) The error signal e(t) is used to control a switch selector system. When e(t) becomes positive the output signal c(t), converted to a square wave c'(t) of known amplitude, is applied to integrator 1. When e(t) becomes negative, the output signal is disconnected from the integrator.

Thus, the integrator input is given by:

$$\begin{cases} i(t) = c'(t) & \text{for } e(t) > 0 \\ i(t) = 0 & \text{for } e(t) \leq 0 \end{cases} \tag{25}$$

This wave form is shown in Figure 7e.

(b) A 90° phase shifter is employed to obtain a control signal p(t) equal in amplitude to e(t). The phase shifter is the same used in the previous method, i.e.

$$G_{ps}(s) = \frac{s - \omega}{s + \omega}$$

(c) When e(t) becomes negative, the integrator output is sampled for the following quarter cycle, while the second control signal p(t) remains positive.

(d) When p(t) becomes negative, it provides a control signal to reset integrator 1 to zero. A quarter-cycle is provided for the reset operation.

(e) As soon as e(t) becomes positive, the series of operations is repeated.

The waveforms appearing at the output of the integrator are given in Figure 7f. Since the sample-and-hold circuit only samples the final value of the integration, its output remains constant after steady-state is attained. From an examination of Figures 7d and 7e, it can be seen that the integrator output reaches a value proportional to the area b minus the area a. Now, if the integrator input has unity amplitude, it can be seen that

$$a = \frac{\phi}{\omega} \tag{26}$$

$$b = \frac{\pi - \phi}{\omega} \tag{27}$$

The final value of the integration is then

$$I = b - a = \frac{\pi - 2\phi}{\omega} \tag{28}$$

and thus, the phase angle ϕ is obtained from the relation

$$\phi = -\frac{I\omega - \pi}{2} \tag{29}$$

This computation is performed by means of the circuitry shown in the block diagram of Figure 8.

Similar reasoning applies to the leading phase shift situation, in which case the area a in Figure 7e is positive instead of negative. This polarity change can be used with additional logic to indicate lead or lag.

III. Advantages and Disadvantages

This sampling method has the clear advantage of requiring no complex equipment such as multipliers for phase determination. The computer does not need to be reset to observe changes resulting from variation of frequency or system parameters. Since relays and diodes are relatively inexpensive, a special purpose computer for this purpose could be constructed. Theoretically, the method can be extended to continuous variation of frequency.

The application of this method to the computation of phase and gain for a simple system, with transfer function $G(s) = (1 + \tau s)/s^2$, resulted in the curves of Figure 9 and Figure 10. In these two

curves the experimental data are given by plotted points and the theoretical curves are drawn for comparison.

Conclusion

The two methods described above show that it is practical to obtain Bode and Nyquist information on the analog computer from closed loop simulations.

References

1. Fuchs, A.M. "Control System Test Equipment, Part II: Complete Test and Evaluation Systems", Control Engineering, v.6, pp. 125-132 (May 1959)

2. V. L. Larrowe, M. M. Spencer, S. R. Lampert, University of Michigan, "Application of a Gas Turbine Engine Performance Computer to a Twin Spool Engine", Report IP - 250, (Nov. '57)

   The circuit is also described in "The Simulation Council Newsletter", Instruments and Automation, v. 31, pp. 1691-1696, (Oct. 1958)

3. Cowley, Percy, "The Application of an Analog Computer to the Measurement of Process Dynamics", ASME Paper No. 56 - IRD - 20 (1956)

4. Goldberg, E. A. "An Analog Computer Nyquist Plotter", IRE Convention Record (1960)

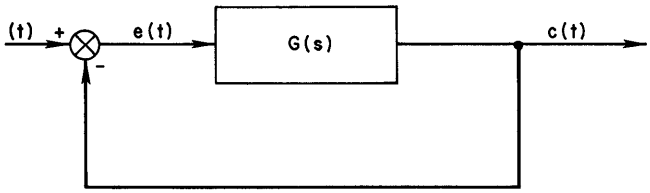
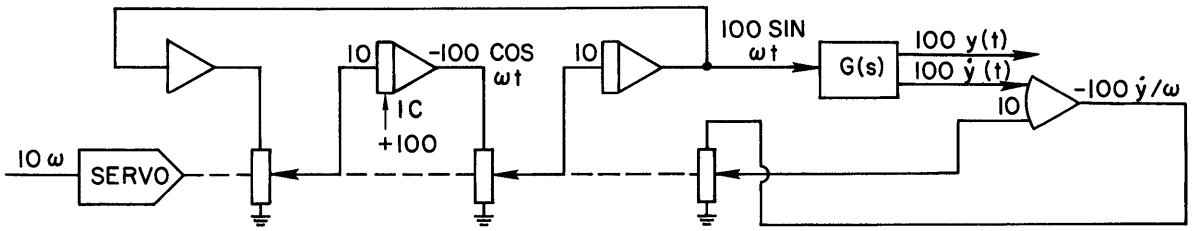


Figure 1
SIMPLE FEEDBACK SYSTEM



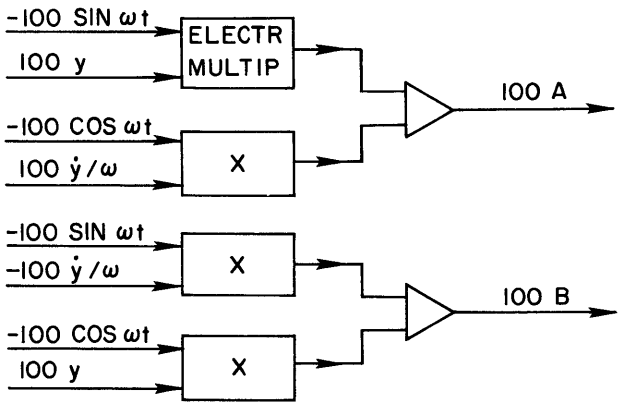


Figure 2
OPEN LOOP COMPUTER DIAGRAM

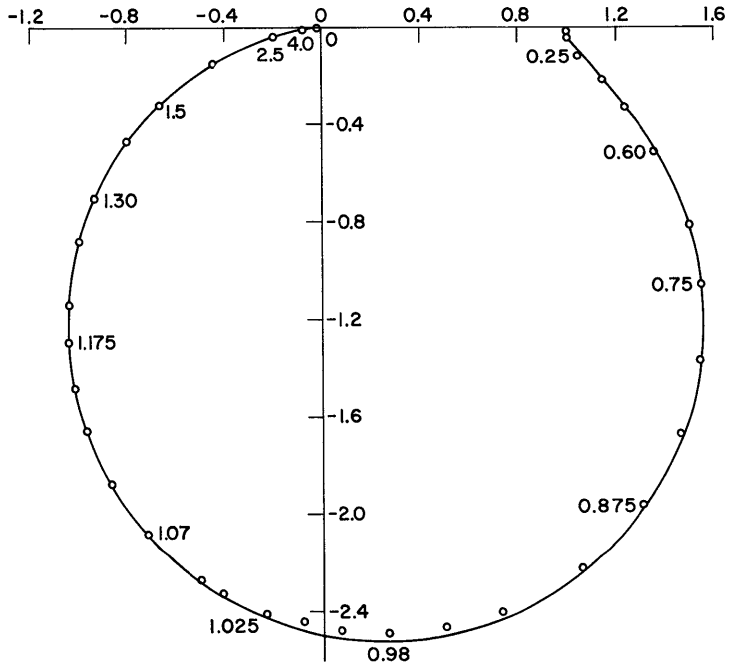


Figure 3
OPEN LOOP NYQUIST PLOT

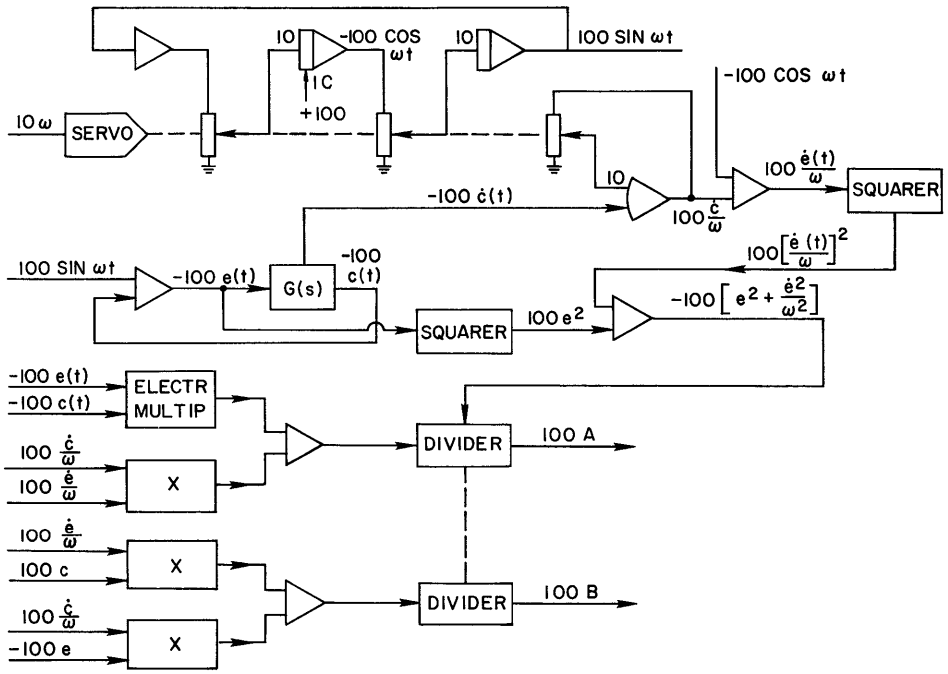


Figure 4
CLOSED LOOP COMPUTER DIAGRAM

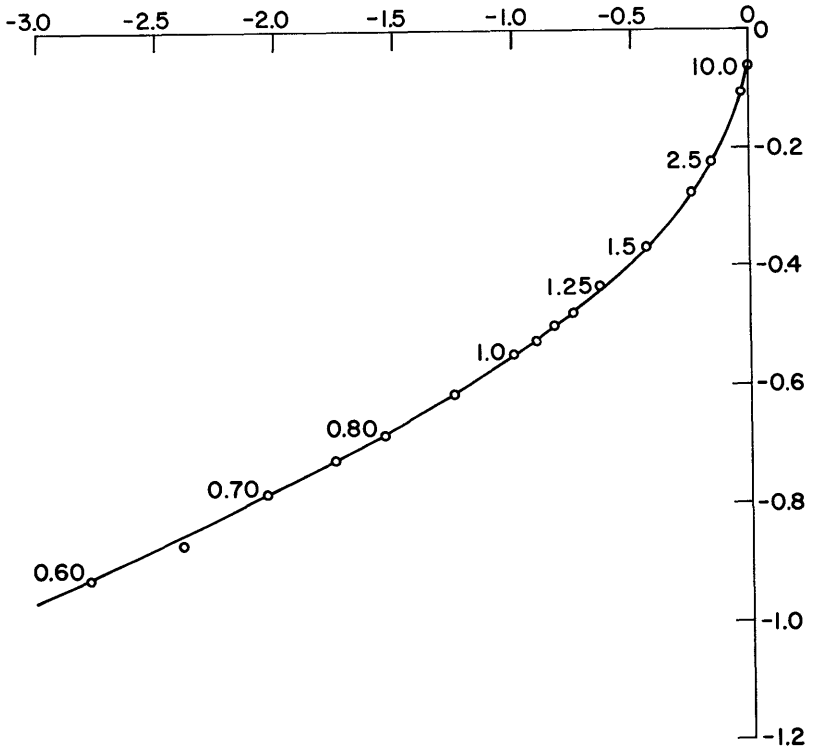


Figure 5
CLOSED LOOP NYQUIST PLOT

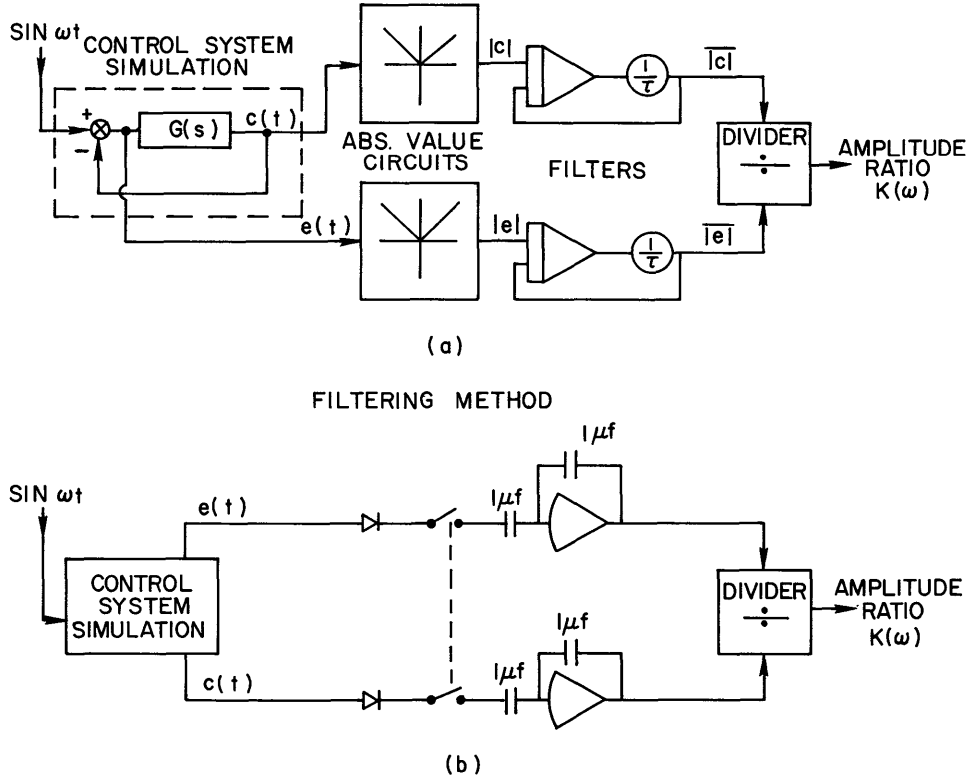


Figure 6
TWO WAYS OF DETERMINING AMPLITUDE RATIO

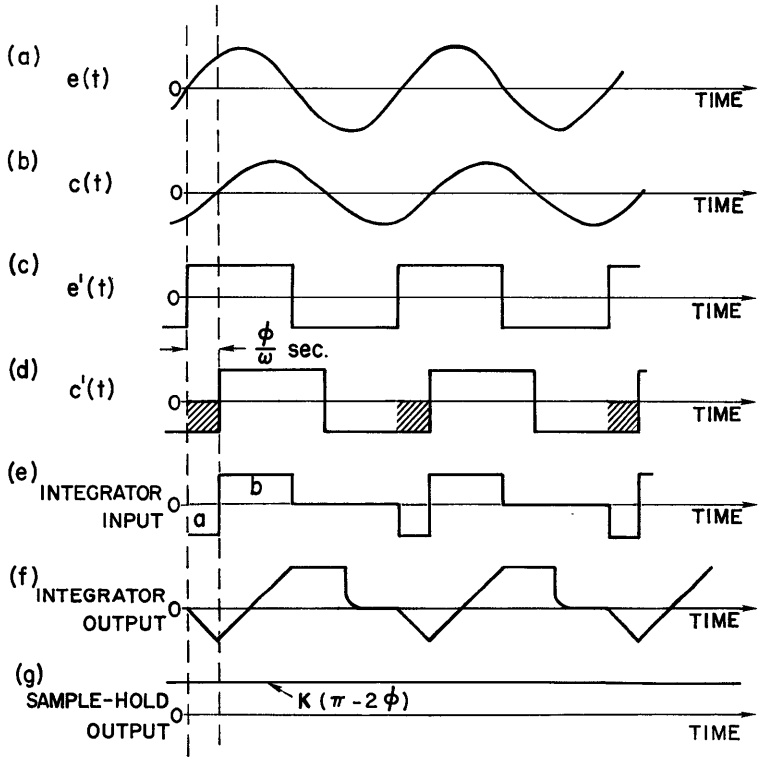


Figure 7
STEADY-STATE WAVE FORMS IN SAMPLING METHOD

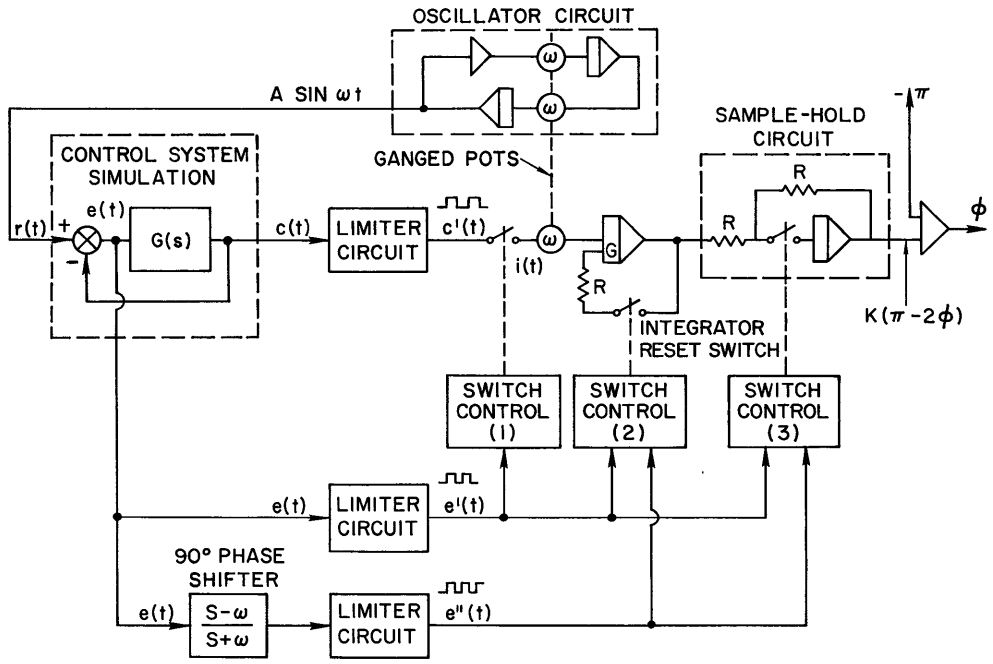


Figure 8
SWITCHING AND SAMPLING METHOD OF PHASE MEASUREMENT

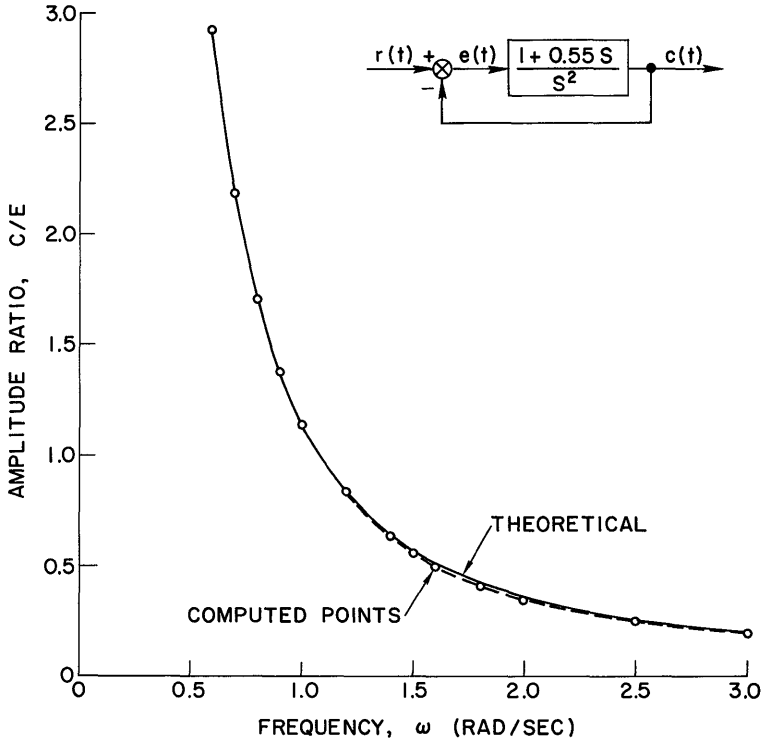


Figure 9
THEORETICAL AND COMPUTED AMPLITUDE RATIO
$[G(j\omega)]$ FOR THE SYSTEM SHOWN ABOVE

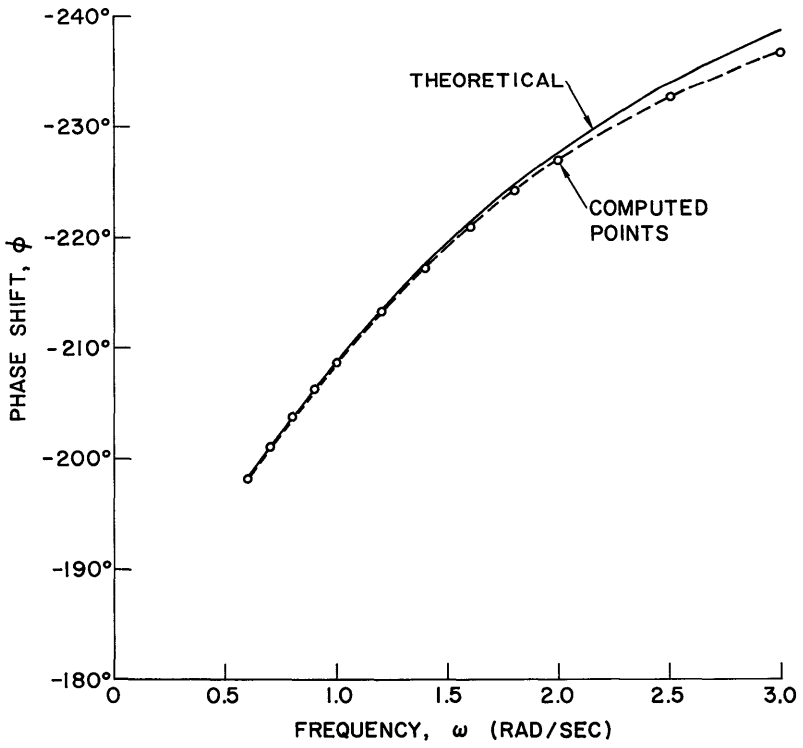


Figure 10
THEORETICAL AND COMPUTED PHASE SHIFT FOR EXAMPLE SYSTEM

ON THE REDUCTION OF ERROR IN CERTAIN ANALOG COMPUTER CALCULATIONS BY THE USE OF CONSTRAINT EQUATIONS

Robert M. Turner
Lockheed Aircraft Corporation - Missiles and Space Division
1123 North Mathilda Avenue, Sunnyvale, California

This paper describes a method of reducing the error in certain analog computer calculations by the negative gradient technique where one has additional information about system performance in the form of constraint equations. In general, the technique leads to correction terms which are introduced into the system differential equations. In some cases, however, simulation is obtained by operating directly with the constraint relations themselves.

The theory is developed and several examples discussed at some length in order to show both the manner in which the corrections may be introduced, and the character of the corrected solutions.

Theory

Let the equation set I represent the system under investigation; i.e., it is the basic differential equation set describing the physics of the situation. Since any nth order differential equation may be written in the form of n first order equations no generality is lost and considerable convenience gained by adopting this presentation. The x_r are independent or driving functions.

$$\text{I}\quad\begin{cases}\dot{y}_1 = f_1(x_1,x_2,\ldots x_p;y_1,y_2,\ldots y_n)\\ \dot{y}_2 = f_2(x_1,x_2,\ldots x_p;y_1,y_2,\ldots y_n)\\ \cdot\\ \cdot\\ \cdot\\ \dot{y}_n = f_n(x_1,x_2,\ldots x_p;y_1,y_2,\ldots y_n)\end{cases}$$

Let the equation set II describe the constraints in (or to be imposed upon) the physical system I.

$$\text{II}\quad\begin{cases}\epsilon_1 = \epsilon_1(x_1,x_2,\ldots x_p;\dot{y}_1,\dot{y}_2,\ldots\dot{y}_n;y_1,y_2,\ldots y_n)\\ \cdot\\ \cdot\\ \cdot\\ \epsilon_q = \epsilon_q(x_1,x_2,\ldots x_p;\dot{y}_1,\dot{y}_2,\ldots\dot{y}_n;y_1,y_2,\ldots y_n)\end{cases}$$

The set II is written such that ϵ_j equals zero for all j. Now since the x_r are considered to be independent variables or driving functions, any deviation of the ϵ_j from zero must be due to incorrect values obtained from the y_i^u in the solution of set I, (where u = 0 for y_i and 1 for $\dot{y}_i$).

Since we wish to consider all the ϵ_j simultaneously, we take as our optimization criteria the minimization of a non-negative function E of ϵ_j where

$$E = \sum_{j=1}^{q} (\epsilon_j)^2 \qquad (1)$$

Now for any given value of x_r and y_i^u we may find the change required in each of the y_i^u making up E by considering E as a surface in Euclidean N space. The vector giving the direction of greatest change in E is the gradient vector, ∇E. Since we are interested in the direction of greatest decrease of the function, we turn our attention to the negative gradient, which has the components

$$-\nabla E = 2\sum_{j=1}^{q} \epsilon_j \frac{\partial \epsilon_j}{\partial y_i^u} \text{ for } \begin{cases} u = 0,\ 1\\ i = 1,2,\ldots n\end{cases} \qquad (2)$$

Then we may assign to each Δy_i^u, where Δy_i^u is defined as the correction term to be added to the y_i^u obtained from the solution of I, the value

$$\Delta y_i^u = -\sum_{j=1}^{q} K_j \epsilon_j \frac{\partial \epsilon_j}{\partial y_i^u} \text{ for } \begin{cases} u = 0,\ 1\\ i = 1,2,\ldots n\end{cases} \qquad (3)$$

That is, we add to each y_i^u a value proportional to the component of $-\nabla E$ in the y_i^u direction. K_j is a weighting factor determined by the importance attached to the particular ϵ_j. Depending upon the process of gradient determination and subsequent correction insertion one has either a true gradient method or steepest descent method. For the applications in this paper, the gradient is determined and the corrections added continuously. Thus no sequential series of steps is taken along one gradient before the next is determined.[1,2]

If we add these corrections directly and continuously to the positional values of the y_i^u, each member of the corrected differential equation set becomes

$$\begin{aligned}\dot{y}_i' &= f_i[x_1,x_2,\ldots x_p;y_1,y_2,\ldots y_n] + \Delta\dot{y}_i\\ y_i &= \int \dot{y}_i' dt + \Delta y_i\end{aligned} \qquad (4)$$

where $\dot{y}_i'$ is the corrected rate term.

Note that Δy_i is not the integral of $\Delta\dot{y}_i$ as each arises independently from (3). We call this the E_p correction of I.

This method is in general quite difficult to implement on an analog computer due to the formation of algebraic loops in the generation of the error

terms. We may, however, take a different point of view and consider each Δy_i^u to arise from a change in the defining rate of y_i^u; i.e., we put

$$\frac{d}{dt}(\Delta y_i^u) = -\sum_{j=1}^{q} K_j \epsilon_j \frac{\partial \epsilon_j}{\partial y_i^u} \quad \text{for} \begin{cases} u = 0,1 \\ i = 1,2,\ldots n \end{cases} \tag{5}$$

This also will give to each y_i^u a correction in the desired direction but in quite a different manner due to the integration process. Thus each member of the corrected differential equation set becomes

$$\dot{y}_i = f_i(x_1,x_2,\ldots x_p;y_1,y_2,\ldots y_n) + \Delta y_i + \int \Delta \dot{y}_i \tag{6}$$

since when $u = 1$ in (5) a second derivative term in Δy_i arises which cannot be inserted in $\dot{y}_i$ without increasing the order of the system. We call (6) the E_R correction of I.

For certain algebraic applications (see Example 1) the system order is indeed increased, but in general, this is not desirable. Direct positional insertion may be used, however, so that both E_P and E_R corrections are applied in the same system. In this case, (6) becomes

$$\dot{y}_i = f_i(x_1,x_2,\ldots x_p;y_1,y_2,\ldots y_n) + \Delta y_i + \Delta \dot{y}_i \tag{7}$$

Example (4) considers this in some detail.

Simply stated, equation (3) shows that corrections may arise for all functions and their derivatives which occur explicitly in the constraint equations. Equations (4), (5), (6) and (7) show how the corrections can be added to the system I.

We shall next illustrate E_P and E_R corrections as applied to certain problems of a general nature arising in analog simulations. The E_R simulation is usually quite easy to obtain from the original system mechanization, since in general it only involves adding the correction terms into the several integrators. This, of course, enables one to compare corrected and uncorrected performance quite readily. E_P correction on the other hand is not so straight forward, and as previously noted, generally gives rise to stability problems on analog equipment.

Examples

Example 1

Consider the desired calculation

$$y_1 = \frac{x_2}{x_1} \tag{1.1}$$

Utilizing the zero order constraint, form

$$y_1 x_1 - x_2 = \epsilon \tag{1.2}$$

Taking

$$y_1 = y_1(0) + \Delta y_1 \tag{1.3}$$

Differentiating and applying (5)

$$\dot{y}_1 = \frac{d}{dt}(\Delta y_1) = -K_1 \epsilon_1 \frac{\partial \epsilon_1}{\partial y_1} = -K_1 \epsilon_1 x_1 \tag{1.4}$$

Thus for quasi-static x_1, x_2

$$y_1 = \frac{x_2}{x_1} + \left[y_1(0) - \frac{x_2}{x_1} \right] e^{-K_1 x_1^2 t} \tag{1.5}$$

For $x_1 > 0$ we note the possibility of incorporating it into the constant K_1. This saves one multiplication; i.e., put

$$\frac{d}{dt}(\Delta y_1) = -K_1 \epsilon_1 \quad \text{for} \quad x_1 > 0 \tag{1.6}$$

Table I shows the type of convergence to the correct solution for several cases where $x_2 = x_1$. Only the exponents are shown since the solution form is the same as (1.5). Note that for the case $x_1 = x_2 = \sin \omega t$ (1.4) will not give correction until t becomes large enough to offset $\frac{\sin 2\omega t}{2\omega}$.

Further investigation of the (non-linear) equations (1.4) and (1.6) is required to establish convergence criteria for arbitrary x_1 and x_2. Equation (1.4) is mechanized in Figure (1.2) with the aid of the symbols in Figure (1.1).

TABLE I

Using (1.4)	Using (1.6)
x_1 and x_2 constant: $-K_1 x_1^2 t$	$-K_1 x_1 t$
$x_1 = x_2 = t$: $-K_1 t^3/3$	$-K_1 t^2/2$
$x_1 = x_2 = \sin \omega t$: $\frac{K_1}{2}(t - \frac{\sin 2\omega t}{2\omega})$	Not applicable since x_1 is not always > 0
$x_1 = x_2 = A + \sin \omega t$: $-K_1\left[A^2 t + t/2 + 2A/\omega - 2A/\omega \cos\omega t - 1/4\omega \sin 2\omega t\right]$	For $A > 1$ $- K_1[At + \frac{1}{\omega}(1-\cos \omega t)]$

In Figure 1.2 G is to be as large as possible since it effectively increases $K_1 . x_1, x_2, N_1$ and N_2 are to be taken as quasi-static where N_1 and N_2 are errors which exist at the points indicated. The instantaneous differential equation is

$$\dot{y}_1 = -K_1(\epsilon_1 + N_1)x_1 + N_2 \tag{1.7}$$

which yields the instantaneous solution form

$$y_1 = y_1(0)e^{-K_1 x_1^2 t} + \left[\frac{x_2}{x_1} - \frac{N_1}{x_1} + \frac{N_2}{K_1 x_1^2}\right](1 - e^{-K_1 x_1^2 t}) \tag{1.8}$$

Attenuation of the error through the first multiplication channel is as $\frac{1}{x_1}$ and through the second by $\frac{1}{K_1 x_1^2}$.

Because of this noise attenuation it becomes practical to use division in forming special

functions; for example:

Given $\dot{y}$ to form $z = \ln y$ we note

$$\dot{z} = \frac{\dot{y}}{y} \tag{1.9}$$

Here $\dot{z}$ is formed by the division circuit, then integrated to give z. The method is useful provided z is used ultimately in the equation for $\dot{y}$ (i.e., the z integration is not open loop). A particular advantage obtains in that the range of the ln function is determined merely by an initial condition on the integrator.

Notice that when (1.6) is applicable, two independent x_2/x_1 and x_3/x_1 divisions are obtainable with one multiplier. For either (1.4) or (1.6), $\dot{y}$ is available, though greatly attenuated and noisy.

Example 2

Inverse function generators in the feedback loop of high gain amplifiers are frequently used to generate monotonic functions having steep slopes. The method is advantageous because of the reciprocal relation between the slopes. Approaching this problem from the point of view of a constraint relation yields the following.

Let $y = f(x)$ be the desired function, given x.

Now

$$f(\tfrac{1}{y}) = x \quad \text{and} \quad \epsilon \equiv f(\tfrac{1}{y}) - x \tag{2.1}$$

Using (5):

$$\frac{d}{dt}(\Delta y) = \dot{y} = -K\epsilon \frac{\partial \epsilon}{\partial y} \tag{2.2}$$

For $y = \sqrt{x}$ we see that $f(\frac{1}{y}) = y^2$ and so

$$\dot{y} = -2K\epsilon y;\ y(0) = \sqrt{x(0)} \tag{2.3}$$

$$\epsilon = y^2 - x \tag{2.4}$$

Performance is good except near $x = 0$. (2.3) is mechanized in Figure 2.1.

Example 3

The problem of finding R, sin φ and cos φ given x and y frequently arises in coordinate transformations. When $\dot{\phi}$ is not available and the frequency too high for standard circuits, the following method has been used with success.

Since

$$\left.\begin{aligned} x^2 + y^2 &= R^2 \\ x &= R\cos\varphi \\ y &= R\sin\varphi \end{aligned}\right] \tag{3.1}$$

we put

$$\epsilon_1 = R\cos\varphi - x \tag{3.2}$$

$$\epsilon_2 = R\sin\varphi - y \tag{3.3}$$

Note that

$$R = x\cos\varphi + y\sin\varphi \tag{3.4}$$

Thus ϵ_1 and ϵ_2 enable us to find sin φ and cos φ; (3.4) determines R.

Using (5)

$$\frac{d}{dt}(\Delta\cos\varphi) = -2K_1\epsilon_1 \frac{\partial \epsilon_1}{\partial \cos\varphi} \tag{3.5}$$

$$\equiv -K\epsilon_1 \text{ for } R > 0$$

$$\frac{d}{dt}(\Delta\sin\varphi) = -2K_1\epsilon_2 \frac{\partial \epsilon_2}{\partial \sin\varphi} \tag{3.6}$$

$$\equiv -K\epsilon_2 \text{ for } R > 0$$

(which are mechanized in Figure 3.1)

The previous examples have illustrated methods of using the system equations themselves to form error (or constraint) equations. Minimizing these led directly to a solution of the system equations.

We shall next consider two examples of systems which are amenable to error correction in the more general sense of constraining the solution of I to satisfy the relations in II by the use of E_P and/or E_R corrections.

Example 4

Amplitude stabilization of the harmonic oscillator using essentially E_R type correction has been treated extensively in the literature.[3,4] However, as this second-order system affords a particularly simple vehicle for obtaining qualitative solutions which aid in understanding the roles of the correction terms, it will again be considered here. To this end, then, let it be desired to generate the sine and cosine of the angle φ where $\dot{\phi}$ is given.

$$y_2 = \sin\varphi \rightarrow \dot{y}_2 = \dot{\phi}\cos\varphi \tag{4.1}$$

$$y_1 = \cos\varphi \rightarrow \dot{y}_1 = -\dot{\phi}\sin\varphi \tag{4.2}$$

Thus the system equations are :

$$\dot{y}_2 = \dot{\phi}y_1 \tag{4.3}$$

$$\dot{y}_1 = -\dot{\phi}y_2 \tag{4.4}$$

The constraint relations we adopt here are:

$$\epsilon_1 = y_1^2 + y_2^2 - 1 \tag{4.5}$$

and

$$\epsilon_2 = y_1\dot{y}_2 - y_2\dot{y}_1 - \dot{\phi} = \dot{\phi}\epsilon_1 \tag{4.6}$$

ϵ_2 is a measure of system rate error, since

$$\frac{d}{dt}\left[\tan^{-1}\frac{y_2}{y_1}\right] = \frac{y_1\dot{y}_2 - y_2\dot{y}_1}{y_1^2 + y_2^2} - \dot{\phi} \tag{4.7}$$

Thus

$$y_1\dot{y}_2 - y_2\dot{y}_1 = \dot{\phi}(y_1^2 + y_2^2) \tag{4.8}$$

and

$$y_1\dot{y}_2 - y_2\dot{y}_1 - \dot{\Phi} = \dot{\Phi}\epsilon_1 \tag{4.9}$$

Using ϵ_1 to obtain Δy_1 and Δy_2:

$$\Delta y_1 = -2K_{01}\epsilon_1 y_1 \tag{4.10}$$

$$\Delta y_2 = -2K_{02}\epsilon_1 y_2 \tag{4.11}$$

Using ϵ_2 to obtain $\Delta\dot{y}_1$ and $\Delta\dot{y}_2$:

$$\Delta\dot{y}_1 = K_{11}\dot{\Phi}\epsilon_1 y_2 \tag{4.12}$$

$$\Delta\dot{y}_2 = -K_{12}\dot{\Phi}\epsilon_1 y_1 \tag{4.13}$$

The weighting functions are deliberately identified in the manner shown in order to determine the effect of mismatch; always an important consideration in analog work.

Using (7), the corrected system equations become:

$$\dot{y}_2 = \dot{\Phi}y_1 + \Delta y_2 + \Delta\dot{y}_2 \tag{4.14}$$

$$\dot{y}_1 = \dot{\Phi}y_2 + \Delta y_1 + \Delta\dot{y}_1 \tag{4.15}$$

Carrying out the work assuming quasi-static ϵ_1 (which amounts to treating it as sign invariant during the instantaneous integration process) yields

$$y_2 = Ae^{-\alpha t}\sin\omega t \tag{4.16}$$

$$y_1 = e^{-\alpha t}\sin(\omega t + \varphi) \tag{4.17}$$

where

$$A = 1 + \frac{\epsilon_1}{2}(K_{11} - K_{12}) \tag{4.18}$$

$$\alpha = (K_{01} + K_{02})\epsilon_1 \tag{4.19}$$

$$\omega = \dot{\Phi}[1 - \epsilon_1(K_{11} + K_{12})] \tag{4.20}$$

$$\varphi = \tan^{-1}\frac{\omega}{\epsilon_1(K_{02} - K_{01})} \tag{4.21}$$

and where second-order terms have been dropped.

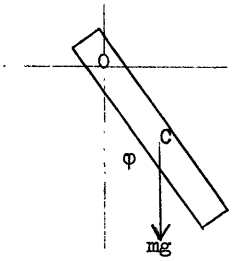
An analysis of (4.16) and (4.17) shows that the system process is such as to decrease the absolute value of ϵ_1. (4.18) and (4.21) give the detrimental effects of mismatch. Note that damping arises only from E_R, and rate correction (to first order terms) from E_P.

The mechanization of (4.3) and (4.4) with E_R corrections arising from ϵ_1 is shown in Figure 4.1. Stable frequencies in excess of 150 cps have been generated using this technique.

Example 5

This final example will treat with the compound pendulum of Figure 5.1 having a total angular excursion of $\pm\pi/2$ radians. The non-linear differential equation describing this system may be solved by elliptic integrals.

Although quite simple to mechanize on an analog computer, the errors after 5 seconds are prohibitive for a pendulum having a 5 cps frequency. The introduction of two constraint relations is necessary to preserve solution integrity.



m = mass of pendulum
l = length of pendulum
O = point of suspension
C = center of mass
k_o = radius of gyration about O
a = OC
mg = force due to gravity

Figure 5.1

Parameters and Definitions

m = 1 gram	$A = \frac{ga}{ko^2}$
l = 1.2 cm	$B = 1/2\ mk_o^2$
a = .346 cm	G = mga
$k_o^2 = .2394$	$y_2 = \sin\varphi$
$g = 980\ cm/sec^2$	$y_1 = \cos\varphi$

These parameters give a 5 cps frequency for an excursion of $\pm\pi/2$ radians. Initially the pendulum is at $\varphi = +\pi/2$ which is defined as the point of minimum kinetic energy.

The energy equation is

$$B\dot{\Phi}^2 + G(1 - \cos\varphi) = G \tag{5.1}$$

Differentiating this leads to the system equations

$$\dot{\Phi} = \omega$$
$$\dot{\omega} = -Ay_2 \tag{5.2}$$

where y_2 and φ are related by

$$\dot{y}_2 = \dot{\Phi}y_1$$
$$\dot{y}_1 = -\dot{\Phi}y_2 \tag{5.3}$$

y_2 is the driving function for (5.2). Any errors arising in $\dot{\omega}$ and $\dot{\Phi}$ (we do not employ φ in the mechanization) arise from the product Ay_2 and the subsequent integration.

$\dot{\Phi}$ is the driving function for (5.3). We know that amplitude stabilization of (5.3) is desirable, but here (5.1) imposes yet another constraint upon the y_1 formed in (5.3).

Using both the total energy and the amplitude stabilization relation, we define

$$\epsilon_1 = y_1^2 + y_2^2 - 1 \tag{5.4}$$

$$\epsilon_2 = B\dot{\phi}^2 - Gy_1 \tag{5.5}$$

from which one obtains (using 3):

$$\begin{aligned} \Delta\dot{\omega} &= 0 \\ \Delta\omega &= \Delta\dot{\phi} = -2K_2\epsilon_2 B\dot{\phi} \\ \Delta\varphi &= -K_2\epsilon_2 y_2 G \end{aligned} \tag{5.6}$$

$$\begin{aligned} \Delta\dot{y}_2 &= \Delta\dot{y}_1 = 0 \\ \Delta y_2 &= -2K_1\epsilon_1 y_2 \\ \Delta y_1 &= -2K_1\epsilon_1 y_1 + K_2\epsilon_2 G \end{aligned} \tag{5.7}$$

Using (7):

$$\begin{aligned} \dot{\phi} &= \omega + \Delta\varphi + \Delta\phi \\ \dot{\omega} &= -Ay_2 + \Delta\omega \end{aligned} \tag{5.8}$$

$$\begin{aligned} \dot{y}_2 &= \dot{\phi}y_1 + \Delta y_2 \\ \dot{y}_1 &= -\dot{\phi}y_2 + \Delta y_1 \end{aligned} \tag{5.9}$$

Note that an attempt to include either $\Delta\varphi$ or $\Delta\dot{\phi}$ in the $\dot{\phi}$ equation leads to an unstable algebraic loop since ϵ_2 contains $\dot{\phi}^2$.

With this restriction, the corrected system is

$$\begin{aligned} \dot{\phi} &= \omega \\ \dot{\omega} &= -Ay_2 - 2K_2\epsilon_2 B\dot{\phi} \end{aligned} \tag{5.10}$$

$$\begin{aligned} \dot{y}_2 &= \dot{\phi}y_1 - 2K_1\epsilon_1 y_2 \\ \dot{y}_1 &= -\dot{\phi}y_2 - 2K_1 y_1\epsilon_1 + K_2\epsilon_2 G \end{aligned} \tag{5.11}$$

Any attempt to implement (6.10) and (6.11) leads immediately to scaling difficulties due to the disparity of B and G. Further, the units of $\Delta\omega$ and Δy_j make impossible a consistent definition in the units of K_2. Actually, this is to be expected for our treatment of K_i has neglected any mention of u and i which certainly must be considered if any rational system of units is to be maintained in the several corrections.

The dilemma confronting us is quite fundamental, for if the K_i are to be determined by a consideration of i and u as well as j, what are to be the weights attached to them? Unless this is known, the corrections will not lie along the negative gradient as desired. We do know, however, that these newly considered constants will all be positive. Thus, no matter what weights are attached to them, the resulting correction vector will decrease E though certainly not in general along its negative gradient. For small E this is probably not significant, but in any case the remedy seems to be to make all the K_j and their factors as large as possible in order to hasten the (less efficient) minimization process.

Thus, except in simple systems where no inconsistency in units arises for the correction terms, we shall follow this rule.

This has been done for (5.10) and (5.11) which are mechanized in Figure (5.2). Deviations from the theoretical solution were monitored for a variety of cases and the results are given in Plates I and II.

Plate I shows the results after 5 seconds of solution time of

1. No correction
2. Amplitude correction into the cosine only
3. Amplitude correction into the sine only
4. Amplitude correction into both the sine and cosine
5. Correction in the potential energy only
6. Correction in the kinetic energy only
7. Correction in both kinetic energy and potential energy.

All of these solutions had errors growing with time.

Plate II gives the results of applying corrections arising from both ϵ_1 and ϵ_2. All of these solutions are stable, but notice that they may be substantially in error unless the corrections are added as specified by (5.10) and (5.11).

In accordance with the theory of keeping K_i as large as possible here, it was noted that decreasing K_1 or K_2 degraded system performance.

References

1. A. Feldbaum, "An Automatic Optimizer", Automatika i Telemekhanika 19:731-43, Aug. 1958. See "Automation Express", Nov. 1958.
2. H. Meissinger, "Meissinger on Steepest Descent", "Instruments and Automation", Sept. 1958.
3. J. Corbett, et al, "Stabilization of Computer Circuits", WADC Tech Report 57-425, Nov.1957.
4. R. Howe and E. Gilbert, "Trigonometric Resolution in Analog Computers by Means of Multiplier Elements", IRE Transactions, Vol.EC 6, June '57.

Acknowledgement

The idea for this paper arose as a result of investigating the division circuit illustrated in Example 1. This circuit is due to R. R. Favreau, and came to my knowledge from W. Brunner, both of Electronic Associates, Inc., Princeton, New Jersey. To Mr. Brunner I am also indebted for many helpful criticisms and comments which have influenced the final form of presentation.

SYMBOLS

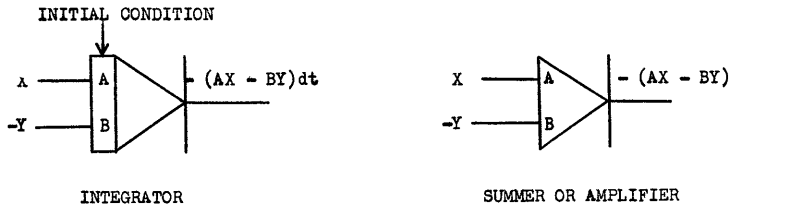


FIGURE 1.1

MECHANIZATION OF (1.5)

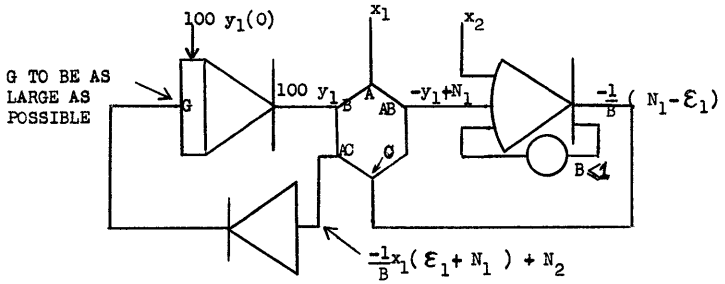


FIGURE 1.2

MECHANIZATION OF (2.3)

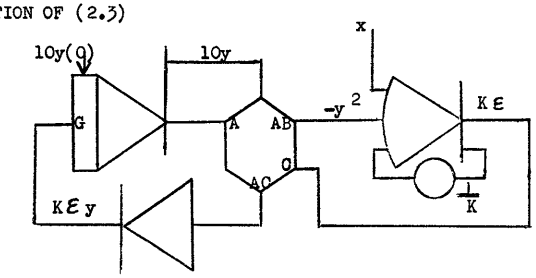


FIGURE 2.1

MECHANIZATION OF (3.5) AND (3.6)

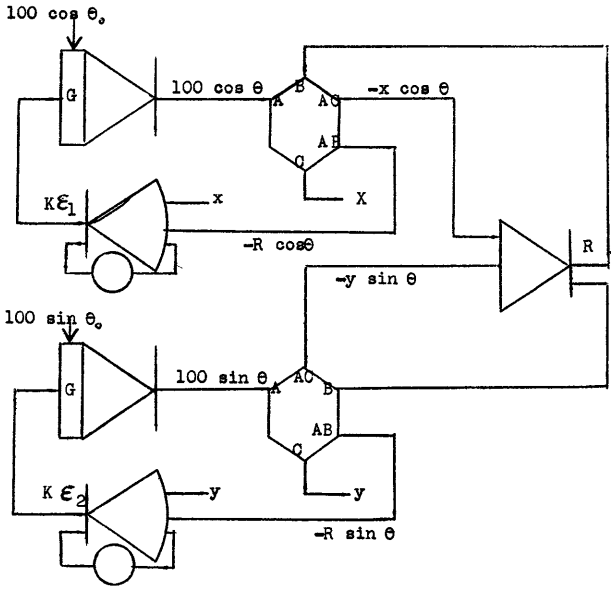


FIGURE 3.1

MECHANIZATION OF (4.3) AND (4.4)

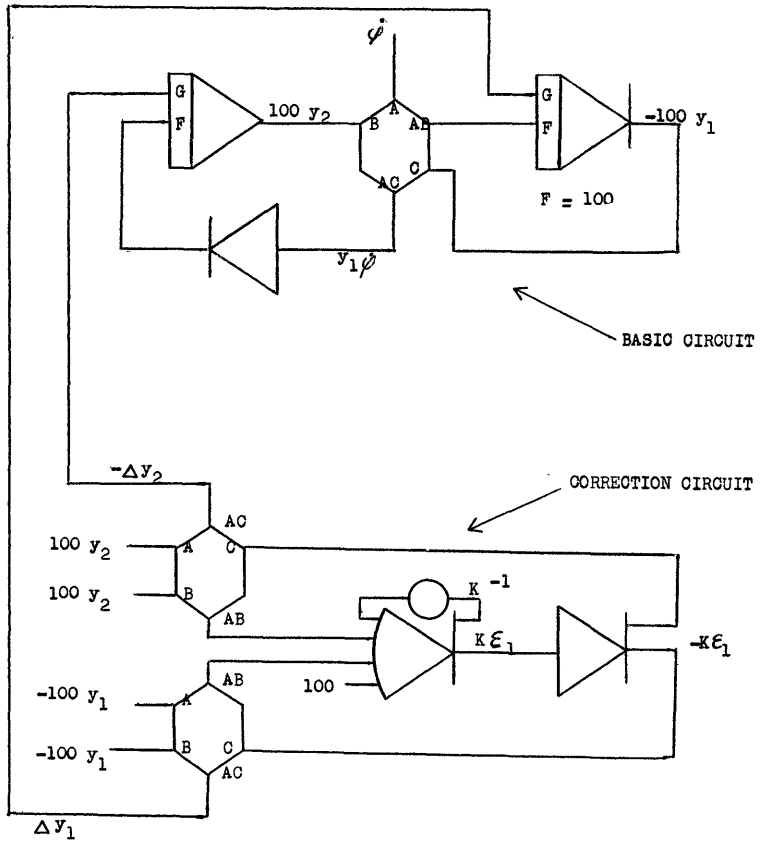


FIGURE 4.1

MECHANIZATION OF (5.10) AND (5.11)

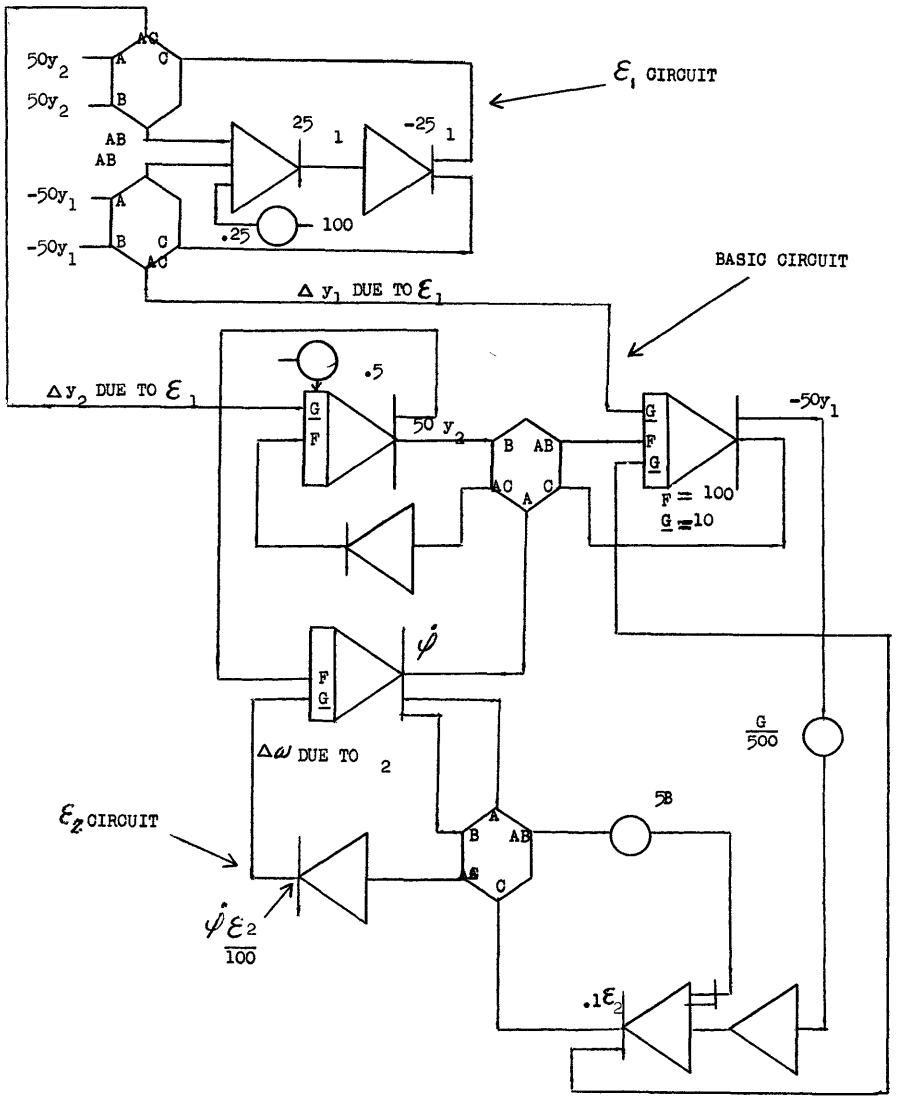
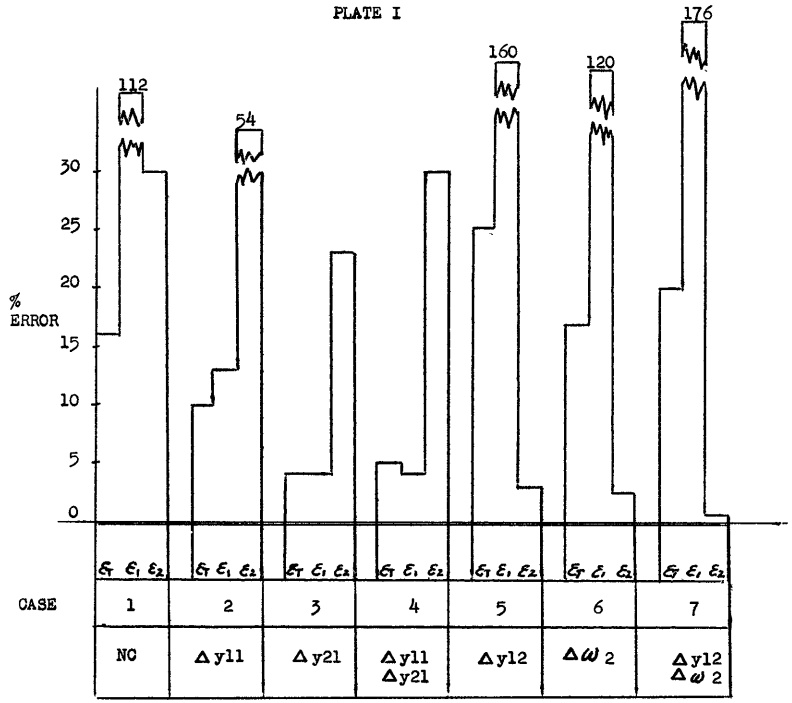


PLATE I



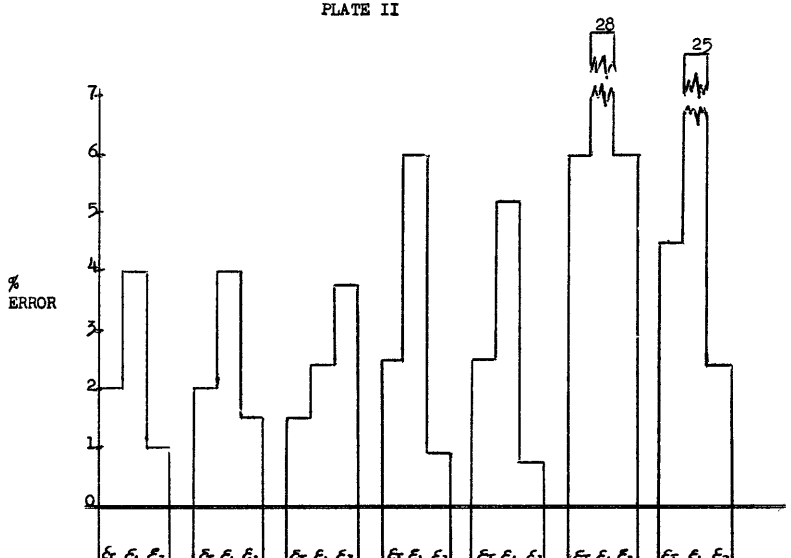
CASE	1	2	3	4	5	6	7
	NC	Δy11	Δy21	Δy11 Δy21	Δy12	$\Delta\omega_2$	Δy12 $\Delta\omega_2$

NC means no correction was added
y11 means that the correction in y_1 due to $\mathcal{E}_1$ was added, etc.

To define the % Error, $\mathcal{E}_1$ was referred to 1.
$\mathcal{E}_2$ was referred to 339 (the maximum energy).
$\mathcal{E}_T$, THE ERROR IN PERIOD, was referred to .2 seconds.

ALL READINGS MADE AT 5 SECONDS

PLATE II



CASE	1	2	3	4	5	6	7
	Δy11 Δy12 Δy21 $\Delta\omega_2$	Δy11 Δy12 Δy21	Δy11 Δy21 $\Delta\omega_2$	Δy12 Δy21	Δy12 Δy21 $\Delta\omega_2$	Δy11 Δy12	Δy11 Δy12 $\Delta\omega_2$

SEE PLATEII FOR AN EXPLANATION OF SYMBOLS

THESE READINGS REMAIN THE SAME FOR SEVERAL MINUTES

THE USE OF PARAMETER INFLUENCE COEFFICIENTS IN COMPUTER ANALYSIS OF DYNAMIC SYSTEMS

Hans F. Meissinger
Hughes Aircraft Company
Culver City, California

Summary

A new computer technique is described which yields the partial derivatives of problem variables with respect to pertinent system parameters simultaneously with the solution of the original system differential equations. These derivatives, known as parameter influence coefficients, are valuable to the analyst in enhancing his understanding of system characteristics. If the problem solution $x(t,\lambda)$ and the parameter influence coefficient $\frac{\partial x}{\partial \lambda}(t,\lambda)$ is known for a particular operating point where $\lambda = \lambda_o$, then it is possible to make a first order prediction of system behavior at a neighboring point having the new parameter value $\lambda_1 = \lambda_o + \Delta\lambda$. Similar predictions can be made if not one but several parameters are to be varied. Thus, the knowledge of parameter influences often helps to reduce the total number of computer runs required in a parametric system study. Typical applications of the technique are: linear extrapolation in the neighborhood of a known solution; determination of design tolerances of a system; prediction of critical parameter values and stability boundaries. The most useful application pertains to systems disturbed by random noise where normally a very large number of computer runs would be required to analyze the system on a statistical basis in a variety of operating conditions. Several illustrative examples are presented in the paper.

Introduction

In the analysis of dynamic systems it is frequently required to determine response characteristics not only for selected operating conditions but rather for a range of conditions over which certain system parameters can vary. If the system is described in terms of a set of differential equations

$$\frac{dx_i}{dt} = f_i(x_1, x_2, \ldots, t; \alpha_1, \alpha_2, \ldots), \quad i = 1, 2, \ldots, n \tag{1}$$

where $\alpha_1, \alpha_2, \ldots, \alpha_m$ are system parameters of particular interest, then any information obtainable from the computer which serves to enhance one's knowledge of the system response as function of these parameters will be valuable. Let the solution of the differential equations (1) obtained for a prescribed set of parameters and initial conditions be expressed as

$$x_{i_o} = x_{i_o}(t; \alpha_1, \alpha_2, \ldots), \quad i = 1, 2, \ldots, n \tag{2}$$

Then the partial derivatives

$$\frac{\partial x_{i_o}}{\partial \alpha_1}, \frac{\partial x_{i_o}}{\partial \alpha_2}, \ldots, \frac{\partial x_{i_o}}{\partial \alpha_m}, \quad i = 1, 2, \ldots, n$$

which will be referred to as parameter influence coefficients will be of considerable interest. They can be used to predict system performance in the neighborhood of the known solution x_{i_o} by first order approximation and to describe system sensitivity to certain design changes. If a parametric study of the system is required, usually a time-consuming task even on modern electronic computers, the parameter influences help reduce the total number of computer runs which must be made to explore a specified region of the parameter space. The situation is analogous to the simple task of drawing a curve either through a set of given points or through a set of points at which tangential slopes are known. In the latter case a much smaller number of points is needed to perform the task with the same degree of accuracy. How many data points may be saved through the additional information contained in the tangential slopes is a question which cannot be decided a priori; this depends on the nature of the curve and on the spacing of the points themselves. Evidently, the parameter influence coefficients give the analyst some degree of insight into trends of performance variation and can guide him in the search for parameter values of particular interest, for example those parameter combinations which promise greater system stability, greater yield of a chemical process, or greater effectiveness of control, as the case may be. Such orientation is especially desirable where many parameters of interest must be varied, and where intuition fails to indicate a clear direction for an experimental search procedure.

This paper discusses a method for obtaining parameter influences by solving a set of auxiliary differential equations, known as sensitivity equations, simultaneously with the original system equations. These sensitivity equations are derived from the original equations by a simple differentiation process prior to programming the computer. Obviously then, the additional information sought can only be gained at the expense of greater program complexity. However, the advantages of this trade-off will become apparent in the following sections.

The use of sensitivity equations was proposed some time ago by Murray and Miller[1] in connection with error analysis of solutions obtained by analog computers. The theory developed by

these authors has a direct bearing on the subject of this article, although the objectives and the field of application have been broadened considerably in the present discussion. In practice, the concept of sensitivity equations has remained largely unused, perhaps owing to the specialized application for which it was originally suggested. In an article published recently by Margolis and Leondes[2] a sensitivity equation is used in developing an adaptive control system capable of sensing parameter influences for feedback purposes.*

The purpose of this paper is to indicate, in terms of some typical applications, the practical usefulness of the parameter influence method and to stimulate further development in this direction. With the aid of this method a computer study involving parameter variations can be organized to follow a more systematic search pattern than has been possible heretofore. The need for an improved design of computer experiments is being felt throughout the computer engineering field. In principle, the parameter influence method which promises to yield such an improvement is applicable to digital as well as analog computers, although it will be presented here only in terms of analog operations. Time savings may prove even more significant in the digital field in view of the relatively large amount of time required for a single solution.

The following list of possible applications of the technique to studies of dynamic systems will illustrate its usefulness:

1. Prediction of solutions in the neighborhood of a known solution by a linear extrapolation method.

2. Determination of parameter tolerances on the basis of linear prediction. Detection of critical parameters.

3. Application to statistical studies: Evaluation of the influence of randomly distributed system constants or initial conditions. Extrapolation of results obtained with random noise signal inputs.

4. Optimization of system parameters in accordance with specified performance criteria using gradient techniques.

5. Analysis of solution sensitivity to computer errors.

6. Determination of system stability boundaries.

7. Variation of the time of occurrence of specified conditions, variation of rise times, settling times.

8. Solution of boundary value problems in ordinary differential equations.

Although only a few of these applications can be discussed in the space of this article the computer approach will be evident from the examples chosen. The emphasis of the discussion will be placed on underlying mathematical considerations. Some limitations of the method will be noted, and areas requiring further research effort will be indicated.

Derivation of Parameter Influence Coefficients

The principle of the technique will be explained first in terms of a simple second order differential equation. It will then be generalized for application to more complex systems.

Consider the inhomogeneous linear equation in x

$$\frac{d^2x}{dt^2} + \mu\frac{dx}{dt} + \lambda x = f(t) \qquad (3)$$

with specified initial values $x(0) = a$ and $\frac{dx}{dt}(0) = b$. The solution x is desired for several values of the parameter λ and may be considered as a function of two variables, $x = x(t,\lambda)$. From a known solution curve obtained for one parameter value λ_o a neighboring curve for λ_1 can be determined by extrapolation in terms of $\Delta\lambda = \lambda_1 - \lambda_o$. Thus

$$x(t,\lambda_1) = x(t,\lambda_o) + \left(\frac{\partial x}{\partial\lambda}\right)_{\lambda_o}\Delta\lambda + \left(\frac{\partial^2 x}{\partial\lambda^2}\right)_{\lambda_o}\frac{\Delta\lambda^2}{2!} + \ldots \qquad (4)$$

The number of terms needed in this expansion for a satisfactory approximation of $x(t,\lambda_1)$ depends on the magnitude of $\Delta\lambda$ and on the behavior of the solution x and its partial derivatives with respect to λ in the region of interest. Only first order extrapolations will be considered here.

The partial derivative $\partial x/\partial\lambda$, itself a function of t and λ, is a parameter influence coefficient of first order. Other parameter influences pertaining to equation (3) are the derivaties $\partial x/\partial\mu$, $\partial x/\partial a$, and $\partial x/\partial b$, the latter two terms representing the sensitivity to changes in initial values.

A simple operation performed on the differential equation (3) makes it possible to program the problem for simultaneous computer solution of x and $\partial x/\partial\lambda$. Differentiation of (3) with respect to λ yields**

$$\frac{\partial^3 x}{\partial\lambda\partial t^2} + \mu\frac{\partial^2 x}{\partial\lambda\partial t} + \lambda\frac{\partial x}{\partial t} + x = 0 \qquad (5)$$

* The referenced article makes use of some earlier unpublished notes by the author[3] in which the parameter influence method was discussed.

** Since x is a function of λ as well as t, the symbol $\partial/\partial t$ must be used in place of d/dt in formulating equation (5).

Interchanging the order of differentiation of x and using the notation $u = \partial x/\partial \lambda$, one obtains the differential equation in u

$$\frac{\partial^2 u}{\partial t^2} + \mu \frac{\partial u}{\partial t} + \lambda u = -x \qquad (6)$$

where u must satisfy the initial conditions $u(0) = 0$, $\frac{\partial u}{\partial t}(0) = 0$, since the initial values of x and $\partial x/\partial t$ do not depend on λ. Equation (6) is known as the sensitivity equation of the system with respect to the parameter λ. Similarly, if the parameter influence of μ is desired, a new equation in $v = \partial x/\partial \mu$, viz.

$$\frac{\partial^2 v}{\partial t^2} + \mu \frac{\partial v}{\partial t} + \lambda v = -\frac{\partial x}{\partial t} \qquad (7)$$

is derived, where v obeys the initial conditions $v(0) = 0$, $\frac{\partial v}{\partial t}(0) = 0$.

In the simple case considered here the homogeneous part of the sensitivity equations is identical to that of (3). The terms -x and $-\partial x/\partial t$ which appear as forcing functions in equations (6) and (7), respectively, provide the coupling of the sensitivity equations to the original equation (3). There is, of course, no coupling in reverse. Figure 1 shows the computer circuits which yield x and u; the two circuits may be interpreted as master and slave, respectively. Owing to the identical structure of (6) and (7) only one slave circuit is necessary to obtain u and v. These functions can be computed in successive operation, simply by using either x or $\partial x/\partial t$ as the coupling term.

If the initial conditions a and b are the parameters of interest it is seen that no coupling terms appear in the sensitivity equations. In the case of $\partial x/\partial a = w$ the homogeneous differential equation

$$\frac{\partial^2 w}{\partial t^2} + \mu \frac{\partial w}{\partial t} + \lambda w = 0 \qquad (8)$$

with initial conditions $w(0) = 1$, $\frac{\partial w}{\partial t}(0) = 0$ is obtained which can be solved simply by operating the original computer setup once more but without the driving function f(t) and with appropriately modified initial values.

It should be noted that the use of the parameter influence method is not restricted to linear systems. If, for example, the original differential equation (3) is augmented by the term νx^3, the sensitivity equation with respect to ν assumes the form

$$\frac{\partial^2 z}{\partial t^2} + \mu \frac{\partial z}{\partial t} + \lambda z + 2\nu x^2 z + x^3 = 0 \qquad (6a)$$

where $z = \partial x/\partial \nu$. The coupling terms in this example are, themselves, nonlinear functions of the dependent variable x, and the structure of the secondary computer circuit differs from the primary one.

Proceeding now to the general problem of an n^{th} order system of differential equations with m parameters $\alpha_1, \alpha_2, \ldots, \alpha_m$

$$\frac{dx_i}{dt} = f_i(x_1, x_2, \ldots, t; \alpha_1, \alpha_2, \ldots), \quad i = 1, 2, \ldots, n \qquad (1)$$

the sensitivity equation with respect to α_k yielding the derivative $u_{ik} = \frac{\partial x_i}{\partial \alpha_k}$ may be formulated as follows

$$\frac{\partial}{\partial \alpha_k}\left(\frac{\partial x_i}{\partial t}\right) = \frac{\partial u_{ik}}{\partial t} = \frac{\partial f_i}{\partial x_1} u_{1k} + \frac{\partial f_i}{\partial x_2} u_{2k} + \cdots + \frac{\partial f_i}{\partial \alpha_k} \qquad i = 1, 2, \ldots, n \qquad (9)$$

The initial conditions are $u_{ik}(0) = 0$ provided none of the initial values of the original differential equation are considered as parameters. This formulation applies to linear as well as nonlinear systems.

It will be observed that in order to study the influence of one parameter α_k it is necessary to program a complete set of n sensitivity equations (9) even though this parameter may appear explicitly only in one equation of the original system (1). If, for example, α_k occurs in the j^{th} term f_j only, then the j^{th} sensitivity equation exhibits the coupling term $\partial f_j/\partial \alpha_k$ whereas $\partial f_i/\partial \alpha_k = 0$ for $i \neq j$. The remaining sensitivity equations nevertheless contain the influence of α_k implicitly in terms of the variables u_{1k}, u_{2k}, ... which establish intercoupling with the j^{th} equation.

These facts are illustrated by a simulation problem involving missile guidance. Figure 2 shows a block diagram of the system containing a nonlinear feedback term $F(\Theta;k)$ representative of radome refraction of the line-of-sight from missile to target. The influence of parameter k upon guidance accuracy is to be investigated. The problem variables V, γ, Θ, y, σ, σ_a and η are defined in the diagram; k is called radome error slope. It is seen that the computation of the parameter influence of k involves only one coupling term $\partial F/\partial k$ between the primary and the secondary simulation loop, whereas the partial derivatives of all problem variables with respect to k, i.e., V_k, γ_k, Θ_k, etc., are required for completing the computer program as indicated in the bottom half of Figure 2. In the guidance study depicted here the statistical effect of radar noise on missile dispersion at impact is analyzed. Since a large number of computer runs is necessary to make statistically significant predictions of rms-miss for any given value of k, a parametric study of the influence of k on miss M proves very time-consuming. Using the parameter influence technique for determining miss variation with k a linear extrapolation from computed sample points will result in a substantial reduction of time required for conducting this study.

From a standpoint of economy of computer operation the technique would not seem to be very attractive since approximately twice the ordinary amount of computing equipment is required to obtain one set of parameter influences, $\partial x_i/\partial \alpha_k$, together with x_i. Fortunately, however, a slight modification of these circuits can often produce other influence coefficients $\partial x_i/\partial \alpha_j$ if the parameter α_j occurs explicitly only in one or a few of the original equations. The time saved in parametric studies of systems having random noise input signals nevertheless may justify this technique in spite of increased program complexity.

A problem of theoretical as well as practical interest in any application of the parameter influence technique is the possible occurrence of resonance phenomena when the sensitivity equations are coupled to the original system of equations[1]. In the simple case presented by equations (3), (6) and (7) it is seen that the secondary system is excited by the output of the primary system having the same natural frequency $\omega \cong \sqrt{\lambda}$. The secondary system will begin to oscillate in resonance if the damping coefficient is small. For systems exhibiting neutrally stable or unstable modes of oscillation the computer circuits for the sensitivity equations can therefore be operated for a limited period of time only bevore an overload condition is reached. The technique remains applicable until this time. However, the following considerations must be kept in mind: As the amplitude of the parameter influence coefficients increases with time, the instantaneous values of the original solution x(t) become more sensitive to parameter variations. Furthermore, higher order parameter influences which are subject to the same resonance effects tend to become relatively important. Hence, the quality of extrapolation based on first order parameter influences is degraded in any system exhibiting unstable characteristics. This fact will be considered in greated detail in the next section.

Other Applications of the Method

The use of parameter influences in the analysis of statistical problems involving random noise disturbances was described in the preceding paragraphs. If the initial values or some parameters of the dynamic system are known to be randomly distributed in a specified manner, a prediction of the statistical characteristics of the output functions at any instant of time is possible on the basis of parameter influence coefficients. For example, the mean and standard deviation of the miss distance of a projectile corresponding to random dispersion of launch conditions may be obtained by a single computer run through first order extrapolation from a nominal flight path. The accuracy of such a prediction will be influenced by the interval over which the initial values or parameters are expected to vary, by the time of flight, and by the problem sensitivity in general. Spot checks of the accuracy in typical cases will be required.

Some other typical applications of the parameter influence method merit further discussion. Examples considered in this section are selected from the topics listed in the Introduction and include extrapolation from a known solution curve, optimization of system parameters, and determination of stability boundaries by systematic search. To simplify the mathematical discussion the second order linear differential equation (3) will be considered again in these examples; the forcing term f(t) will be omitted. Using dot symbols to denote time derivatives the system equation and sensitivity equation are, respectively,

$$\ddot{x} + \mu\dot{x} + \lambda x = 0 \tag{3a}$$

$$\ddot{u} + \mu\dot{u} + \lambda u = -x \tag{6}$$

subject to initial conditions $x(0) = a$, $\dot{x}(0) = 0$, $u(0) = 0$, $\dot{u}(0) = 0$, where as before u is defined as $u = \partial x/\partial \lambda$.

Extrapolation

Results of extrapolation from a known solution obtained on the computer for the value λ_o are shown in Figure 3 as a family of output curves with λ as parameter. The diagram also shows accurate solution curves obtained by actual parameter variation in the system. Similarly, Figure 4 shows a family of curves obtained by variation of the parameter μ, starting from a selected value μ_o. In both cases a high quality of approximation is observed over a limited time interval depending on the magnitude of the parameter changes $\Delta\lambda$ and $\Delta\mu$. The time-dependent increase in approximation error is inherent in the nature of the first order extrapolation scheme. A simple discussion, applying to the case of undamped oscillations ($\mu = 0$), will help to clarify this fact. The solution has the form

$$x = a \cos \omega t \tag{10}$$

where $\omega = \sqrt{\lambda}$. The sensitivity equation with respect to ω thus becomes

$$\ddot{u} + \omega^2 u = -2\omega x = -2\omega a \cos \omega t \tag{11}$$

and yields the parameter influence

$$u = -at \sin \omega t \tag{12}$$

A first order extrapolation centered at ω_o would yield the approximate solution

$$x_e(t,\omega) = a \cos \omega_o t - at(\omega - \omega_o)\sin \omega_o t \tag{13}$$

where the higher order terms of the power series expansion with respect to $(\omega - \omega_o)$ are neglected. The error of this approximation involves the terms

$$\frac{\partial^2 x}{\partial \omega^2}\frac{(\omega-\omega_o)^2}{2!} + \frac{\partial^3 x}{\partial \omega^3}\frac{(\omega-\omega_o)^3}{3!} + \cdots$$

$$= -a\frac{t^2(\omega-\omega_o)^2}{2}\cos \omega_o t + a\frac{t^3(\omega-\omega_o)^3}{6}\sin \omega_o t + \cdots \tag{14}$$

The time interval during which the second order error term remains within specified bounds is seen to be inversely proportional to the frequency difference $(\omega - \omega_o)$ used in the extrapolation procedure. Boundaries of fixed approximation errors in a frequency versus time diagram are illustrated in Figure 5.

Parameter Optimization

An application of the method to optimization of system parameters is illustrated by the process of matching the output $x(t)$ of a system with two unspecified parameters against a known function $x_i(t)$ in terms of least-square errors. This problem arises, for example, if test results obtained in the field are to be duplicated by laboratory experiment with a corresponding mathematical model of the system having adjustable parameter settings. Once again, the second order equation (3a) is used as model; the parameters μ and λ are assumed as unknown.

The optimization criterium is given by

$$h(\mu,\lambda) = \int_0^T \left[x(t;\mu,\lambda) - x_i(t)\right]^2 dt \qquad (15)$$

To minimize this function a gradient technique[4,5] can be employed where the values of μ and λ are varied in proportion to the respective components of the gradient vector $\nabla h(\mu,\lambda)$. This technique assures a steepest descent to the minimum value of h along a path perpendicular to the contours $h(\mu,\lambda)$ = constant in the μ-λ-plane.

In order to compute the gradient components

$$\frac{\partial h}{\partial \mu} = 2 \int_0^T \left[x(t;\mu,\lambda) - x_i(t)\right] \frac{\partial x}{\partial \mu}\, dt \qquad (16)$$

$$\frac{\partial h}{\partial \lambda} = 2 \int_0^T \left[x(t;\mu,\lambda) - x_i(t)\right] \frac{\partial x}{\partial \lambda}\, dt \qquad (17)$$

the parameter influences $\partial x/\partial \mu$ and $\partial x/\partial \lambda$ are required. They are obtained as continuous functions of time together with the solution $x(t)$. The computer simultaneously evaluates the integrals (16) and (17). The sequence of operations consists in a systematic step-by-step variation of parameter values μ and λ in accordance with the gradient direction established at the end of each computing cycle. The process converges to the desired minimum of h. Figure 6 shows typical contours and gradient lines obtained in the solution of this optimization problem.

Stability Boundaries

The problem of finding stability boundaries of a dynamic system in the parameter space arises in many phases of control system design and analysis. The technique described here provides a systematic search pattern for repeated trials on the computer. Predictions based on parameter influence coefficients are made before each new trial run to assure rapid convergence of the trial sequence to a point on the stability boundary. Only one parameter is varied in this sequence of operations.

In the case of a damped oscillatory system one may proceed by selecting two or more successive maxima or minima of a solution curve and the corresponding values of the parameter influence coefficient available from the computer. Predicted variations of these maxima or minima are then plotted against the parameter in question, and a point, or a cluster of points, of intersection of the different extrapolation lines is determined. This intersection designates the parameter value at which, according to first order prediction, the successive maxima or minima would have remained equal. Thus, a first approximation of the stability boundary is obtained. Subsequent trials are made, if desired, to yield successively better approximations. It can be shown that this step-by-step approach is, in fact, an example of Newton's iteration procedure and converges rapidly to the desired boundary point, under certain conditions regarding the initial choice of the parameter value. The nature of this convergence will be discussed in the Appendix.

An example of the iterative approach is presented in Figure 7 which shows solutions of the linear differential equation (3a) having the damping factor μ as parameter. Starting with a positive damping value, $\mu_1 = 0.10$, the sequence of solutions proceeds rapidly to the vicinity of the boundary case where $\mu_B = 0$. The example indicates that the first iteration overshoots to a slightly negative value of μ; this error is almost completely eliminated in the second step. Figure 8 illustrates the extrapolation graphically in terms of predicted maxima plotted versus μ. It is seen that the initial choice, $\mu_1 = 0.10$, is corrected by the incremental change $\Delta\mu_2 = -0.116$ which gives rise to the slightly unstable second trial solution shown in Figure 7.

In general, the successive parameter increments $\Delta\mu_i$ to be used for the i^{th} trial run may be computed in accordance with the extrapolation scheme illustrated in Figure 8. Depending on whether the intersection of two or more extrapolation lines are to be taken into account, the procedure is formulated as follows

$$\Delta\mu_i = -\left[\frac{x_{max_n} - x_{max_{n+1}}}{v_n - v_{n+1}}\right]_{i-1} \qquad (18)$$

for one intersection point

$$\Delta\mu_i = -\left[\frac{1}{N}\sum_{n=1}^{N} \frac{x_{max_n} - x_{max_{n+1}}}{v_n - v_{n+1}}\right]_{i-1} \qquad (19)$$

for N intersection points

where x_{max_n}, $x_{max_{n+1}}$, etc. denote consecutive maxima and v_n, v_{n+1}, etc. the corresponding parameter influence terms. These expressions will be derived in the Appendix.

In systems where two or more parameters define the stability boundary the process must be repeated for various parameter combinations. At each of these combinations the prediction of the boundary point is performed in terms of the same parameter since the computer program contains the sensitivity equation (or equations) for that parameter.

Applications Involving Time of Occurrence

A type of problem frequently encountered in computer applications involves the time of occurrence T of a specified condition and the variation of T as a result of parameter changes. Examples are: settling time or time of occurrence of the first maximum of a transient in the study of control system response; time of the nth zero-crossing of a non-monotonic or oscillatory variable; boundary-value and eigenvalue problems where a parameter must be adjusted until the solution meets a specified end condition. In a manner similar to the systematic search for stability boundaries described above, the parameter influence technique may be employed to reduce the number of computer runs which would normally be required in a trial and error procedure.

Let the problem solution $x = x(t,\lambda)$ attain a specified value C at the time $t = T$. Then the equation

$$x(T,\lambda) = C \tag{20}$$

is an implicit representation of the time of occurrence T as a function of the parameter λ. From the theory of implicit functions one obtains the result

$$\frac{\Delta T}{\Delta \lambda} \cong - \frac{\frac{\partial x}{\partial \lambda}}{\frac{\partial x}{\partial t}} = - \frac{u(T,\lambda)}{\dot{x}(T,\lambda)} \tag{21}$$

which may be directly applied to predict the necessary change in λ to alter T by ΔT on the basis of known values of u and $\dot{x}$, provided the derivative $\dot{x}$ does not vanish. The variables u and $\dot{x}$ are assumed to be available in the computer program. The first order extrapolation represented by equation (21) can be used in an iteration procedure designed to converge to the desired final value of T.

Conclusions

Computer programming techniques for obtaining parameter influence coefficients have been discussed, and applications which illustrate this approach to the analysis of system performance have been presented in the foregoing sections. The new method promises more effective utilization of the computer by reducing the effort required in conducting comprehensive parameter studies or in searching for optimum parameter combinations of a system. Savings in machine time and operators' time are realized through a greater yield of useful information from each computer run. The cost of additional program complexity or increased number of computer channels, however, and the greater burden of checkout and trouble-shooting in the case of analog operations must be weighed against the advantages gained by the method. In statistical evaluation studies of dynamic systems, known for being extremely time-consuming, the method offers definite promise. In digital computation, where a search for optimum solutions, stability boundaries, etc., cannot conveniently be guided by the analyst without much loss of time, a systematic search pattern based on parameter influence data is particularly attractive. Development along these lines is currently in progress. In view of the limited experience gained in applying the method to examples of the type described above, a conclusive evaluation of net advantages cannot be made at this time.

Judging strictly from a viewpoint of time savings, it appears that the analog computer field can profit more immediately from increased operating speeds available in certain types of computer models. Repetitive analog computers, operating at rates of up to 50 solutions per second, are capable of handling statistical evaluation studies in a remarkably short time. It is expected, nevertheless, that parameter influence techniques will find increased application on analog computers of the conventional real-time variety which are available in much larger numbers.

Mathematically, the question arises whether parameter influences cannot be equally well obtained without increased program complexity by a finite difference approach. Such an approach requires pairs of computer solutions in close vicinity to approximate the partial derivatives in question. This procedure is feasible, if somewhat more time-consuming, except in the case of random noise studies. Here the finite differences obtained from two adjacent samples of runs would be made highly inaccurate by the confidence intervals attached to the individual results.

The question of extrapolation accuracy, discussed in the sections on Derivation of Parameter Influence Coefficients and Other Applications of the Method, is of fundamental interest and must be considered in each individual application. In theory, higher order parameter influences could be used to reduce the error inherent in first order prediction, and these terms could be obtained on the computer in a manner similar to that described for the first order influence coefficients. In practice, this procedure leads to a prohibitive increase in program complexity and is not considered justified since accuracy of prediction is not the primary purpose of the approach.

Acknowledgements

The author wishes to acknowledge many valuable suggestions made by his colleagues A. J. Zeichner and L. Levine in early discussion of the above concepts. S. Dunin contributed pertinent comments relating to practical applications. Numerical results presented in the article were obtained with the aid of computing facilities at the Hughes Aircraft Company, under a company-sponsored research program.

Appendix

Convergence of the Iterative Method for Finding the Stability Boundary of an Oscillatory System

The iteration procedure, illustrated by Figures 7 and 8, for finding stability boundaries will be investigated in this Appendix, and criteria for its convergence will be discussed. Equations (18) and (19) which give the incremental change $\Delta\mu_i$ of the parameter μ after the $(i-1)^{st}$ trial will first be derived. The predicted μ-value at which two consecutive maxima, x_{max_n} and $x_{max_{n+1}}$, become equal is found at the intersection of the two lines of extrapolation

$$\bar{x}_n = x_{max} + \left.\frac{\partial x}{\partial \mu}\right|_n \Delta\mu$$

$$\bar{x}_{n+1} = x_{max_{n+1}} + \left.\frac{\partial x}{\partial \mu}\right|_{n+1} \Delta\mu \qquad (A1)$$

Hence, by equating these expressions and using the notation $\partial x/\partial\mu = v$, one obtains

$$x_{max_n} + v_n \Delta\mu = x_{max_{n+1}} + v_{n+1} \Delta\mu \qquad (A2)$$

from which the equation

$$\Delta\mu_i = -\left[\frac{x_{max_n} - x_{max_{n+1}}}{v_n - v_{n+1}}\right]_{i-1} \qquad (18)$$

immediately follows. If more than two consecutive maxima are used in the extrapolation, an average of the predicted (distinct) intersection points is formed which yields equation (19) as a generalized form of (18). For linear oscillations of the type depicted in Figure 7 it is advantageous to use minima as well as maxima in the extrapolation procedure in order to obtain twice as many sample points from each computer run. The minimum x_{min_k} and the corresponding slope v_k are used with appropriate sign reversals when a comparison with the preceding maximum is made. The algorithm equivalent to equation (18) thus takes the form

$$\Delta\mu'_i = -\left[\frac{x_{max_n} + x_{min_{n+1}}}{v_n + v_{n+1}}\right]_{i-1} \qquad (A3)$$

where the subscripts n, n+1, etc., refer to consecutive extreme values irrespective of their character as maxima or minima. This modified prediction formula was used in obtaining the results shown in Figures 7 and 8.

The nature of the iteration procedure can be more readily interpreted after introducing the simplifying notation

$$a_n = x_{max_n} - x_{max_{n+1}} \qquad (A4)$$

For any given subscript n the difference term a_n is a function of μ only. Using this definition, a simple expression for the slope difference $v_n - v_{n+1}$ can be derived, viz.,

$$v_n - v_{n+1} = \left.\frac{\partial x}{\partial \mu}\right|_n - \left.\frac{\partial x}{\partial \mu}\right|_{n+1} = \frac{\partial}{\partial \mu}(x_{max_n} - x_{max_{n+1}}) = \frac{\partial a_n}{\partial \mu} \qquad (A5)$$

Hence, the prediction formula (18) may be rewritten in the form

$$\Delta\mu_i = -\left[\frac{a_n}{\frac{\partial a_n}{\partial \mu}}\right]_{i-1} \qquad (A6)$$

which is Newton's familiar iteration formula for finding the zero of the difference function $a_n(\mu)$. Thus, the successive extrapolation for finding the stability boundary point $a_n = 0$ proves to be nothing else but an application of Newton's iteration procedure.

Figure 9 shows typical a_n-curves plotted against μ for n = 1, 2, 3 and 4. The curves were obtained by comparing consecutive amplitudes (i.e. maxima and minima) from a family* of solutions $x(t,\mu)$ of the linear differential equation

$$\ddot{x} + \mu\dot{x} + \lambda x = 0 \qquad (3a)$$

with $x(0) = a$, $\dot{x}(0) = 0$.

It is important to determine over which range of μ the iteration process can be expected to converge to the stability boundary located at $\mu_B = 0$ in the case under consideration. Inspection of Figure 9 indicates that convergence to this point is limited to values of μ at which the a_n-curves have a positive slope since the slope at the zero-crossing is positive. All a_n-curves exhibit a maximum at some value $\mu_{max,n} > 0$. Hence, it is seen that convergence to the stability boundary is assured only if the iteration process is started at a point μ_o located to the left of the respective maximum. (The iteration would proceed in the opposite direction if the initial point were located on the other side.) Starting the iteration with a case of positive damping, $0 < \mu_o < \mu_{max,n}$, it is observed that, owing to the convex character of the a_n-curves, the first iteration step always overshoots into the region of negative μ from

* Solutions such as those shown in Figures 3, 4 and 7 are representatives of the family.

where the convergence to $\mu_B = 0$ is quite rapid. The sequence of iterated solutions depicted in Figure 7 confirms this observation. On the other hand, if the iteration is started in the unstable region ($\mu_0 < 0$) convergence is always assured regardless of the position of the starting point.

Inspection of the a_n-curves reveals, furthermore, that the location $\mu_{max,n}$ of the maxima moves rapidly in the direction of $\mu = 0$ for increasing n-values, i.e. the range of convergence in the stable region of μ is narrowed down if amplitudes other than the first few are picked for the prediction process. In the case of the first amplitude pair, n = 1, the convergence range is exceptionally large extending to values in the neighborhood of critical damping, $\mu_{crit} = 2\sqrt{\lambda}$.

The above considerations are based on the case of a linear second order system but apply more generally to higher order systems exhibiting several distinct modes of oscillation. This is true because a particular mode of interest has the same structure as the single mode of oscillation characterized by the solutions shown in Figure 7. In the cases of practical interest a dominant mode of oscillation exists for which the stability boundary is being sought. In the presence of initial transients the amplitudes of the dominant mode will only be detected after some time interval. Hence, based on the information obtained from the a_n-curves of Figure 9, there exists only a small range of positive damping over which convergence to the stability boundary is assured. In these cases an iterative approach from the unstable region is preferable.

Figure 9 illustrates one other interesting fact which concerns the choice of single vs. multiple intersections, i.e. the use of equation (18) vs. (19), respectively. It is seen that, except in a close vicinity of the boundary point $\mu_B = 0$ where all a_n-curves have a common tangent, the multiple intersection approach yields a cluster of increments $\Delta\mu_i$. While this disparity of predictions tends to degrade the rapidity of convergence of the iteration process, an averaging of predicted values is desirable whenever the read-out accuracy of successive maxima is limited, i.e. in the majority of analog computer operations.

It is expected that the essential features of this analysis carry over into applications involving non-linear oscillations, although this has not been studied in detail. In view of the different laws governing the growth or decay of amplitudes in non-linear oscillations, the a_n vs. μ curves may look substantially different from those shown in Figure 9. However, as long as a_n is a continuous and well-behaved function of the parameter μ near the stability boundary point, the iteration process in general is a convergent one and thus can be used to find the boundary.

References

1. Miller, K. S. and Murray, F. J., "The Mathematical Basis for the Error Analysis of Differential Analyzers", M.I.T. Journal of Mathematics and Physics, Nos. 2 and 3, 1953.

2. Margolis, M. and Leondes, C. T., "A Parameter Tracking Servo for Adaptive Control Systems", 1959 IRE-WESCON Convention Record, Part 4, pp. 104-115, San Francisco, California, August 1959.

3. Meissinger, H. F., "An Abridged Simulation Technique for Problems Involving Random Noise Signals", unpublished notes, Hughes Aircraft Company, Culver City, California, December 1957.

4. Courant, R., "Variational Methods for the Solution of Problems of Equilibrium and Vibration", Bulletin of the American Mathematical Society, Vol. 49, pp. 1-23, 1943.

5. Tompkins, C. G., "Methods of Steep Descent", Modern Mathematics for the Engineer, E. F. Beckenbach, Editor, McGraw-Hill Book Company, Inc., New York, pp. 448-479, 1956.

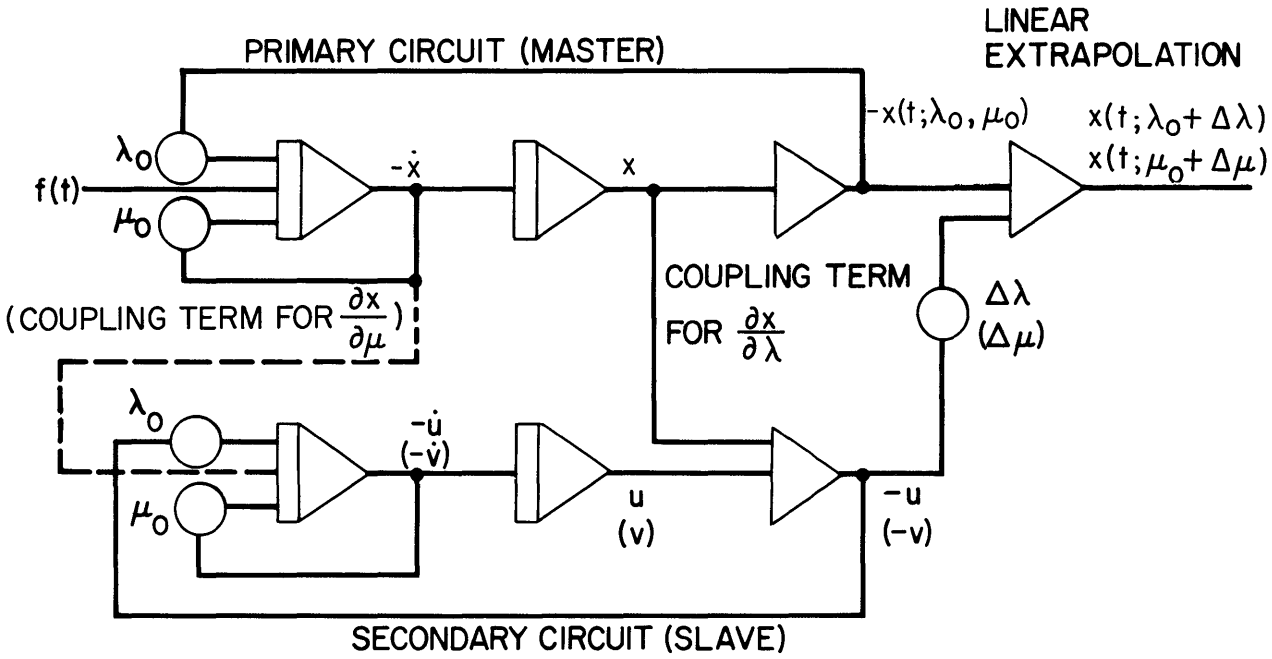


Fig. 1. Computer diagram for determining parameter influences $u = \frac{\partial x}{\partial \lambda}$ or $v = \frac{\partial x}{\partial \mu}$ in a linear second order system.

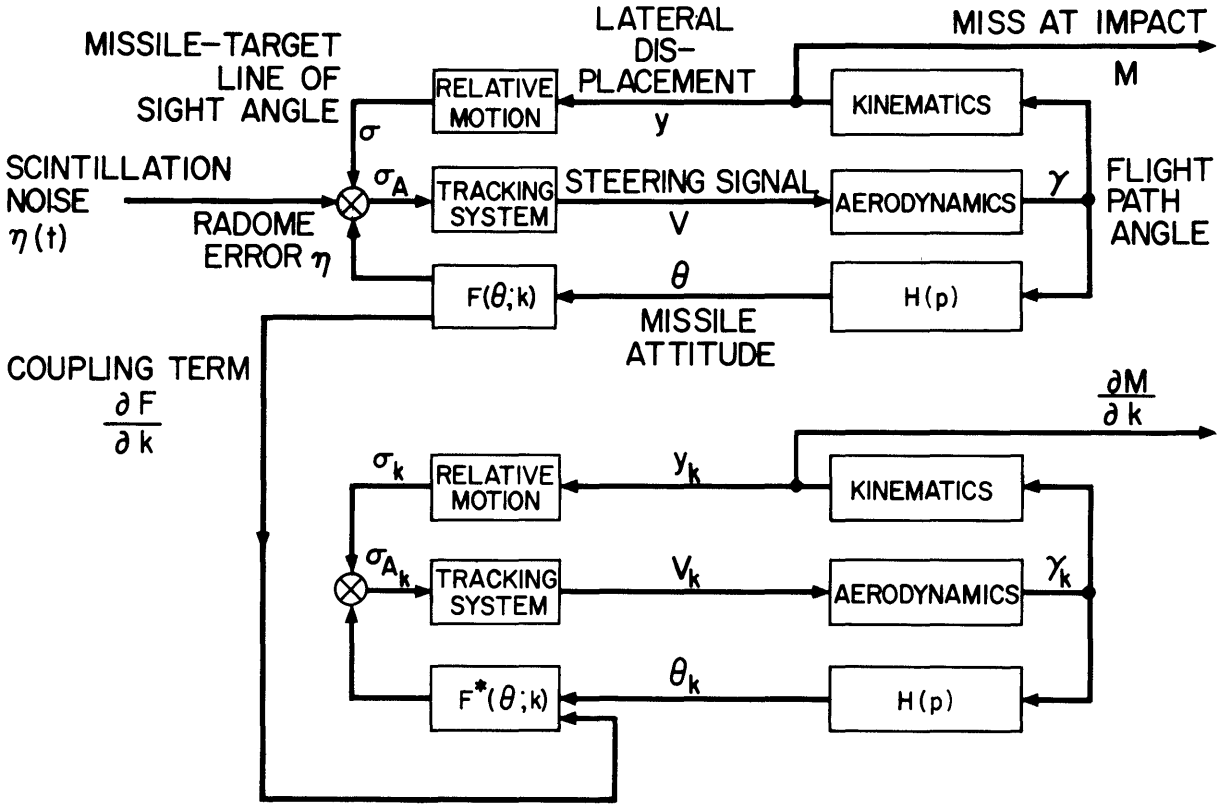


Fig. 2. Determination of miss M and influence coefficient $\frac{\partial M}{\partial k}$ in a non-linear guidance problem.

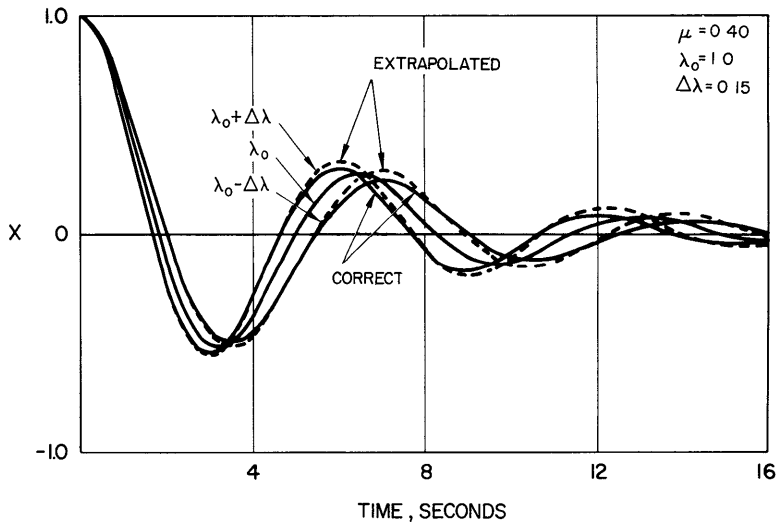


Fig. 3. Extrapolated solutions in the vicinity of a known solution, with λ as parameter.

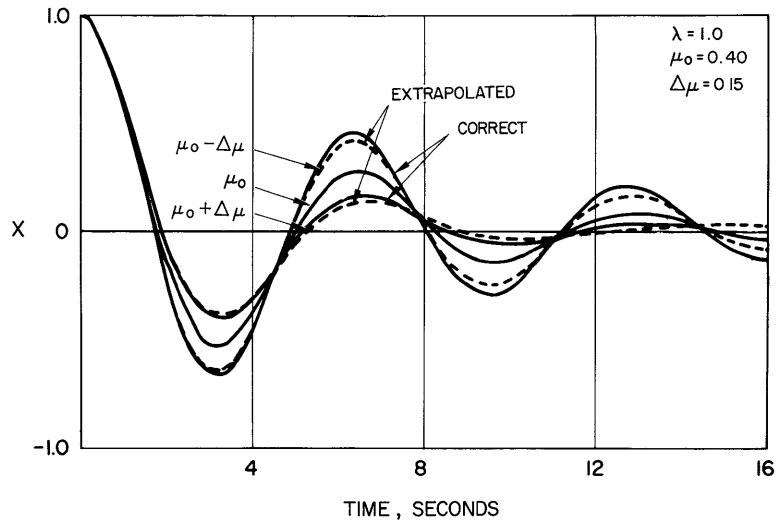


Fig. 4. Extrapolated solutions in the vicinity of a known solution, with μ as parameter.

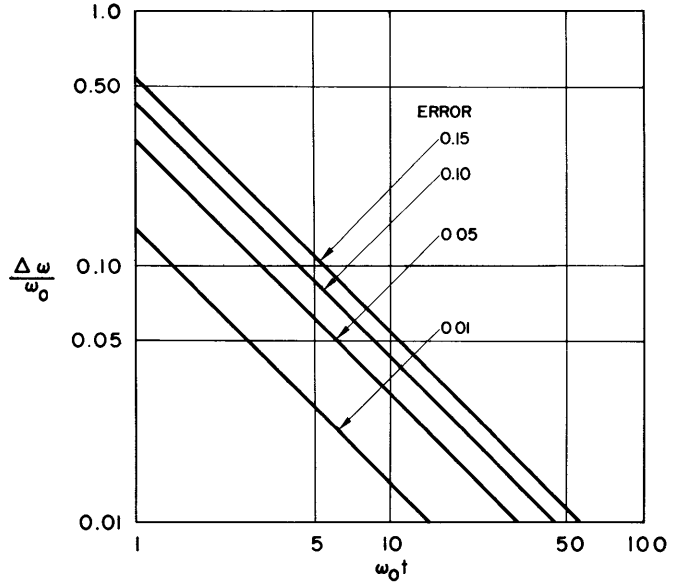


Fig. 5. Extrapolation error due to second order influence term as function of frequency and time.

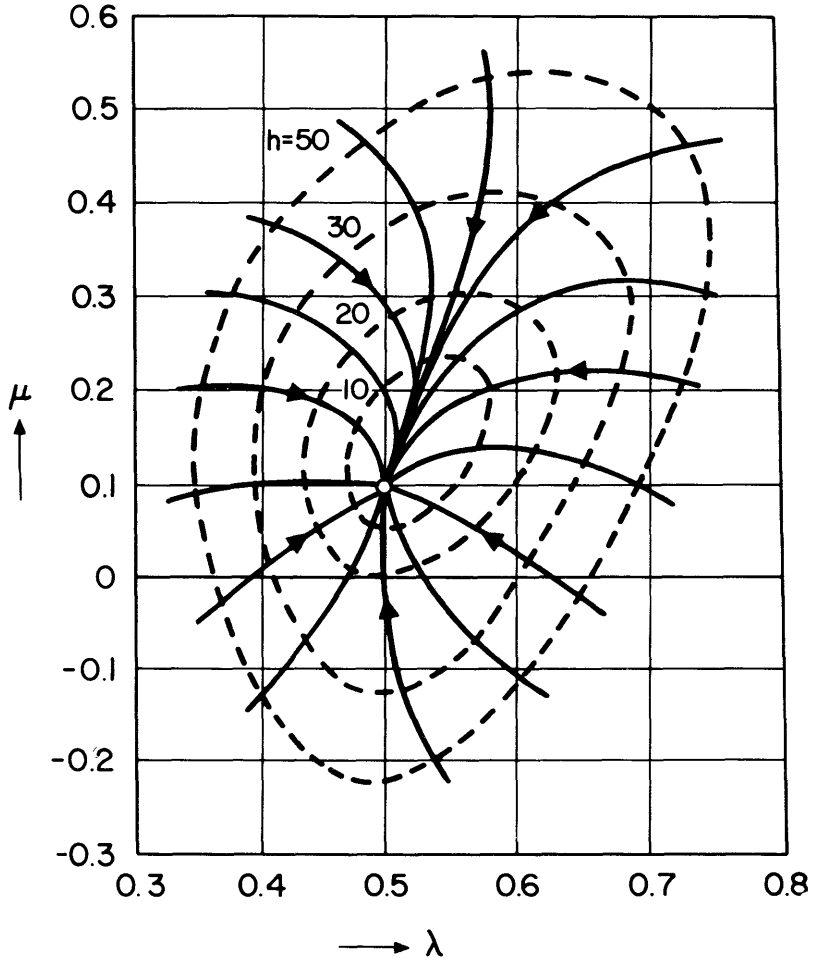


Fig. 6. Gradient and contour curves of the function $h(\lambda,\mu)$ obtained in parameter optimization by steepest descent.

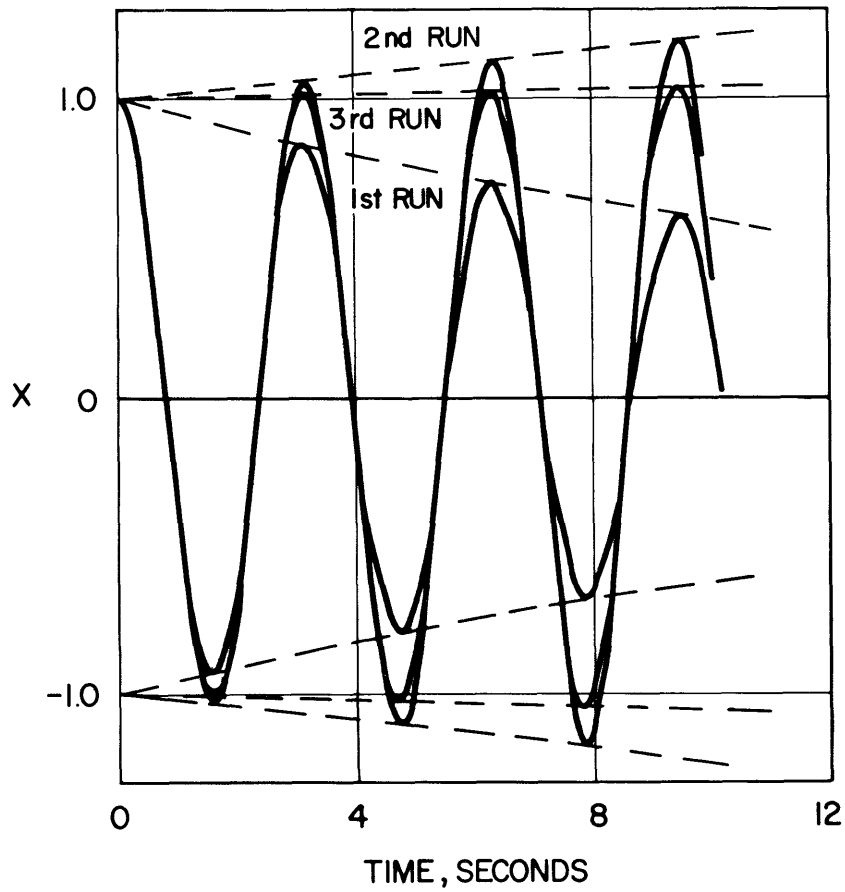


Fig. 7. Iterative solutions yielding stability boundary.

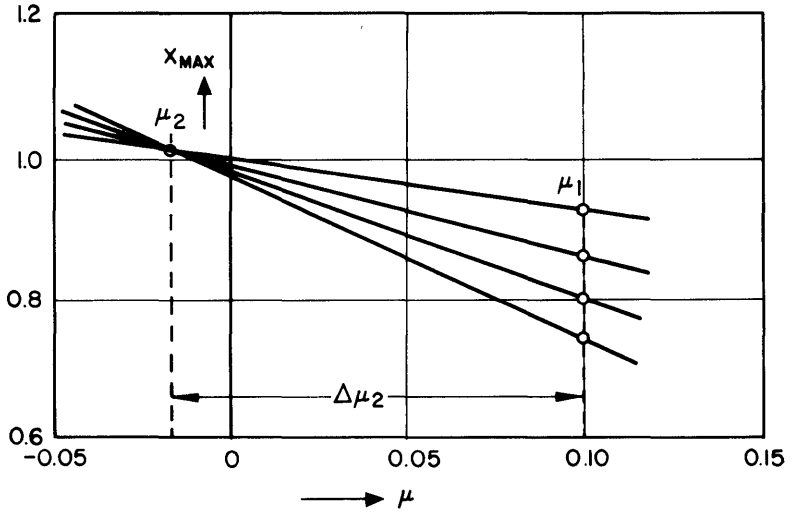


Fig. 8. Determination of stability boundary by extrapolation.

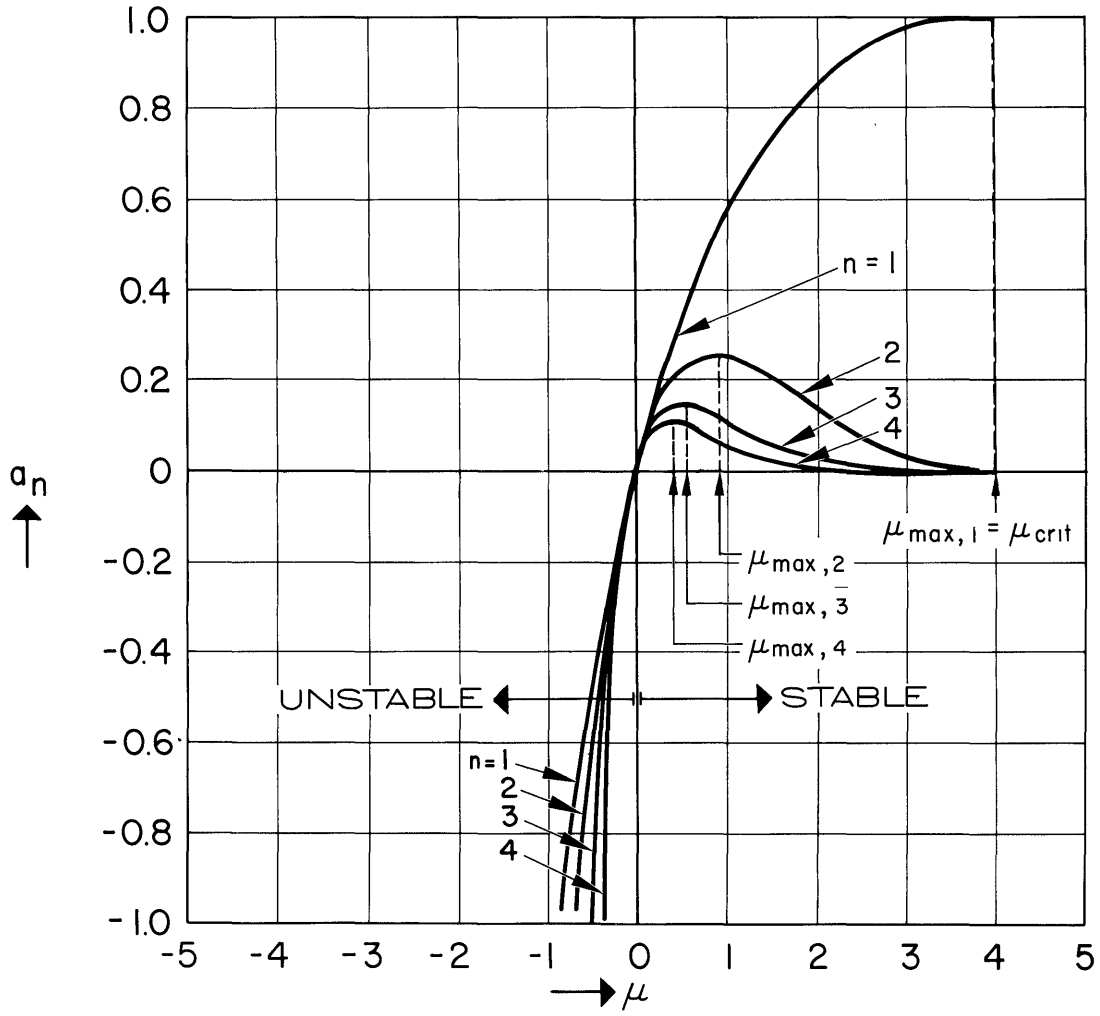


Fig. 9. Amplitude difference a_n vs. damping factor μ for linear oscillatory system.

DATA PROCESSING--WHAT NEXT?

John M. Salzer
Director, Intellectronics Laboratories
Ramo-Wooldridge
Canoga Park, Calif.

Summary

Looking into the future of data processing technology, it is possible to identify major trends in specific areas, viz., components and circuits, data systems and equipments, programming and language problems, human factors and overall system design, and learning and cerebral systems. The real advances and the real broadening of the field are directly tied to our ability to make data processing machines more truly the direct extensions of the human intellectual process.

Introductory

Outside of proving the author wrong years hence, crystal gazing has two important functions: to assess and summarize present viewpoints, and to stimulate thinking in new directions.

In the dynamic and developing field of data processing, looking at the future is particularly difficult and also particularly interesting. For an industry that started to find itself little more than a decade ago, economic growth and technical progress have been spectacular. From the early relay machines performing their operations at humanly perceivable speed, through the long hot spell of electron-tube giants, to the ultra high-speed magnetic-semiconductor computers the road has led through an astonishing number of difficult engineering achievements. Direct coding of simple numerical methods led to the complex structures of automatic assembly routines and common language programs, and thus the new science of programming came of age. The use of these data processors extended from scientific calculations--for which they were originally intended--into all facets of business, military, industry, and government. From isolated research projects there has grown a billion-dollar industry.

The end is not in sight! There will be more, there will be bigger, there will be faster, and there will be more useful systems. But what will they be? In looking at the future, we are really examining the needs of the present because if we can assume that our endeavors will be in the direction of fulfilling these needs, then we can proceed to look at the technical feasibility and the schedule of their fulfillment.

The data processing field is a combination of many efforts, whose somewhat fragmented examination will not immediately give the total picture that we desire. However, let us try to synthesize the overall endeavor after separately discussing:

*Components and circuits;

*Data systems and equipments;

*Programming and language problems;

*Human factors and overall system design;

and

*Learning machines and cerebral systems.

Circuits and Components

There is no question that progress will continue to be made toward smaller and faster components and circuits. For some time to come limitations in this respect will be the invention and utilization of materials, development of automatic and carefully controlled production techniques, and in general the ingenuity and resourcefulness of the designers. But eventually, and even as we go along, more fundamental limitations having to do with basic considerations in information theory will appear. Signal-to-noise ratio and heat dissipation will combine to make some dreams difficult to realize, others impossible. It can be shown that to transmit a bit of information, some minimum energy, however small, is required. As more bits are to be moved in a given time, the energy per second or power will go up. To dissipate this energy might require means which can invalidate the advantage of speed and miniaturization. (Figure 1.)

Recent inventions in molecular electronics, or molectronics, have aroused considerable interest. Without a doubt this method has tremendous promise. Practical and useful circuits will, in the near future, literally be "grown" for incorporation into actual equipment. But we will not grow complete computers for many years, and probably we should never do so.

Of more immediate interest are tunnel diodes, which serve as an example of the kind of developments to be expected in the semi-conductor field. The geometry and construction of the tunnel diode suggest some short-term possibilities of multi-electrode tunnels for more complex circuit structures.

Thin film memories certainly have a future. What is needed here are means of making measurements during the fabrication process so that it can be controlled; furthermore, efficient means should be found for making some of the selection circuitry itself of the thin film type. Thin film memories have been a long time coming and there is some question whether, by the time they become practical, other developed methods will not have filled the gap.

The potential of cryogenic circuits would be vastly more powerful, were it not for the inherent volatility of information. To avoid the embarrassing forgetfulness a computer could experience due to a low supply of liquid helium, many problems will first have to be solved. In addition to practical and reliable cooling devices, the combined use of cryogenic and non-volatile circuitry might offer one potential measure of safety. For short-term use such as a missile flight volatility is not

a serious handicap.

In the area of miniaturization some more immediate methods and results can be seen. The micro-modules that have been developed are certainly effective but use components with leads, and therefore require laborious automatic or manual means of putting them together and soldering them. Recently some advances were made in the utilization of leadless components and of entirely new ways of plugging circuit modules into each other. These methods, which may be practical today, permit economical circuit construction. (Figures 2 and 3.) The components would be mounted within non-phenolic boards permitting the stacking of the circuit modules within any distance from each other. Components themselves are already so small (Figure 4) that the problems of miniaturization will have more to do with the interconnections and the heat dissipation than the size of the components themselves.

Data Systems and Equipments

While the trend toward bigger, faster and more capable computers is likely to continue for some time, this trend will not be able to keep up with the ever increasing complexity of applications. Under such conditions, a problem will have to be segmented and performed on separate computers operating simultaneously. However, this approach might put serious limitations on the user, unless vastly improved communications and interconnections are possible among the various units of a large system. This is then the new direction: systems consisting of any number of different units (large and small computers, buffers, all kinds of memories, input and output units, displays, analog and digital units, etc.); high-speed automatic and completely flexible interconnections among the various units or modules of the system; and finally high-speed and automatic communication between systems.

Presently, there are many systems which can be called modular in the sense that various peripheral units can be added to the system for improved capability. But usually these systems have most of their controls centralized in the single computer whose failure will stop the operation of the whole system. A system whose controls are distributed over a large number of units, whose only central element is an electronic switching network, and whose modules can be automatically interconnected and reconnected in almost any manner has been developed and named "polymorphic" or many-shaped. (Figure 5.) This concept of system design will spread not only to newly designed equipment but also to the consolidation of many presently available modules into polymorphic complexes.

All digital computers so far marketed are one of two types: so-called algebraic or general-purpose (GP) and digital differential analyzer (DDA). It is somewhat disappointing to state that no fundamentally new and practical approach to digital computation looms on the horizon. As noted above, the innovations will arise in the manner in which these conventional computing and control units are used and interconnected. However, an extension of computer organization techniques along the lines of a variable instruction repertoire is a distinct possibility. Such a computer will be able to act as any of several different computers so that it can accept programs prepared for other computers and perform them at the same speed. This can be a significant advantage in a polymorphic system consisting of different types of units and in general in breaking down the language barriers between computers.

Strides will be made in the communication of data within and without a system. Higher speeds in electronic switching, error detection and correction equipment, message traffic consoles, message generators are among the devices to be used with increased frequency. Communication between the human and the machine will receive vastly increased emphasis, as will be noted later.

Programming and Common Languages

Advances in programming are extremely important but seldom spectacular. This has been the case in the past, and this will be the case in the future. A massive and steady effort is essential to enhance the usefulness and effectiveness of the ever-increasing number of data processing systems.

There is a bewildering variety of computers, order structures, and codes, with the result that no individual can thoroughly know more than a few. Consequently, there exists a tendency for programmers to become specialists in particular computers, or series of computers. This trend will continue for some years, but it will be partially counteracted by the more general use of automatic techniques in programming. The sheer volume of programming to be done will make it mandatory to develop more advanced machine methods to let the computer help prepare its own programs. Years from now such self-programming techniques will incorporate automatic learning methods which will be discussed in a later section.

There will be, and there should be, a steady effort toward standardization of codes, instructions, flow diagrams, program bookkeeping aids, etc. However, one must not expect rapid solutions; progress will be step-by-step, and problems will be met one-by-one. The design of common languages is an encouraging step in the direction of standardization, but this itself points up two difficulties: first, there are "languages" in plural, which indicates the limitations on their being "common"; second, the very use of common languages points up the difficulty of standardizing problem languages and machine languages themselves.

It seems that general purpose (stored program) computers will often be applied in specialized areas of application mainly because programming and other supporting activities will be aimed in such directions. The programming and applications engineers will explore certain uses which then become the special purpose of the general purpose computer. This points up the fact that the importance of the programming support in relation to the hardware effort itself will increase.

Programming methods will also have to keep pace with the new developments in computer organization and system uses. In the flexible multicomputer systems mentioned in the preceding section one of the computers acts as master and determines which units of the polymorphic system will carry out what task and when. The design of the master control programs is a new and challenging programming endeavor.

Operational systems are often used by a large number of humans who are at best only moderately trained in computer and programming technology. Means should be provided for them to get at the programs readily and to perform modifications with relative ease. This problem in programming has hardly been touched, but its solution would result in efficient adaptive man-machine systems.

Another need in intellectronic systems is a quick machine response to random and variable requests of the operators. A programming technique which may offer a solution is the use of programs which are segmented into their component parts. These parts would be put together in accordance with the task to be performed and at the time that the task is to be performed. We can think of this method as on-line assembling and compiling, the need for which will increase with real-time applications of data processors. Where the inputs to the data processing systems are random requests coming from various sources and exhibit significant variations from time to time, the on-line composition of a program to handle each specific request may prove more efficient than the storage of individual programs to answer all possible requests.

Human Factors and Overall System Design

Data processing systems--particularly real-time operational systems--will be more truly the servants and the extensions of the human intellect if the interrelationships between man and machine are given appropriate consideration at all stages of system and equipment design. In intellectronic systems the human needs and the operator characteristics are basic factors.

In the early stages of computer development the design of memory, arithmetic and control circuits and equipment was the first consideration, and the matter of input and output often became an afterthought. This trend has, of course, been reversed and ever increasing attention is being paid to the input and output problems and the convenience of the operator. Yet, the operator must often be carefully trained in order to make it possible for him to use present-day machines appropriately and adequately. As more systems are used in real time, the careful training of the operator in order to permit him the use of the system is not always feasible. It is necessary, therefore, to make designs which allow the use of the system by a human only moderately trained.

As man becomes a "black box" in the overall system, human factors studies will become an integral part of system design. Two aspects of human factors can be distinguished: human engineering, which concerns itself with the relationship of the human and specific piece of equipment such as a display device, or an input console; and man-machine system studies, which concern themselves with the overall interactions in a system complex consisting of humans and equipments. The latter aspect of human factors is gaining increased significance both in the military and in the industry.

Think of using the library one day in somewhat the following manner. You sit down at a desk-like piece of furniture (Figure 6) and you insert your library card in an appropriate slot. Then you type out the subject matter for which you are searching. Note that you do not look up any indexes, glossaries, books on codes, synonyms, etc. Having typed in your request, an abstract flashes on the screen for you to examine. From the nature of this material you determine how well or poorly you might have stated your request. For example, if the abstract indicates that your request must have been too vague, you may type in additional words or descriptions for increased clarity. As you keep on examining abstracts, you keep modifying your requests as necessary. Finally the retrieval reduces to, say, a dozen relevant abstracts. You go back and re-examine these abstracts, which appear by easy push-button control. This more thorough examination might reduce your total interest to three articles, or books, which you then indicate by a push-button as being your selection. The selected material slides down the chute at the left lower section of your desk. You remove your library card from the slot, pick up the books, walk out; you are already charged for them.

The type of human factors design consideration that must go in the future library system will be part of the design of many intellectronic systems. Such designs are not as far in the future as one might think; they are starting to be applied in military command systems and they will be shortly applied in management systems of business enterprises. Teams of engineers, psychologists, physiologists, and other specialists will rapidly become part of any large data processing design and applications effort.

Part of any systems effort is the design and development of individual items of equipment. In systems where the human is also an element, his design and development should also become part of the systems activity. Corresponding to the design in the case of a human is the selection of the human from the random population, while the development effort in this case consists of the training of the human. Both the selection and training of personnel must therefore become an integral part of the system design effort. It should never be the design engineer himself, or his colleagues, who try out the system for its operational feasibility; rather the type and calibre of personnel who will actually use the system must also be the participant in the experiment. In this manner one has a greater assurance that the system will serve its operational purpose.

Considering the training problem one can readily see how the difficulty of a training method can react on the design of the overall man-machine complex. There must be a balance of trade-offs between the complexity of the system and

procedures, and the difficulty of training the operator to use the system. Once the system is designed and operational, an important task is the training of operators to use these systems. From the point of view of future development in this area we will see more automatic aids for training people. Training machines will be special-purpose devices having the appearance of the actual human factors consoles and displays to be used in the system, but having simple special-purpose storage and data processing capabilities which can simulate the actual overall data processing system.

The topic of training brings us naturally to the educational field. Our new capabilities in data processing and human engineering are bound to have a long-range effect on our educational methods and indeed, on our whole educational system. Already, television and closed circuit television have both demonstrated the tremendous potential of new techniques in education. Incidentally, the successful introduction of such methods depends to a large degree on human acceptance. The psychologist's concern about how the student reacts to a two-dimensional teacher has just as much to do with the success of this new method as the solution of the technical problems of transmitting a picture. In an effort to find out more about the human element in an educational system, a large number of measurements have already been made and such new scientific disciplines as neurophysiology, and psychophysiology have been identified. Emotional and physiological reactions are measured in order to optimize the student-machine-teacher systems and methods. The bookkeeping and student control problems themselves of an educational institution also deserve the fullest consideration of data processing technology. Training aids, study rooms, and decentralized user consoles for centralized libraries are some of the items subject to automation in future intellectronics systems for education.

Learning and Cerebral Systems

One of the most fascinating and interesting research activities of our field will concern self-adaptive and statistically structured automata. Presently this whole problem area is being attacked on at least three different fronts: (1) cybernetics, which leans on neurology, physiology, and biology to study the detailed microscopic behavior of complex human and animal systems; (2) psychology, which is interested in human behavior in the macroscopic or overall black-box sense, and (3) data processing, which investigates programmed or structured automata from the point of view of their similarity with living cerebral systems. The interaction among these three fronts have been noteworthy, and will become much more substantial in the future. Progress in this field will be difficult to come by, but the results could be quite spectacular. It is a field where, hopefully, more of our talent will become active, and where more research dollars could be rightfully spent.

Considerable work has been performed on structured machines in which a certain number of inputs are connected to a certain number of outputs through redundant paths. The nature of these paths can be adjusted on the basis of performance and this provides the system with adaptive or learning capability. Often these systems are simulated on a programmed computer. The structures so far examined have been very simple compared to the complexity required to simulate a cerebral system. But even so, the preliminary results are interesting and promising.

As the number of inputs and outputs in a structured machine increases, the complexity of the device assumes astronomical proportions. A truly meaningful learning device would have billions of bits of information stored, and these cells would have to be interconnected by channels having complex adaptive mechanisms. To implement such a device, component and circuit development would have to advance by orders of magnitude. Yet, we do not have to proceed all the way. It is feasible to construct within years structured learning machines of complexities which, although inadequate for practical use, will be meaningful for significant contributions to this area of research. It is time to apply appropriate circuit talent directly to the problem of implementing, at least in a semi-practical fashion, the next degree of complexity of such machines. The kind of circuits that will be required will probably not be straight binary or straight digital, but will exhibit mixtures of continuous and discontinuous behaviors. It is a new horizon for the circuit specialist.

Closely related to the field of learning machines is pattern perception. We are starting to learn quite a bit about what parameters and characteristics are important in describing a pattern. Thus, we know in some cases that a point-by-point description of a shape or event is unnecessary, and we further learn about what parameters are necessary to describe the event. What is less known is how to measure these parameters directly. At the present time we are still forced to measure a shape or event point by point, and to use automated inductive reasoning in deriving or synthesizing the desired parameters. Again, it will be a mixture of digital and analog techniques as well as a mixture of definitive and statistical considerations that can provide us with some answers.

Four stages of development can be distinguished in the pattern automation field: pattern following, pattern generating, pattern matching, and pattern recognizing. Pattern following is a straight analog servo problem extensively exploited in the machine tool control field over the past years. Pattern creation, or generation, is again used in the machine tool field. For example, a circular pattern is generated by defining such parameters as the center of the circle, its radius, and the beginning and end of the arc. Pattern matching serves to automatically determine the similarity of two patterns. For instance, using optical means, it is possible to recognize whether a specific word or set of specific words is, or is not, contained on a printed page and where on that printed page it is contained. Correlation techniques have been developed to get a certain consistency and reliability in recognizing predefined patterns, which may be keywords or signatures. The importance of this in library search and

retrieval, as well as in the field of business and banking, is readily visualized. Pattern recognition is the last and ultimate step. It is quite conceivable that pattern matching techniques in combination with feed-back or trial-and-error methods, may become the mechanism by which patterns can be recognized much before straightforward pattern recognition methods are fully developed. At any rate, this area poses a real challenge and the promise of leading to most significant future results.

Learning and adaptive systems are rapidly becoming not only important catchwords to attract contractual support, but also practical devices that might improve the performance of control and data processing systems. For instance, in the field of machine translation of languages, the translations are now scrutinized by human linguists who find the shortcomings and change the program in order to improve the automatic translation. The next step is the design of programs that can improve their translating capability upon the mere indication by the human that a shortcoming of a specific sort has been observed. The same comments are applicable to indexing and abstracting systems which would constantly monitor the satisfaction or dissatisfaction of the human consumer in order to improve their methods of indexing and retrieval. These methods could also be applied to the development of common languages.

The tasks ahead of us in this field are tremendous and require the support of substantial research efforts over periods of time, not to be considered short.

Summary

When we call ourselves a computer society, most of us realize that the word "computer" is used to cover a multitude of related activities. In trying to describe the broadening horizon of our activities, we started to use expressions such as "data processing", "cybernetics", "information systems", "automation", and more recently, "intellectronics". These words are not just words: they are coded symbols of our professional aims.

From our brief review of things to come, we can conclude without a doubt that the advances in our technology will be startling, and indeed they seem unlimited. Computers will be small enough to become vastly more useful in many new applications, and powerful enough to perform almost any conceivable task. Programs will become more generally useful to facilitate the solution of a greater variety of problems. But the real advances and the real broadening of our field are directly tied up with our ability to make these machines more truly the direct extension of the human intellectual process.

Computers are sometimes named "bright but ineffective children". When we consider the time required for a new data processing installation to become fully effective, or the time to prepare a new problem for the computer, or the difficulty of translating a program prepared for one computer for use on another, or the length of training required to make the operator properly conversant with the machine, or the procedure required to make even a simple change in a program while it is in the computer, or the unnatural manner in which the computer would go about recognizing a pattern, or the clumsy indirect means of communication between the human and the data processor, then we must admit that there is a lot to be done to gain effectiveness. These problems are our new challenges, and their solution will constitute the milestones of this developing technology.

The effort that will be put into new and miniaturized equipment, greatly advanced programming techniques, vastly improved communications between machine and machine, man and machine, machine and man, and perhaps man and man, statistically structured information systems, will be justified--in the final analysis--by the economies and the effectiveness of future intellectronic systems. The increased scientific and engineering support needed by this field will presumably come from the vastly increased utilization of automated systems in the military and in the industry, or indeed, by the general public. The applications willcover a broad spectrum. Data processing and computing aids will permeate all facets of our life. But progress will be marred by an effect which industry does not await with pleasant expectations. The ratio of research-scientific-engineering dollars to be expended for each production dollar, and perhaps even for each dollar saved in operation, will continue to increase. This is the price of automated intellect, and this is the price which must stay commensurate with the conveniences gained and with the growth of our economy.

The wider application of automated aids to our intellectual processes might have some interesting effects on our approach to problem solution. The tendency of not developing a broad base of fundamental scientific know-how but rather depending on the ready means of having reference to relevant material and eventually having at our disposal--pardon the expression--"thinking aids", is a truly significant trend of our times. Today we might say "I will read it when I need it," and tomorrow we might say "why read it when I can get the answer to my question directly?" Such tendency is very much in line with the growing complexity of our everyday life.

It is hard to see where this intellectual revolution of the Twentieth Century will lead. For it is important to maintain the proper perspective on what it is we are up to. Let us not propose to install a super computer in the White House to run the country by direct control. Nor let us squander our energies and resources in providing the housewife with a computer to let in the cat in the morning. Between profundity and trivia, let us remember the human element and steer a prudent course to the benefit of our society. Only the practical fulfillment of sound human needs can guarantee the healthy growth of our technology.

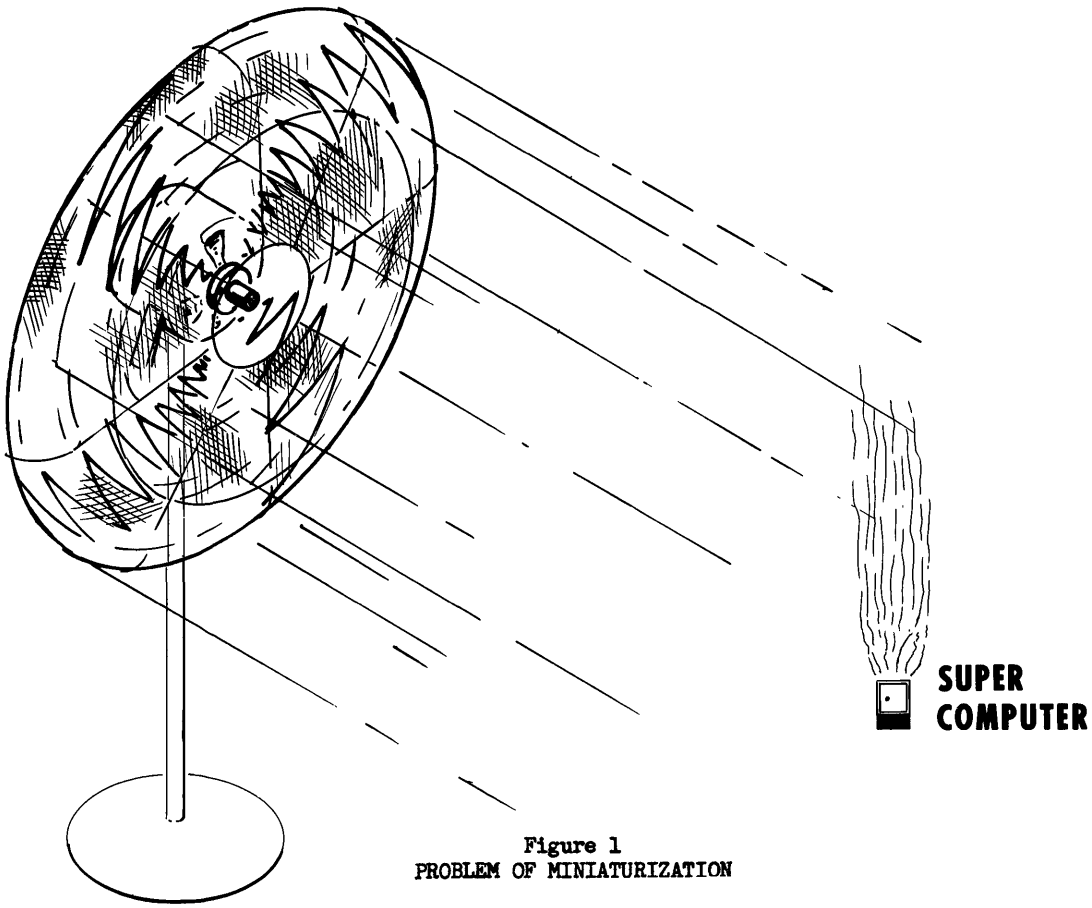


Figure 1
PROBLEM OF MINIATURIZATION

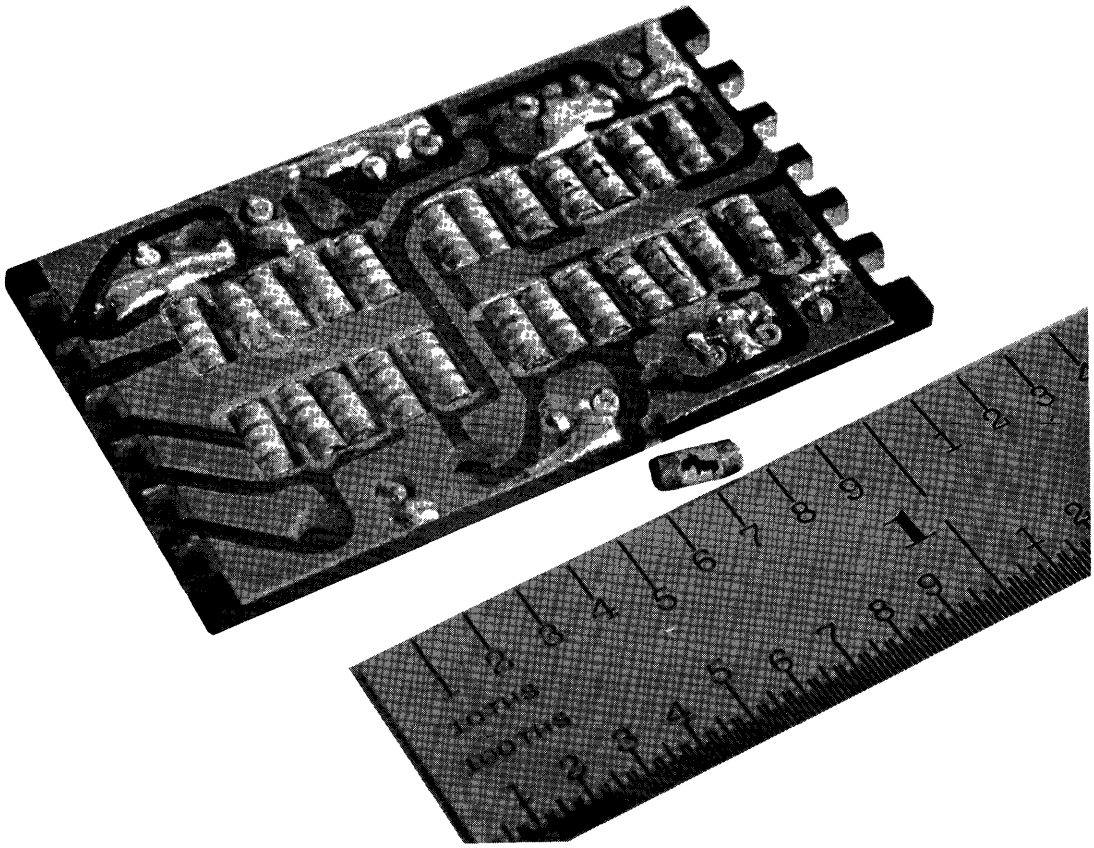


Figure 2
MICROMINIATURIZED CIRCUIT BOARD

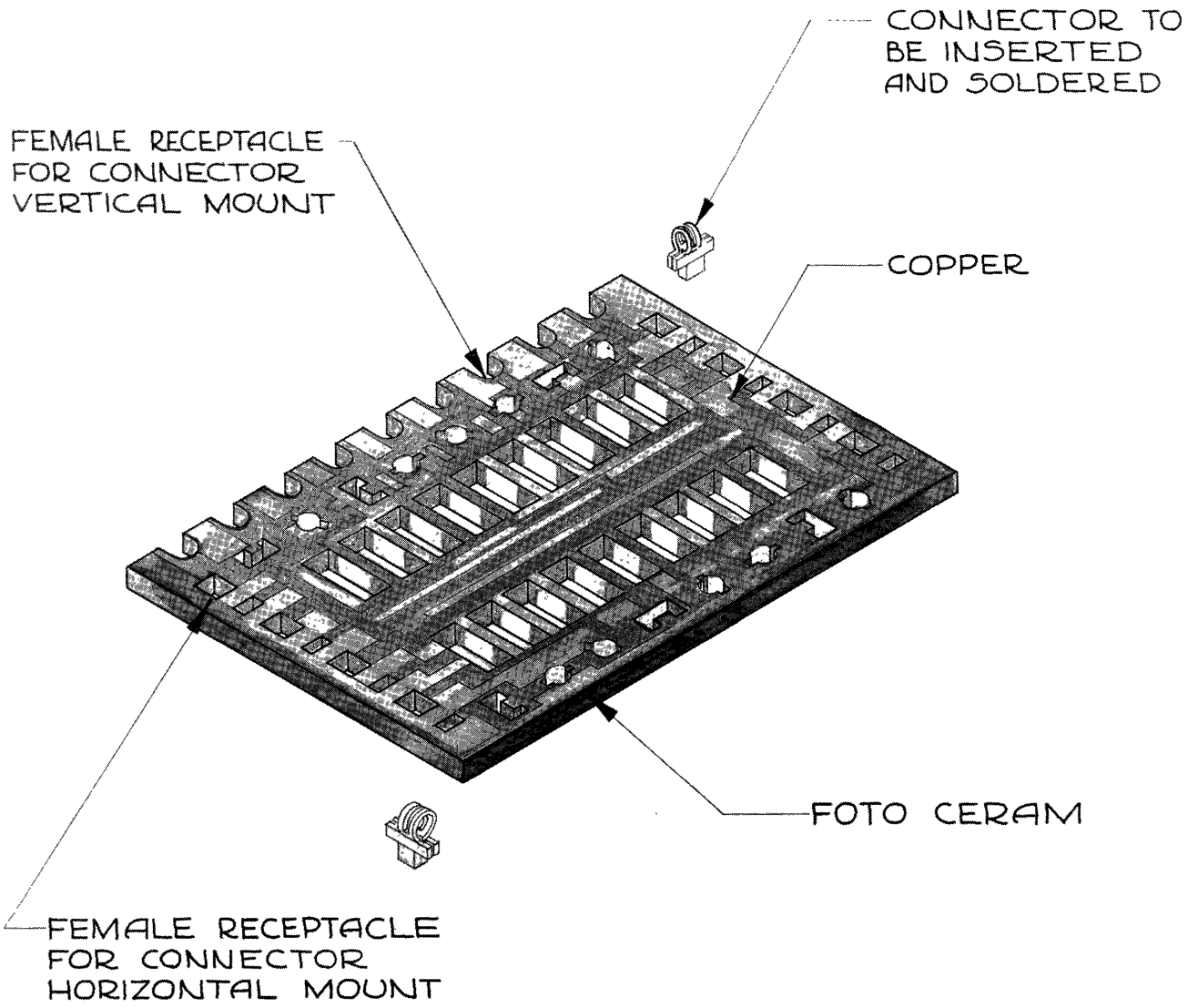


Figure 3
CONNECTORS FOR MICROMINIATURIZED CIRCUIT BOARDS

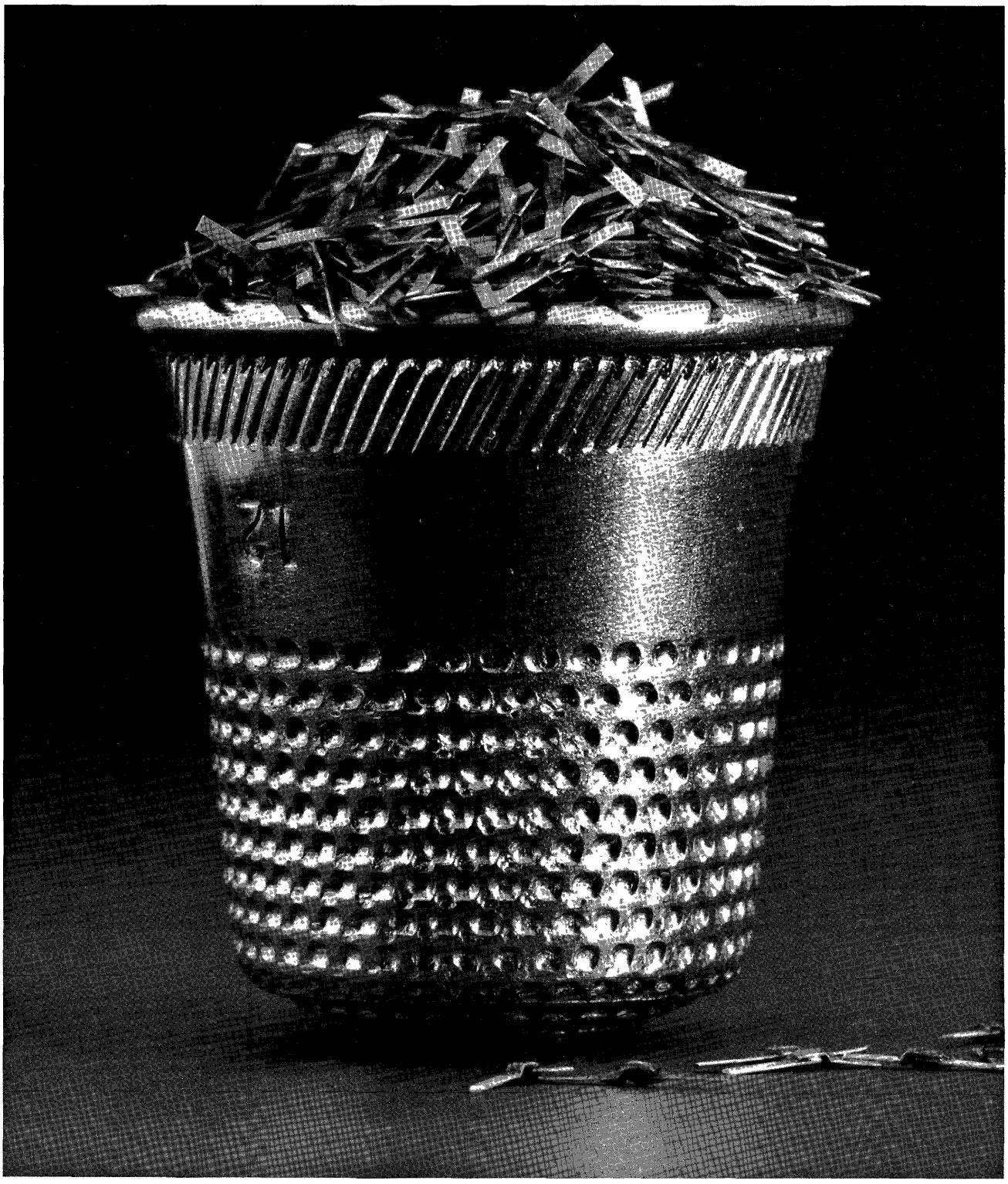


Figure 4
A THIMBLEFUL OF MICROMINIATURIZED DIODES

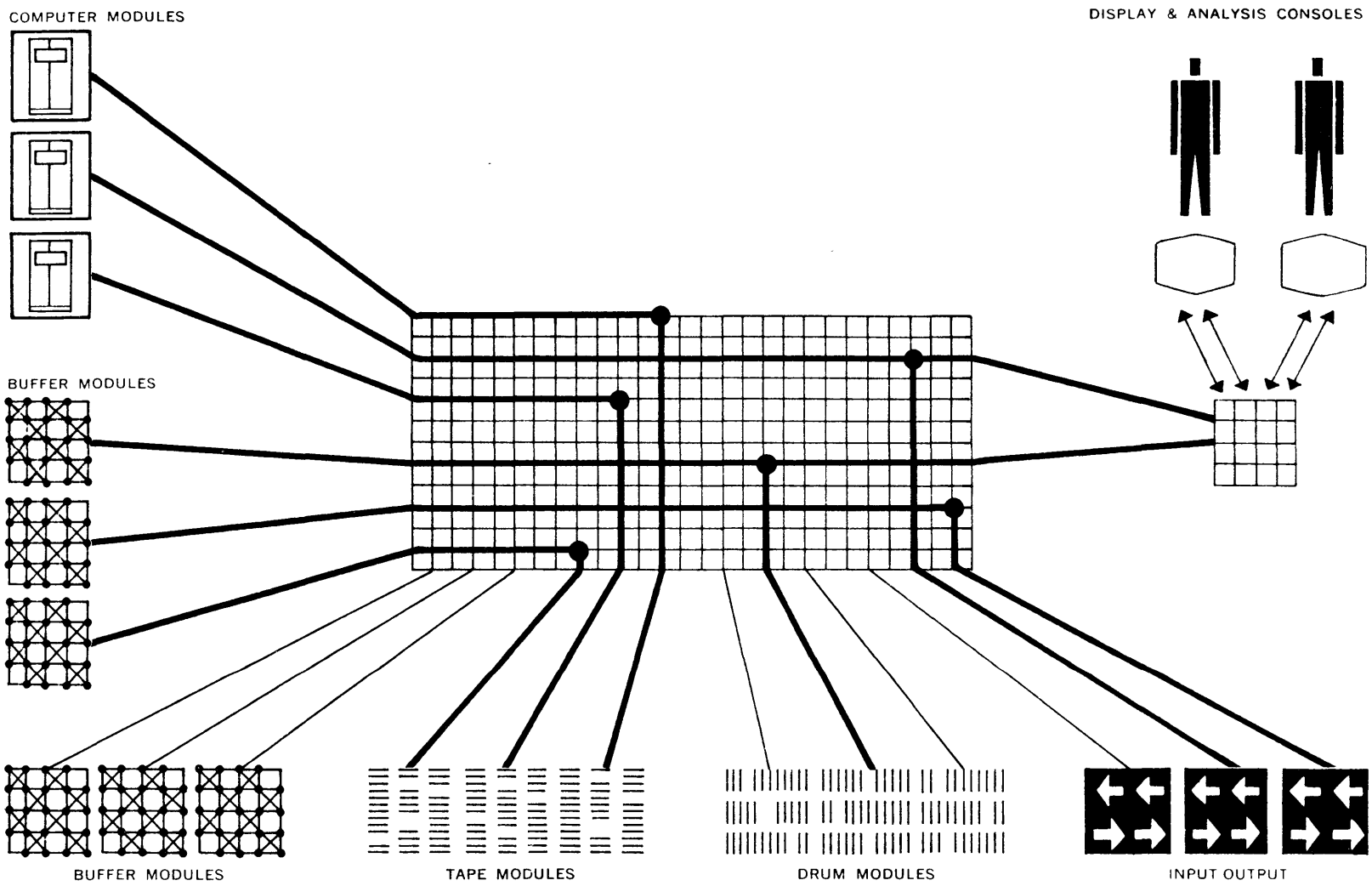


Figure 5
POLYMORPHIC DATA PROCESSING SYSTEM DIAGRAM

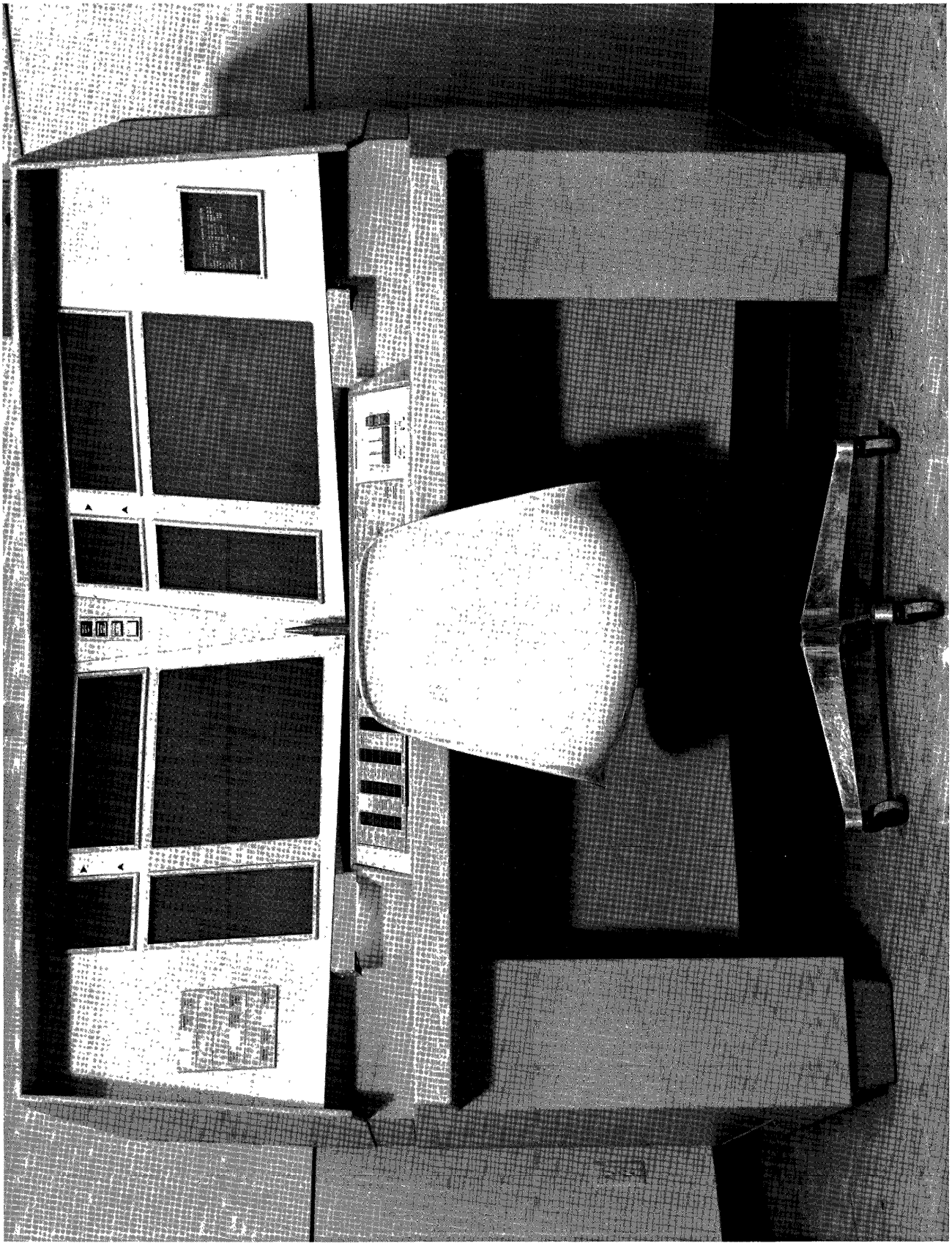


Figure 6
DISPLAY AND ANALYSIS CONSOLE

THE OUTLOOK FOR MACHINE TRANSLATION

Franz L. Alt
National Bureau of Standards
Washington, D. C.

Introduction

The idea of using electronic digital computers for language translation seems to have arisen about 1946; it was first brought to widespread attention in a memorandum by Warren Weaver[1] in 1949. Today there are about a dozen projects in the United States devoted to machine translation, mostly translation from Russian to English. Many additional projects are located in the Soviet Union, a few in other countries. Many of these projects have been in operation for a number of years. Yet today there is nobody in a position to feed some foreign text into a computing machine and produce automatically an acceptable translation. This is not to say that there have been no accomplishments. Some achievements may be noted, in some other areas there has been considerable progress; and it seems plausible that in a few years completely automatic translation of fair quality will be possible.

Most projects in the United States have been devoted to translation from Russian to English, although French and German have also been considered as source languages. In some projects, studies of the English language have been conducted in the hope of benefiting machine translation. The effort in the Soviet Union, from the scant information available, seems to be of a magnitude comparable to that in the United States, but spread over more language pairs. Although the two major projects in Great Britain have considered translation from French, German and Russian into English they have concentrated mainly on general questions. Not only do different projects work on different language pairs, but they frequently attack different portions of the translation problem. Many limit themselves to source texts in specific fields, such as chemistry or mathematics; some concentrate on the theories of linguistic structures; some on the compilation of a dictionary; some on methods of transcription of the original text onto punched cards or other machine media; some on so-called automatic programming for language translation, which in this case means the creation of general-purpose machine programs which can perform a variety of linguistic operations. When they deal with the central problems of machine rules for analyzing the source words and synthesizing target sentences, there are many differences in the methods which are used. In order to appreciate these differences, we shall first have to survey the main difficulties in the way of translation and the methods that have been proposed to overcome them.

Methods of Machine Translation

When the idea of automatic translation emerged, at first only word-for-word translation was considered. The inadequacies of this plan were obvious. They manifested themselves principally in the facts that some source words had more than one target word associated with them, and that the word order in the target language often has to be different from that in the source language. Speaking of multiple target words and of word order, however, is not the most useful approach to classification of problems. The change in word order can be understood by analyzing the grammar of a sentence. Multiple target words can be either different grammatical forms belonging to the same stem, or words having entirely different meanings. Thus, in the main, the problems encountered in translation are classified into syntactic and semantic problems.

The first attack on semantic problems was outlined in the memorandum by Warren Weaver[1] in 1949. Early ideas ran as follows: A word like "nucleus" might have one meaning in a context of physics, another meaning in a context of biology, etc. There are other words which occur only in contexts of physics, still others which occur in biology, and so on. In the dictionary, each word would be coded according to the fields in which it is used. If a word of multiple meaning is encountered, a number of neighboring words both before and after the questionable one would be searched in order to get a "majority opinion" concerning the field with which the present text is concerned.

This idea of word classification by context does not dispose of the problem entirely. The choice among multiple meanings may be determined not by context but by certain successions of words. For instance, the same Russian word may have to be translated into English as "prove" in the sentence "Prove the theorem of Pythagoras" or as "show" in the sentence "Show me how to construct a regular pentagon". Attempts are being made to establish large-scale statistical information about pairs of words occurring together. Another approach has been to attempt a more refined classification of word meanings, resulting in something like an oversized Thesaurus. Still other investigators maintain that the magnitude of the semantic problem can be greatly reduced by attempting whenever possible to find neutral translations which will cover as many meanings of the source word as possible.

The three approaches to the semantic problem which we have just outlined might be named the statistical, systematic and empirical approach. The same three methods of approach can be distinguished in dealing with the syntactic problem.

Here, the statistical approach consists in searching through large amounts of source texts and enumerating the frequency of certain word sequences. For instance, how often does an adjective precede a noun, how often does it follow it? The systematic approach attempts to set up a system of rules--in other words, a machine program --which analyzes the syntactic structure of each

source sentence. That is to say, it identifies subject, predicate, direct object, etc. of each sentence or clause. This frequently has to be preceded by a grammatical analysis of each word, just as conventional grammar is divided into morphology and syntax, the former dealing with the inflectional forms of each word, the latter with the function of each word in the sentence. Finally, the empirical approach starts by selecting a few very simple rules for translation, tries them on a body of text and notices where they fail, corrects the rules or introduces new ones to cope with the observed failures, tries the revised rules on a larger body of text, and so forth.

From a slightly different viewpoint we may distinguish between the use of conventional grammar and the design of new systems of grammar (or "linguistic structures", as they are called), which are intended to be better suited to mechanical analysis than is conventional grammar.

There are other differences among the various groups working on machine translation. For instance, in designing the dictionary or glossary, some propose to list in the glossary every inflectional form of every source word, while others propose to list only the stem, or equivalently some canonical form such as the infinitive of a verb and the nominative singular of a noun. Russian nouns have a dozen inflectional forms, adjectives and verbs many more. Thus the size of the required glossary is greatly affected by this decision.

Other differences are found in limiting the scope of a machine translation project. Some groups are satisfied to translate into a kind of pidgin English. Some are resigned to leaving certain semantic ambiguities unresolved and printing out multiple meanings. Some are willing to admit failure in a small percentage of all cases. Some will even admit undetected errors in the translation (The latter is a point of view which I consider dangerous). Some propose to use a man-machine partnership rather than letting the machine do the entire job. In these cases the machine prepares certain aids to translation--at best a kind of preliminary draft, and these are used by a "post editor" in producing a polished translation. Pre-editors are less frequently contemplated, but a certain amount of pre-editing may be combined with manual key punching of the text.

It is my feeling that, in dealing with the syntactic problem, the empirical approach is most likely to give an early appearance of success, by leading in a short period of time to a system for producing less accurate but serviceable translations. It will prove quite difficult to improve the quality of translations later on. The systematic approach may take a little longer in achieving its first concrete results, but these will be of higher quality and will lend themselves readily to further improvement by adding more and more refined rules of grammatical analysis. The statistical approach to syntax seems to me to be far from successful. On the prospect of success in the semantic problem I have no present opinion at all.

Example - the Approach of the National Bureau of Standards

The problems associated with mechanical translation may perhaps be better appreciated by looking in detail at one example. For convenience I propose to use the work now in progress at my own organization.

The project at NBS, under the direction of Mrs. Ida Rhodes, concentrates on problems of syntax and dictionary organization. In its approach to syntax it stays close to conventional grammar. We may visualize the proposed machine program as consisting of two parts, the first concerned with glossary look-up and morphological analysis, the second with syntactic analysis. We look upon constructing such a machine code as similar to setting up a mathematical theory of language. Some idiosyncrasies of the project arise from the fact that its staff consists mostly of mathematicians, and coincidentally we are using mathematical texts for our experiments.

The dictionary look-up proceeds on the assumption that, in any foreseeable machine scheme, memory capacity will be a bottleneck. At this point of our exposition this is a mere assumption, but we shall see soon that it is well founded. It will be impossible to store the entire dictionary in the internal memory of the machine. It will be stored on an external medium, for instance on magnetic tape or on a magnetic disc memory. Since reading from an external medium into the computer is relatively slow, every effort is made to store the dictionary as compactly as possible. This desire explains some of the unusual features of the machine program.

The program starts by reading a few hundred words of source text into the machine. Each word is decomposed into its inflectional ending, prefixes, suffixes, and the root. Endings, prefixes, and suffixes are identified by comparison with stored lists, and are replaced by numerical codes. The roots are put in alphabetical order and are then looked up in the dictionary by a single pass through the external medium stored in the dictionary. There, we find the source words stored in alphabetical order of the roots, listing under each root those combinations of prefixes and suffixes which occur in the source language in combination with this root, each followed by grammatical information about the source word and by one or more English equivalents. This information is read into the internal memory.

The foregoing exposition is oversimplified in several respects. Sometimes the machine obtains, in place of a prefix, a group of letters which only looks like a prefix but which grammatically is not one. Such a group is called a pseudo-prefix. Similarly there are pseudo-suffixes. What is left after splitting them off is called the pseudo-root. For instance, if the method were applied to English, the word "conifer" would be split into the pseudo-prefix "con", the pseudo-suffix "er" and the pseudo-root "if". Another point that has been oversimplified is the arrangement of the dictionary. Although we called it "in alphabetical order of the roots" the program actually uses a more elaborate and more economical scheme.

The information obtained from the dictionary, together with the previously identified ending, results in a morphological description of the word in question, which is stored for later use. In the second part our code works on one sentence at a time and establishes the elements of each sentence, such as subject, predicate, direct object, etc. In this process we use a device which, to the best of my knowledge, is unique. As each word is identified, it is used to "predict" other sentence elements. For example, a transitive verb predicts a complement in the accusative case, etc. All these predictions are stored and each new word is compared with them. If no match is found, this very fact is stored in the expectation that it will be resolved by a subsequent word of the sentence. Thus, throughout this part of the program we have two pools of stored information which we call foresight and hindsight, and which assist in determining the syntactical function of each word in the sentence. This syntactical information, in turn, indicates how our translation should differ from word-for-word translation. For example, it indicates changes in word order, the insertion of English prepositions like of or by, etc.

This account of the method being developed at NBS is necessarily brief and omits many essential features, for instance, how we deal with clauses and phrases within a sentence, or with word forms which are morphologically ambiguous. What I have said will, however, suffice for our present purposes. To date the dictionary look-up and morphological analysis has been coded, and the syntactic analysis is in process of being programmed.

Survey of Projects

The most important ones among the mechanical translation projects in Western countries will be enumerated here, in geographical order from West to East, and each project briefly characterized. For a recent complete survey, see e.g. Bar-Hillel.[2]

1. University of Washington, one of the oldest projects, concerned with translation from German and Russian to English. Extensive dictionary of inflected forms. Word-for-word translation, study of selected grammatical and semantic problems.

2. University of California, Berkeley, one of the youngest projects. Russian to English. Emphasis on economy in the use of the dictionary.

3. Rand Corporation, Russian to English, empirical and statistical approach, some work on semantic problems. Large corpus of transcribed Russian material in Physics and Mathematics.

4. Ramo-Wooldridge, Russian to English, empirical approach, emphasis on programming methods and on display of results in a form which facilitates continuing revisions. In its approach to grammar this project is close to one of the groups formerly at Georgetown University.

5. University of Texas, a newcomer. German to English, concentration on grammatical problems.

6. Wayne State University, a new group, Russian to English, statistical approach. Cooperates with Ramo-Wooldridge and with a group formerly at Georgetown University.

7. Georgetown University, one of the oldest groups, at one time comprised four separate projects. One of these is disbanded, its ideas being continued at Ramo-Wooldridge and Wayne. Another, Russian to English on an empirical basis, recently moved from Georgetown to a private corporation, CEIR.

A third project at Georgetown, called "General Analysis Technique", Russian to English, has made considerable progress with syntactic analysis, which is being developed in part empirically and in part systematically, staying fairly close to conventional grammar. A good-sized dictionary has been assembled and a large corpus of text examined. Heavy reliance is placed on post-editing.

The fourth of the Georgetown projects works on French to English, is strictly empirical, lays stress on advanced programming techniques.

There are also small-scale efforts devoted to Chinese and Arabic.

Most projects at Georgetown are concerned with source texts in the field of chemistry.

8. National Bureau of Standards, a relatively recent project, Russian to English, systematic approach to grammatical (morphological and syntactic) analysis, staying close to conventional grammar. Emphasis on economical use of dictionary. Uses source texts in mathematics.

9. University of Pennsylvania, systematic approach to grammatical structure of languages, esp. English.

10. IBM Corporation, concerned mostly with hardware, with some supplementary systems studies.

11. Massachusetts Institute of Technology, one of the oldest projects, has pioneered in developing new theories of the grammatical structure of languages. Also works on syntactic problems of German-to-English translation and on general-purpose programming systems for machine translation in general.

12. Harvard University, Russian to English, has compiled a large dictionary and worked systematically on methods and machine codes for dictionary compilation and updating, and on word-for-word translation. More recently concerned with grammatical analysis on lines similar to National Bureau of Standards, and with general theory of language structures. A related project is in operation at the Arthur D. Little Company.

13. Birkbeck College, University of London, apparently the oldest group in the field. German, French and some Russian to English. Morphological analysis of source words, some syntax, largely empirical.

14. Cambridge University, England. General linguistic theory, semantic problems.

15. University of Milan, Italy. A highly theoretical, long-range approach.

There are a few other projects, either too small or too new to be characterized separately. They include one at the National Physical Laboratory in England and one in France.

Economic Considerations

We now turn to the discussion of the cost of using electronic computers for language translation. Cost depends on the system used, i.e. on equipment and methods. Although we shall mostly

deal with cost in dollars, there are elements of cost not readily expressible in terms of money, such as the damage caused by time delays or by erroneous translation.

There are so many uncertain and unpredictable factors in the situation that we can discuss only orders of magnitude. In some cases we can estimate within a factor of 2 or better; in other cases we are lucky to come to within a factor of 10 of true cost. Sometimes, rather than assign a cost to a process, we shall state the conditions under which this process becomes competitive with others.

Human Translation

The cost of translating "by hand", i.e. by human translators, seems to range from 0.25 to 4 cents or more per word. If quoted per page, one hears figures ranging from $12 to $25 per page. The cost depends on the qualifications of the translator, on the technical difficulty of the text, and on the degree of perfection, smoothness of style and appearance desired. It is cheaper in foreign countries than in the United States. As an indication of order of magnitude we may use the figure of 1 cent per word. In this connection "word" means the usual average of five characters followed by one space. To avoid confusion with the "machine word", the unit of memory space in computers, some authors use the word "grubit" for the latter. It happens that on many machines a word corresponds to about six characters.

Apart from the monetary cost, human translation suffers from its slowness and from the scarcity of experienced translators, especially for technical subjects. The quality of translation is variable, and from time to time gross errors appear which could be avoided by translators with greater technical competence or by the use of very detailed technical dictionaries. By and large, however, the quality of human translation is far superior to anything in sight of mechanical translation.

The cost of mechanical translation may be divided into the two elements of initial cost and current cost.

Initial Cost

Under this heading we shall discuss the cost of developing programs and codes for automatic translation.

As always in cost accounting questions, there is some ambiguity about where some specific items should be charged. In the present problem, it could be argued that the initial cost of developing the hardware should be treated as an element of the initial cost of translation. It is, however, my contention that mechanical translation will, if at all, be performed by general-purpose equipment which would be developed in any case, regardless of its application to automatic translation.

Current spending in the United States for the development of machine programs for translation probably amounts to between 2 and 3 million dollars per year. The effort maintained in the U.S.S.R. is of the same magnitude, though perhaps at lower cost. Including projects in other countries, an amount between 5 and 10 million dollars per year is being invested in this work. One may speculate that at the present rate it will take at least five years before the system is really perfected, notwithstanding certain glowing newspaper accounts about systems supposedly now in operation. It is true that certain simple-minded things can be done right now, certainly the word-for-word translation and probably something a little better. It is also very probable that some concrete and respectable progress will be visible two or three years from now but I should estimate at least five years before a really satisfactory system is reached. Even this may be an underestimate. Before we are through, the world-wide investment in these programs may be $50 million.

Now there are good reasons for saying that this investment need not be amortized. Present research in automatic translation is of great scientific interest, will result in deep insights into the working of the human mind and in great improvements in our use of computers in general, and is worth being carried on for its own sake. But let us take the narrow viewpoint and insist on amortizing it. The question is, over how long a period of time or what volume of work? Let us assume that, after a number of systems have been perfected and the best of them selected, we may wish to translate, over a number of years, a total of 40,000 volumes averaging 500,000 words each, or a total of 20 billion words, so that the amortization cost per word is 1/4 cent. (We may disregard interest cost for our purpose.) Forty thousand volumes is the size of a medium-sized technical library; in a large technical library probably the new accessions alone total 10,000 volumes in one or two years. Remember, also, that translation will be made from and into several languages. Thus, even if amortization is extended over a small number of years, it will cost only a small fraction of a cent per word. In other words, work on automatic translation systems is an excellent investment.

Equipment Costs

The current cost of machine translation will depend on the hourly cost of the machines used and the amount of machine time for translation.

These factors, of course, vary greatly from one machine to another. As a starting point, I propose to discuss the most advanced machines on which complete performance and cost specifications are available, machines like the Stretch computer of IBM or the LARC of Sperry-Rand. These computers come with a variety of optional equipment, and depending on this we may estimate their hourly cost at between $1,000 and $2,500. (This is intended to be first shift rental, without overhead or other charges.) A little later we shall be more specific about the kinds of optional components we may wish to include in a system designed optimally for language translation.

These computers are scheduled to perform operations like addition, subtraction, transfers, logical operations, etc. at the rate of about one microsecond per operation. Multiplications and divisions take longer, but these operations are hardly used in a machine program for language

translation.

Apart from the computing machine itself, we have to consider terminal equipment. There seems to be no problem at the output end. Conventional tape-controlled printers can print a thousand words in something less than a minute, at a cost of a fraction of one dollar. More advanced equipment will undoubtedly be even more economical.

The input operations are indeed critical for the success of machine translation. If a Russian text had to be punched into cards, say, by hand, the cost of this operation and of checked and revised punching to the required degree of accuracy would be almost comparable to the entire cost of human translation. Fortunately there seems to be reason to expect that reading machines, capable of scanning printed pages of Russian texts and recording them on some mechanical medium suitable for machine input, will be available in the next few years. Some rudimentary machines are already in existence.[3] We have no specifications, nor reliable cost figures, for an entirely satisfactory system, but it seems likely that the cost of operating this equipment will be considerably less than the cost of key punching.

Machine Time Requirements

I do not yet know how many instructions will be required for the entire translation program. It will certainly be several thousand, it may be several tens of thousands. Not all of these will be executed for every source word which is to be translated. Most of them will come into play only occasionally, while some will be executed repeatedly for every word. Let us estimate that 10,000 instructions have to be executed for each word to be translated. On next-generation computers the execution time of most instructions is about one microsecond, so roughly speaking we can translate at the rate of 100 words per second. This translation process has to be overlapped by reading from the dictionary. It has been estimated that the dictionary will have to contain 50,000 Russian stems, and that even in a highly condensed version of storing in the dictionary such as that of NBS, about 20 machine words will be required for each Russian stem. This is a total of one million machine words, and is probably also on the high side. There is no difficulty about storing one million machine words on a magnetic disc memory. To read through this dictionary in alphabetical order requires about one second, and during that second the machine can simultaneously translate about 100 words. Thus, the optimal procedure will appear to be to read the source text into the machine in batches of about 100 words, alphabetize them (or rather their roots), read through the dictionary and extract the information pertaining to these 100 words, and then proceed to translate while the next hundred words are being read in and looked up. Internal memory required is only 2,000 words for the dictionary information pertaining to 100 source words, storage space for all instructions required by the program, and some temporary working storage. A total of 32,000 words of internal memory is probably adequate. The hourly cost of a machine with 32,000 words internal memory and a magnetic disc memory is on the order of $1,000. The machine translates at the rate of 100 words per second, or at a cost of 1/3 of a cent per word. This, as we said before, compares with one cent per word by human translator. Since all our estimates are quite crude, all we can say is that the cost of machine translation by this scheme is probably not higher than by human translator.

To see how these economic factors might depend on our choice of equipment, let us suppose that we wish to use the same translation system on one of today's machines, like an IBM 704 or a Univac. Here we have no disc memory, so that the dictionary has to be stored on magnetic tape. This will take something like half an hour for reading into the machine. The internal memory of these machines is at most 32,000 words, of which we can assign at most 20,000 for copying the pertinent portions of the dictionary, reserving the remainder of the memory for instructions and working space. Twenty thousand machine words of memory will hold the dictionary information corresponding to 1,000 source words. Thus, the procedure would be to read the text in batches of 1,000 words, alphabetize them, copy from the dictionary and translate. The translation time for 1,000 words, at 10,000 instructions each, is about five minutes. Thus the computer is poorly used, most of its time is spent in reading through the dictionary over and over again, and the total time required to translate 1,000 words is over half an hour, giving a cost of about 15 cents a word. Here again, since all our estimates were quite crude, we can only say that machine translation by means of today's computers is probably more expensive than by human translator, perhaps by an order of magnitude.

There are other factors to be considered. Many instructions are used quite infrequently. For example, the Russian word for "and" requires a whole subroutine all by itself. Probably many other words require special treatment which has to be incorporated into special subroutines, and these might in the end account for a large portion of the instructions used. There is no reason why such subroutines could not be stored in the dictionary with the word to which they pertain. Thus, we can to some extent trade dictionary space for internal memory capacity. The same can be done in reverse order by selecting a fairly large number of frequently occurring words and storing all dictionary information about them in the internal memory, reserving the external dictionary for the less frequently used words.

It may turn out that our extreme insistence on conserving space in the dictionary has been unnecessary. Perhaps we can loosen up a bit, waste a little space in the dictionary and store information in less concentrated form. This may result in a dictionary several times larger, but on future machines all reading from the external dictionary will be overlapped by computing, and the computing time would actually be reduced by this more generous allotment of external storage space.

Our present crude estimates of the requirements for different translation systems do not enable us to choose definitely a best among them.

Several of them appear to yield cost figures of the same order of magnitude. Whichever will be the final choice, it does seem likely that some system will be found whereby machine translation will be considerably more economical than human translation.

References

1. Weaver, W., Translation. Machine Translation of Languages (W. N. Locke and A. D. Booth, eds.), Wiley, New York, 1953.

2. Bar-Hillel, Y., The present status of automatic translation of languages. Advances in Computers 1, Academic Press, New York, 1960.

3. See, for instance, a series of papers on this subject in Proceedings of the Eastern Joint Computer Conference 1959, in press.

COMPUTERS FOR FIELD ARTILLERY

by

Lieutenant Colonel Louis R. van de Velde, Artillery,
Chief of Research and Review Division
Department of Gunnery
U. S. Army Artillery and Missile School
Fort Sill, Oklahoma

A FORWARD OBSERVER FIRES ON A TARGET

A Field Artillery forward observer moving with an infantry company in the attack throws himself down behind a small outcropping of rock. He glances at the plastic coated map in his right hand; he sets the dials on a small device he holds in his left hand. He presses a button. In less than 3 minutes lethal volleys from two batteries of Field Artillery fall on the area occupied by the enemy recoilless rifles and automatic weapons which have been holding up our advance. The enemy weapons are silenced. The observer again turns dials on his hand message generator and presses the "SEND" button. He has reported the results of the fire.

Three miles to the rear, the operations officer in the Artillery battalion fire direction center turns to his battalion commander and says, "That completes the fifteenth fire mission we have fired today without adjustment, but with excellent effect. Our observers are reading the maps well; and of course, our computers are computing the firing data with deadly precision. The results of that mission will by now be stored on the magnetic tapes at Division Artillery Headquarters and soon will be input to the periodic intelligence program."

Computers Moving Into The Field Artillery

Yes. Computers are moving into the Field Artillery; and we are studying the implications and applications of the computers to Field Artillery. It is my purpose to tell you something of how we expect to fit computers into tactical field operational doctrine and training in the years immediately ahead. I am in the field artillery and in a strict sense I am qualified only to speak of the applications for computers in that combat arm of the Army. However, the Artillery's computer effort is a part of the total effort and I believe you would like to have a brief glance and the big picture first, before we narrow down to a single combat arm.

The Army's ADP System

The Army has undertaken an ambitious Automatic Data Processing System development program in which we foresee having the field army served by a number of computers working in a plan combining independent and integrated action. The total number of tasks foreseen for our ADP system numbers in the area of one hundred and ranges from the complex mathematics of our required ballistic solutions, computed entirely in the forward areas, to the data processing of logistical requirements dependent on a network of computers and entry devices.

I am sure that you can all form an idea of the tremendous potential savings in time and money available in a computer system encompassing an entire theater of operations, and even the continental United States, that is, the "Zone of the Interior". In past wars we have had cumbersome depot systems stretching from the home factory to the forward combat zone. From the large depots at the port of debarkation a series of successively smaller depots stretch forward. Maintaining supplies to fill this long pipeline was a formidable task. Knowing what you had available was an equally formidable task. Recall that where industry is faced with this same problem, industry is not faced with constant movement of depots, movement of the forward end of the pipeline, rotation of personnel, and accident as grave as bombing and shelling; nor is industry faced with a communication system that is in a state of great flux. In sum, then our depot system and supply problem in a theater of operations has always been inevitably costly and inefficient. Enter data processing. With rapid accounting, with speedy and accurate reply to queries from anywhere in the system, and perhaps best of all, with powerful and valid extrapolation and prediction of requirements we can have good management and timely execution; we can probably cause whole lines of depots to fall. We can save the huge quantities of stocks consumed in just filling the pipeline.

We have great hopes for the data processing possibilities ahead for us in a field army - in material accounting and in personnel and operational accounting of all kinds.

Now, having sketched out for you the larger picture, the field army and the whole scheme of computers we plan for it from the large scale to small scale, let me return to my own field and sketch out what we see in - "COMPUTERS FOR ARTILLERY".

Computers in the Field Artillery

The Field Artillery supports the other two ground gaining combat arms - Infantry and Armor - and is in turn supported by the remaining arms and services. Therefore, by logical extension of my remarks you can readily derive from an understanding of the artillery's computer problems an understanding of the place of computers in the full

field army.

Among the combat arms - Infantry, Armor, and Artillery - the Artillery holds a unique place with respect to automatic digital computing. The Artillery stands at that point in the field army at which technical, mathematical computing meets automatic data processing - the instrument of logistics, battlefield management, and command. (Figure 1).

Requirements for Mathematical Computing

First, we will look at the Artillery's requirements for mathematical computing. Second, we will glance briefly at the type of computers we have in mind and a possible scheme of location; third, we will look at the Artillery's requirements in data processing; and finally, we shall look at a portion of the complete system within the Field Artillery as it might look a few years hence.

Our requirement for technical, mathematical computing falls into three fields: survey (where are we on the ground?); meteorology (how strong is that head wind?); and firing data (at what angle shall we point these cannons and rockets?). (Figure 2).

Finding the answer to each of these questions presents a straightforward mathematical problem. Survey data and meteorological data are normally furnished to the artillery battalion by higher headquarters, and so will be discussed when we come to a discussion of the complete artillery system of computers. However, fire control data, based on survey and meteorological inputs, is computed at the artillery battalion and battery - down where the howitzers, guns, and rockets can be found. We will begin at the forward end and discuss fire control.

At this point I think I must take a minute to define briefly the organizational structure of the artillery so that we may use three simple terms - battery, battalion, and division artillery for the rest of the paper. The basic firing unit in the Field Artillery is the battery, which we can say includes 6 howitzers (guns) and is represented by the symbol shown in Figure 3.

The next larget unit is the Field Artillery battalion which includes two or three batteries and is shown in figure 4.

And finally, the third and last unit we need to consider is the division artillery. A division artillery will control a number of battalions, say five or six. Figure 5 shows the organization of a division artillery.

Fire Control Computer

What computers are we planning for our mathematical requirements and where will we place them? For use within batteries and battalions and primarily in the production of firing data we are now testing a small ruggedized computer, FADAC, Field Artillery Data Automatic Computer. FADAC has a rotating memory in the 4096 word class. The computer is designed to give us a complete ballistic solution (integration of half a score of differential terms representing the interacting factors of angle of departure, velocity, mass, weather, and so on) in less than the time of flight of the projectile. We expect great accuracy from the solution because the program combines the best of proving ground methods (full mathematical solution to the equations of motion) and the best of practical experience. The computer will store, and constantly update, the empirically derived difference between predicted and proven results. The end result, we hope, will be first-round accuracy for our cannons and improved accuracy for rockets. It is also adaptable to other problems and can be termed a general purpose computer. For all of our artillery computers we intend to have removable and changeable labels on our operating panels so that the soldier operator can talk exclusively in terms of his own problem. Reading in a new program is accompanied by attaching a new set of labels for the operator's buttons. We thus have the advantages of being special purpose at the moment and general purpose through the day.

Response To Fire Requests

You must bear in mind that in ground combat we normally fire cannon as area weapons. That is, we mass the fire of a dozen or more cannon on small areas (100 meter, 200 meter circles). By first-round accuracy we then mean that the impact pattern of the first volley fired by the battery or battalion will include the target and produce lethal effect against it. (Figure 6 and 7).

Let us accept that immediate, accurate response to a requirement to fire is the Field Artillery's primary interest in computers. How does this affect the development of a computer system? We can return to the forward observer we met at the beginning of this talk for an answer to the question.

First let us look at the sequence in the complete process of firing and then consider the actors who are responsible for action at each step. In Figure 8 are the elements of the problem. Here is an enemy target; the target is seen by an observer; the observer requests the battalion to take an action; the battalion provides an officer who makes a decision; the decision assigns the firing to one or more batteries; data is computed; and the batteries fire.

All of these actions can be speeded by their inclusion in a complete computer system. A complete system can help us avoid error. However, only the central computer is indispensible to the achievement of first-round accuracy; we must have a computer for the mathematics of the ballistic solution.

An Automatic Fire Request/Response System

Let us look again at the process, with automatic devices in operation throughout. Figure 9 shows the full span. First is the observer's request for fire. The observer uses an automatic device to report the location of the target. (Marked "Message Entry" on Figure 10). He could use the telephone or radio with only a small price in speed (and, of course, a risk of human error in number transcription - coordinate location of the target, and so on). Second is the decision and order to fire. We receive the mission and display it electronically on a device at the battalion which I shall call an electronic map and switch device (Figure 11).

The configuration of this device is under study. Although the display is automatic, human decision (to fire or not to fire, and how much) remains with the gunnery officer. Transmission of the gunnery officer's commands to the firing battery is automatic. Removing the automatic aspects and returning to inspection of a paper map (followed by telephone or radio to the battery to fire) would cost in time but not accuracy (discounting transcription error). And finally, there is at the battery a display unit on which can be displayed a complete set of fire commands and data for the guns. The complete set includes gunnery officer instructions to the battery, and the output of the computer located at the battery or battalion, or both. With only a small loss in speed (and again, possible human error) instructions and data can go to the guns by voice (telephone or radio). We retain accuracy as long as the data is generated by computer.

Recapitulating, we can say that without loss in first-round accuracy we can delete from the system the observer's message entry device, the electronic map, and switch device, and the battery display unit. We believe in the possibilities of a complete system; but our greatest tactical gain - accuracy - can come from a decentralized array of ballistic data computers not linked by data transmission systems (other than the old fashioned kind developed by Alexander Graham Bell).

We wish to play the game both ways. We want the benefits of a complete system if we can have them; but knowing the uncertainties of the battlefield we wish small computers well forward - in batteries and battalions - to compute firing data. This concept of being able to work our computers either in a system or independently we call the federal-state approach.

Implications Of Increased Accuracy

There are a number of significant implications in a more accurate solution to the cannon and rocket fire control problem. The first is that the Artillery will be able to fire many more "immediate fire for effect" missions, missions in which no adjustment by the observer is required even though no recent registration ("zeroing in") has been accomplished. The obvious gain in effect on the target is perhaps matched by the time and ammunition saved. Observers can fire many more missions in a given period of time.

A reduction in the number of registrations (a procedure for determining corrections of the moment) will bring a number of benefits. Logistically we will save in the ammunition moved forward. How many thousands of rounds were consumed in Korea and World War II in registering batteries and re-registering them repeatedly? We know the cost of one round of ammunition at the factory in Virginia or Ohio. But what is the cost of that round delivered to the 38th parallel in Korea? I do not mean to imply that registration will not be necessary at times; but the requirement should be materially reduced since the computer gives us a pure ballistic solution in which errors are removed or isolated to a degree not now possible using the approximations presented in firing tables.

Fewer registrations will mean more surprise. Units can move into new positions and be ready to render effective support without that tell-tale pre-firing registration procedure. Tactical mobility will be increased. In the past, impending darkness and other restrictions on registration in a new area have often influenced the displacement of Artillery units. Time will be saved; registrations can be time consuming. Under present techniques missions sometimes have to wait or registrations have to be interrupted.

Unobserved Fires

In addition to the response to requests for fires called in by observers, the Field Artillery frequently has to place fires on areas not under the observation of a trained artilleryman. The fires may be on call from supported units or may be part of a predetermined schedule of fires. In any event the utmost in accuracy is required. In the past this accuracy has not been easy to achieve. The normal procedure is to record basic data for all targets and update the data every two hours as new meteorological information is received. The bookkeeping can become extensive. The pressure of events can make it impossible for fire direction personnel to find time to update data for their on-call fires; and even when manual procedures are perfectly executed the results are limited to firing table accuracy.

With a fire control computer we will be able to store the location of many targets. Basic data for fire on the targets can be generated. Correction factors determined from firing can be applied to new conditions. Receipt of data on newly determined meteorological conditions will be followed by an immediate updating of fire commands for all on call fires. The battalion will be able to repeat fires on old targets or place them on new targets nearly as rapidly as the guns can be laid (aimed).

The handling of relatively large target lists is a problem in data processing and fire planning. I should like to defer discussion of these

topics until later and complete the treatment of technical, mathematical computing requirements.

Missile Computers

Missile fire control also presents an obvious application for high speed computers. One missile fire control computer - the Jukebox for Redstone - has been delivered and is in the hands of operational units. The Jukebox computes, in five minutes, the complex mathematical problem requiring two trained operators working with electric desk calculators two and one half to three hours to complete by hand. In the case of missile computers the gain is in speed. The hand solution produces accurate results for the Redstone but is intolerably slow. Every missile type will have to be assigned an appropriate fire control computer. Because all computers are potentially general purpose machines, the same computer that computes the fire control problem for one missile can be reprogrammed to compute the problem for another. There are technical limitations in some cases, of course. But there will inevitably be a trend toward standardization and limitation of types of digital computers for the Artillery.

Topographic Survey Computing

Topographic surveys in the battle zone are accomplished by the combined effort of survey teams at successive organization levels. The battalion has a survey function, the division Artillery has a broader function, and so on. Accurate and rapid exchange of information between the survey echelons is imperative. Therefore the Artillery survey requirement presents a problem in both straightforward mathematical computing and in data processing.

The end product of the Artillery survey system is the precise location in three dimensions of points in the areas of the Artillery battalions, and the target area, plus the establishment of a reference direction. To reach this end the survey system must build a network of computations and establish a set of centers for exchange of data. There is currently under development a computer-supported system for this task. Regardless of whether the complete integrated task can be performed automatically in the near future, isolated computers at the appropriate various levels, supported by conventional communications, the "federal-state approach", will speed the production of data required for accurate artillery fire.

We are studying message entry schemes and believe that the battalion survey requirements might be met by a message entry device having access to the battalion computer in the battalion operation section, the fire direction center; or it might be better to allocate a computer to survey at battalion under some conditions. In any event there could be associated with the battalion computer in the fire direction center, a survey panel for the processing of survey problems when the scope of these exceeds the capability of the fire control panel. At division artillery level it is possible that the survey requirement, with particular reference to the work of advance parties during displacements, will be such that a small computer individually vehicularly mounted must be assigned to survey. This computer would itself be served by both message entry devices in the hands of survey parties and by the survey control panel on the computer. The small survey computer would, of course, have access to the medium size computer at division artillery headquarters.

Meteorological Data For Fire Control

The preparation of meteorological data for the artillery has been developed and refined over the years. Yet, the timely preparation of the required data by hand necessitates the use of graphic methods and approximations that cannot possibly give the precise results to be expected from a high speed machine. The application of computers to generation of meteorological messages for artillery fire control is an immediate prospect. The dissemination of the results automatically through the system is the obvious corollary.

Meteorological data is generated at division artillery and higher headquarters. The "met" sections, almost alone among our Artillery elements, have an on line computing problem with a real-time consideration. (There are radar tracking problems with a real-time aspect). We track a rising balloon which is constantly transmitting to the ground station. We can solve this problem either by the assignment of a small computer to the metro section (which may wish to locate at a point well away from division artillery headquarters), or we can use real-time registers and interrupt features on a medium scale computer at division artillery, or we can digitize the data from the analog input at the met station and transmit it to the medium scale computer at division artillery.

The computer requirement for mathematical computing in a division artillery then looks something like what is shown in Figure 12.

Data Processing Requirements

What are some of the Field Artillery applications in the data processing field? Many of the applications are straightforward, business accounting type problems, challenging because of their magnitude and because of the requirement to bring together the input from many sources. Other applications are less straightforward - or at least less obvious in their outline - and are therefore even more challenging to the artilleryman concerned with problem formulation and analysis. In the second group, there is particularly fire planning.

Fire Planning by Computer

The Artillery and Missile School has a program which can be placed on the commercial model computer at Fort Sill (a Bendix G-15D) to produce a schedule of non-nuclear fires at division

artillery level. For example, one problem we run plans the fires of 21 batteries against 97 targets in execution of a schedule of firing that occupies the batteries for one hour.

Schedule of Fires

This is a first bite into a larger problem. Fire planning includes much more than producing a fixed schedule of fires. However, the Artillery School's program works; it demonstrates that fire planning in terms of a fixed schedule of fires can be accomplished in a few minutes on a small tactical type computer. The implications are enormous. If the laborious process of fire planning can be reduced to an operation requiring on the order of ten minutes or less, the reaction time of large units can be drastically reduced. This may materially change the tactics of divisions and larger units.

When visitors to Fort Sill are first shown the fire planning program they frequently display certain incredulity. The objection is usually made that there is either some hocus-pocus about the operation or the results are poor indeed; for no machine can have the battle experience (and judgment derived therefrom) that we can expect from commanders and staff officers.

The objection is valid - to a point. However, it breaks down for two reasons. First, many of the operations in fire planning that we view as the exercise of judgment involve not judgment but routine response to a routine situation. And second, the machine can be made to accept human judgment in a number of ways: in establishing initial criteria; in modifying criteria in response to human direction when intermediate answers are presented to the operator for inspection; or through processes that, with a slight stretching of terms, can be called self education of the computer.

Judgment In Fire Planning

Consider the first point - that we often call the exercise of judgment what is in fact merely a standardized response to a standardized situation. For example, a fire plan to support a division Artillery non-nuclear schedule of fires is being prepared. The staff officer scheduling the fires of the available Artillery battalions has run down his target list and has come to a target labelled "machine gun". Map coordinates are listed, recent intelligence indicates the weapon is most probably where shown; the weapon is on high ground and in the general line of advance of the attacking force. The staff officer produces response A; that is, he schedules a certain caliber and volume of fire against the enemy weapon.

A moment later the staff officer comes to another item on the target list that meets the same general description. The planner again has response A. There may be a difference between the two enemy weapon positions; but by the time fire is planned this difference has left the intelligence picture. The difference may have been too subtle to permit reporting in the acceptably brief report procedures. It probably follows that the difference between the two weapons was also immaterial from a tactical point of view. The point is that the staff officer fire-planner has a standard reaction to a standard piece of intelligence. Much of fire scheduling is along these lines. Many officers drawing up schedules of fire have had only a general view of the terrain.

Admittedly, there is quite a difference between a machine gun in the line of advance (or with a field of fire covering the line of advance) and a gun answering precisely the same description but located at a point too distant to permit it to fire on the attack element. But this is geometry and can readily be resolved by the machine.

The objection may be that the computer's reaction is too rigid, no room is permitted for variations in plan, situation, or the many imponderables of the battlefield. The answer is that more refinements will be built into our program as thought is given to them; and, perhaps more important, the program can permit modifications of criteria by the commander at any time. A particular enemy is particularly resistant to previously accepted standard quantities and types of fire. Very well, the commander directs that the standard reaction to a particular type of target be increased, doubled, or changed in degree in any way the battle experience and judgment of the commander or his staff suggest.

Determining Who Should Hit

The fire scheduling problem is composed of two phases. First, the machine must determine which units can hit which targets; then the machine must determine which units should hit which targets. Determining who CAN hit is straightforward mathematics. Locations of gun and target are compared, azimuths of fire are considered and a tabulation of the results stored. Considering one target at a time the machine examines each firing unit in the appropriate category that can hit that target and determines if the desired effect can be achieved by assigning the unit to fire on the target. Successively, acceptable units are assigned until the desired level of damage can be predicted. This is the who SHOULD hit phase. It presents the more subtle and interesting problem.

Inputs: "Target Word"; "Fire Unit Word"

To begin the problem we present the computer two sets of organized inputs - data with respect to the targets, known as the "target word"; and data with respect to the fire units, known as the "fire unit word." The commander's criteria are set in as constants and can be changed with each plan.

Target Word

The target word consists of ten pieces of data, as shown in Figure 13.

The concentration number is a tag that will permit identification of the target on the final fire plan. The tag will also permit deleting the target from the target list at any time prior to computing the fire plan. The coordinates are standard military map grid coordinates (a cartesian system) and form the basis for the determination of CAN hit lists by the computer.

The next seven items in the target word are what permit the blend of routine reaction and judgment (or commander's preference) in the machine developed plan. The seven items are all determining factors in the who SHOULD hit consideration. The final item, the date time group is necessary for intelligence processing.

Target Types

If there is to be a standardized reaction to targets there must be standardized targets. The working group at the Artillery and Missile School have developed a standard list of targets composed of three classes and, within those classes, a total of ten types. We consider that all nonnuclear targets presented to a ground force for planned fires can be categorized as falling within one of the types of targets shown in Figure 14.

Total Target Description

In addition to affixing a type number to each target it is assumed that the observer or other acquisition source will give some indication of the size of the target. An index of the magnitude can be conveyed under two convenient designators: scale, a measure of physical dimensions; and density, a measure of the number of personnel within the target area. Necessarily, both designators must be simple and stated in general terms. It has been tentatively decided to use two scales, and two densities. Scale 1 is a target, varying from a virtual point to approximately 120 meters in diameter; and scale 2 is a target of 121 to 240 meters in width, the depth remaining at 120 meters. The depths and widths are associated with the normal dispersion pattern from average battery gun positions for the calibers under consideration. Large target areas are attacked with conventional artillery only after they have been broken down into smaller targets of the scales mentioned.

With ten target types, 2 scales and 2 densities (density 1 is considered normal, density 2 a thickly packed target such as an assembly area) there is ample opportunity for adequate target description for immediate response to requests, and for intelligence file purposes. Ten target descriptions - more than ample.

Reliability is a reflection of the assurance that the target is in fact as reported. Targets are "confirmed" or "suspect".

Priority, Phase and Repeat

Priority has been established as 1, 2, or 3 for the machine. Priority 1 targets are those that must be hit or the fire plan is considered unacceptable and the machine halts, awaiting new instructions. Before halting, the machine attempts to shift the center line on which the battalions and batteries are laid to see if a slight change will increase the unit's capabilities. Priority 3 targets are those that can be most readily discarded. This leaves, by forced definition, priority 2 targets as those that should be hit if possible, and should not be discarded until after priority 3 targets are discarded. Priority can be determined by the computer from the other parameters. There is provision for override so that the local situation and the commander's desires may govern.

The remaining two items in the target word determine whether the target is fired on early, late, or in the middle of the schedule of fires; and whether or not fire is repeated on the target. Each target is tagged with the desired action as it is entered into the machine. If the commander wishes to make it a matter of standard procedure that certain types of targets are always fired on in a certain phase of the fire plan, this can be built into the program. The time required to input target data will be reduced accordingly. The same applies to the repeat requirement. Fire can be repeated once, twice, or, in long plans, three times.

Fire Unit Word

It is also necessary to furnish the computer data on the fire units. The "fire unit word" consists of six parts. I shall not discuss them in detail as the purpose of each piece of data is fairly obvious. The fire unit word is shown in Figure 15.

Supporting the Fire Plan Formulator

The Fire Plan Formulator, the name given our program, modified and revised as it will be at the Artillery and Missile School, will produce a fire plan in less than a quarter of an hour - perhaps in as little as 5 minutes on a small high speed tactical machine. The Fire Plan Formulator needs to be supported by devices and procedures that work together as a system to provide an updated target list from which fire plans can be made on call. The Formulator also needs the support of devices and procedures that will take a finished fire plan and disseminate it to the many points on the battlefield where it will be used.

Such an integrated scheme of devices and procedures is in prospect for the Artillery. It is part of the Army's Automatic Data Processing System. The Army's effort to develop an automa-

tic data system for the field army (ADPS) is being especially directed toward the Field Artillery subsystem which will be first subsystem developed.

The Artillery Subsystem for ADPS

The U. S. Army Electronic Proving Ground, Fort Huachuca, Arizona is responsible for executing the analysis and programming of the full ADP system. The Field Artillery is actively supporting development of the Artillery subsystem.

In Figure 16 I have shown part of the Artillery subsystem up through division artillery as it may appear when allocations of computers are ultimately determined. Recall that a division is a body of men in the 10,000 to 15,000 class. The Division Artillery consists of six battalions and a Division Artillery Headquarters. A possible computer system for one battalion and the division Artillery is shown in Figure 16.

Note again our familiar forward observer, the battalion fire direction center, and the firing batteries in which are located the guns; notice the survey computer; and the division Artillery computer.

To perform the fire data computation (ballistic solution) which was discussed in the beginning of the paper we have the computer in the battalion - here. We also plan computers for fire control at the batteries. But now we have discussed fire planning and we have implied other applications, ammunition accounting, target list updating, survey, and others. The central processor (computer) at battalion must be fortified.

We will probably require memory increments or augmentation in some form to raise the capacity of the memory to 8 or possibly 12,000 words. In the forward areas such as battalion we do not believe that we can ever work with the fragility inherent in magnetic tape drives as we now know them. Memory augmentation might be through the use of special ruggedized canisters carrying endless loop magnetic tapes; or possibly the time requirement for program change will be sufficiently generous to permit the use of endless loop paper tape canisters.

We have under development for the ADP system, small core memory computers (known as COMPAC) which have a capability for the addition of memory modules in units of 4096 words at a time. This may be an unnecessarily costly means of memory augmentation for us. Just what computers and peripheral equipment will be used in the small units cannot be stated with any certainty at this time.

There is one interesting feature that we have introduced here and that is the multiple console concept. Around a single central processor (core memory, parallel computer) we have three consoles, or operating panels. The panels are labelled with the precise problem parameters of "fire control", "operations", and "survey" so that our soldier computer operators can be skilled artillery men first, and computer operators second. We cannot afford long periods of training for computer operators. A real time interrupt feature on COMPAC permits apparent multiple entry to the computer.

At the division artillery we have a medium scale computer known as the BASICPAC. This computer has a core memory of 4096 words which is readily expandable by the addition of memory modules of 4096 each. In addition to the 4 to 8 standard magnetic tapes available to it, the machine will handle half a dozen input/output devices. The computer is further fortified and brought to the medium class by the numerous index and real time registers.

A Complete System

Here we begin to have the heart of a true, relatively full computer system. The division artillery computer can query and respond to querries of the six battalion computers and to the survey computer working directly out of the Division Artillery Headquarters. The division artillery computer receives data for computation from the met station, or it receives for retransmission a met message computed at the met station. Our system now gives us survey, and meteorology - the full set of mathematical requirements less fire control which is accomplished at lower levels. The BASICPAC also gives us a capability for fire planning, target list processing, intelligence functions, ammunition accounting, march planning, and numerous administrative and logistical functions.

We now begin to look like part of a system. BASICPAC communicates with the Division Headquarters, with each of its own six battalions, and with the next higher Artillery Headquarters (known as Corps Artillery). Unquestionably, the system will give us great power.

As you can see, we in the Artillery have our work cut out for us in keeping one jump ahead of the hardware, in analyzing our problems, and in planning the integration of the data handling of each problem into our system. We understand and appreciate the power of this great new weapon. It is up to us to do some hard thinking and produce the best possible methods for using it. We intend to try.

Among the combat arms

—Infantry, Armor, and Artillery—

the Artillery holds a unique place with respect to automatic digital computing. The Artillery stands at that point in the field army at which technical mathematical computing meets automatic data processing— the instrument of logistics, battlefield management, and command.

Figure 1

NEED MATHEMATICAL COMPUTING FOR——

SURVEY (Where are we?)

METEOROLOGY (How strong is the wind?)

FIRING DATA (At what angle shall we point our cannon and rockets?)

Figure 2

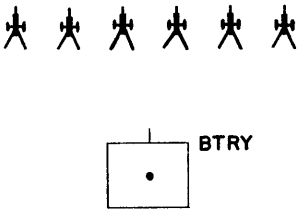


Figure 3

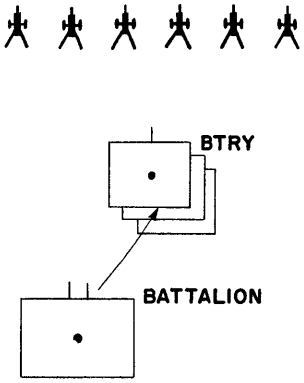


Figure 4

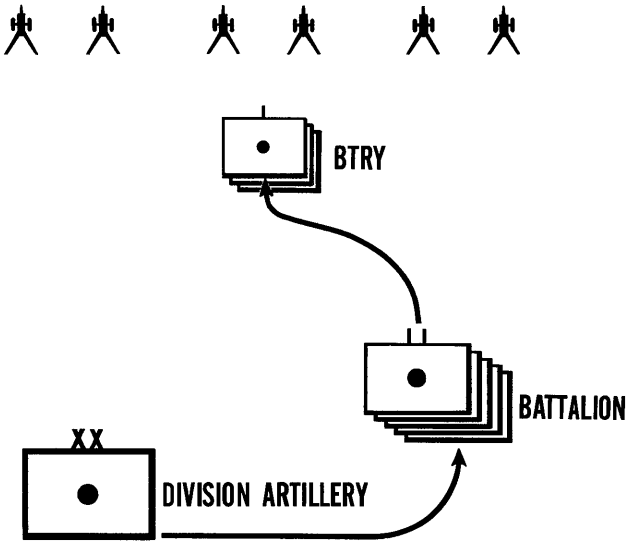
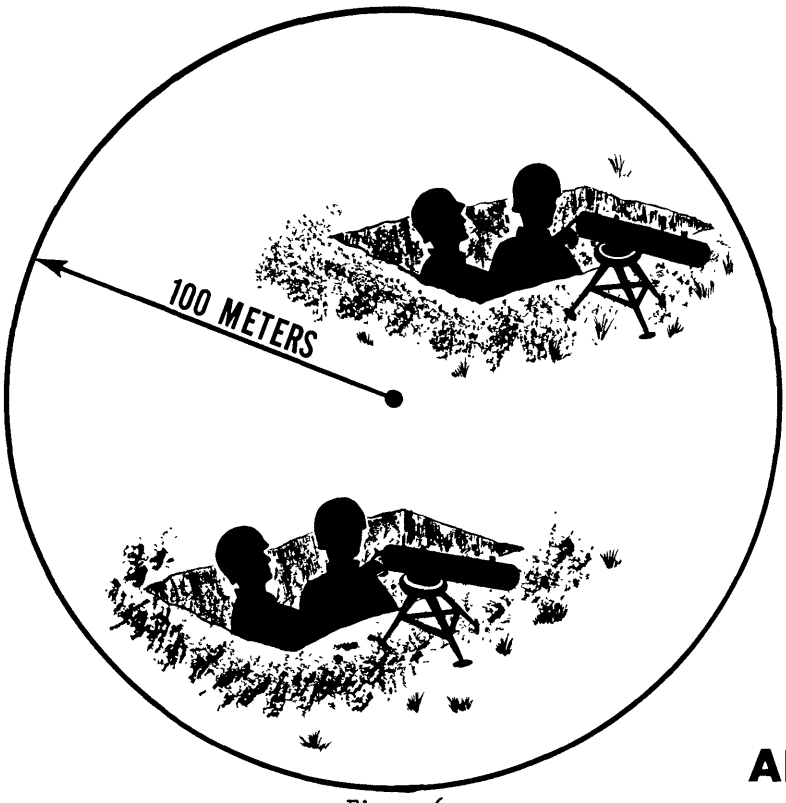


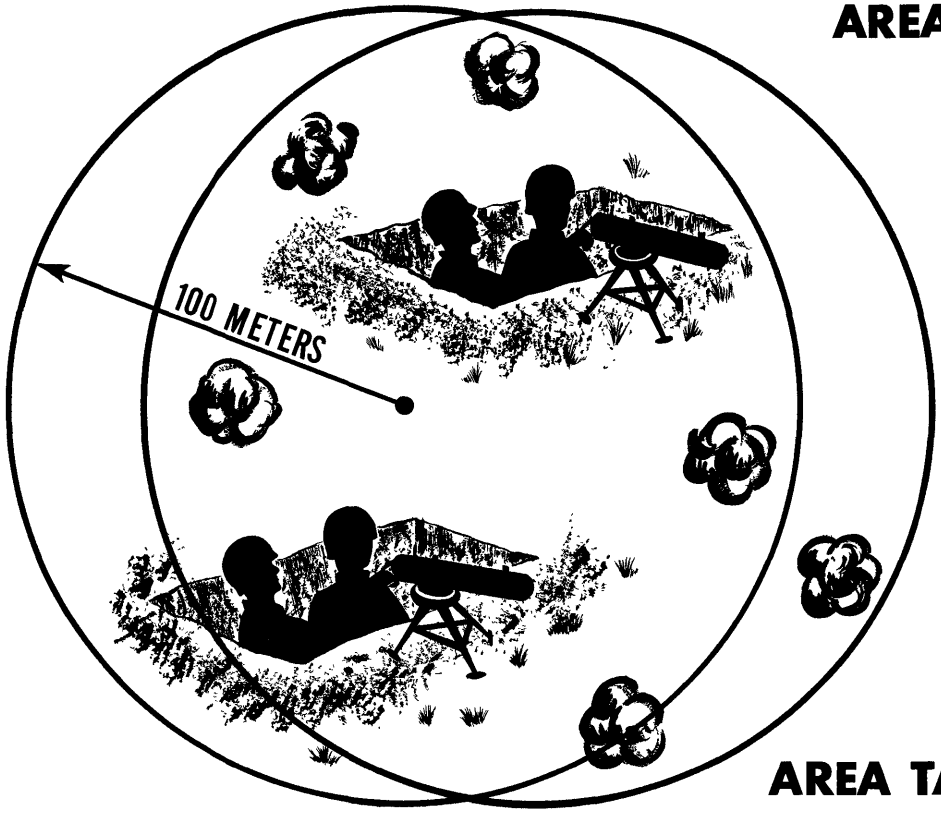
Figure 5



AREA TARGET

Figure 6

AREA FIRE



AREA TARGET

Figure 7

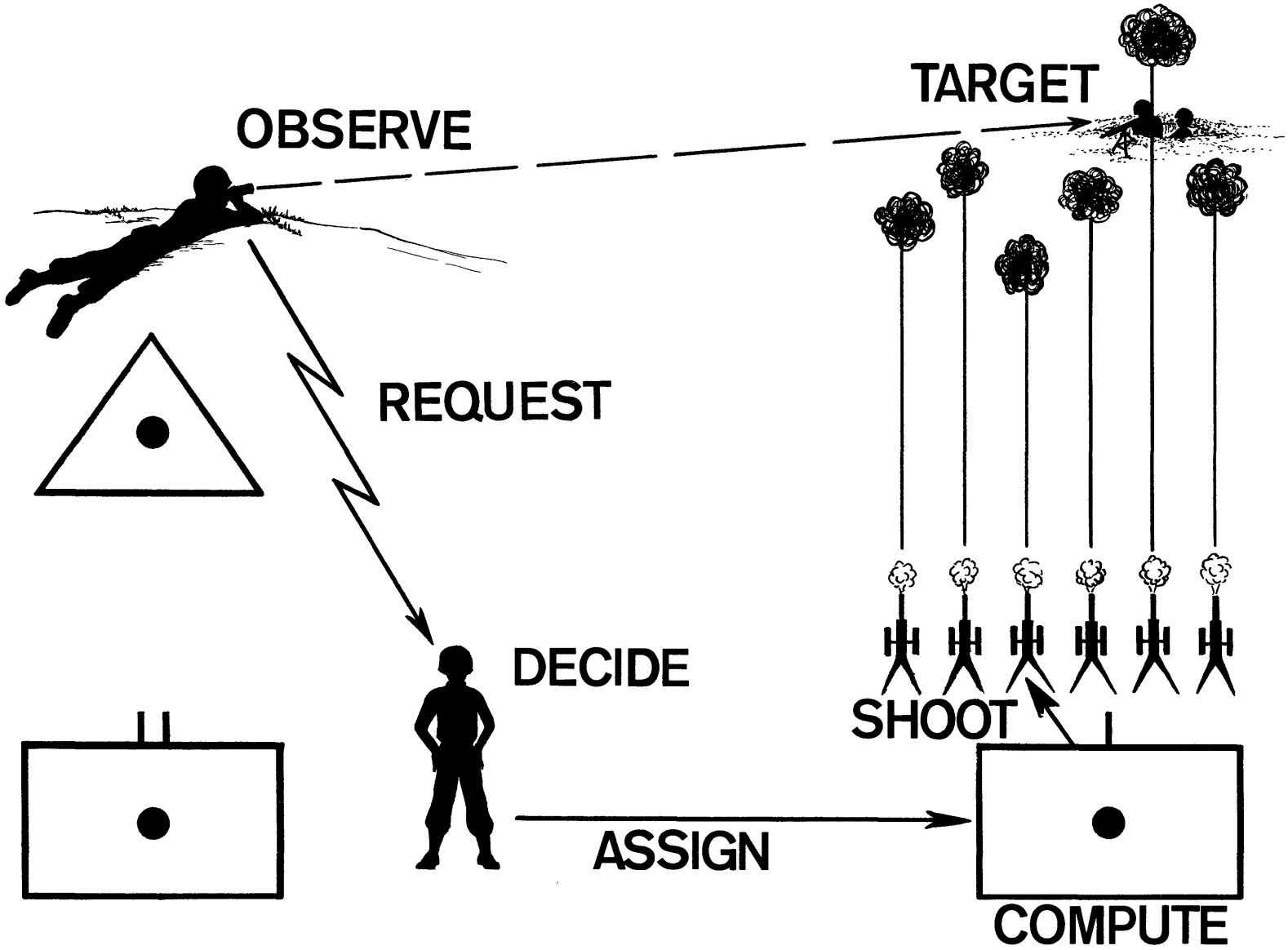


Figure 8

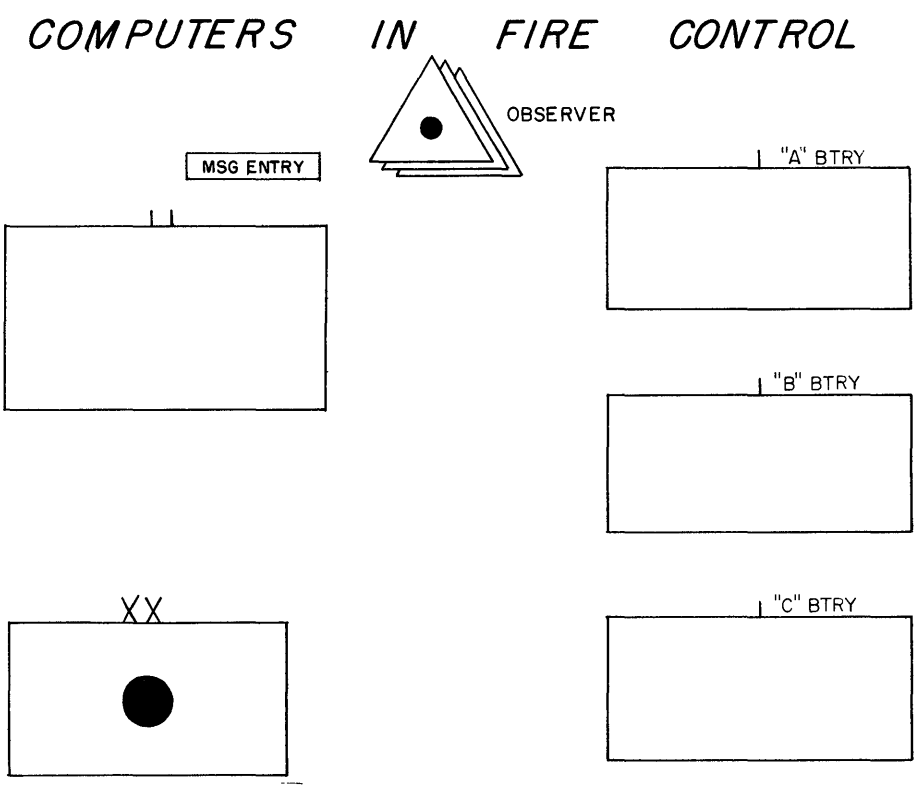


Figure 9

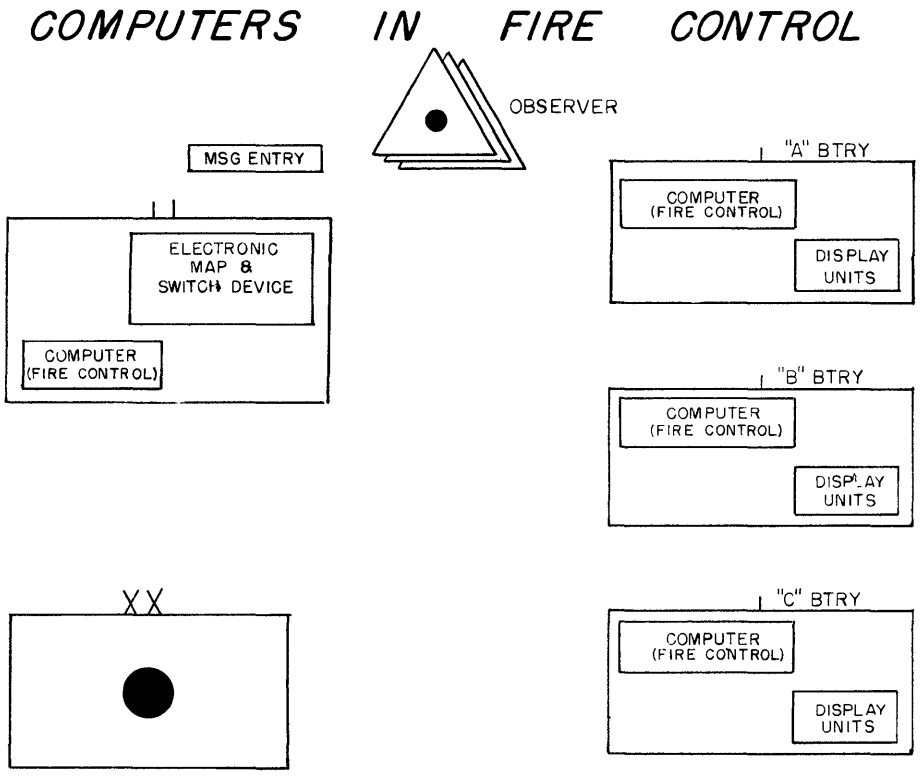


Figure 10

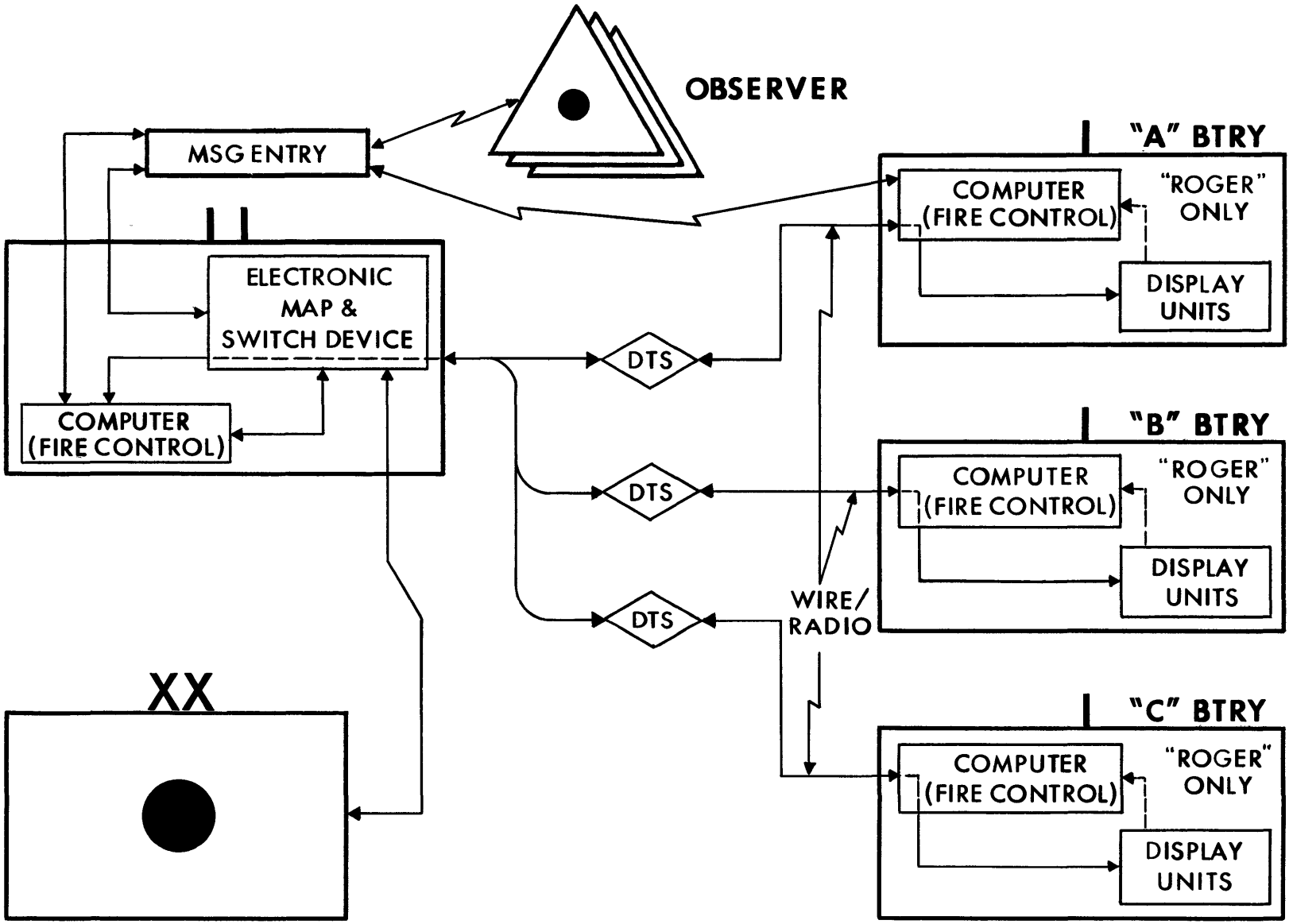

COMPUTERS IN FIRE CONTROL
OBSERVER
MSG ENTRY
ELECTRONIC
MAP &
SWITCH DEVICE
COMPUTER
(FIRE CONTROL)
DTS
DTS
DTS
WIRE/
RADIO
"A" BTRY
COMPUTER
(FIRE CONTROL)
"ROGER"
ONLY
DISPLAY
UNITS
"B" BTRY
COMPUTER
(FIRE CONTROL)
"ROGER"
ONLY
DISPLAY
UNITS
"C" BTRY
COMPUTER
(FIRE CONTROL)
"ROGER"
ONLY
DISPLAY
UNITS
XX

Figure 11

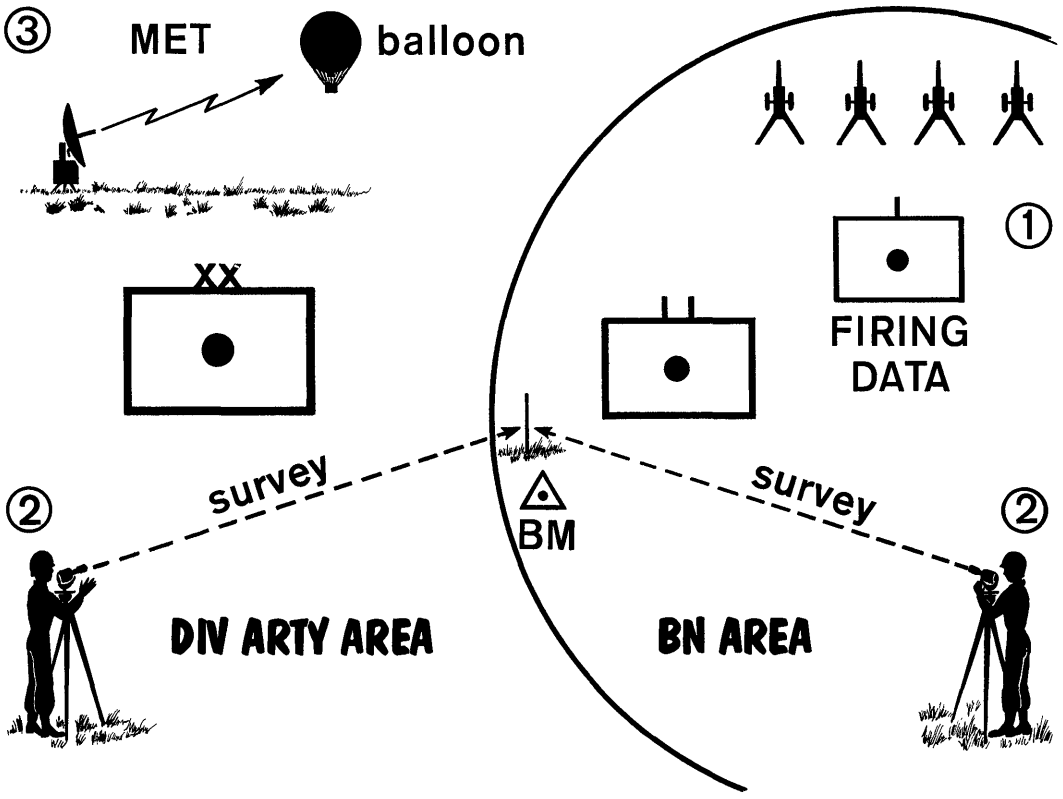


Figure 12

TARGET WORD

1. CONCENTRATION NUMBER
2. COORDINATES
3. TYPE
4. SCALE
5. DENSITY
6. RELIABILITY
7. PRIORITY
8. PHASE
9. REPEAT
10. DATE/TIME

Figure 13

TARGETS

CLASS	TYPE
I - PERSONNEL IN OPEN	0. INFANTRY IN OPEN 1. OP IN OPEN
II - PERSONNEL IN SHELTER	2. INFANTRY DUG IN 3. COMMAND POSTS 4. INFANTRY WEAPONS 5. OP DUG IN
III - PERSONNEL/MATERIEL	6. MORTARS 7. ARTILLERY 8. ARMOR 9. SUPPLY DUMPS

Figure 14

FIRE UNIT WORD

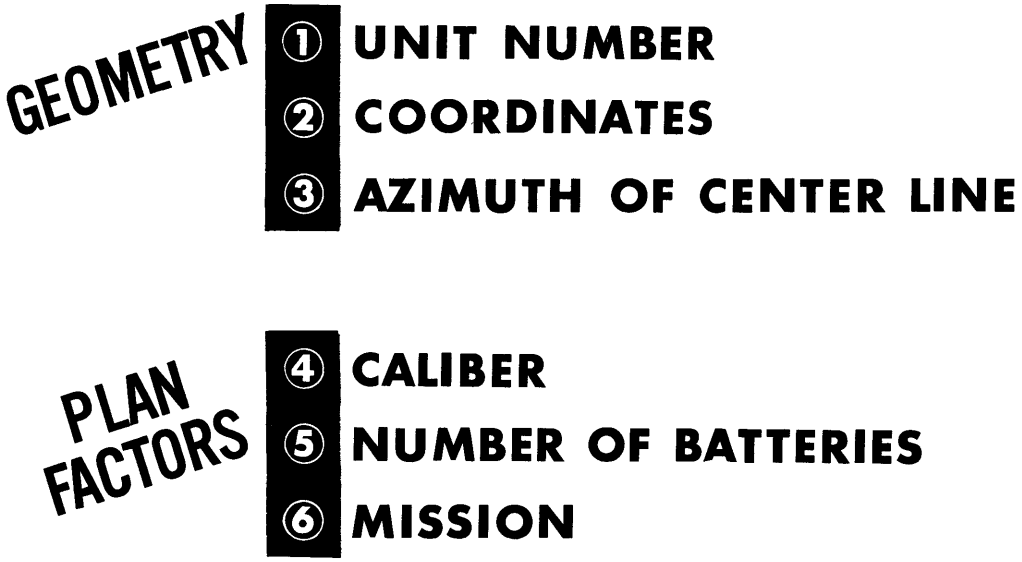


Figure 15

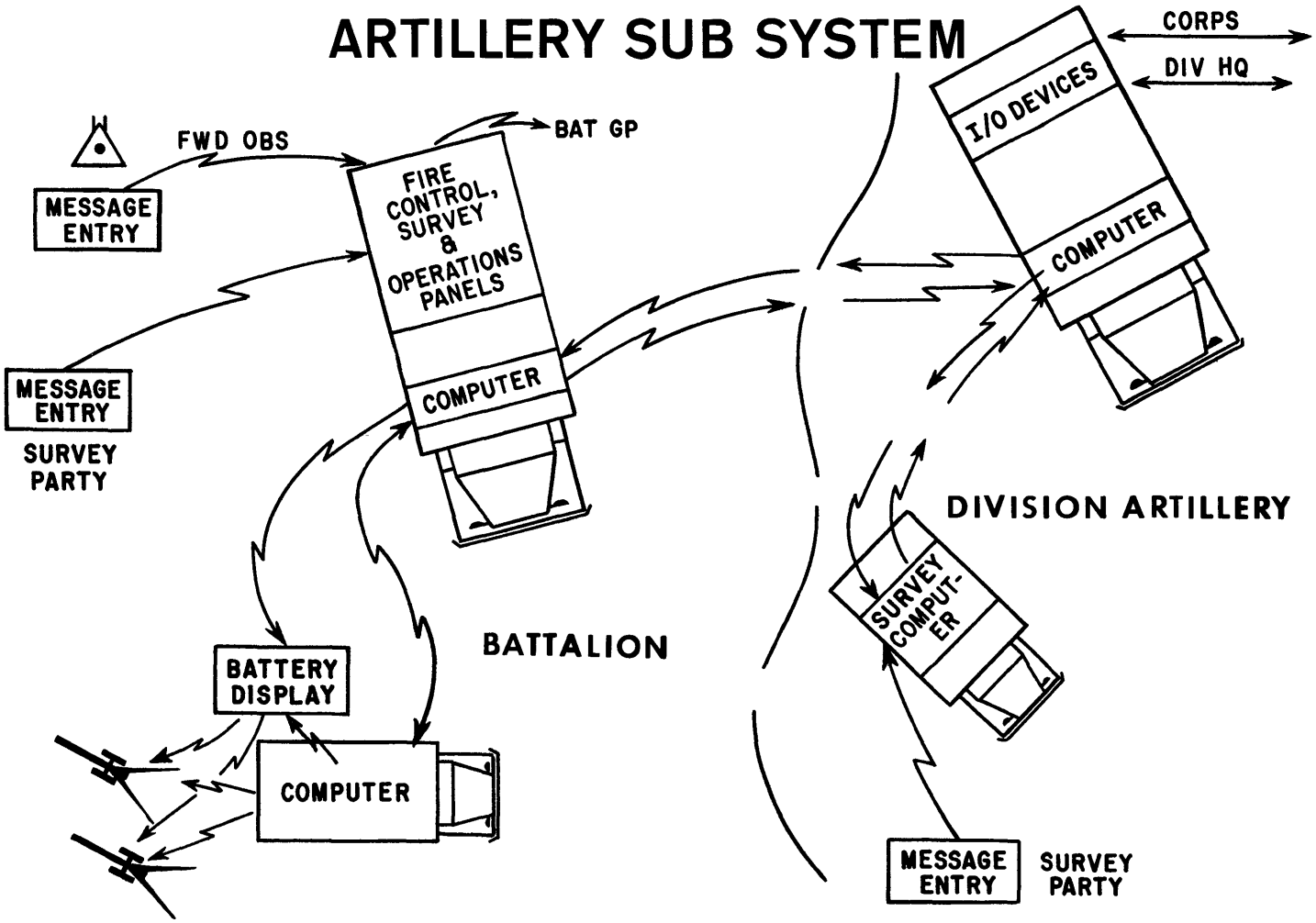

ARTILLERY SUB SYSTEM
CORPS
DIV HQ
I/O DEVICES
COMPUTER
FWD OBS
BAT GP
MESSAGE ENTRY
FIRE CONTROL, SURVEY & OPERATIONS PANELS
COMPUTER
MESSAGE ENTRY
SURVEY PARTY
DIVISION ARTILLERY
SURVEY COMPUT-ER
BATTALION
BATTERY DISPLAY
COMPUTER
MESSAGE ENTRY
SURVEY PARTY

Figure 16

COMMUNICATIONS WITHIN A POLYMORPHIC INTELLECTRONIC SYSTEM

George P. West, Ralph J. Koerner
Intellectronics Laboratories, Ramo-Wooldridge
Canoga Park, California

Summary

A multiple-computer system, employing several computers of intermediate capacity, provides many system advantages over conventional large-scale computing systems. A multi-computer system requires effective communication of data and control signals between modules of the system. This paper briefly describes the RW-400 Data Processing Central, a new polymorphic data processing system. The communications problems associated with such a system are discussed, and a high-speed electronic switch employing multiple-aperture magnetic cores to provide system communication is described. The utilization of this high-speed switch in providing efficient data communication, as well as effective control of the system modules, is explained.

With increased automation of large-scale data processing applications, the trend has been to larger memories, higher clock rates, and increasingly complex arithmetic units. This approach is not entirely satisfactory, particularly from cost and reliability standpoints. A multiple-computer approach employing several computers of intermediate capacity provides many system advantages. A novel organization of more or less conventional computational elements, communicating in a common format by means of a high-speed electronic switching network, provides a data processing system which can assume many different forms. Such a polymorphic system can be more nearly tailored to the requirements of a particular application, since storage capacity and arithmetic capability can be adjusted independently, in relatively small increments to match the requirements at hand. Furthermore, changing operational requirements can be matched by expanding the system economically, without obsolescence of existing equipment or programs, merely by adding additional computers or memory devices. Finally, the system can be programmed to adjust its own performance to meet rapidly changing operational requirements by shifting computational capacity from one assignment to another.

The RW-400 system is a multiple-computer system of this type, capable of simultaneously processing multiple tasks with extremely short time requirements for the completion of each task. The Computer Modules of the RW-400 system employ a powerful two-address order code providing three optional forms of each arithmetic instruction. These options may be utilized to specify the contents of the accumulator as one of the operands, providing in effect a two-plus address structure. A magnetic core memory of 1024-word capacity, and 10-microsecond memory cycle, is provided with each Computer. Control and synchronization of computations is exercised by a generalized program interrupt feature. Each Computer is capable of recognizing one or more external interrupt conditions to force a branch in the computer program. Each Computer is provided with a masking register which can be loaded by the computer program to specify which of the 13 possible external interrupt conditions, as well as which of the internal interrupt conditions, will be effective at any instant of time.

The system employs novel magnetic core buffer memories (Buffer Modules) to augment the memory provided with each Computer Module. The Buffer Modules make possible rapid, efficient transferral of data and programs within the system. Each Buffer Module is capable of independently executing a stored program which permits it to gather data from magnetic tapes, drums, or the system input-output buffers. Similarly, the Buffer Module can distribute results under the control of its own stored program. The Buffer Module may also be used to augment the memory of any of the Computer Modules; in this mode, operands taken from the Buffer memory may be utilized by the Computer Module almost as rapidly as operands from the Computer's internal memory.

The system employs magnetic tapes, magnetic drums, printers, plotters, and input-output buffers as independent modules. All of these modules communicate with one another by means of a high-speed electronic switch, similar in many respects to a telephone exchange. Figure 1 illustrates a typical system configuration. The Central Switching Exchange can be tailored to the needs of the system, and may easily be expanded to accommodate system growth. The remainder of the paper discusses the design objectives for an adequate central switch, and develops a switch design which fulfills these design objectives.

The data switch must provide multiple paths so that each Computer can communicate without delay or interference. Control of the switch points must be rapid (less than 100 microseconds) and convenient so that Computer time is not wasted in manipulating the switch. The switch must test for a busy condition before making the connection, to prevent interference with communications in progress. The switch should be reliable, providing alternate paths for communication so that failure of a cross-point does not disable the system. The switch should be modular to provide for system growth. Since new problems must be checked out while the system is in normal operation, a means of restricting the accessibility of modules must be incorporated in the design. For example, in checking out a program that involves a Computer, a Buffer and two Tape Modules, it is necessary to restrict the switching capability so that the Computer and Buffer can control connections only between the modules being code-checked. This feature

prevents the new program from interfering with normal system operation.

The RW-400 data switch employs transfluxors (double-aperture cores) as the switching element. The principle of operation is analogous to that of a transformer connection which can be disabled by saturating the iron to prevent the flux linkage of normal transformer action. Some of the desirable properties of the transfluxor switching element are extreme reliability, small size, low cost, low power consumption, connection memory, ease of control, and bilateral signal flow characteristics. Figure 2 is a diagram of a transfluxor crossbar switch.[1] In this array, the transfluxors are selected by coincident currents.

Assume a "connection" to be made and signal flow to be taking place between Signal Input 2 (S_2) and Signal Output 3 (O_3) through core 10. Signal flow consists of a train of alternately positive and negative pulses of current along S_2 which are inductively coupled by core 10 to O_3; O_3 in turn is attached to the load.

To "disconnect" or open the path between S_2 and O_3, a blocking pulse of current is introduced along blocking line B_3. This blocking pulse blocks all possible signal paths for O_3, specifically cores 9, 10, 11, and 12.

To establish a new connection, between S_3 and O_3 for example, core 11 must be unblocked. This unblocking is accomplished by passing coincident unblock currents through lines UB_3 and B_3.

A transfluxor plane is provided for each signal line required between the computation modules. The RW-400 switch provides separate line groups employing 18 lines for communication in each direction. Consequently, the RW-400 switch employs 36 transfluxor planes. Control of the switch is obtained by providing an input commutator which samples each of the active input lines in turn. The commutator position controls one axis of the coincident-current selection, while the other axis is specified by a Command Output instruction from the Computer or Buffer being sampled by the commutator. An additional transfluxor plane is provided to record the status of each cross-point. In this way, it is possible to test the additional plane to determine whether the connection requested is available or busy. If the device requested is busy, the connection request is rejected without interfering with the communication already in progress. This basic switch configuration meets all of the design objectives except two, which are discussed below.

Control of the switch is rapid. Since the commutator speed depends upon the number of connection requests which must be serviced, the service time for a switch instruction will vary. Average service time is estimated to be about 50 microseconds. The switch tests for a busy condition before making a connection. Magnetic switching elements promise high reliability. Alternate paths are available by employing Buffer Modules in the transfer. The switch is modular in the sense that all circuits for a given set of input or output lines are packaged on a single 19-inch insert card. A cabinet with power supplies and all the transfluxor planes forms the basic switch. As additional modules are required in the system, line circuits are added as required. Line circuits constitute about one-half of the cost of the switch.

One of the design requirements which is not met by this simple design is the ability to restrict the setting of some of the cross-points. This design requirement to inhibit certain connections could be met by adding another transfluxor plane in which allowed cross-points would be recorded by blocking or unblocking the transfluxor corresponding to the cross-point. The memory plane would then be tested to see if the connection is allowed, in the same way that a test for a busy circuit is made.

The other design requirement which is not met is harder to rectify with this simple design. Control of the cross-points of the switch in this design is not particularly convenient, since the addresses used in the switching commands must correspond to the physical location on the switch matrix of the module selected. A design which would permit the programmer the use of arbitrary tags in his program, for the modules he intends to use, would be much more useful. When the program is complete, the tags used by the programmer could be entered in a memory in the switch before the program is executed. In this way, the switch could properly interpret the tags and select the desired module. In other words, a system of symbolic addressing for the switch would be much more convenient than an absolute addressing scheme. A very rapid table look-up operation is necessary to interpret the symbolic addresses within the switch.

A transfluxor memory device, called an Interrogation Module, which performs this table look-up very rapidly, has been developed by Ralph Koerner of Ramo-Wooldridge. The transfluxor memory employs two cores to store each bit position, one for each of the two possible states of the bit. The bits of the words which constitute the table are entered in parallel along one axis of the memory plane, while word positions along the other axis of the plane are designated by a read-in signal. Figure 3 illustrates the principle of operation of a three-bit Interrogation Module. The normal or clear position would have all cores unblocked. Words are entered into the memory in such a way that a logical "one" unblocks the "one" core and blocks the "zero" core of the pair of cores associated with the bit position, while a logical "zero" blocks the "one" core and unblocks the "zero" core. If, now, the complement of a number is used to interrogate the memory along the vertical lines threading all word positions, all of the output lines will have signals coupled to them by the transfluxors except that line corresponding to the memory position which stores the word exactly matching the word at the input of the interrogation unit. For that word only, the input signals will all traverse blocked cores, since the stored word and the interrogating word are complements. This Interrogation Module can be incorporated in the design of the switch to provide symbolic addressing; but even more important, it greatly simplifies the problem of interrupt communication within the system.

Interrupt communication is necessary to coordinate, synchronize, and control the operations of the individual Computers to provide the best

overall performance. The interrupt feature is a means of forcing a branch in the computer program by the use of an external electric signal such as is generated by depressing a push-button. In the RW-400 system, most interrupt signals originate from Computer or Buffer Modules; however, other devices such as the input-output buffers, provide signals which can be utilized as interrupt signals. When interrupted, a Computer must determine the meaning of the interrupt signal and its source. Some information as to intent or source of the interrupt can be conveyed by the bit position in which the Computer receives the interrupt. Unless a directed system of interrupts is employed, the bit assignments are rapidly used up specifying which Computer or Buffer should respond to the signal.

As an additional means of coordinating system actions, a digital clock is provided which can be read by any Computer or Buffer. The clock is more useful when it generates an interrupt signal at the end of a prescribed interval of time, directing it to the appropriate Computer or Buffer. The directed interrupt is an important system requirement.

Figure 4 illustrates a switch design which employs an Interrogation Module to control the simple data switch described earlier. The commutator samples the input lines in turn, routing the address tag specified by the Command Output instructions to the Interrogation Module. In general, the commutator position determines the selection along one axis, while the output of the Interrogation Module determines the selection along the other axis. The Interrogation Module is provided with additional word positions which are used to route interrupt signals to specified Computers. Note that the digital clock is sampled along with the Computers and Buffers at the input to the switch. By placing a time value in one of the interrupt words of the Interrogation Module, an interrupt is generated when the digital clock value equals the pre-stored time. In this way, Computer interrupts can be generated at specific time intervals. In normal system operation, Buffer Modules read or write the tape or drum records so that Computers usually accept data from Buffer Modules or deliver data to Buffers. For this reason, it is convenient to permit a Buffer to connect itself to a Computer if the Computer connection is not busy. When the Buffer stops its self-instruction program, interrupting the Computer, the Computer can use the results immediately without loss of time manipulating the switch. This type of connection is obtained by assigning either Computers or Buffers entry to both the x- and y-axis of the switch. The figure shows Buffers on both axes of the switch. If there are fewer Computers than Buffers in the system, the Computers would receive this double assignment so as to conserve the number of y-connections available to the system.

The Interrogation Module provides a ready means of restricting accessibility of certain cross-points of the switch. The word length is increased beyond that required for the selection address, and the extra bits are interrogated by the commutator position, while the selection address interrogates the rest of the word. Selection is inhibited unless the correct commutator code is stored in the memory of the Interrogation Module. Note that, by interrogation with "zero's" on both wires of a bit position, no signal is generated at the output, regardless of how the cores are set. In this way, it is possible to ignore certain fields of the interrogation word. If the commutator position interrogation occurs using an assigned bit for each commutator position, more than one device can be permitted to control a given switch selection. This is accomplished by setting into the commutator field of the Interrogation Module the proper bits for that selection word. In this way, identical switch address tags may be used concurrently in two different computer programs without confusion, since each will be assigned a different commutator bit by the Master Computer in setting up the assignment table. The bit will differentiate between the two Computers which are in simultaneous operation; thus, the switch can distinguish between the two identical address tags by determining which Computer is using the tag.

The use of the Interrogation Module in conjunction with the transfluxor switch provides a very convenient means of solving the difficult communication problem associated with data switching and interrupt control within a polymorphic computer system.

References

1. J.A. Rajchman and A.W. Lo. "The Transfluxor," Proceedings of the IRE (March 1956), pp. 321-332.

2. J.A. Rajchman and A.W. Lo. "The Transfluxor," Proceedings of the IRE (March 1956), pp. 321-332.

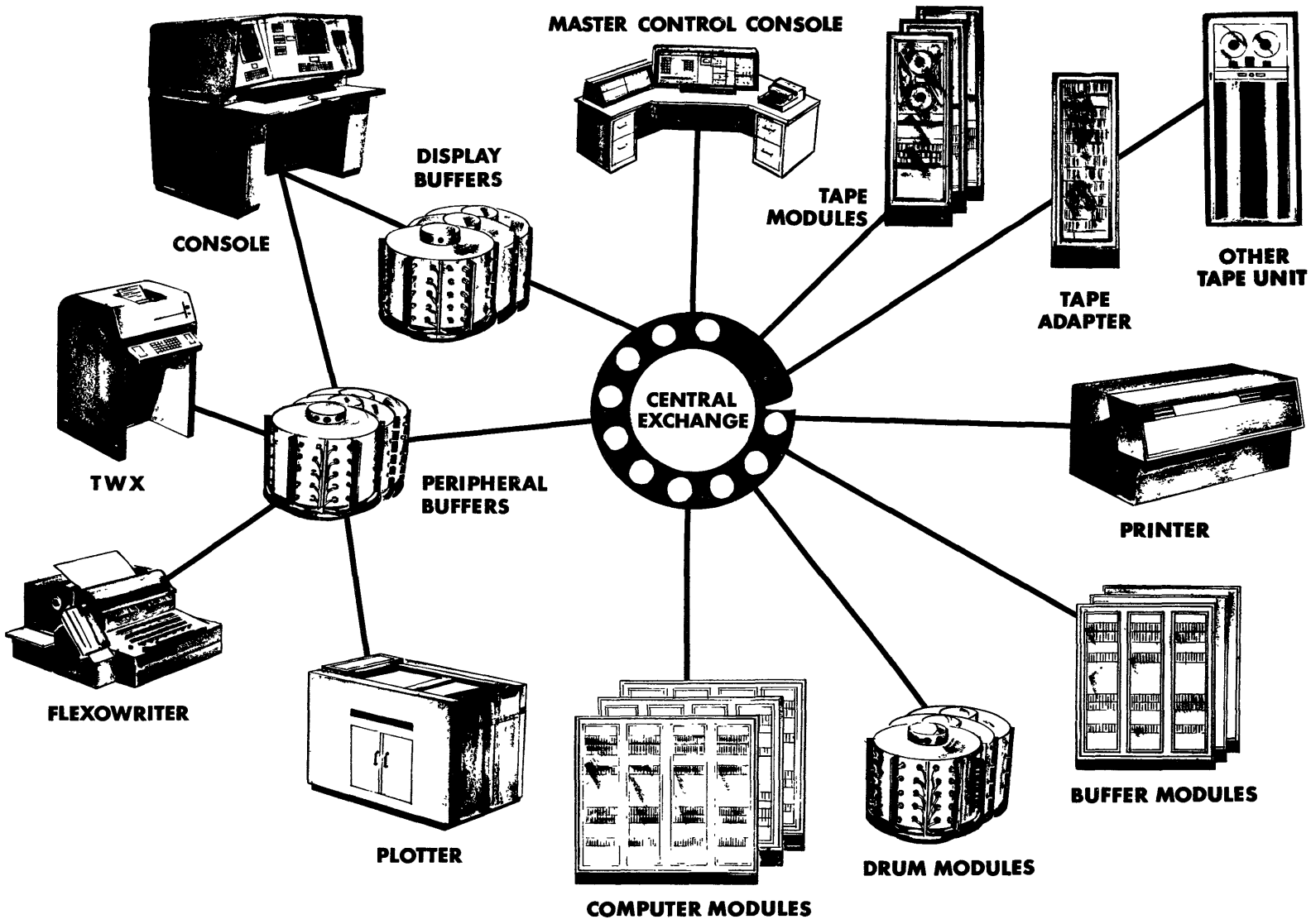


Figure 1
SYSTEM DIAGRAM

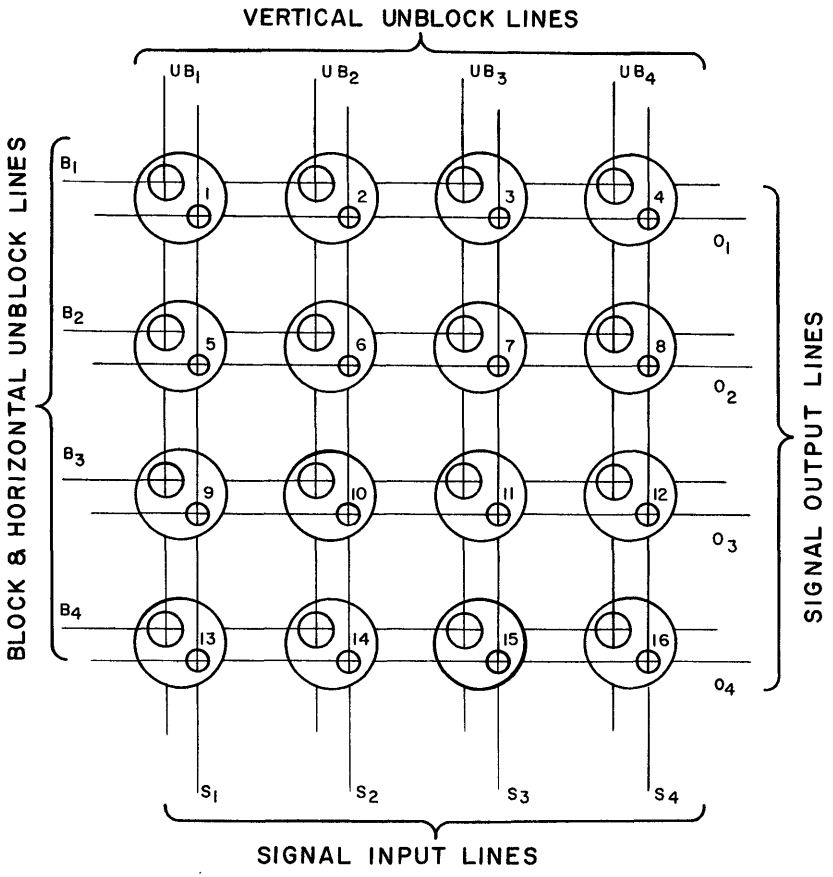


Figure 2
TRANSFLUXOR CORSSBAR SWITCH (SEE REFERENCE NO. 2)

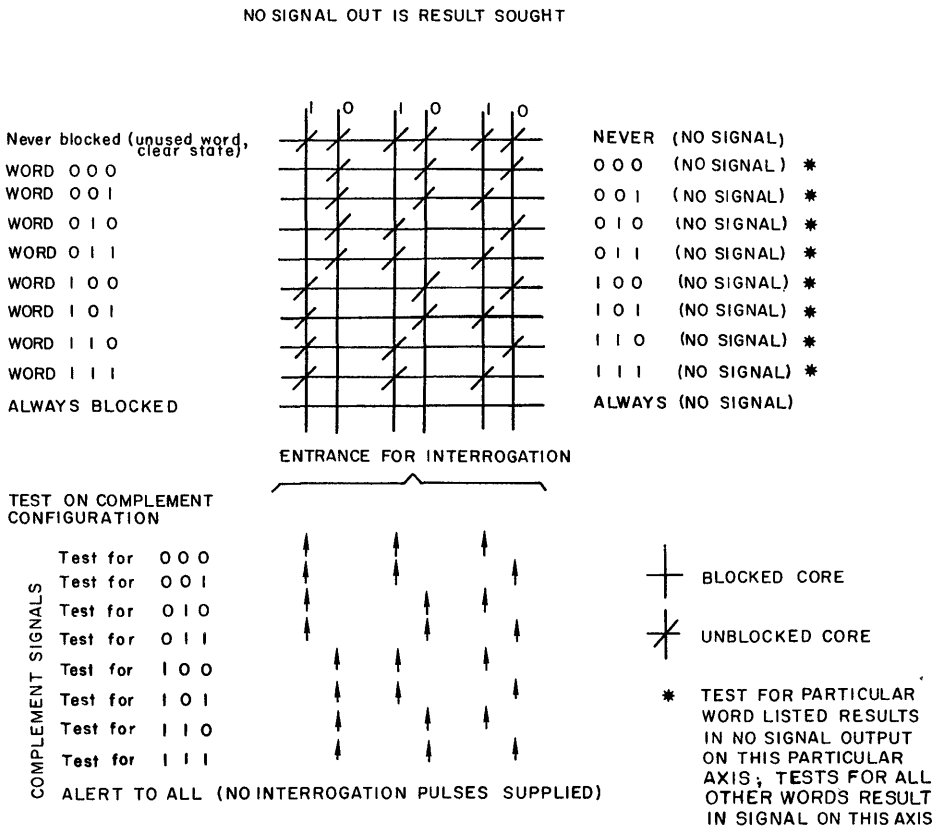


Figure 3
THREE-BIT INTERROGATION MEMORY

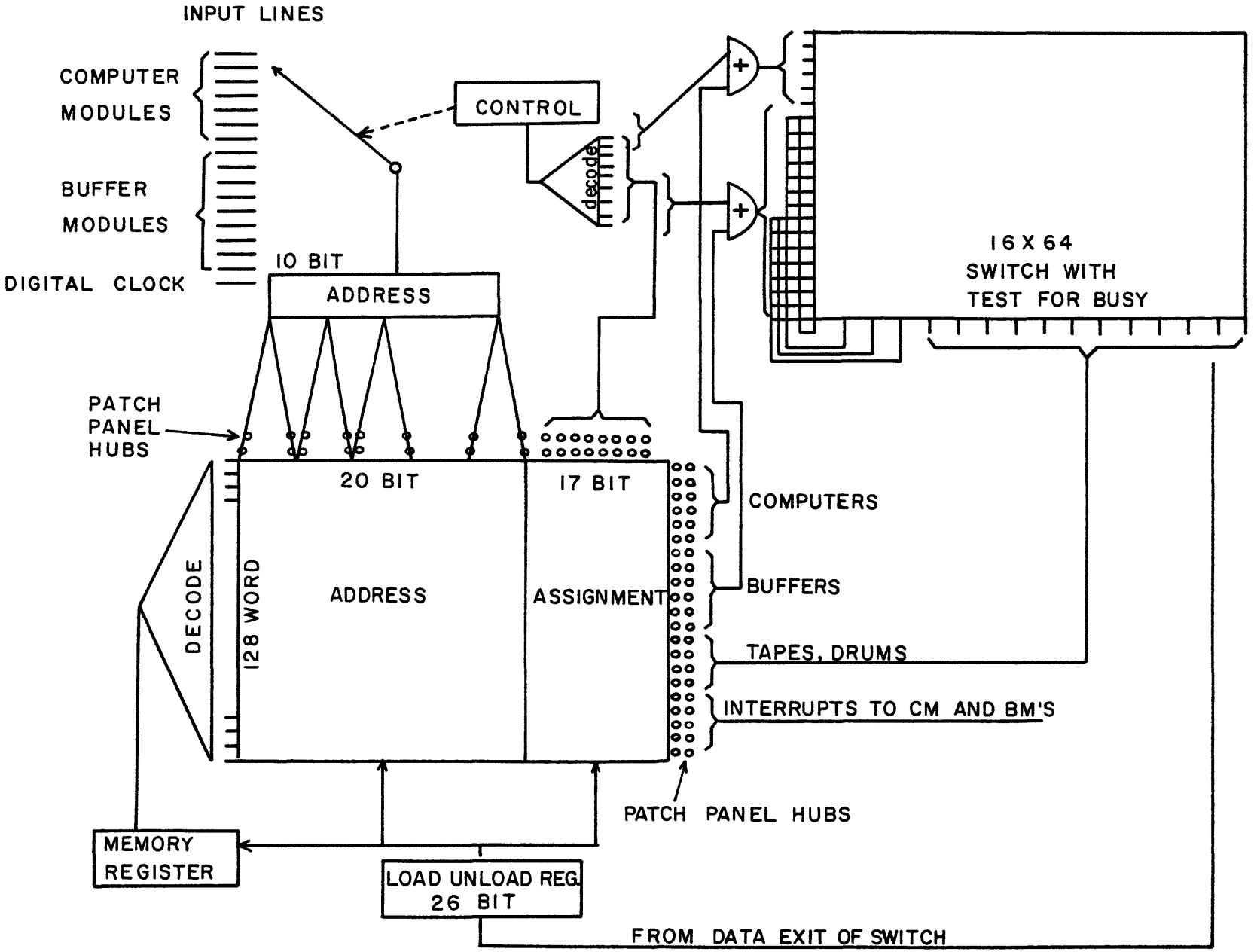


Figure 4
SWITCH DESIGN EMPLOYING INTERROGATION CONTROL

ENCODING OF INCOMPLETELY SPECIFIED BOOLEAN MATRICES

T. A. Dolotta
E. J. McCluskey, Jr.

Princeton University
Princeton, New Jersey

Summary — The problem discussed in this paper is a generalization of the problem of choosing the binary codes for the various operations of a digital computer with a view towards minimizing the gating circuitry in the computer's central control. The general procedure is a systematic method for the simultaneous encoding of a large number of arbitrary Boolean functions in an economical manner. The method is applicable to those cases where the designer is allowed to choose the independent variables so as to minimize the circuit which realizes the given Boolean functions. Other applications of the procedure are indicated. The method given includes a means for determining a lower bound on the cost of the circuit which realizes the given Boolean functions. The entire procedure is systematic and lends itself well to mechanization via a digital computer program. A discussion of the advantages and limitations of the method presented here is included, and the method is compared with a previously published procedure for attacking the same problem.

Introduction

In the design of the control part of a digital computer, there may often arise the following problem: We are given the number of bits reserved for the operation code, and also the various control signals that have to be generated for each operation, and we want to choose the codes for the various operations so that the control signals can be obtained from the operation codes by a simple and cheap network.

We will assume that we have m operations to encode with n bits, (of course, $m \leq 2^n$), and that we have to generate p control signals. For each operation, we can then specify which control signals must be ON (i.e., equal to "1"), which must be OFF (i.e., equal to "0"), and which may be either ON or OFF (i.e., "don't cares"). We now formally state the problem.

I. Problem Statement

Assume that we have p columns, F_1 to F_p, each column containing m symbols. There are 3 allowable symbols, namely 0 (zero), 1 (one), and $\emptyset$ (don't care). We consider each column as a Boolean function of n independent variables. If any column, F_i contains only zeros and ones, and if $2^n = m$, then we will say that F_i is completely specified. Otherwise, F_i is incompletely specified. (We do not consider the trivial case where F_i is made up entirely of don't cares; i.e., is completely unspecified.) Our problem is so to choose the n independent variables that the p columns can be synthesized from these variables in a simple and cheap fashion under a set of fairly realistic assumptions. We also want to obtain a lower bound on the cost of synthesizing the p functions.

II. Assumptions and Definitions

1. The number of independent variables, n, has to be specified externally, subject to the restriction that $2^n \geq m$.
2. The n independent variables, denoted by $x_1, x_2, \ldots, x_n$, are available together with their complements at no cost.
3. The p functions $F_1, F_2, \ldots, F_p$ are to be synthesized from the independent variables by using 2 stage diode logic only. (Inverters are not allowed.)
4. The cost of synthesizing these functions will be defined as the number of diodes required by the circuit designed under the above assumptions.
5. Given a column, F_i, we define the complement of F_i, denoted by $\bar{F}_i$, as the column obtained by changing all the 0's in F_i to 1's and all the 1's to 0's. The $\emptyset$'s are not affected.
6. F_i will be called a 0(1)-column if it contains no 1's (0's).
7. F_i will be said to be specified in any of its m positions if that position contains a 0 or a 1. Otherwise (if the position contains a $\emptyset$), F_i is said to be unspecified in the particular position.
8. Given two columns, F_i and F_j, there will exist the intersection of F_i and F_j, written as $F_i \cdot F_j$, if and only if F_i and F_j agree (have the same symbol) wherever both F_i and F_j are specified. $F_i \cdot F_j$ $(= F_j \cdot F_i)$ will be defined as a column of length m which agrees with both F_i and F_j wherever either is specified and contains $\emptyset$'s everywhere else. In figure 1, columns A and B have an intersection, which is shown as column D; columns A and C' also intersect. Columns B and C do not intersect. (We write this as $B \cdot C = \Lambda$)

A	B	C	D	E	F	
0	0	$\emptyset$	0	0	1	$D = A \cdot B$
1	$\emptyset$	0	1	1	$\emptyset$	
1	1	0	1	1	0	
$\emptyset$	1	1	1	1	0	
0	0	1	0	0	1	
$\emptyset$	1	$\emptyset$	1	0	0	
$\emptyset$	$\emptyset$	0	$\emptyset$	$\emptyset$	1	

Figure 1

9. Three columns, say F_i, F_j, and F_k will have an intersection, $F_i \cdot F_j \cdot F_k$, if and only if $F_i \cdot F_j$, $F_i \cdot F_k$, and $F_j \cdot F_k$ all exist. The

intersection will agree with all three columns wherever any of the three is specified. The extension to the intersection of any number is obvious.

10. Given two columns, F_i and F_j, F_i will be said to include F_j if and only if F_j agrees with F_i wherever the latter is specified. We write this relation as $F_i \supset F_j$. In Figure 1, we see that $A \supset E$, and $B \supset F'$. If $F_i \supset F_j$, we will call F_i the including column, and F_j the included column. It is obvious that F_i has at least as many $\emptyset$'s as F_j.
11. Below we list some of the more obvious and useful facts which follow from the above definitions:

    a. If $A \supset B$, and $B \supset A$, then $A = B$.
    b. If $A \supset B$, and $B \supset C$, then $A \supset C$.
    c. If $A \supset B$, $B \supset C$, and $C \supset A$, then $A=B=C$.
    d. If $A \supset B$, and $B \cdot C$ exists, then $A \cdot C$ exists.
    e. If $A \supset B$, then $A \not\supset B'$, unless A is completely unspecified.
    f. If $A \cdot B$ exists, then $A \cdot B'$ does not exist, unless A or B or both are unspecified in every position.
    g. Obviously, if $A \supset B$, then $A \cdot B = B$.
    h. If $A \cdot B = B$, then $A \supset B$.
    i. If $A \supset B$, then $A' \supset B'$.
    j. If $A \cdot B$ exists, then so does $A' \cdot B'$, and $(A \cdot B)' = A' \cdot B'$.
    k. If $A \cdot B$ exists, let $A \cdot B = C$; then $A \supset C$ and $B \supset C$.

12. For any column, F_i, let
    $N_{i,0}$ = number of zeros occurring in F_i
    $N_{i,1}$ = number of ones occurring in F_i
    $N_{i,\emptyset}$ = number of $\emptyset$'s occurring in F_i
13. Any column, F_i, will be called codable if and only if

$$N_{i,0} \leq 2^{n-1} \quad \text{and} \quad N_{i,1} \leq 2^{n-1}$$

    Otherwise the column will be called not codable.
14. We assume that constant signals representing "0" and "1" are available to us.
15. Column F_i will be said to cover column F_j if and only if either $F_j \supset F_i$ or $F_j \supset F_i'$

III. Procedure

The procedure that we will follow in order to obtain an encoding of the various operations is probably best explained by first considering a concrete example. Figure 2 shows a 6 x 7 matrix: the six rows correspond to six operations which we must encode,

Oper. \ F_i	F_1	F_2	F_3	F_4	F_5	F_6	F_7
C_1	0	$\emptyset$	0	1	0	0	1
C_2	1	1	0	1	$\emptyset$	1	$\emptyset$
C_3	$\emptyset$	$\emptyset$	1	0	$\emptyset$	$\emptyset$	$\emptyset$
C_4	0	0	1	1	$\emptyset$	0	1
C_5	0	1	1	1	0	1	$\emptyset$
C_6	1	0	0	1	1	$\emptyset$	1

Figure 2

and the seven columns represent seven control signals which must be generated from the operation codes. Each operation must be represented by a three-bit code word. The manner in which the rows and columns are numbered is arbitrary.

We should first examine the matrix row by row to insure that no two rows have an intersection. Should the above not hold, i.e., should there be two rows which have an intersection, then this would mean that the two operations corresponding to the two rows are not distinct, and could be replaced by one operation. In a normal situation, this would not be the case.[1] We note that in Figure 2, there are no two rows which have an intersection.

We now focus our attention on the columns of our matrix. We first note that column 7 is a 1-column. Therefore we need not worry about encoding it, since we can obtain the corresponding control signal by using a constant "1" signal in its place. We therefore discard the 7th column.

At this point we examine the remaining six columns to determine whether there exist any inclusions between any two columns. We find that $F_5 \supset F_1$. We note that wherever the control signal represented by F_5 is needed, we may with impunity replace it by the signal corresponding to F_1, since F_5 is the same as F_1 wherever F_5 is specified. And since F_1 has to be synthesized, we needn't synthesize F_5, but merely use F_1 in its place. Therefore we discard F_5, and concern ourselves with the remaining five columns, namely F_1, F_2, F_3, F_4, and F_6. We call this set of five columns Set A.

We now examine these five columns to determine whether there exist intersections between any two of these columns. We find that $F_2 \cdot F_6$ exists. Let us define $F_a = F_2 \cdot F_6$. We observe that if instead of encoding F_2 and F_6, we encode F_a, then we can replace F_2 and F_6 by F_a, since the latter agrees with both F_2 and F_6 wherever either is specified, so that we will satisfy the original specification. Thus we now have only four columns to encode, namely F_1, F_3, F_4, and F_a. Figure 3 shows these four columns.

Oper. \ F_i	F_1	F_3	F_4	F_a
1	0	0	1	0
2	1	0	1	1
3	$\emptyset$	1	0	$\emptyset$
4	1	1	1	0
5	0	1	1	1
6	1	0	1	0

Figure 3

We should note parenthetically here that in our original matrix (Fig. 2) there also existed the intersections $F_2 \cdot F_5$, $F_2 \cdot F_7$, $F_4 \cdot F_7$, and $F_6 \cdot F_7$. All of these, however, involve F_5 and F_7 which have been discarded, and therefore these four intersections are of no further interest to us.

At this point we would like to identify each of the four columns in Figure 3 with one of the three independent variables or a complement of such a var-

[1] This remark may not apply in other applications to which our encoding procedure might be put; and it is conceivable that we might want to have two or more distinct operations even though one might suffice.

iable. Should this be possible, we would then have an ideal (or zero cost) encoding in the sense that each of the seven original signals could be obtained from the independent variables or a constant signal without any intervening logical network. If an ideal encoding is to be possible, then we must be able to identify each one of our gating functions with at least one of the independent variables (or its complement). An ideal encoding very often is not possible for one of two reasons. First of all, in order to be able to identify four columns with 3 independent variables, we must identify at least two of the four columns with one of the independent variables; if this is to be possible, then there must exist at least one inclusion or intersection relation within our four columns. However, we know that this is not the case, and therefore at best only three of the four columns can be identified with the independent variables, and the remaining column will have to be synthesized with a logical network. Secondly, if a column is to be made into (i.e., identified with) an independent variable, then that column must be codable. This follows from the fact that if we write down the 2^n possible n-bit words, we note that each bit position contains exactly 2^{n-1} "1" 's, and 2^{n-1} "0" 's. Since in our case n = 3, we know that no column which contains more than four "1" 's or "0" 's can be used as an independent variable. We note in Figure 3 that column F_4 is not codable. We conclude on the basis of either or both of the reasons stated above that an ideal solution is not possible in our example, and that at least one 2-diode gate will be needed to synthesize the signal represented by F_4.

We now identify the three remaining columns (namely 1, 3, and a — calling them Set B) with the three independent variables in any arbitrary fashion. All columns which are members of Set A but not of Set B constitute Set C; in this case Set C contains column 4. Figure 4 shows such an assignment, with ϕ 's replaced by blanks.

Oper. \ x_i	x_1	x_2	x_3	
1	0	0	0	
2	1	0	1	
3		1		$F_1 = x_1$
4	0	1	0	$F_3 = x_2$
5	0	1	1	$F_a = x_3$
6	1	0	0	

Figure 4

It appears at first that we can choose any of the following code words for row 3: 010, 011, 110, 111. However, 010 and 011 are used already by rows 4 and 5, respectively; our choice is limited to 110 and 111. It can be seen that either of these two choices will give a valid encoding, in the sense that no code word appears more than once. Also, it turns out that either choice will give a simple encoding for F_4. Arbitrarily, we chose 111 as the code word for row 3. Figure 5 shows that final encoding, together with the expressions for the control signals in terms of the independent variables. Of course, we can use the three independent variables in any order we wish, and we may complement any or all of them, so that there are many codes which are symmetries of our code and

Oper.	x_1	x_2	x_3	
1	0	0	0	$F_1 = x_1$
2	1	0	1	$F_2 = x_3$
3	1	1	1	$F_3 = x_2$
4	0	1	0	$F_4 = x_1' + x_2'$
5	0	1	1	$F_5 = x_1$
6	1	0	0	$F_6 = x_3$
				$F_7 = 1$

Figure 5

which give the same type of solution to this problem.[2] This fact is especially useful if we want to avoid the use of a specific code word (such as an all-zero code word), because it is possible to eliminate such a word by complementing one or more of the independent variables whenever $m < 2^n$. In our example, complementing of x_3 will result in the elimination of the all-zero word. If this is done, then of course x_3 and x_3' have to be substituted for one another in all the expressions where they occur.

We must realize that when we ascertained that columns F_1, F_3, and F_a were all codable, this was only a necessary, but not a sufficient condition for these columns to make a valid code. In Figure 1, columns C, D, and E are all codable, but they do not make a valid code, since the code 011 occurs twice. (For a code to be valid, it is necessary and sufficient that no code word appear more than once.)

We see that we have obtained here as cheap a 3-bit encoding of our original matrix as is possible, since we have in fact met our minimum estimate of cost. It should be apparent that this is not always the case, and that there may arise other problems when we attempt to encode very large matrices. Consideration of these points as well as a more rigorous and complete statement of our procedure will be the subject of the next sections.

IV. General Considerations

In this section we will discuss some possible situations which have to be considered in order to make our procedure complete.

1. Inclusions. Any time we find that two columns are in the following relationship $F_i \supset F_j$ we may always replace F_i by F_j and discard F_i. On the other hand, if the relationship is $F_i \supset F_j$ then F_i may be discarded only if F_j or $F_j \cdot F_k$ (where F_k is any other column or intersection of columns) is used as an independent variable. Otherwise, we must synthesize F_i; this restriction would not hold if we allowed the use of inverters in addition to 2-stage diode logic.

---

[2] Since there are n! ways of ordering n columns, and 2^n ways in which n columns may be complemented, there are $n! \cdot 2^n - 1$ codes which are symmetries of any n-bit code; some of these may be identical with the original code.

2. Intersections. Only those intersections which are codable are of interest to us when we are choosing the columns to be used as independent variables. Non-codable intersections are only useful in the sense that they can be used sometimes to get a cheaper network when the columns which make up the intersections are to be expressed as functions of the independent variables. For instance, if we have two non-codable columns which have an intersection, it may sometimes prove cheaper to synthesize the intersection as one function rather than the two columns as two separate functions. For this reason, non-codable intersections are of interest only if they contain no primed columns. (Again, this last statement would not be true if we allowed the use of inverters.)

It sometimes happens that we find that the following (codable) intersections exist: $F_i \cdot F_j$ and $F_i \cdot F_k$ (but $F_j \cdot F_k = 0$). In that case, we would use F_j and $F_i \cdot F_k$ (or $F_i \cdot F_j$ and F_k) rather than $F_i \cdot \bar{F}_j$ and $\bar{F}_i \cdot F_k$, since in any case we cover all 3 columns, but F_j has at least one more $\emptyset$ than $F_i \cdot F_j$, and it is in general desirable to have many $\emptyset$'s in the columns used as the independent variables, because this gives us more freedom in "tailoring" the independent variables.

It may occur that $F_i \cdot F_j = F_i \cdot F_j \cdot F_k$; this means that $F_i \cdot F_j \subset F_k$. In such an instance we would usually consider the intersections as including F_k, namely as $F_i \cdot F_j \cdot F_k$ rather than as just $F_i \cdot F_j$.

3. Validity. We have given above the necessary and sufficient condition for a code to be valid. It is quite simple to ascertain whether n columns make a valid code if there are no $\emptyset$'s in these columns. If very many $\emptyset$'s do appear, however, it becomes more tedious to determine whether a code can be made valid. However, it is never necessary to resort to enumeration, since the problem of determining whether a code can be made valid is equivalent to the "marriage problem" and can be solved by the algorithms given in references 6, 7, and 8.

4. Cost. Once a valid code covering as many of the columns of our original matrix as possible is found, it is possible to determine (from the number and type of columns which are not covered by this code) the minimum cost of realizing our circuit (in terms of either diodes or diode gates). A more detailed discussion of this particular procedure is given in the Appendix.

V. Summary of Procedure

In this section we give a list of the steps which we execute in encoding a matrix.

1. Discard all 1-columns and 0-columns.
2. Find all inclusion relations of the $F_i \supset F_j$ type which exist among the columns remaining after Step 1. Discard all the including columns found in this manner. Call A the set of columns remaining now.
3. Make a list of all the inclusions of the type $F_i \supset F_j$ which exist in A and for which F_j is codable.
4. If it is desired to eliminate redundant rows from our original matrix, then examine the rows for such redundancies and eliminate them. The procedure here consists essentially of finding all inclusions among rows (of the type $c_i \supset c_j$), discarding all including rows, and then of finding all the intersections (which contain no primed rows) among the remaining rows and of combining the rows so as to get as small a number of rows as possible. When this is done, we may find that we need fewer bits than n to encode our matrix, since by eliminating rows we have decreased the value of m. If, however, the smaller number of bits is used in our code, we will often find that the cost of the final circuit will be higher than if we used the original values of m and n.
5. In Set A, find all pairwise intersections which either contain no primed columns, or are codable (or both).
6. Using the results of Step 5, form all the maximal intersections from Set A which again are either codable or contain no primes.
7. Using the results of Step 6, form two sets:
   Set B will contain all the maximal codable intersections, as well as all the codable columns which are members of Set A but are not covered by any intersections included in B.
   Set C will contain all the maximal non-codable intersections found in Step 6 as well as all those non-codable columns in A which are not covered by intersections included in Set C. These two sets cover all the columns in A.
8. From Set B, choose a set of n members. Call this set D.[3] D is so chosen that all the columns of A which do not appear in D can be covered with a minimum number of members of Sets B and C. Set D is then modified using the results of the second paragraph of Section IV-2 above, so that no column of A is covered by more than one member of D. Should there be more than one possible choice of members of D, then we choose the members so as to get as many $\emptyset$'s in D as possible.
9. Check Set D for validity. If it is not valid, repeat Step 8 and choose the next best set for D. Repeat Steps 8 and 9 until a valid set is found. In executing Steps 8 and 9 we may be able to use some of the relations obtained in Step 3 above to eliminate or simplify some members of Sets B and D, as indicated in Section IV-1 above.
10. At this point we can compute a minimum bound on the circuit cost (see the Appendix).
11. Once a valid set is found, we may have to replace some of the $\emptyset$'s by 1's or 0's to assure the validity of the set. The remaining $\emptyset$'s are replaced by 1's or 0's so as to make easy

[3] If B has n members, then Sets B and D are identical. If B has less than n members, we add arbitrary columns to B. This is illustrated in Section VI below. If B has less than n members, and Set C is empty, this implies that there are redundant rows in our matrix.

the encoding of the columns which are not identified with the independent variables (but of course still preserving the validity of the code).

12. If the number of possible valid D sets is small, or if this procedure is carried out on a computer, we may want to try several of these sets in order to obtain a better solution. The amount of enumeration here is usually quite small and fairly manageable, especially on a computer.
13. Once a code is obtained, we encode all the members of Set C as well as those members of Set B not included in our final D set. There may be here more than one possible set of columns we may choose for encoding, as indicated in the first paragraph of Section IV-2. The encoding proper is done by standard minimization methods for Boolean functions, and is well described in literature. (See, for instance, references 1 and 2.)

VI. Illustrative Example

The problem which we are treating here was first proposed by E. Hirschhorn (Ref. 3). The procedure proposed by Hirschhorn is somewhat intuitive and depends strongly on the designer's ingenuity. Hirschhorn gives an example in that paper, and we will treat the same example in order to afford us some amount of comparison between the two methods.

The matrix is shown in Figure 6. The value of n is set at the minimum, namely 5. Hirschhorn finds that if a straightforward code is used (i.e., 00001 is assigned to row 1, 00010 to row 2, 00011 to row 3, etc.), then 45 diodes and 16 gates, or 39 letter symbols (or contacts) are required to synthesize the ten control signals. Hirschhorn's procedure reduces these numbers to 24 diodes and 9 gates, or 25 letter symbols. Both we and Hirschhorn design the circuit for each control signal separately, rather than designing one multiple output network for all the control signals.

Op. (C's)	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}
1	1	1	1	0	ø	0	ø	ø	ø	ø
2	1	1	0	ø	0	1	ø	ø	ø	ø
3	1	0	1	1	ø	ø	ø	ø	0	ø
4	1	0	1	ø	0	ø	ø	ø	ø	ø
5	1	0	1	1	ø	ø	1	ø	ø	ø
6	1	0	1	ø	0	ø	ø	ø	ø	ø
7	1	0	1	ø	ø	ø	ø	0	ø	ø
8	1	0	1	1	0	ø	ø	ø	ø	ø
9	1	0	1	ø	ø	ø	1	ø	ø	1
10	1	0	1	0	0	ø	0	ø	ø	ø
11	1	0	1	ø	ø	0	1	ø	0	ø
12	1	0	1	1	0	0	1	ø	ø	ø
13	1	0	1	ø	ø	0	ø	0	ø	ø
14	1	0	1	0	0	ø	ø	0	ø	ø
15	1	0	ø	0	ø	ø	ø	0	ø	0
16	1	0	ø	1	0	ø	ø	ø	ø	ø
17	1	0	ø	0	ø	ø	1	1	ø	ø
18	1	0	0	0	0	0	ø	ø	ø	1
19	0	0	0	1	ø	1	1	ø	ø	ø
20	0	0	1	ø	0	ø	ø	ø	1	ø

Figure 6

We will now treat this example by our method.

1. F_5 is a 0-column; it is discarded.
2. There are no inclusions.
3. There are no inclusions.
4. Hirschhorn did not attempt to eliminate any rows. To afford a better comparison, we will do likewise. However, we should point out that the number of rows can be reduced down to seven, by making use of the following intersections which exist in the matrix:

$$c_3 \cdot c_4 \cdot c_5 \cdot c_6 \cdot c_7 \cdot c_8 \cdot c_9 \cdot c_{11} \cdot c_{12} \cdot c_{13} \cdot c_{16}$$
$$c_7 \cdot c_{10} \cdot c_{13} \cdot c_{14} \cdot c_{15}$$
$$c_{17} \cdot c_{18}$$

These are not the only intersections which occur in the matrix, but they are sufficient to show that the following transformation could be made:

c_a replaces c_1
c_b " c_2
c_c " $c_3 \cdot c_4 \cdot c_5 \cdot c_6 \cdot c_8 \cdot c_9 \cdot c_{11} \cdot c_{12} \cdot c_{16}$
c_d " $c_7\ c_{10}\ c_{13}\ c_{14}\ c_{15}$
c_e " $c_{17} \cdot c_{18}$
c_f " c_{19}
c_g " c_{20}

Therefore n could be made as small as 3.

5. Set A consists of all the columns except F_5. The following pairwise intersections exist (we omit the letter F, and use the subscripts only):

$1 \cdot 9'$
$3 \cdot 8'$
$4 \cdot 9'$
$6 \cdot 8,\ 6 \cdot 9,\ 6 \cdot 10'$
$7 \cdot 8,\ 7 \cdot 9',\ 7 \cdot 10$
$8 \cdot 9,\ 8 \cdot 9',\ 8 \cdot 10$
$9 \cdot 10,\ 9 \cdot 10'$

Columns 1 and 2 are not codable, since $2^{n-1} = 16$ for $n = 5$, and $N_{1,1} = 18$ and $N_{2,0} = 18$; therefore we discard the intersection $1 \cdot 9'$, since it is not codable and contains a primed variable.

6. The following intersections are maximal and codable (by the rules of section II-11 above):

$3 \cdot 8'$
$4 \cdot 9'$
$6 \cdot 8 \cdot 9$
$6 \cdot 9 \cdot 10'$
$7 \cdot 8 \cdot 9' \cdot 10$

7. Set B is identical to the list of intersections immediately above. Set C contains columns 1 and 2.
8. We note that we can cover all of B with four columns. In fact, we can choose any of the following

sets:

a.	3 • 8', 4 • 9', 6 • 10', 7	31
b.	3 • 8', 4 • 9', 6, 7 • 10	31
c.	3 • 8', 4, 6 • 9 • 10', 7	31
d.	3 • 8', 4, 6 • 9, 7 • 10	31
e.	3 • 8', 4, 6 • 10', 7 • 9'	31
f.	3 • 8', 4, 6, 7 • 9' • 10	31
g.	3, 4 • 9', 6 • 8, 7 • 10	29
h.	3, 4 • 9', 6 • 10', 7 • 8	29
i.	3, 4 • 9', 6, 7 • 8 • 10	30
j.	3, 4, 6 • 8 • 9, 7 • 10	29
k.	3, 4, 6 • 9 • 10', 7 • 8	29
l.	3, 4, 6 • 8, 7 • 9' • 10	29
m.	3, 4, 6 • 9, 7 • 8 • 10	30
n.	3, 4, 6 • 10', 7 • 8 • 9'	29
o.	3, 4, 6, 7 • 8 • 9' • 10	30

Next to each row, we have listed the sum of all the $\emptyset$'s in the four columns. Thus we should first choose D from among the first six possibilities. At this point we can further narrow down our choice by considering the columns not covered by D. By examining column 1, we note that if the code words for rows 19 and 20 were to be adjacent (i.e., if they differed in only one bit position), then we could synthesize F_1 with only one gate. Similarly column 2 shows us that we want row 1 and row 2 code words to be adjacent. As a result, we would like to avoid the following intersections in our code:

4•9, 7•9', 7•9'•10, 7•8•9', and 7•8•9'•10

because the columns resulting from these intersections differ in positions 1 and 2, or 19 and 20. (This is also true of any intersection which covers column 3, but since we want to include this column in our code, we cannot do anything about it.). Thus our first choice narrows down to two sets:

3•8', 4, 6•9•10', 7
and 3•8', 4, 6•9, 7•10

9. Both of the above sets are valid, so we'll try the first one. Figure 7 shows these four columns and the fifth column (x_5) needed to make a 20-word code. Blanks are used instead of $\emptyset$'s, and the underlined entries have been made in accordance with the considerations given above to facilitate the encoding of columns 1 and 2. We must make sure that such replacements of $\emptyset$'s with 0's and 1's do not make our code invalid. In our case the code is still valid.
10. Since we have found a code which takes care of all but the first two columns, we conclude that the minimum cost for our final circuit will be at least two 2-diode gates, or 4 diodes and 2 gates, or 11 letter symbols. (See Appendix for details of this particular step.)
11. We now fill out the remainder of our code in any arbitrary fashion, with the only restriction being that the code has to be kept valid, i.e., no two rows may be assigned the same code word.
12. We may, if we wish, try one or more of the remaining fourteen possible D sets, to determine whether a better code is possible. We will not do this, since, as we will see in the next step, our solution is quite good.
13. Since we have used 20 of the 32 possible code words, the remaining 12 words become in effect "don't cares" and may be used to simplify the expressions for F_1 and F_2. Figure 8 shows the code filled out as per Step 11 above, and the resulting expressions for the 10 gate signals. We see that our solution requires 7 diodes and 2 gates, or 14 letter-symbols. This is quite close to the minimum cost bound as determined above, and considerably better than the solution proposed by Hirschhorn, which requires 24 diodes and 9 gates, or 25 letter symbols.

Oper.	x_1	x_2	x_3	x_4	x_5
1	1	0	0	<u>0</u>	<u>0</u>
2	0	<u>0</u>	1	<u>0</u>	<u>0</u>
3	1	1	0		
4	1				
5	1	1		1	
6	1				
7	1				
8	1	1			
9	1		0	1	
10	1	0		0	
11	1		0	1	
12	1	1	0	1	
13	1		0		
14	1	0			
15	1	0	1		
16		1			
17	0	0		1	
18	0	0	0		
19	0	1	1	1	1
20	1	<u>1</u>	1	<u>1</u>	<u>1</u>

$x_1 = 3 \cdot 8'$
$x_2 = 4$
$x_3 = 6 \cdot 9 \cdot 10'$
$x_4 = 7$

Figure 7

We should note that while performing Step 8, we could have also observed that since with six columns we could have covered all the members of Set A, and that if we were to use six bits, all the columns of Set A would be codable, so that it is conceivable that with n = 6 we could have obtained an ideal solution, provided we could have made a valid code. In our case this is not possible for n = 6, because if we use columns 1, 2, and 3 (or 3•8') as three of the six code columns (and we must do this to obtain an ideal solution), then we note that rows 3 through 14

would be identical in these 3 columns, so that there would only be eight possible code words that could be inserted into these twelve positions. If we were willing, however, to use n = 7, then an ideal solution would be possible.

This type of consideration is quite valuable in determining, for a particular problem, what the "exchange ratio" is of cost versus code bits (in excess of the minimum number of code bits).

Op.	x_1	x_2	x_3	x_4	x_5	
1	1	0	0	0	0	
2	0	0	1	0	0	
3	1	1	0	0	0	
4	1	0	0	0	1	$F_1 = x_2' + x_3' + x_4' + x_5'$
5	1	1	1	1	0	$F_2 = x_2' \cdot x_4' \cdot x_5'$
6	1	1	0	1	1	$F_3 = x_1$
7	1	1	1	0	0	$F_4 = x_2$
8	1	1	1	0	1	$F_5 = 0$
9	1	0	0	1	0	$F_6 = x_3$
10	1	0	1	0	1	$F_7 = x_4$
11	1	0	0	1	1	$F_8 = x_1'$
12	1	1	0	1	0	$F_9 = x_3$
13	1	1	0	0	1	$F_{10} = x_3'$
14	1	0	1	1	0	
15	1	0	1	1	1	
16	0	1	0	0	0	
17	0	0	0	1	0	
18	0	0	0	0	1	
19	0	1	1	1	1	
20	1	1	1	1	1	

Figure 8

VII. Discussion

In order to make this presentation more complete, we will mention here a few considerations pertinent to the use of our encoding method.

First of all, we should realize that while we discussed the procedure in terms of design of central control of a digital computer, this is by no means the only application to which our method can be put. Another possible use it has is in the synthesis of the output networks of a sequential switching circuit. In fact, we can use this procedure whenever we need to simultaneously encode any number of Boolean functions with some number of variables.

As is true for most procedures of this type, our method will be better suited to some problems than to others. In particular, it lends itself better to encoding matrices which contain many "don't cares" than to completely specified ones. Since the method does not try all the possible assignments (there are 1.4×10^{23} possible distinct assignments for a 20-word code which uses 5 bits — see ref. 5), we are not guaranteed that our solution will be as good and cheap as possible, except in the cases where we meet our lower bound on the cost; in most cases, however, we do come quite close to this bound, and thus get quite a good solution. It is also true that this bound cannot always be met, but it does give us an idea of how good our solution is.

The main advantage of our procedure is that it is quite systematic, and lends itself well to being programmed on a digital computer. Such a mechanization might in fact greatly enhance the usefulness of our method by allowing us to try several of the possible codes as determined by our procedure (see Sec. IV, Step 12).

Finally, we should point out that this problem has also been treated from a somewhat different point of view by E. J. Schubert (Ref. 4).

VIII. Conclusion

A method has been presented for simultaneously encoding an arbitrary number of Boolean functions in an economical manner. Included is a means for estimating a lower bound on the circuit cost, so that it is possible to obtain some indication of how good a particular encoding is. The entire procedure is quite systematic, and can be mechanized on a digital computer. The application of this procedure to the synthesis of gating signals in the central control of a digital computer is given, and other applications are indicated. The advantages and limitations of the method are discussed.

Appendix

We have defined above the cost of a circuit as being the number of diodes needed to construct that circuit in a 2-stage form. We will give here a method for estimating the lower bound on the cost of a circuit in terms of both the diodes and the gates needed to realize the circuit in 2-stage form. The reason for including the cost in terms of gates is that in some realizations (for instance, when resistor-transistor logic is used) the number of gates or packages is of more interest than the number of diodes.

Let us define P_d and P_g as the minimum costs in terms of diodes and gates, respectively, of our circuit. We determine these values in the following fashion:

1. Consider Sets B and $\bar{B}$, as defined in the text above. Let b be the minimum number of columns needed to cover all those columns in $\bar{B}$ which are not covered by the largest valid D set, then it is obvious that we will need at least b gates and 2b diodes to realize the signals corresponding to all the columns in Set B.
2. Consider a column, F_i, which is not codable. We can easily count the $N_{i,o}$ and $N_{i,1}$ for this column, and

$$N_{i,\emptyset} = 2^n - N_{i,o} - N_{i,1}$$

Let $N_{i,x}$ stand for either $N_{i,o}$ or $N_{i,1}$. If the following relation holds:

$$N_{i,x} \leq 2^s \leq N_{i,x} + N_{i,\emptyset} \qquad (1)$$

where s is an integer, $0 \leq s < n$, then this means that it might be possible to make $N_{ix} = 2^s$ by replacing the proper number of $\emptyset$'s by 1's (or 0's). If this is possible and is in fact done, then F_i can be realized with a single gate and n-s diodes. Thus we want s as large as possible. As an example, consider F_1 in our illustrative example:

$N_{i,o} = 2 \qquad N_{i,1} = 18 \qquad N_{i,\phi} = 12 \qquad n = 5$

$2 \leq 2^s \leq 14$

Thus s can be 1, 2, or 3. If we choose s=3, then conceivably we could synthesize F_1 with one gate and n - s = 2 diodes. If relation (1) above does not hold, but the following relation holds,

$$N_{i,x} \leq 2^s + 2^t \leq N_{i,x} + N_{i,\phi} \qquad (2)$$

where t is an integer, $0 \leq t < s$, then the number of 1's (or 0's) in F_i can be made equal to $2^s + 2^t$, by substituting the proper number of 1's and 0's for the ϕ's in F_i.[4] If this is done, then the following possibilities exist:

a) If s = n-1, then we can synthesize F_i with 2 gates and n - t + 1 diodes. As an example, consider a case where $N_{i,1} = 17$, n=5. We see that s = 4 (= n - 1), and t = 0. The function $f = x_1 + x_2 x_3 x_4 x_5$ is seen to contain 17 points, and can be realized with 2 gates and n - t + 1 = 5 - 0 + 1 = 6 diodes.

b) If $s < n - 1$, and t = s - 1, then we can realize F_i with 2 gates and n - s + 2 diodes. For instance, if $N_{i,1} = 12$, n = 5, then s = 3, t = s - 1 = 2, and the function $f = x_1x_2 + x_1x_3 = x_1 \cdot (x_2 + x_3)$ can be realized with 2 gates and n - s + 2 = 5 - 3 + 2 = 4 diodes.

c) If $s < n - 1$, $t < s - 1$, 3 gates are needed and 2n - s - t + 1 diodes. Let, for an example, $N_{i,1} = 10$, n = 5; then s = 3, t = 1. The function $f = x_1x_2 + x_1x_3x_4$ contains 10 points. It can be realized with 3 gates and

2n - s - t + 1 = 10 - 3 - 1 + 1 = 7 diodes.[5]

The formulas given above are fairly easy to derive. However, it becomes quite difficult to generalize this procedure to the cases where F_i doesn't satisfy either relation (1) or relation (2) above. At present, it appears best to set the lower bound in such cases to 3 gates and 6 diodes (since each gate has at least 2 inputs).

3. If we now consider Set C as defined in the text, we can obtain the minimum bound on the cost of covering all the columns in C by considering all the possible combinations of columns and/or intersections which cover all the columns in C, getting the bound on each set, and choosing the lowest of these bounds. Let this lowest bound be R_d and R_g, for diodes and gates, respectively. For instance, if a Set C consisted of $F_1 \cdot F_2$, $F_2 \cdot F_3$, and F_4, then we would have to compute the following sums, where Q(i) stands for the cost of F_i as determined by Step 2 above:
   a. Q(1·2) + Q(3) + Q(4)
   b. Q(1) + Q(2·3) + Q(4)
   c. Q(1) + Q(2) + Q(3) + Q(4)

   This is done both for the diodes and gates. We then pick the lowest of these sums as R_d and R_g respectively.

4. Our minimum cost bounds are:

$$P_d = R_d + 2b$$
$$P_g = R_g + b$$

We must note here that the minimum cost bounds have been computed on the assumption that the networks which realize the various signals resresented by the columns in B and C sets are synthesized separately for each such column. This is often the case in many applications, since it makes the design and trouble-shooting easier (a fault in any part of such a network affects only one gating signal). It is, however, possible that in certain cases a saving could be obtained by designing only one multiple output network for all the gating signals. It must of course be realized that the minimum cost bound as computed by the method given above is not necessarily realizable so that in some cases there may exist no solution which meets that bound. However, this bound does give a fair indication of how good any given solution is, and in those cases where it is met, it guarantees that we have indeed an optimum solution.

Acknowledgment

Part of the research presented in this paper was done while both authors were employed by the Bell Telephone Laboratories in Whippany, N. J., and the remainder of the research was supported by the Higgins Funds at the Department of Electrical Engineering of Princeton University.

References

1. E. J. McCluskey Jr., "Minimization of Boolean Functions," Bell Sys. Tech. J., vol. 35, pp. 1417-1444, November 1956.
2. S. H. Caldwell, "Switching Circuits and Logical Design," John Wiley & Sons, New York, N. Y., Chapter 5, 1958.
3. E. Hirschhorn, "Simplification of a Class of Boolean Functions," Journal of the A.C.M., vol. 5, pp. 67-75, January 1958.
4. E. J. Schubert, "Simultaneous Logical Equations (Matrix Logic IV)," AIEE Paper No. 59 - 1192, to be published.
5. E. J. McCluskey Jr., and S. H. Unger, "A Note on the Number of Internal Assignments for Sequential Switching Circuits," I.R.E. Trans. on E. C., vol. EC - 8, December 1959.
6. P. R. Halmos and H. E. Vaughan, "The Marriage Problem," Amer. J. Math., vol. 72, pp. 214-215, 1950.
7. H. W. Kuhn, "The Hungarian Method for the Assignment Problem," Naval Research Logistics Quarterly, vol. 2, Nos.1 and 2, pp.83-97, 1955.
8. M. M. Flood, "The Traveling Salesman Problem," Operations Research, vol. 4, pp.61-75, 1956.

---

[4] One should make t as large as possible.

[5] In all three of the above examples it has been assumed for the sake of simplicity that $N_{i,\phi} = 0$.

A BUILT-IN TABLE LOOKUP ARITHMETIC UNIT

R. C. Jackson, W. H. Rhodes, Jr., W. D. Winger
Data Systems Division, Poughkeepsie, N. Y.

and J. G. Brenza
Advanced Systems Development Division, Peekskill, N. Y.

International Business Machines Corporation

Summary

A novel approach to arithmetic operations in digital computers is described which requires fewer stages of logic and fewer components than conventional adder circuitry. This is accomplished by using the decimal digits in an arithmetic field to complete memory addresses which in turn are used to reference a set of answers prestored in magnetic core storage. Multiple non-contiguous memory addresses stored in the machine hardware then reference the various digit positions of the sum, difference, or partial product at the proper time in the arithmetic cycle to develop the result. The accuracy of all results is easily verified by simply performing an automatic validity check on the data paths within the computer system.

The machine operator need not concern himself with the "hardware" method used within the computer, since the arithmetic commands commonly found in digital computers(Add, Subtract, etc.)are available in the instruction set for the system shown; table lookup circuitry does the work.

Introduction

Arithmetic operations (addition, subtraction, and multiplication) in the computer system described in this paper are performed without the use of conventional adder circuitry as found in most stored program computers. This is done by making reference at the proper time in the arithmetic cycle to a set of answers prestored in an assigned table area in magnetic core storage. Using this method, arithmetic results are obtained with considerably fewer components and fewer stages of logic than are found in conventional adder circuitry. Furthermore, a completely checked result is obtained, since a sum, difference, or product is created solely by routing the proper data onto the data paths of the machine; here it can undergo a parity check, character by character.

Before a detailed description of arithmetic operations can be given, it will be necessary to review a few design features of this system which make table lookup arithmetic feasible. A partial data flow schematic is shown in Figure 1. (For reasons of clarity, only required components are included; input/output, timing, and control are implied.)

The main storage medium is a three-dimensional magnetic core array, 12 planes deep and containing 20,000 decimal digits. It is addressed serially by digit from a single transistorized 5-digit Memory Address Register (MAR). The array is driven from two 10 x 10 matrix switches which select one "column" of 12 cores to be read from memory to the transistorized Memory Buffer Register (MBR). Since digits in this system consist of 6 bits (4 data bits, 1 parity check bit, and 1 flag bit), the state of these 12 cores read from memory by one read command is placed in MBR as two digits. (If the digit at memory location XXXXN, where N is even, is addressed for read-out, the digit at memory location XXXXN + 1 will be read out as well; if the digit at memory location XXXXN + 1 is addressed for read-out, the digit at memory location XXXXN will be read out as well.) This two-sequential-digit output from memory using only one address is of great importence in multiplication. Its application will be seen later. When the two digits have been received by the MBR, further selection of the digit is made, based on the units position of the address in MAR, and it is ascertained which of these digits is to be sent to the one-

digit transistorized Memory Data Register (MDR). The digit may then be sent from the MDR to other machine locations.

Three-hundred-twenty memory locations are assigned as a "table area": a 20-digit area from memory locations 00080 through 00099 to receive the product in multiplication; a 100-digit area from memory locations 00300 through 00399 for storage of an addition table for use by the computer during the execution of all arithmetic instructions (Figure 3); and a 200-digit area from memory locations 00100 through 00299 for storage of a multiplication table, for use by the computer during the execution of the multiply instruction (Figure 7). All of these "table area" locations are in addressable memory.

Table lookup arithmetic requires that multiple noncontiguous memory addresses be available during the operation so that factors can be obtained and answers can be written back into memory at the various digit positions of the partial product (or sum or difference) at the proper times. This is accomplished through the use of a two-dimensional core array called Memory Address Register Storage (MARS, Figure 1) which supplies the Memory Address Register (MAR) with the proper address required in each machine cycle. The eight 5-digit registers in MARS which are available for MAR to use in addressing memory are Instruction Address Registers 1 and 2 (IR-1 and IR-2), Operand Address Registers 1, 2, and 3 (OR-1, OR-2, and OR-3), and Product Address Registers 1, 2, and 3 (PR-1, PR-2, and PR-3). Their use will be discussed later. In any one machine cycle, MAR is controlled to accept an address from one of the MARS registers. During the machine cycle, the 5-digit address in MAR is returned to a MARS register or registers by way of the Increment Switch. In passing through the increment switch the address from MAR may be incremented by 1 or 2, or decremented by 1, or may bypass the switch and remain unchanged. The MARS register to which the MAR address is returned may be the MARS register from which the address originally came, or a different MARS register, or the original and a different MARS register. The control for these operations is determined by the operation to be performed.

The remaining hardware required for describing table lookup arithmetic consists of a one-digit Multiplier Register, a two-digit Digit Register, two translators called True/Complement Switches 1 and 2, two Plus 1 Switches, and a Doubler. The registers serve as temporary storage devices for digits; the true/complement switches provide either the true numeric value of a digit supplied to them or its nines complement; the plus 1 switches increment by one the output of true/complement switch 1 and/or the tens digit of the doubler output; and the doubler does just what its name indicates, i. e., it supplies a two-digit output twice the value of its one-digit input. Table lookup arithmetic does not require the use of counters, shift registers, word registers, accumulator registers, or adders.

Along with the above "hardware" description, it should be noted that the instruction set of this computer system consists of two-address, 12-digit instructions, so that the locations of both fields involved in an arithmetic operation are specified in one instruction. During instruction interpretation time (I-time), these two 5-digit addresses are loaded into OR-2 and OR-1 for use during instruction execution time (E-time). All arithmetic operations are performed serially, one digit at a time, on data words (fields) which are variable in length. Each field is defined by placing a flag bit over the high order digit of the field. The units digit of a field may also have a flag bit, but this is used to denote that the field is negative; if the flag on the units digit is missing, the field is considered positive. (Note: Since flags have this dual function in arithmetic fields, the minimum length of a field is two digits. No restriction exists on the maximum length, except what is practical for the stored program being executed.)

Addition

Let us now follow the logic of the addition of two positive fields as it would occur in a typical Add instruction. The general approach taken by logical circuitry within the machine is the use of corresponding digits of the addend and augend to form a unique address in core storage at which is stored the sum of the two digits selected. This sum then replaces the augend digit in memory, and the procedure is repeated using the next higher order digits of the two fields.

Assume that I-time for the instruction has been completed so that the 5-digit memory addresses of the addend and augend have been stored in MARS registers OR-1 and OR-2, respectively. The address of the augend stored in OR-2 has also been stored in OR-3, should recomplement be required. In Figure 1, it can be seen that in the first machine cycle the address contained in OR-1 is read out of MARS

to MAR. Using this MAR address, the digit in the units position of the addend is read from memory to MBR and thence to MDR. From here it is placed on the data bus and stored temporarily in the digit register (units). Simultaneously, the address in MAR is passed through the increment switch, decremented by 1, and replaced in OR-1 as 09174 (i. e., the address of the tens digit of the addend).

At the beginning of the second machine cycle the address contained in OR-2 is read out to MAR. The digit at this address (the units position of the augend) is then read from memory to the MBR and then the MDR. The address from MAR bypasses the increment switch and is replaced unchanged in OR-2. After memory has been read in this cycle (shown in Figure 2), MAR is reset and the augend digit from MDR passes through true/complement (T/C) switch 2. The true output of T/C switch 2 is stored in the tens position of MAR. Simultaneously, the addend digit previously stored in the digit register (units) is passed through T/C switch 1 and the true output, bypassing the Plus 1 Switch, is placed in the units position of MAR. The remaining positions of MAR are filled with fixed digits; zeros are forced into the ten-thousands and thousands positions, and a "3" is forced into the hundreds position.

This address is then used to read a digit from the addition table stored in memory (solid coordinates in Figure 3) to the MBR and then the MDR during the third machine cycle (Figure 4). This is the units digit of the sum.

The fourth machine cycle finds OR-2 again read out of MARS to MAR. The augend digit at the corresponding memory location is replaced by the units digit of the sum which was stored in MDR during the previous machine cycle. The address from MAR simultaneously passes through the increment switch, is decremented by 1, and is replaced in OR-2 as 14623 (i. e., the address of the tens digit of the augend). One add cycle has now been completed and one digit of the sum developed.

The procedure is now repeated in the next four machine cycles, developing the sum of the tens digit of the addend and augend (the addresses of these digits are now contained by OR-1 and OR-2, respectively). A carry occurs using this pair of digits which is detected and handled in the following manner: After the proper digits of the addend and augend are routed to MAR, the manufactured address is used to read the second digit of the sum (flag bit included) from the addition table in memory (dashed coordinates in Figure 3) to MDR. The flag bit stored with this digit denotes a carry resulting from the addition of the two digits. (Detection of this flag on read-out sets a logical path to provide for the inclusion of the carry when the hundreds digits are added.) The sum digit from MDR (without the flag) is stored back in memory per OR-2, replacing the tens digit of the augend. At the end of the second add cycle, the MARS addresses are:

MAR	14623
MARS OR-1	09173
MARS OR-2	14622
MARS OR-3	14624

The last digit of the sum is developed in the same manner (Figure 5), except that the flag detection from the previous add cycle causes the output of the digit register (units) to pass through T/C switch 1 and the plus 1 switch to compensate for the previous carry. The address in MAR used to obtain the sum digit is, therefore, increased by one over that which would normally be used. The sum digit is then placed back in memory, replacing the last digit of the augend. The fact that both fields contain a flag bit in the digit position used in this add cycle terminates the add operation, and the computer starts the I-time of the next instruction in the sequence.

Addition of two fields requires 4A machine cycles where A equals the number of digits in the augend.

Addition of two fields with unlike signs and subtraction are carried out by complement addition, but the procedure is only slightly changed from that previously described. If, for example, the augend field had been negative in developing the previous sum, the output of T/C switch 2 would have been the nines complement of its input. The table lookup is then performed as before, using the true value of the addend and the complemented value of the augend for the table lookup address. (Note: To get the tens complement of the field, a carry in is assumed on the first add cycle which is implemented by having the plus 1 switch increment the output of T/C switch 1 by one before the digit gets to MAR.)

Recomplement is performed whenever the signs of the augend and addend are initially different and the sign of the sum is negative. After the sum is completed and it is determined that recomplement is required, the reset address of the units digit of the sum

is transferred from OR-3 to OR-2 and another pass is made through the sum, complementing each digit. This, too, is performed using the table lookup procedure. Each digit of the sum passes through T/C switch 2 whose output, in this case, is the complement of the sum digit input. This output digit is routed to the tens position of MAR and a zero is forced in the units position; the table lookup performed using this address causes memory to yield the complement of the original sum digit to MDR. It is then placed back in memory in the proper sum digit position. The procedure is repeated for each of the sum digits until the recomplemented field is completely developed. Recomplement requires four machine cycles per digit of the sum.

Multiplication

Multiplication utilizes six of the eight MARS registers to store addresses and makes reference to a multiplication table as well as the addition table. The product is built up serially in much the same manner as a third-grader performs his multiplication. Logical circuitry within the machine sequences the use of single digits of the multiplier and multiplicand to form unique core storage addresses, at which are stored the partial products. After a partial product is added (through table lookup) to the product area, the logic determines the next pair of multiplier and multiplicand digits to be used. The process continues building up the product in the product area until each digit of the multiplier has been used with each digit of the multiplicand and the product has been developed.

During I-time in the Multiply instruction, OR-1 is loaded with the address of the multiplier, and OR-2 and OR-3 are loaded with the address of the multiplicand. The product area in memory (locations 00080 through 00099) is cleared to zeros, and address 00099, the units digit of the product, is placed in PR-1.

During the first E-time cycle, the units digit of the multiplier is read from memory per OR-1 to the multiplier register, and OR-1 is reloaded with the address of the tens digit of the multiplier; it remains inactive until each digit of the multiplicand has been used.

On the second machine cycle, the units digit of the multiplicand is read from memory per OR-3 to MDR and thence through T/C switch 2 to the tens position of MAR. OR-3 is simultaneously decremented by one. Figure 6 shows the status of MARS at this point. During this same cycle, the digit from the multiplier register is doubled and sent to the hundreds and units positions of MAR; the hundreds position is increased by one before reaching MAR by passing through the plus 1 switch.

The manufactured address in MAR is then used (third cycle) to read from memory to MBR-Even and MBR-Odd the two-digit product of 6 and 9 (solid coordinates in Figure 7).

[Note: The advantage of having two sequential digits read from memory for each address used is obvious here. It should also be noted that this product is stored in memory with the digits transposed. Since the multiplier digit is always doubled, the address in MARwill always be even, permitting the addressed digit to pass to MBR-Even, to MDR, and thence to the units position of the digit register. The digit from MBR-Odd is transmitted on a separate data bus to the digit register (tens), where the product then appears in proper sequence.]

PR-1 is read out to MAR and the units digit of the product area (initially zero) is transferred to MDR during the fourth cycle. In parallel the MAR address is placed unchanged in PR-2, and also into PR-1 decremented by one (Figure 8). (PR-1 will not be used again until all digits of the multiplicand have been used. It functions then as a reset address for the proper positioning in the product area. Its function will be demonstrated later.) The digit in MDR and the product units digit from the digit register (units) are then used to obtain a sum address in the usual way.

On the fifth cycle the manufactured address in MAR is used to read the sum of 0 and 4 from the addition table in memory to MDR, as described earlier.

On the sixth cycle PR-2 is read out to MAR, then decremented by one and placed back into PR-2 and PR-3 simultaneously. In parallel, the new partial product low order digit in MDR is read back into memory per the MAR address.

PR-3 is read to MAR and placed back in PR-3 unchanged on the seventh cycle, so that the tens digit of the product area can be read from memory to MDR and routed to MAR Tens to help create a sum address as before (Figure 9). The product tens digit from the digit

register (tens) completes the variable portion of the address.

Reading from memory using this address in the eighth cycle obtains the sum of 0 and 5 from the addition table to MDR.

The last cycle (ninth) in this sequence causes the digit in MDR to be placed back in memory using the address from PR-3. The two-digit product of the units digits of the multiplier and multiplicand is now located in the proper position in the product area.

[Note: The interaction of PR-2 and PR-3 is not obvious from this example. Each time a two-digit product of a multiplier digit and multiplicand digit is created, it is necessary to add these two digits serially to the partial product previously developed. PR-2 is used to add the units digit to the partial product, and it then steps to the address of the tens digit. Any carries which may occur from the addition of the units digits can be propagated by increasing the address used in the table lookup of the addition of the tens digits. Carries resulting from the addition of the tens digits, however, may have to be propagated several times before the next sequence is performed. PR-3 is used to accomplish this when necessary (not required in this example) and PR-2 holds its place as a reset address.]

The tens digit of the multiplicand is obtained on the tenth cycle, using the address from OR-3 which is read to MAR, decremented by one, and placed back in OR-3.* The tens digit travels from MDR to MAR Tens as before, and simultaneously the output of the multiplier register and doubler is placed in MAR Hundreds and MAR Units, completing the address (Figure 10). The flag bit on the multiplicand digit is sensed on this cycle, and a flip-flop is set to start the procedure over again with the units digit of the multiplicand when this sequence is completed.

Again, reference is made to the multiplication table in memory (eleventh cycle) using this manufactured address (dashed coordinates in Figure 7). The two-digit product of 4 and 9, with digits transposed, is read to MBR-Even and MBR-Odd and thence to the digit register as previously described.

PR-2 is used in the twelfth cycle to obtain the previously developed partial product tens digit from memory. This digit and the product units digit are used to form a sum address as before (Figure 11).

An add cycle (thirteenth) is again carried out, using this manufactured address to refer to the add table. The one-digit sum is read to MDR from memory. Since the sum of the two digits is greater than 9, a flag (carry indication) is detected which sets up the gating to include the carry on the next add cycle.

Using the address in PR-2, the new partial product tens digit (without its flag) is read back into memory (fourteenth cycle). In parallel with the reading of the digit back into memory, the address is returned to PR-2 and PR-3, decremented by one. At the end of this cycle, the MARS addresses are:

OR-1	00422
OR-2	12047
OR-3	12045
PR-1	00098
PR-2	00097
PR-3	00097

It is then necessary to read out the hundreds digit of the partial product (fifteenth cycle); PR-3 is used. It is placed in MAR and replaced unchanged in PR-3. This digit and the product tens digit, from the digit register (tens), form a new sum address except that gating through the plus 1 switch provides the effect of the carry from the previous add cycle.

The sixteenth cycle of this sequence is then executed, using the manufactured address in MAR (00304) with memory yielding the sum digit to MDR from the addition table.

On the last cycle of this sequence, PR-3 is used to place the new hundreds digit of the partial product back in memory. The partial product is now complete, using both digits of the multiplicand and the units digit of the multiplier.

We now return to the first cycle, where a new multiplier digit is required and OR-1 is

---

* For purposes of clarity in explaining the example chosen, development of the next partial product using the tens digit of the multiplicand is shown as additional unique cycles. Actually, the logic of the multiply timing ring would cause a return to the second cycle above except that (1) the addresses now in MARS would be utilized, and (2) PR-2 would be used in the fourth cycle instead of PR-1.

used to obtain it from memory (Figure 12). The flag is detected on the digit, which sets flip-flops to terminate the execution of the instruction when the flag is next detected in the multiplicand. This new digit is stored in the multiplier register, and OR-3 is reset equal to the content of OR-2, providing the return path to the units digit of the multiplicand.

From this point on the multiplication proceeds, developing the remaining digits of the product in a manner similar to that previously described. It should be noted that PR-1 contains the proper address at which the next digit of the partial product will be developed. PR-2, therefore, will be reset to this address at the proper time and PR-1 decremented by one so that a new reset address is then available. It will not be needed in the development of this product, however, since the complete product will be developed before it is again required.

On the last memory cycle in the development of the complete product, a flag bit is placed over the high order digit of the product. No restriction exists on the maximum length of the product except that only 20 digits of the product area are cleared to zero in the execution of the multiply instruction. Should the product exceed 20 digits, additional programming steps are required to clear to zero the appropriate number of digits in excess of 20 before multiplying.

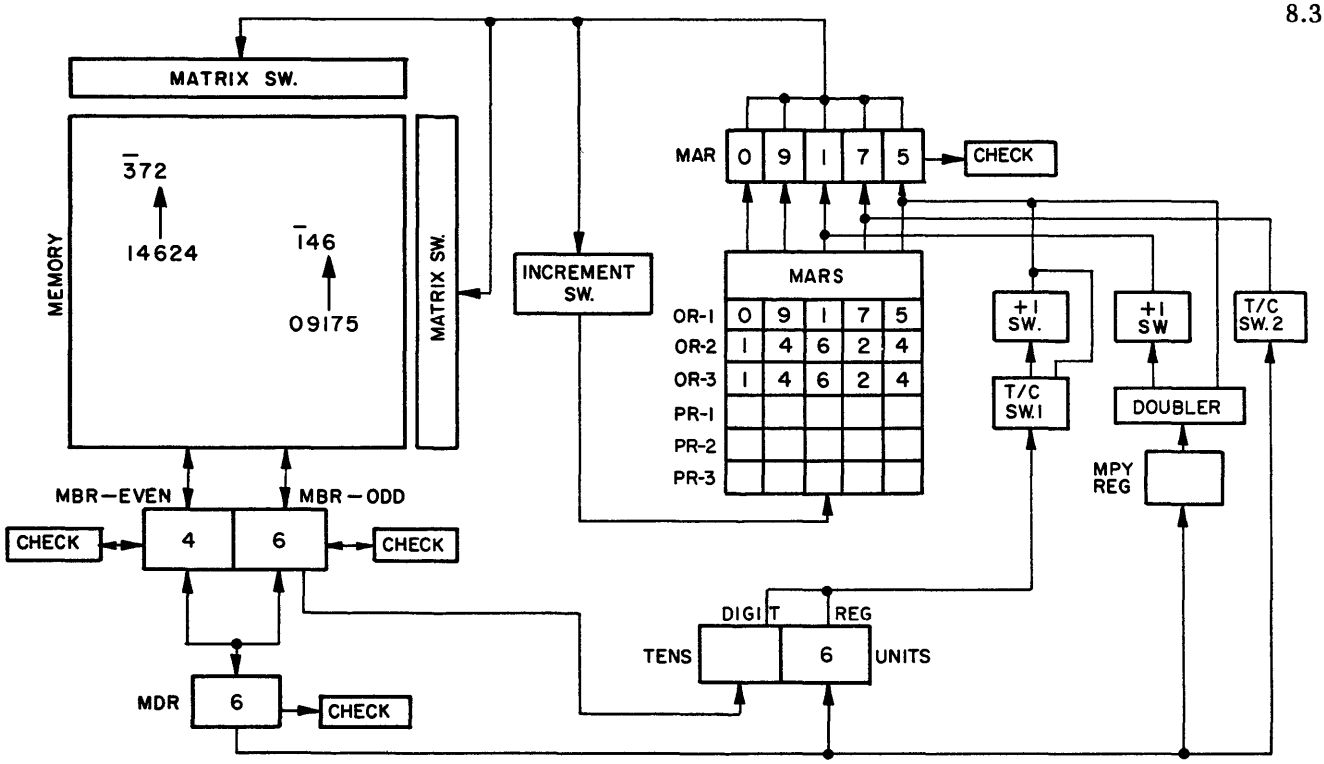
Multiplication of two fields requires approximately $Q(8.4P + 2)$ machine cycles, where Q and P equal the number of digits in the multiplier and multiplicand, respectively.

Conclusion

The system presented above describes the arithmetic unit incorporated in the first engineering model of the IBM 1620 Data Processing System. A refinement of this table lookup arithmetic is included in the production version of the 1620. No division hardware was provided because the use ratio of such an instruction to others was found to be small. Since other features of the 1620 provided convenient subroutine programming, division was included as part of the general program package of mathematical subroutines (sin x, log x, e^x, etc.) included with the system.

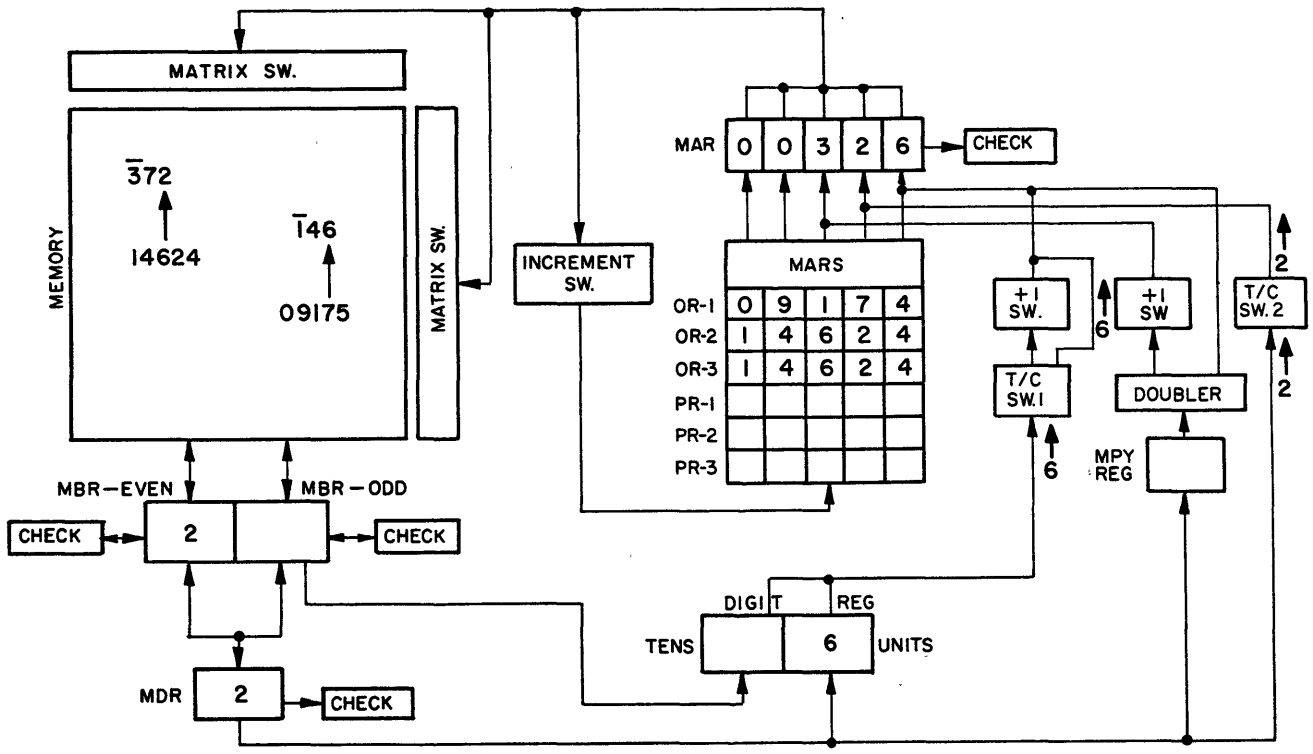
While table lookup arithmetic is employed in the 1620, the arithmetic operations described above are applicable to any system having the necessary organization. It is important to note that the speed of the arithmetic is directly proportional to the machine cycle. Also, the machine organization described in this paper lends itself to a table lookup divide instruction using the existing components; only economic justification and the divide timing ring are required.

Elimination of the expensive adder circuitry heretofore required was made practical by using many of the components needed by the system for other operations. The low cost, high performance core storage, for example, provided the table area and the speed required for efficient arithmetic operations. The MARS array and registers, too, were required for other uses by the instruction set. That being the case, a few extra components and a logical sequence provided the same final result as a conventional adder, but at a reduced cost.



372
+146

Figure 1
DATA FLOW - ADDEND UNITS DIGIT SELECTION



372
+146

Figure 2
DATA FLOW - SUM ADDRESS GENERATION

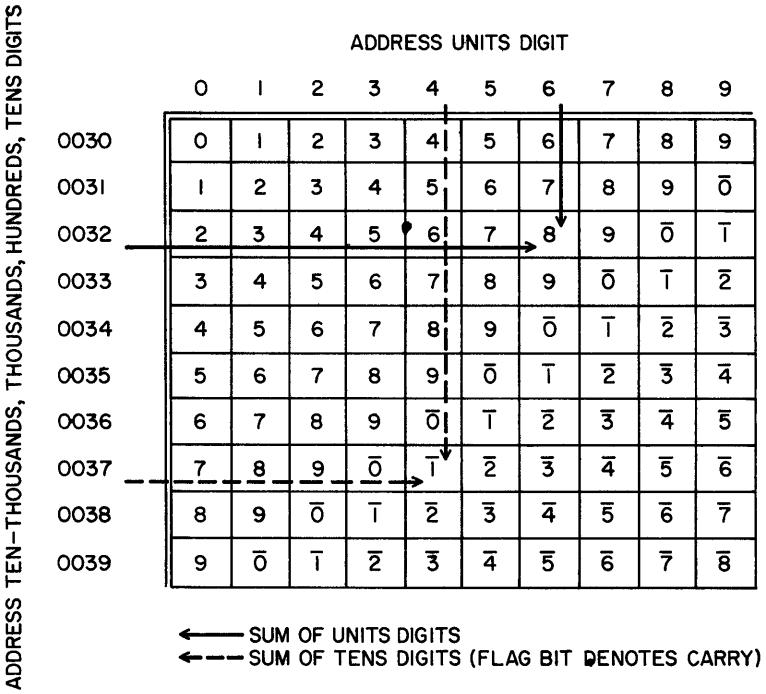
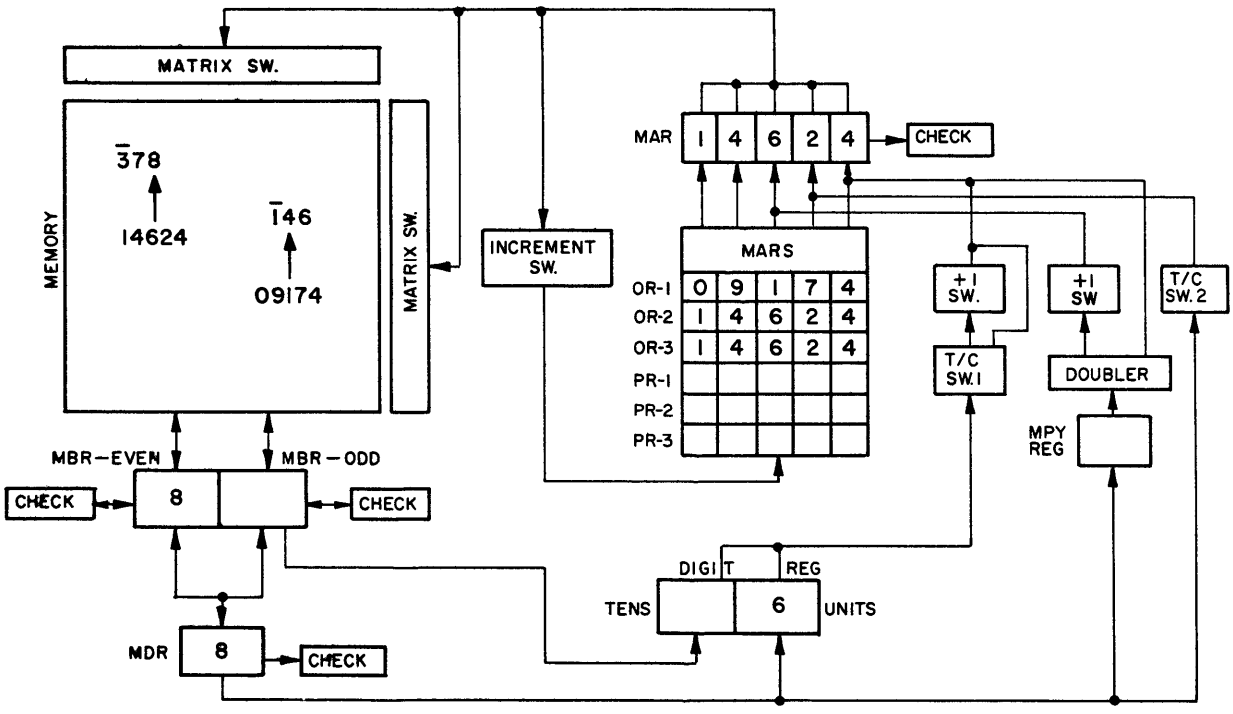
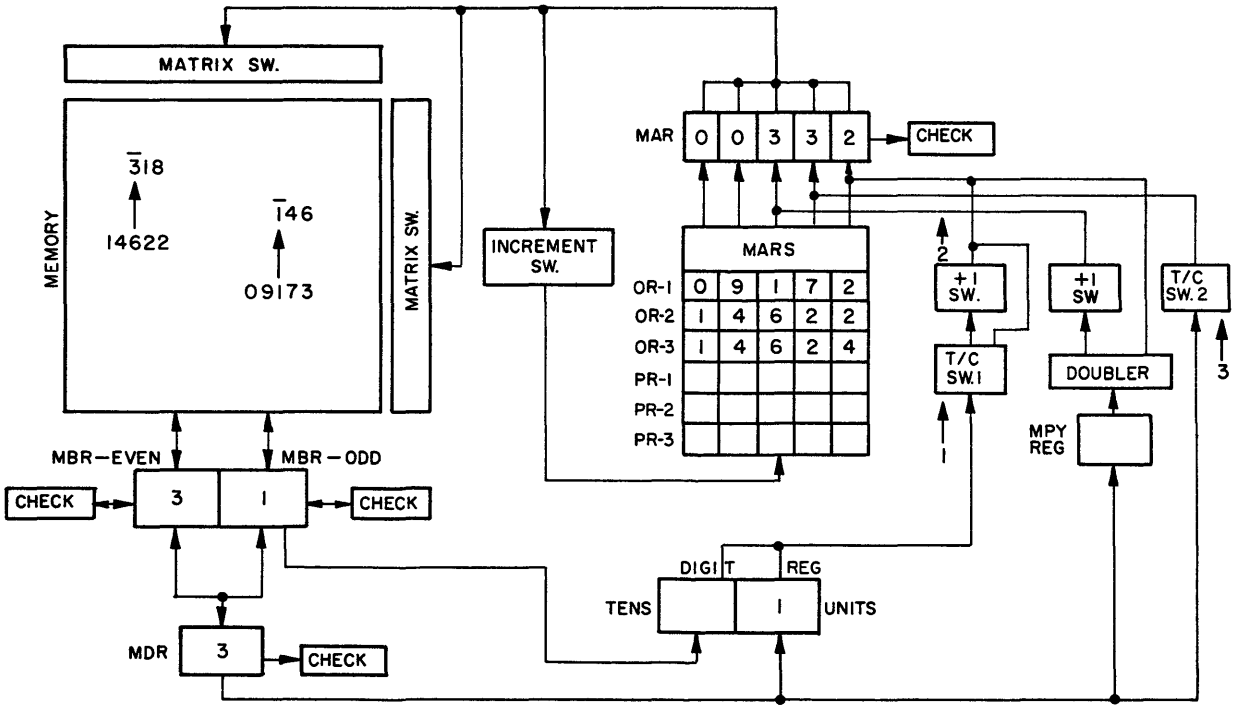


Figure 3
ADDITION TABLE (FOR USE IN EXECUTION OF ALL ARITHMETIC INSTRUCTIONS)



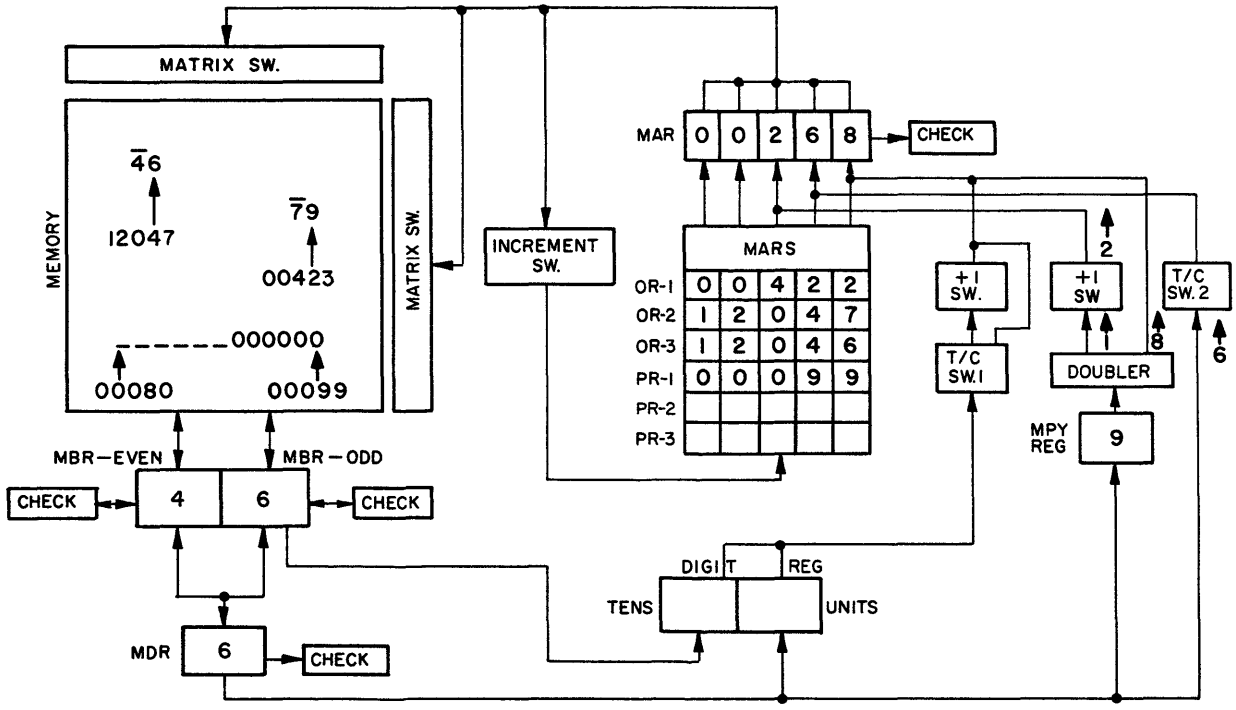
372
+146
8

Figure 4
DATA FLOW - STORAGE OF UNITS DIGITS SUM



372
+146
18

Figure 5
DATA FLOW - SUM ADDRESS GENERATION, WITH CARRY IN



46
X 79

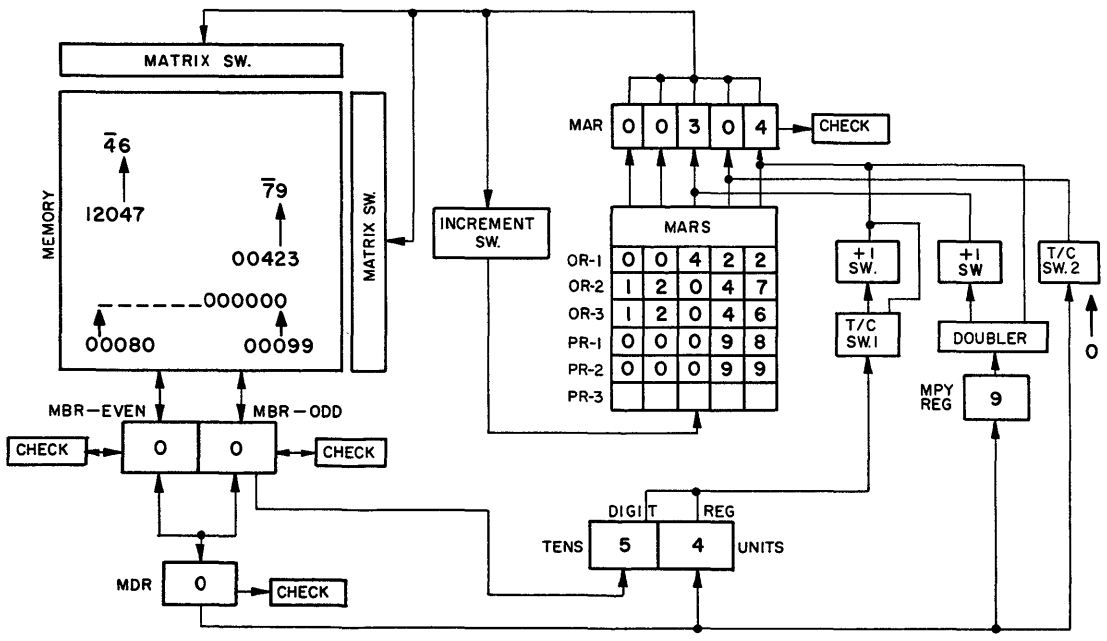
Figure 6
PRODUCT ADDRESS GENERATION (MULTIPLIER
UNITS DIGIT TIMES MULTIPLICAND UNITS DIGIT)

ADDRESS UNITS DIGIT

ADDRESS TEN-THOUSANDS, THOUSANDS, HUNDREDS, TENS DIGITS	0	1	2	3	4	5	6	7	8	9
0010	0	0	0	0	0	0	0	0	0	0
0011	0	0	1	0	2	0	3	0	4	0
0012	0	0	2	0	4	0	6	0	8	0
0013	0	0	3	0	6	0	9	0	2	1
0014	0	0	4	0	8	0	2	1	6	1
0015	0	0	5	0	0	1	5	1	0	2
0016	0	0	6	0	2	1	8	1	4	2
0017	0	0	7	0	4	1	1	2	8	2
0018	0	0	8	0	6	1	4	2	2	3
0019	0	0	9	0	8	1	7	2	6	3
0020	0	0	0	0	0	0	0	0	0	0
0021	5	0	6	0	7	0	8	0	9	0
0022	0	1	2	1	4	1	6	1	8	1
0023	5	1	8	1	1	2	4	2	7	2
0024	0	2	4	2	8	2	2	3	6	3
0025	5	2	0	3	5	3	0	4	5	4
0026	0	3	6	3	2	4	8	4	4	5
0027	5	3	2	4	9	4	6	5	3	6
0028	0	4	8	4	6	5	4	6	2	7
0029	5	4	4	5	3	6	2	7	1	8

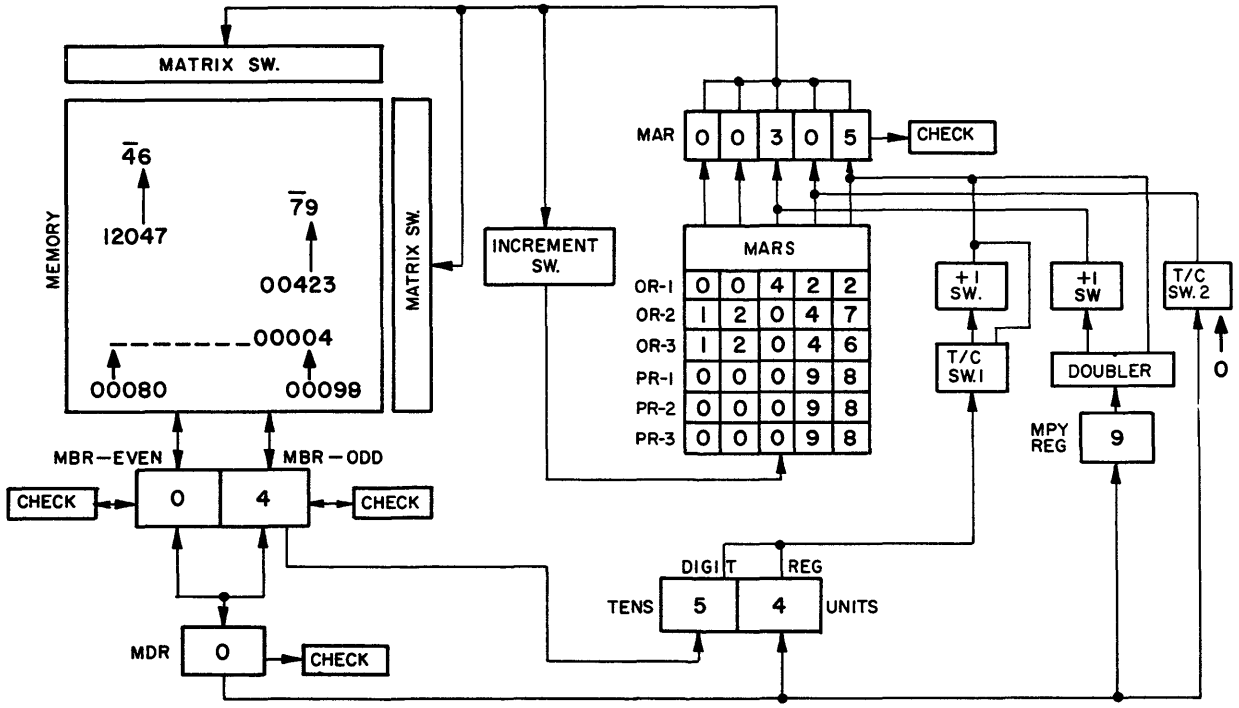
— PRODUCT OF UNITS DIGITS
-- PRODUCT OF MULTIPLIER UNITS AND MULTIPLICAND TENS DIGITS

Figure 7
MULTIPLICATION TABLE (FOR USE IN EXECUTION OF MULTIPLY INSTRUCTION)



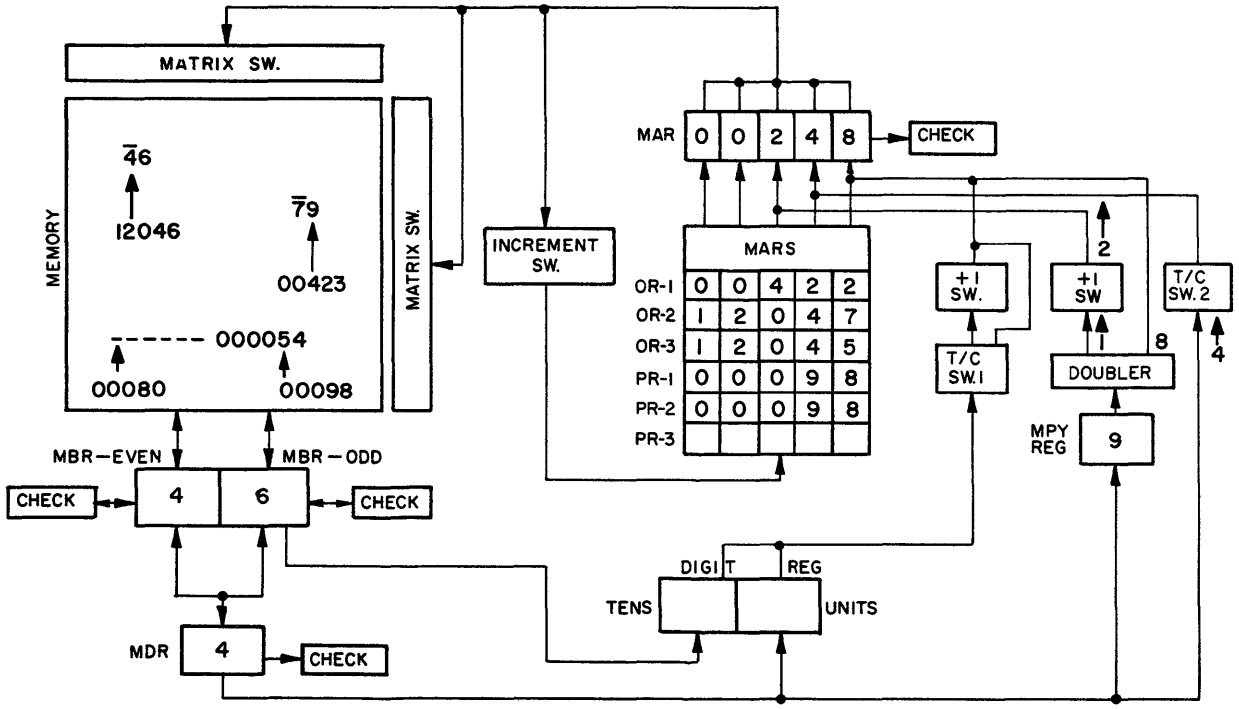
$\times \frac{46}{79}$

Figure 8
SUM ADDRESS GENERATION
(PRODUCT UNITS DIGIT PLUS PARTIAL PRODUCT UNITS DIGIT)



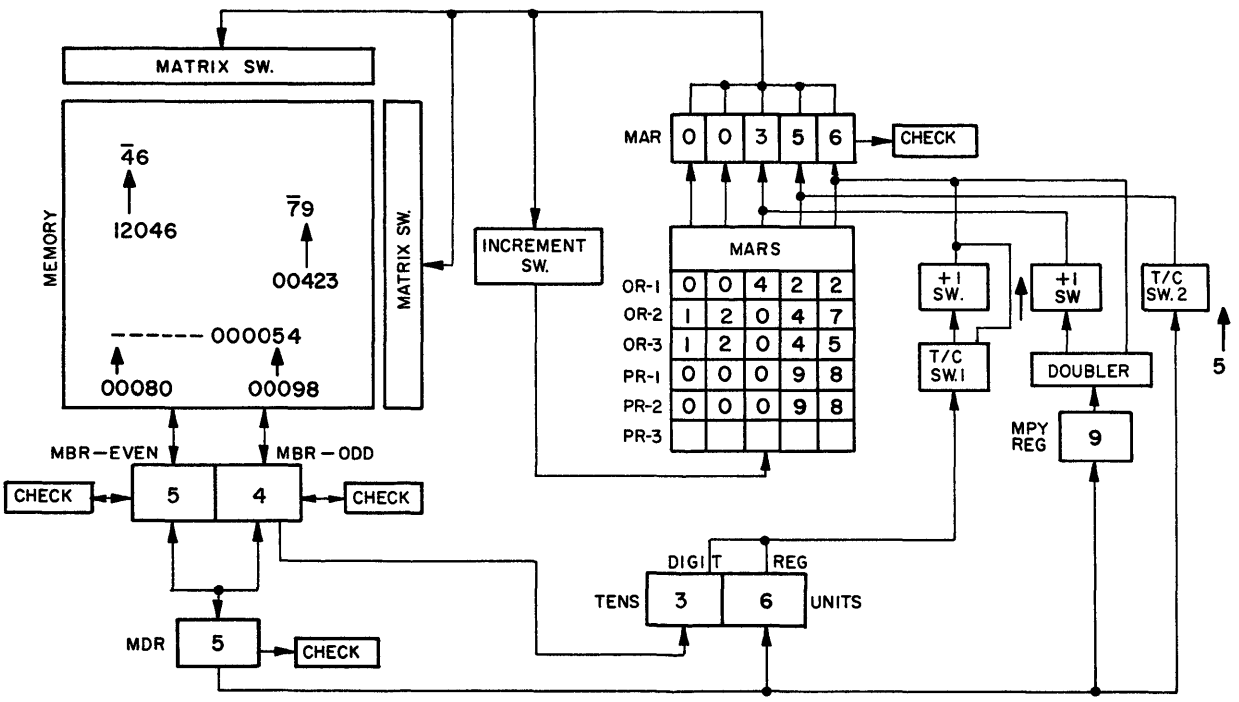
$$\begin{array}{r} 46 \\ \times\ 79 \\ \hline 54 \end{array}$$

Figure 9
SUM ADDRESS GENERATION
(PRODUCT TENS DIGIT PLUS PARTIAL PRODUCT TENS DIGIT)



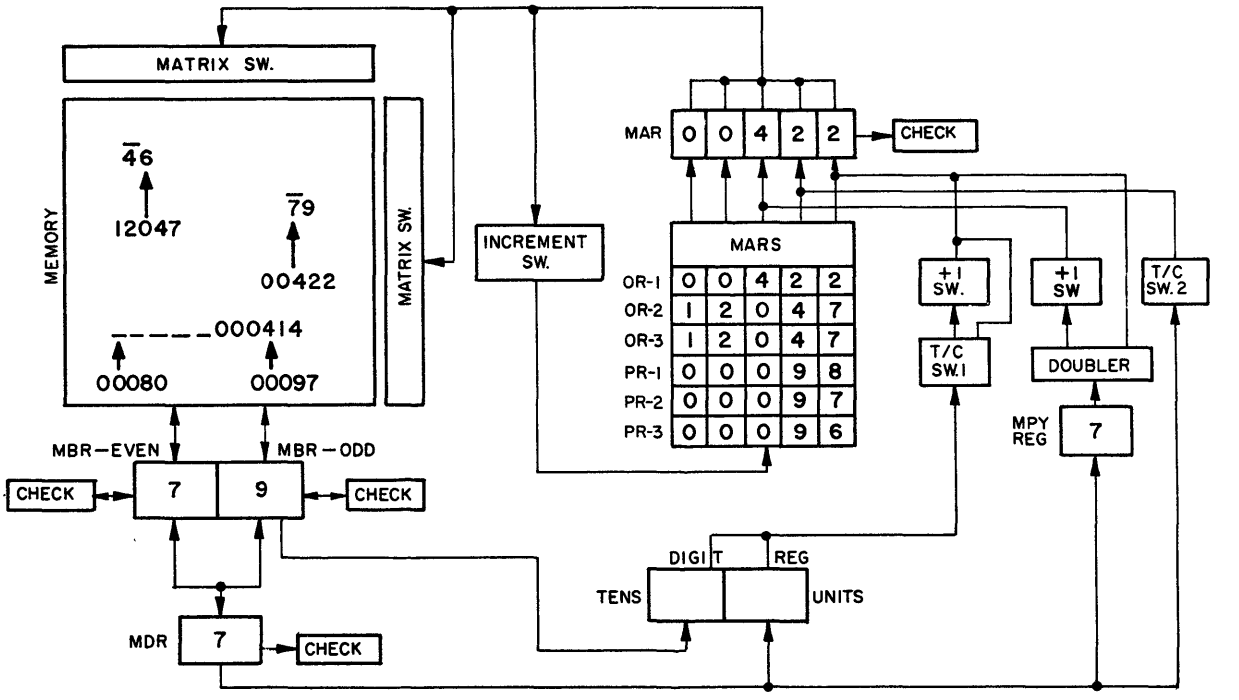
$$\begin{array}{r} 46 \\ \times\ 79 \\ \hline 54 \end{array}$$

Figure 10
DATA FLOW - PRODUCT ADDRESS GENERATION
(MULTIPLIER UNITS DIGIT TIMES MULTIPLICAND TENS DIGIT)



46
x 79
54

Figure 11
DATA FLOW - SUM ADDRESS GENERATION
(PRODUCT UNITS DIGIT PLUS PREVIOUS PARTIAL PRODUCT TENS DIGIT)



46
x 79
54
36
42
28
3634

PARTIAL PRODUCTS
54
414
834
3634

Figure 12
DATA FLOW - MULTIPLIER DIGIT SELECTION

ON MICROELECTRONIC COMPONENTS, INTERCONNECTIONS, AND SYSTEM FABRICATION

Kenneth R. Shoulders
Stanford Research Institute
Menlo Park, California

Summary--Microelectronic data processing systems are analyzed and various requirements for components are considered. The rapid reduction in transmission line cross section upon scaling down causes increased losses in microelectronic systems thus giving rise to the need for high impedance components for non-cryogenic applications. A new component is proposed that seems particularly suited for microelectronic system requirements and fabrication methods. This component is based upon the quantum mechanical tunneling of electrons into vacuum; has an estimated switching time of 10^{-10} seconds; promises immunity to temperature variations; and seems adaptable to self-forming manufacturing methods giving component uniformity. A method of electron beam activitated micromachining for film materials is presented in which a thin chemically resistant film is formed with an electron beam to selectively protect the film being machined during a subsequent molecular beam etching. This high speed process has resolution capabilities of several hundred angstrom units; can process electronically clean materials with minimum contamination; and may ultimately be suited for the economical production of one cubic inch data processing systems having 10^{11} active components.

The Over-All Requirement

We want to build electronic data processing systems that have the complexity of human neural networks but are capable of operating with electronic speed. These machines should be capable of solving very complex problems such as those that arise in nonlinear systems, magnetohydrodynamics, pattern recognition from the real world, self-organization and learning, and should in general, be very useful assistants in our society.

Having once obtained such a machine, the need will invariably arise to make it highly portable so that it can cope with problems where they exist instead of depending on problems being brought to it. Everyone should have such a personal assistant.

The realization of such a machine requires an extremely high degree of organization of matter and may not be economically permissible unless a high speed electronic construction process is used for specifying the end product. Admittedly, this is not the next generation of equipment to be expected. Our aims here are to try to achieve what seems possible with electronics and not base our work upon an incremental advance from present devices.

There has been a tendency lately to conceal the need for high resolution processing methods through the use of such terms as "molectronics," or by talk of distributing this-or-that. There always seems to be a discrete device, even if composed of distributed parts, which serves our requirements in a data processing machine. It seems likely that this will continue for some time.

Our problem then is how to handle material with enough resolution to fabricate 10^{11} devices in a size that is portable and then to find an interesting configuration for these materials so as to give a desired electronic result.

But before we start building things at random, we might do well to look at some of the properties of electrical devices in the micro-world, and thus avoid some lost motion by only working on things that are expected to work. We have picked arbitrarily 10^{11} components for our system. Let us see what this implies in terms of physical size for the machine.

The most highly resolved construction process that we have any control over is what we call electron beam activated micromachining. This process has been shown to give some results in the 100-angstrom-unit region, but let us degrade this to 300 angstroms for the moment and see what we can make with it. If an arbitrary component can be made from about 1000 of these 300 angstrom unit dots, as seen from the plan view, then the outside dimensions of the component would be about one micron. The thickness could be expected to be an equivalent value. If three of these one-micron sized areas are used for interconnections for every component, then we have two microns of linear dimension per component and 10^4 per inch. One square inch of area would contain 10^8 elements. If ten layers of components per module were used there would be 10^9 components. One hundred modules, each being 0.010 inch thick for support purposes, would stack to a one-inch cube having 10^{11} components.

Before we go into the insides of this system, let us look at some external problems. If we wish to pick up optical data we find that a one-inch cube made by stacking wafers has just two surfaces available for communication with optical patterns from the outside world. One of these would be expected to receive light images, and the other to generate light patterns for human use.

There will be a noticeable lack of lead wires at the edges of the modules, compared to the number of components on the module. For communicating between modules, wide-band serial techniques could be used for the few wires available at the edge and the surfaces could be used by allowing light coupling between modules. It might be expected that up to 10^4 parallel paths could be achieved in this way without the use of a lens. The bandwidth would seem to depend upon the light

detector, and may fall in the 10-megacycle range.

In changing the size of things so radically--about three orders of magnitude--we will be well advised to look over old components to see if they can become useful in this size range. For example, we may reconsider a mechanical electrostatic relay because now it could have vacuum contacts and operate at low voltages in the 10-megacycle range. We could look again at conventional vacuum tubes and consider the negligible transit time lag and absence of bothersome space charge effects because of the high fields. Thermal devices will be found to have time constants in the microsecond range.

Scaling of Electrical Properties

Not all effects are beneficial when sizes are reduced. One of the most disturbing changes is the increased electrical loss of conventional wires.

The effect of scaling down in size on certain fundamental electronic components can be shown by reviewing the behavior of inductance, capacity, and resistance. The values of inductance and capacity vary directly with their size, while resistance has an inverse behavior. If all dimensions of a standard sized tuned circuit were reduced by 10^4 to make it about a micron on a side then the resonant frequency would be raised by 10^4. The Q at this new frequency has been decreased by 10^4, and would be reduced by 10^8 at the original frequency. A search for new filtering methods is in order. R-C filters, and methods of producing effective inductance could be used, but their stability has never been adequate. This new filter may be the logic of a pattern recognition machine. Serial communication filtering could be handled by sample data techniques coupled with appropriate quantization and logic circuits--in other words, the digital filter. In short, the communication set of the future would become a computer. These techniques would not be thrifty in their use of components and it should be expected that vast numbers of components would be needed.

In addition to an increased loss for tuned circuits, we find that the same mechanisms are at work to cause excessively lossy transmission lines. For example, a 3000-angstrom-diameter line having a length of one inch would have 100,000 ohms DC resistance. The added high frequency loss of the line would be quite severe. A clear dictate for a micro-component wired to its neighbors in the classical way is that it be a high impedance component. In general, electric field operated devices like tubes and transistors have high impedance while magnetic components have low impedance. This high impedance does not necessarily mean long switching times in microelectronics. Consider that the distributed capacity of a one-micron device may be as high as 10^{-15} farad, and if this is associated with a 100,000-ohm resistor a 10^{-10} second time constant results.

Properties of Materials

We should now look into the properties of materials in the sub-micron region. The resistivity of conductors is very near that of bulk materials for a 3000-angstrom-diameter transmission line,[1] but we are encouraged to find that dielectrics can have breakdown strengths several orders of magnitude higher than bulk materials.[2] Breakdown values of 3×10^7 v/cm for a 2000-angstrom film of aluminum oxide is typical. This is primarily because the thickness is near the value for the mean free path of an electron in the dielectric, and an avalanche is not allowed to occur under such conditions and breakdown is forestalled. One should be motivated to look into high field strength electrostatic devices in order to take advantage of this effect.

Some of the mechanical properties of microstructures seem to make things difficult for us. For example, it seems much more difficult to grow a highly ordered single crystal film than a bulk single crystal, especially on an amorphous or micro-polycrystalline substrate that may be required for multi-layer component construction.

Surface tension forces play a very important role in certain sub-micron configurations; these powerful forces try to form our carefully shaped parts into tiny spheres which may not be useful.[3,4]

When the quantity of material in our component is reduced to the level required, extreme precautions must be taken to prevent either self-diffusion or diffusion of foreign materials. These microelectronic components can no longer average out the effects of foreign material.

> "Well, I haven't looked that closely but the thing that does surprise me is that the effects are as big as they seem to be if indeed they are caused by what I suppose." (anonymous contribution to discussion on surface phenomena)

As the size of our device is reduced the surface-to-volume ratio rises and surface recombination effects may become very difficult to cope with.

This increase of surface-to-volume ratio brings with it a fantastic cooling mechanism for individual micron-sized components. It has been found in relay contact research, and particularly by x-ray micro techniques[5], that the power density into a one-micron area can be 10^8 watts per square centimeter continuously without material decomposition. Of course, it would not be expected that many components in the same area could operate at this level without catastrophic effects.

Tunnel Effect Vacuum Tube

We propose to take advantage of some of the effects found in the micro-world and incorporate them into a component based upon the quantum mechanical tunneling of electrons into vacuum. Figure 1 shows one configuration that may prove interesting.

The operation of this device may be divided into two parts to help our understanding of it--namely obtaining electron emission, and using these emitted electrons.

The cathode properties have been investigated by many workers and there is general agreement that the current density can be 10^7 amps per square centimeter with only very small space charge effects.[6] This is due to the high field

strength of a few times 10^7 volts per centimeter. The current from this source can be varied over seven powers of 10 by a change in field of 2 to 1 at the cathode, implying high gain possibilities. Tests at 10,000 megacycles have verified that there is no detectable deviation from the DC emission characteristics.[7]

The variation of grid voltage would cause the necessary field change at the cathode. A measured velocity distribution of 0.14 electron volts shows that this is not a noisy source.[8]

Our intentions are to form an array of small tips superimposed on the cathode shown in Fig. 1. These tips should be a few hundred angstrom units in diameter in order to obtain emission below 100 volts when applied to the screen grid or anode. The cathode shown in Fig. 1 is nominally 3000 angstrom units wide.

Electron optical considerations dictate how the emitted electrons are to be used. A positive control grid would normally be used, but this would not draw appreciable current because it is located away from the electron path between cathode and anode. Negative grids are possible if they are smooth enough or have high work function so as to prevent emission. It is desirable to collect the electrons at a low plate potential, thus avoiding unnecessary heating. Potentials as low as a few volts would seem possible because the field strength would still be sufficient to allow collection of electron current densities of greater than 10,000 amps per square centimeter without adverse space charge effects.

The screen grid would remain at some level in the 100-volt range acting as a bias to help cause field emission, but not collecting an appreciable current.

The upper limit in switching speed for this device would be set by the allowable power dissipation. Using an anode voltage of 10 volts, a current of 100 microamperes, a capacity of 10^{-15} farad, then this one-micron-sized device would show a switching time constant of 10^{-10} seconds for a power density of 10^5 watts per square centimeter--well within the allowed value for a single component.

Our completed machine composed of tunnel effect elements could have a maximum input power of 100 watts--as determined by heating considerations. The one milliwatt of input per device would allow 10^5 devices to operate simultaneously with a 10^{-10} second switching time which gives a bit rate of 10^{15} per second.

The transit time for electrons would be about 10^{-13} seconds, and this would seem to remove the need for complicated traveling wave devices to obtain high frequency gain.

Environment. Tunnel effect cathodes have been found to be insensitive to variations in temperature between $4^{\circ}K$ and $2000^{\circ}K$. The complete tunnel effect device would seem to be similarly independent of temperature although a high temperature limit is clearly predicted by conduction within the dielectric.

Mechanical forces would not likely effect this microscopic device unless a physical crack resulted in the construction materials.

Ionizing radiation does not affect the cathode properties or other metal electrodes until the dielectric has been severely damaged. The choice of dielectrics for this device is very wide, but materials like aluminum oxide or beryllium oxide, in film form, would be chosen because of their stable characteristics under bombardment.

Reliability. It is obvious that a device is perfectly reliable if none of the material composing it is allowed to move or migrate. This condition for stability is most easily met by using only refractory metals and dielectrics and removing all foreign materials from the system. Tunnel effect devices need only one type of conductor and one type of dielectric; these materials are chosen primarily for their stability, and no compromise has to be made to win some electrical effect. In addition to stable components, it is necessary to have dense layers of encapsulating materials to prevent migration of materials from the outside surfaces into the active areas. Microscopic vacuum and solid state devices suffer to an equal extent from this migration. To assist the application of dense outer layers it is beneficial to use high temperature processing for encapsulation.

Several thousand hours of operating life is presently obtained with gross field emission devices operated at 10^7 amps per square centimeter.[10] The cause for deterioration is inevitably due to contamination of the cathode or sputtering of the cathode with ions formed from gas contaminants that migrate into the area. By reducing the area of the container, a large source of contamination is removed. Lower operating voltages reduce the sputtering effects, and a clean construction process would remove the remaining offenders. The field emission cathodes now in existence are the only experimental examples the electronics industry has which operate with the microscopic areas of $10^{-10} cm^2$ that we anticipate using. We should be warned that the stability problem for all devices of equivalent area is severe, and then take steps toward clean construction methods.

Reproducibility. Vacuum tunnel effect devices have a very high probability of being self-formed into vast arrays having uniform electrical properties. This seems to be a unique advantage not shared by many types of components. The principal requirement for device self-formation is that a chemical process be effected by a significant electrical property of the component in such a way that the component is altered to a new form. The simplest illustration of this effect is the formation of an electrolytic capacitor in which the voltage controls the dielectric thickness. Additional requirements for self-formation are that all chemical residue may be driven from the component and that the uniformity will not be altered by this material being removed.

Cryogenic devices seem to have a negligible possibility of self-formation; magnetic devices a somewhat higher possibility, but still fairly low; semiconductor devices would be next in line; and then vacuum devices, which would be the easiest group because of the constant accessibility of the electronic surface being formed to the chemicals

responsible for forming. It is advantageous to have an electronic device that is insensitive to temperature variations for self-forming methods, because all too frequently chemical action cannot be made to take place at the component operating temperature. Once again vacuum devices, and especially tunnel effect vacuum components show their advantages.

In the self-formation of our tunnel effect components, the device would be made as near as possible to the final shape by electron beam activated micromachining with the many small cathode tips being formed too sharp for use. A voltage would be applied to all electrodes simultaneously in the presence of a molecular beam etchant and at an elevated temperature. The voltage would be raised until the sharpest tips began emitting. These emitted electrons have been found to accelerate the etching of small tips which in turn tends to increase the tip radius and reduce emission. By gradually increasing the applied voltage up to the operating value, all of the excessively sharp tips are degenerated to a constant emission value. The entire array of components would now be heated under ultra high vacuum conditions and sealed with encapsulating layers of materials. This self-forming process has taken into consideration the pertinent spacing variations as well as the cathode properties.

Interconnection. We intend to employ vacuum tunnel effect devices in a system that uses only optical input converters, electroluminescent generators, and a means for interconnecting components. No lumped capacitance, inductance, or resistance seems needed.

Since complete vacuum tunnel effect devices do not exist, detailed electrical characteristics are not known and it would be a waste of time to develop elaborate circuits for this hypothetical device. Still, there are many things that can guide us toward the eventual circuits.

Direct coupling between devices is facilitated by the positive grid characteristics. It has been found that the voltage-vs.-current characteristic of a vacuum tunnel diode behaves like a normal diode with a battery in series with it, thus forestalling conduction until the voltage has been raised appreciably. This bias-battery effect is also beneficial for direct coupling.

When an output is sought that takes either one of two states--B+ or ground--two tetrodes are used in series across the power input, one serving as a switchable load resistor for the other. In the type of microelectronics described here, every effort should be made to conserve energy, and the arrangement just described allows the capacity of the switched point between the two devices to be charged or discharged quickly without using a low value load resistor that requires the constant dissipation of energy for one of the switched states.

If resistors are excluded from our circuits then it will be found difficult to discharge grids under certain conditions. Here we would like to invoke the use of secondary emission.[11,12] The time delay from secondary emission surfaces is so short that it has never been measured. The current density available is arbitrarily high, up to the point that space charge or heating effects set in. The stability for properly cleaned surfaces would be the same as for equally clean field emission cathodes; thus we may feel free to employ this effect in clean microelectronic systems. The effect we seek is to drive a control grid positive when it receives a burst of electrons with enough velocity to exceed the secondary emission one to one point. In this way, a floating electrode may be driven positive or negative by changing the velocity of the electrons or by deflecting a beam of constant velocity to either the high or the low secondary emission areas of the electrode.

We would employ active memories of the negative resistance or dynatron type using secondary electron emission in this microelectronics system. A memory device of this type can have an almost arbitrarily low power consumption--limited by the leakage of the dielectric. The storage capacity of this memory must be charged and discharged by very energetic devices such as tunnel effect tetrodes in order to operate in the 10^{-10} second region.

As an alternate memory method we could use flip-flops having additional tunnel effect tetrodes to replace the conventional plate load resistors. The proper feedback to the variable load resistors serves to provide a low impedance for quick charging and a high impedance for the quiescent period.

These memory configurations reduce the current during the quiescent period to a value just necessary to prevent dielectric leakage from changing the stored value. A large array of these elements in quiescent state would draw a current equivalent to a slightly leaky capacitor of comparable area and thickness. Accidental removal of the voltage source from the system would have no immediate effect.

In future systems we shall want to dispose of some of the wires for interconnecting devices. Besides being a lossy means for conveying information in the micro-world, wires are a very inflexible arrangement for connecting components. We would like to have a completely uniform array of low complexity modules that are not interconnected in the beginning. A "path" should be built as the machine learns, and this path should be built by entirely electronic methods making efficient use of the components. An electrically steerable, periodically focused electron guide, sometimes called slalom focusing,[13] seems to answer our needs. This system makes use of the ballistic properties of free electrons and is able to guide bundles of electrons around devious corners, cross beams in the same plane without interference, obtain persistent current loops, and potentially cause certain beam-beam interactions that can do logic.

One configuration that appears useful is to locate our periodic array of modules--capable of launching, receiving, and steering electrons--between two conductive planes at cathode potential. This layer of components would then contain about half as much material as the wired method. Voids, for the potential electron paths, would replace the wires. The potential of the outer electrode

of our module would guide the electrons, the potential being determined by an internal memory and past history. The module would be expected to carry out steering functions and logical functions independently so as to increase the logical freedom of the system.

Once again our high field strengths would provide the means for suppressing space charge effects. The transit time for a 100-volt beam would be about one hundredth the velocity of light--not a bad price to pay when the dispersion effects on a lossy transmission line are considered.

Having once entered the electron beam realm we are able to use parametric amplifier techniques and achieve low noise amplifiers in spite of high lattice temperature.[14]

Light Coupling. For coupling light into our system, we will invoke nothing more sophisticated than a photosensitive surface connected to the grid of a tunnel effect amplifier, either directly or through secondary emission electron multipliers.

For output displays, field emitted electrons can cause DC electroluminescence effects, or in a less advanced scheme simple bombardment of a film of phosphor will suffice. If our systems are to be immune to temperature effects up into the red heat range, blue emission would seem most interesting. A blue sensitive photo surface has a high work function and can be stable in this temperature range.

Construction Process Requirements

We now need a fabrication method to make our devices. Although vacuum tunnel effect devices have been emphasized we do not intend to exclude any known electronic component from our repertoire.

It may seem at first that we have taken on some especially difficult problems like vacuum encapsulation, but a closer look reveals that this is no more difficult than filling in unused space in a solid state system. Present film methods do not see the problem because the edges of the film are so diffuse that a grading effect occurs between adjacent elements. When components are made that have micron dimensions on all sides, high resolution filling of voids is necessary to prevent a self-induced pin-hole effect. Converting a small open cavity into a small closed vacuum cavity can be done as easily as filling voids in one layer so as to smooth the surface, before proceeding to the next layer. This smoothing seems necessary regardless of the component used.

By being able to support a very thin film inside of a vacuum cavity, infrared or thermal detectors could be partly isolated from the lattice. Mechanical devices like electrostatic relays or acoustical filters also need this isolation.

Material Deposition. We have elected to use vacuum deposition methods to obtain our starting materials because this is a convenient method of producing a wide range of clean materials. Thermal evaporation has been used to produce photo conductors, phosphors, magnetic materials, superconductors, metals, dielectrics, and semiconductors. Reactive deposition methods[15] have produced the same group of materials, but have superior crystalline properties and greater stability. Reactive deposition is usually carried out by reacting some metal halide with a reducing or an oxidizing agent.

These reactions are most conveniently carried out in a high vacuum system as shown schematically in Fig. 2. For the deposition of molybdenum, one of the evaporation sources becomes molybdenum pentachloride and the other is hydrogen, made by thermally decomposing a material like zirconium hydride. The evaporation rate of these materials is monitored by separate ion gages and adjusted by external regulators. Upon reacting at the heated substrate, molybdenum is deposited and hydrogen chloride is pumped away. By this same method, aluminum oxide films can be formed when aluminum chloride and water vapor are reacted. Pin hole free deposits are obtained because there is no ash or lumps from the sources that are not immediately volatilized at the surface, and in addition the high mobility of the molecules on the surface do not cause shadowing effects from dust specks. There is less strain in the deposited films since the substrate can be much hotter for reactive deposition than for a conventional thermal evaporation. This higher temperature is possible because there is a mechanism for carrying away the energy of the condensing molecule when the volatile product of the reaction leaves the surface. Reactively deposited materials can be graded to the underlying film in an optimum fashion--not so little that the films tear or peel, and not so much as to ruin the sensitive electronic surfaces. Doping can be added to semiconductors with reasonable assurance of going in properly.

Material cannot be accurately localized by masking at the surface with reactive deposition, as it can with thermal evaporation. We are thus led to seek a method of selective removal for the films that are produced.

Material Removal. Any film that can be deposited can be etched away in the same vacuum system by using a molecular beam of the proper compound or element. If one of the evaporators in Fig. 2 becomes a source for chlorine, and the previously deposited molybdenum film is heated to around 500°C, the film will be converted into molybdenum chloride which evaporates from the surface. Aluminum oxide films can be etched in a phosgene molecular beam.

Etching in a vacuum system carries many advantages with it. For example, if etching effects are desired that cannot be obtained with molecular beams, then atomic beams can be used. The lack of surfaces between the source and the substrate prevent recombination of the atomic species. A sputtering etch can be easily carried out in the same apparatus.

It is desirable to have an etching reaction as efficient as possible to produce the cleanest samples. Any excess materials, like carrier gases, coming into contact with the surface raise the possibilities of contamination.

The Resist. It will be found that certain thin layers of material can protect the film below

them during etching and thus become a resist. Amorphous silica has been used as a resist for the selective etching of molybdenum and aluminum oxide as well as many others. This silica layer need be only about 50 angstroms thick for the protection of several thousand angstroms of underlying film material. After serving its purpose as a resist, the silica can be removed with a brief hydrogen fluoride molecular beam etch. Many resists are available, and simple chemical selection rules can determine which one is the best choice for any particular requirement.

The resist layers can be thermally evaporated onto a surface through masks, but this would not serve our purposes of having a high speed, electronically controlled, high resolution process. We have found ways of producing a silica resist by bombarding certain silicon-containing organic compounds with electrons.[16] The most common example is the bombardment of tetraethylorthosilicate. This substance is a liquid at room temperature and evaporates to the substrate where it probably forms a multi-molecular adsorbed layer. When this film is bombarded for about one second with a beam of 10 milliamperes per square centimeter current density, enough silica is produced for our purposes. The mechanism for silica production is not known in detail, but it is assumed to be a free radical polymerization of the organic followed by decomposition to silica upon heating to etching temperatures.

This gas-like compound is not easy to handle because it tends to enter the electron lens and contaminate the surfaces. An alternate material that also produces silica, but is easier to handle, is triphenylsilanol. This is a solid that is evaporated to the surface and then exposed as described above. Immediately after exposure no change is evident, but heating causes the unexposed areas to evaporate and the exposed portions to remain as a silica-like deposit.

The efficiency of this reaction is such that an average of one molecule of silica is produced for each electron. More sophisticated methods of resist production have given yields of over 10^3 molecules per electron by causing current multiplication within the sample being exposed.

The resolution of resist production has been seen to exceed 100 angstrom units. This is observed by exposing a surface of resist producing material with an oblique flood electron beam in such a fashion that shadows are cast by objects on the surface. These objects are usually polystyrene spheres about 880 angstrom units in diameter. An electron microscope is used to view the result.

Electron-Beam Activated Micromachining. It can be seen that electron-beam micromachining is the combination of certain methods of deposition, resist production, and etching. Our initial requirement for this fabrication process was to economically produce a one-cubic-inch system of 10^{11} parts being composed of 100 modules; each having 10 layers of 10^8 components per layer. How much time would it take to construct the system? Let us find how long it would take to make a single layer of 10^8 components, and then multiply by 1000.

The substrate must first be very carefully prepared by grinding, polishing, preparing terminals, vacuum firing, re-polishing, and then final cleaning and smoothing in the ultra high vacuum processing chamber. The substrate can be an expensive item.

For each layer of components there may be an average of four depositions and four etchings required. About five minutes per deposition and etching is possible, which includes time for heating and cooling. This totals forty minutes so far.

The 10^8 components per layer and their interconnections can be represented as an array of about 10^{11} 300 angstrom unit dots. This is arrived at by remembering that each component is composed of 10^3 pieces--our basic building blocks. It has been found that high resolution electron optical systems can easily have 10^8 bits in one field of view--a consequence of the small aperture angles used. A gross image could thus contain 10^5 complete components per field. We will be required to mechanically move the substrate in front of the lens to produce our latent-resist image. 1000 steps are needed to make 10^8 components. Each exposure can be made as rapidly as the electrical and mechanical system is able to servo and register on the preceding layers--about one tenth of a second. The high current density of the electron beam is able to expose the image in a negligably short time. One thousand steps would take about two minutes per layer or eight minutes for four layers. We have now accumulated 48 minutes to process one layer of components.

The production of an entire machine would take 1000 times 48 minutes or about 800 hours.

We have elected to use an electron beam to convey information to a surface to describe the patterns of material that will be successively built up to form a module. Besides the obvious advantage of having short wavelength, electron beams have extremely bright sources--i.e., many events per unit time; they are electrically steerable, and they interact well with matter. Beams have been used to analyze the properties of matter such as absorption spectra, crystalline diffraction effects, chemical analysis by x-ray fluorescence,[17] geometrical arrangement,[18] contact potential, and surface cleanliness.[19] In addition, beams have been used in electron mirror microscopes[20] to show voltage, magnetic field, and resistivity. We intend to integrate all of these functions into a single electron optical system so that our machine tool becomes an analytical tool as well.

Electron Optics. The electron optical system employed must integrate the functions of a scanning electron microscope, scanning x-ray fluorescence probe, and mirror microscope. This instrument is required to operate under ultra high vacuum conditions so that conventional electron microscope designs having many rubber gaskets will not do. We have elected to make an all-ceramic electrostatic instrument, with metallized surfaces, in order to obtain the necessary mechanical stability during the vacuum bake-out cycle.

When completed, this instrument should be

capable of machining or viewing structures down to a limit of 100 angstrom units resolution. The x-ray fluorescence probe could measure film thickness to one tenth percent accuracy and be able to carry out a quantitative chemical determination on 10^{-13} grams of material to one percent accuracy. The mirror microscope feature could measure the voltages on any element of our components that lay on the surface. This voltage measuring method could be converted by simple electrical switching into a very wide band oscilloscope to show the dynamic behavior of the components.

In addition to the functions of magnification, demagnification, manipulation, and electron and x-ray detection, the electron optical system must have a pattern generator integrated with it. The simplest means of conveying information to the surface is to use a scanning electron beam, but it will be found that this method is severely bandwidth and energy limited for complex patterns.

If a perfectly repetitive pattern is required for each of our 1000 exposures per layer, then a simple large-scale mask is needed near the electron source. If some sub-module exists in irregular order throughout our system then this can be obtained by a mask and then be deflected electrically to the desired location.

Since this electron beam machining process requires such a low current density, purely electrical methods of pattern generation may be invoked. A slowly scanning input beam can write a charge pattern on a storage screen which in turn controls the transmission of electrons. Ultimately, we would use a vacuum tunnel effect cathode array for our electron source. The emission from discrete areas would be controlled by local grids connected to micro-memory elements which are set by a slowly scanning beam or a microelectronic stepping system to write in the pattern changes that are required. Thus we have components made by electronic micromachining responsible for the building of new systems by the same method. In the end, self reproduction would be a distinct possibility without the use of a lens system, because all copies would be made on a 1 to 1 size basis.

Acknowledgements

We gratefully acknowledge the support of our work by the Information Systems Branch of the Office of Naval Research on Contract Nonr 2887(00) and for the support of the Air Force on Contract AF 19(604)-6114 and the Signal Corps on Contract DA 36-039-SC-84526.

References

1. L. Holland, Vacuum Deposition of Thin Films, p. 233 (John Wiley & Sons, Inc., New York, N.Y. 1956). See section on "Conduction in Very Thin Films" and section on "Mean Free Path of Conduction Electrons."
2. D. A. Powers and A. Von Hippel, Massachusetts Institute of Technology, Laboratory for Insulation Research. Progress Reports show consistent breakdown strengths between 50 and 200 megavolts per centimeter for film thicknesses between 700 and 4000 angstroms using alumino silicate glass.
3. W. P. Dyke and W. W. Dolan, Advances in Electronics and Electron Physics, Vol. VIII, p. 153 (Academic Press, Inc., New York, N.Y., 1956). See the discussion on field emission cathode blunting.
4. E. Mueller, Handbuch der Physik, Vol. 21, p. 202. Springer-Verlag, Berlin (1956).
5. V. E. Cosslett, A. Engstrom, and H. H. Pattee, X-Ray Microscopy and Microradiography, p. 336 (Academic Press, Inc., New York, N.Y., 1957).
6. W. P. Dyke and W. W. Dolan, Advances in Electronics and Electron Physics, Vol. VIII, p. 109 (Academic Press, Inc., New York, N.Y., 1956).
7. J. P. Barbour, F. M. Charbonnier, L. F. Garrett, W. P. Dyke, "On the Application of Field Emission to a Two Cavity Microwave Amplifier." Paper given at the 1958 Field Emission Symposium, University of Chicago.
8. Russell D. Young, Erwin W. Mueller, "Experimental Determination of the Total Energy Distribution of Field Emitted Electrons." Paper given at 1958 Field Emission Symposium, University of Chicago.
9. W. P. Dyke and W. W. Dolan, Advances in Electronics and Electron Physics, Vol. VIII, p. 120, 121, Appendix II (Academic Press, Inc., New York, N.Y., 1956).
10. E. E. Martin, J. K. Trolan, and B. W. Steller, "Research on Field Emission Cathodes," Scientific Report 5, Contract AF 33(616)-5404, Linfield Research Institute, McMinnville, Oregon (1959).
11. H. Bruining, Physics and Application of Secondary Electron Emission, McGraw-Hill, New York, N.Y. (1954).
12. A. M. Skellett, "The Use of Secondary Electron Emission to Obtain Trigger or Relay Action," J. Appl. Phys., Vol. 13, p. 519 (August 1942).
13. J. S. Cook, R. Kompfner, W. H. Yocom, "Slalom Focusing," Proc. of the Inst. of Radio Engineers, Vol. 45, p. 1517 (1957).
14. C. Burton Crumly and Robert Adler, "Electron Beam Parametric Amplifiers," p. 73, Electronic Industries (November 1959).
15. C. F. Powell, I. E. Campbell, and B. W. Gonser, Vapor-Plating, John Wiley & Sons, Inc., New York, N.Y. (1955).
16. D. A. Buck and K. R. Shoulders, "An Approach to Microminiature Printed Systems," proceedings of the Eastern Joint Computer Conference (December 1958) special publication T-114.
17. V. E. Cosslett, A. Engstrom, H. H. Pattee, X-Ray Microscopy and Microradiography, Academic Press, Inc., New York, N.Y. (1957).
18. V. E. Cosslett and P. Duncumb, "A Scanning Microscope with Either Electron or X-Ray Recording," p. 12, Academic Press, Inc., New York, N.Y., Electron Microscopy--Proceedings of the Stockholm Conference (September 1956).
19. H. E. Farnsworth, R. E. Schilier, T. H. George, and R. M. Burger, "Application of the Ion Bombardment Cleaning Method to Titanium, Germanium, Silicon, and Nickel as Determined by Low-Energy Electron Diffraction," J. Appl. Phys., Vol. 29, p. 1150 (1958).
20. Ludwig Mayer, J. Appl. Phys., Vol. 26, p. 1228 (1955); Vol. 28, p. 975 (1957); Vol. 29, p. 658 (1958); Vol. 29, p. 1454 (1958).

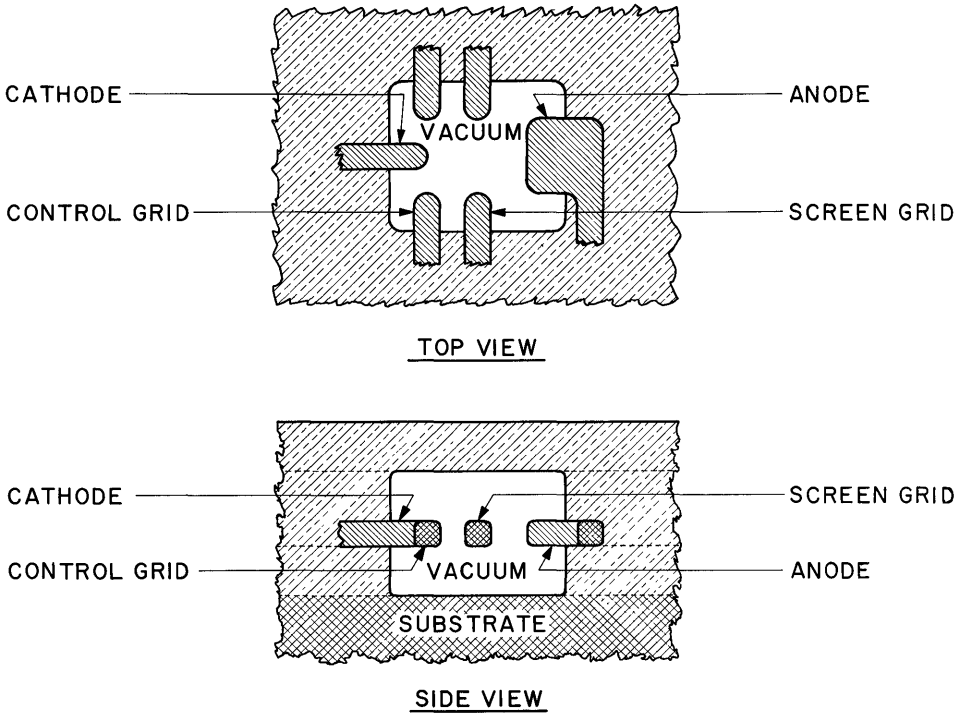


Figure 1. Tunnel Effect Vacuum Tetrode

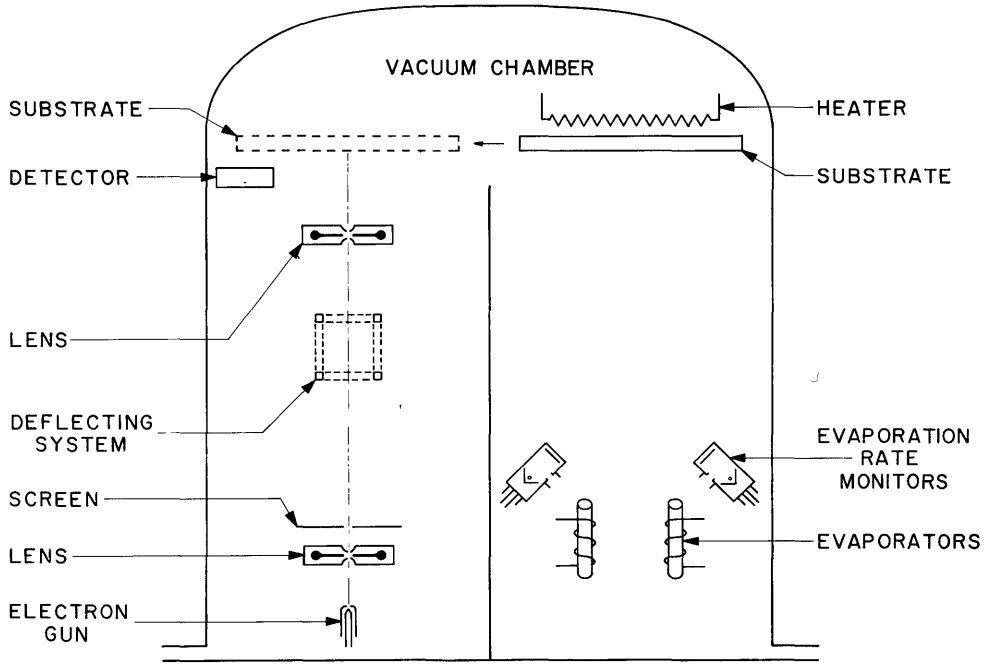


Figure 2. Processing Chamber

ITERATIVE CIRCUIT COMPUTERS

John H. Holland

The University of Michigan
Ann Arbor, Michigan

SUMMARY

The paper first discusses an example of a computer, intended as a prototype of a practical computer, having an iterative structure and capable of processing arbitrarily many words of stored data at the same time, each by a different sub-program if desired. Next a mathematical characterization is given of a broad class of computers satisfying the conditions just stated. Finally the characterization is related to a program aimed at establishing a theory of adaptive systems via the concept of automaton generators.

I

INTRODUCTION

Computers constructed of hundreds of millions of logic and storage elements will require an organization radically different from present computers if the elements are to be used efficiently in computation. It should be possible to process arbitrarily many words of stored data at the same time, each by a different sub-program if desired. In addition the structure of the computer should be iterative or modular in order to allow efficient use of template techniques in its construction.

The present paper contains a mathematical characterization of a broad class of computers satisfying these conditions. This class, with appropriate interpretation of the symbols, includes representatives structurally and behaviorally equivalent to each of the following types of automata:

1) Turing machines (with 1 or more tapes)[12,9]
2) Tessellation automata (Von Neumann, Moore)[14,7]
3) Growing logical nets (Burks-Wang)[2,1]
4) Potentially-infinite automata (Church)[3]

The class also contains automata which, in various senses, are generalizations of each of these four types.

Ultimately, for the designer, the value of such a characterization depends upon whether or not it can actually suggest designs for solid-state computer. Section II of this paper discusses a possible abstract prototype for such a computer--it is one of many alternatives which can be defined and investigated with the help of the characterization. (A similar computer is discussed in greater detail in the Proceedings of the 1959 Eastern Joint Computer Conference)[5]. Section III summarizes the mathematical characterization and considers its interpretation. Section IV relates this paper to a program (begun by the author in 1958) which has as its objective a theory of adaptive systems.

II

AN ITERATIVE CIRCUIT COMPUTER

The computer outlined in this section is presented primarily to suggest something of the class of computers included in the characterization summarized in the next section. At the same time, however, the order code, addressing schemes, etc. where chosen to reflect their counterparts in present computers. In this sense, the computer can also be thought of as a prototype of a practical computer--assuming that the large numbers of components required can be provided economically. Because the computer can execute an arbitrary number of sub-programs simultaneously, and because the sub-programs are spatially organized, its operation is of course considerably different from present computers.

The computer can be considered to be composed of modules arranged in a 2-dimensional rectangular grid; the computer is homogeneous (or iterative) in the sense that each of the modules can be represented by the same fixed logical network. The modules are synchronously timed and time for the computer can be considered as occurring in discrete steps, t = 0, 1, 2, ...

Basically each module consists of a binary storage register together with associated circuitry and some auxiliary registers. At each time-step a module may be either active or inactive. An active module, in effect, interprets the number in its storage register as an instruction and proceeds to execute it. There is no restriction (other than the size of the computer) on the number of active modules at any given time. Ordinarily if a module M(i,j) at coordinates (i,j) is active at time-step t, then at time-step t+1, M(i,j) returns to inactive status and its successor, one of the four neighbors M(i+1,j), M(i,j+1), M(i-1,j), or M(i,j-1), becomes active. (The exceptions to this rule occur when the instruction in the storage register of the active module specifies a different course of action as, for example, when the instruction is the equivalent of a transfer

instruction).

The successor is specified by bits s_1, s_2 in M(i,j)'s storage register. If we define the line of successors of a given module as the module itself, its successor, the successor of the successor, etc., then a given sub-program in the computer will usually consist of the line of successors of some module. Since several modules can be active at the same time the computer can in fact execute several sub-programs at once. We have noted parenthetically that there are orders which control the course of action--there are also orders equivalent to store orders which can alter the number (and hence the instruction) in a storage register. Therefore, the number of sub-programs being executed can be varied with time, and the variation can be controlled by one or more sub-programs.

The action of a module during each time-step can be diveded into three successive phases:

1) During phase one, the initial phase of each time-step, a module's storage register can be set to any arbitrarily chosen value and its auxiliary registers to any desired condition. The numbers and conditions thus supplied are the computer's input. Although the number in the storage register can be arbitrarily changed at the beginning of each time-step, it need not be; for many purposes the majority of modules will receive input only during the first few moments of time ("storing the program") or only at selected times t_1, t_2, . . . ("data input"). Of course, some modules may have a new number for input at each time-step; in this case the modules play a role similar to the inputs to a sequential circuit.

2) During phase two, an active module determines the location of its operand set, the set of storage registers upon which its instruction is to operate. This the module does by, in effect, opening a branching path (sequence of gates) to the operands. The path-building action depends upon two properties of modules:

First, by setting bit p in its storage register equal to 1, a module may be given special status which marks it as a point of origination for paths; the module is then called a P-module.

Secondly, each module has a neighbor, distinct from its successor, designated as its predecessor by bits q_1, q_2 in its storage register; the line of predecessors of a given module M_0 is then defined as the sequence of all modules $[M_0,M_1,\ldots,M_k,\ldots]$ such that, for each k, M_k is the predecessor of M_{k-1} and M_{k-1} is the successor of M_k. Note that the line of predecessors may in extreme cases be infinitely long or non-existent. The line of predecessors of an active module ordinarily serves to link it with a P-module (through a series of open gates). During the initial part of phase two the path specification bits $y_0,\ldots,y_n$ and $d_0,\ldots,d_3$ in the storage register of an active module M_0, are gated down its line of predecessors to the nearest P-module (if any) along that line. The path specification bits are then used by the P-module to open a branching path to the operand set of the active module.

Each path must originate at a P-module. The modules belonging to a given path can be separated into sub-sequences call segments. Each segment consists of y modules extending parallel to one of the axes from some position (i,j) through positions $(i+b_1, j+b_2)$, $(i+2b_1, j+2b_2)$,. . . , $(i+(y-1)b_1, j+(y-1)b_2)$, where $b_1 = \pm 1$ or 0 and $b_2 = \pm (1-b_1)$; the module at $(i+yb_1, j+yb_2)$ will be called the termination of the segment. Each module possesses four *-registers and if the module belongs to a segment in direction (b_1,b_2) the appropriate *-register, $(b_1,b_2)^*$, is turned on gating lines between (i,j) and $(i+b_1, j+b_2)$. Since each *-register gates a separate set of lines, a module may (with certain exceptions) belong to as many as four paths. Once a *-register is turned on it stays on until it is turned off; thus a path segment, once marked, persists until "erased".

Each segment of a path results from the complete phase two action of a single active module; however, since a path may branch, more than one segment may result in one time-step from the action of a given active module. After the digits $y_n,\ldots,y_0$, $d_3,\ldots,d_0$ are gated to the nearest P-module along the line of predecessors of the active module, new segments are constructed at the termination of each branch of the path originating at the P-module. Note that, because of the branching, there will be more than one path termination.

Branching is controlled by the digits $d_3,\ldots,d_0$. To each of the four digits $d_3,\ldots,d_0$ corresponds one of the four neighbors at each branch termination. If $d_i=1$ then, when the path is extended, at each existing path termination a new branch will be sent through the i^{th} neighbor parallel to the axis.

Path extension takes place only when bit $y_n=0$; then bits $y_{n-1},\ldots,y_0$ determine the common length of the new segments and bits $d_3,\ldots,d_0$ determine their directions. If $y_n=1$ then final path segments, if any, in the directions specified by $d_3,\ldots,d_0$ are erased (bits $y_{n-1},\ldots,y_0$ not being used in this case). In order to prevent interference of one path with another, or with itself, a set of priority and interlock rules are required. These rules will not be specified here but the interested reader can see a complete set of such rules for a similar computer in the 1959 E.J.C.C. paper cited previously.5

3) During phase three, an active module executes the instruction contained in its storage register. This involves the following modules: the active module itself holds the order code in bits i_2, i_1, i_0 of its storage register; the storage registers of the modules terminating the nearest path contain the set of words to be operated on (the operand set); finally there must be a module which serves as arithmetic unit. In order to serve as an arithmetic unit, bits (p,a) in the storage register of a module must first be set to the value (0,1), giving the module special status--A-module status. (Note that this means a module in P-module status, p=1, cannot be an A-module). If

M(i,j) is an active module then the first A-module along its line of predecessors serves as the arithmetic unit.

A short, though representative, set of orders follows:

(i) Execution of OR/ADD causes the following sequence of actions: first the numbers stored at the modules in the operand set are transferred down the branches of the path toward the P-module; as these numbers meet at branch-points a resultant is formed equal to the bit-by-bit disjunction of the incoming numbers (i.e. bit j in the resultant is 1 only if at least one of the incoming numbers has 1 at position j); when the final resultant is formed at the P-module it is transferred along the line of predecessors to the nearest A-module; there the number is added to whatever number is in the storage register of the A-module. Note that this sequence of actions takes place wholly within phase three of the time-step in which the instruction is executed.

(ii) Execution of AND/ADD proceeds just as OR/AND except that a bit-by-bit conjunction (output bit is 1 only if all corresponding input bits are 1) takes the place of bit-by-bit disjunction.

(iii) Execution of STORE causes the number in the storage register of the nearest A-module to be transferred to the storage registers of all modules in the operand set.

(iv) Execution of TRANSFER ON MINUS depends upon the number in the storage register of the nearest A-module. If, in this number, $y_n=0$ then at the end of phase three the active module becomes inactive and its successor becomes active. If $y_n=1$ then each of the modules in the operand set, rather than the successor, become active.

(v) NO ORDER causes the execution phase to pass without the execution of an order.

(vi) STOP causes the active module to become inactive without passing activity on to its successor at the next time-step.

With the exception of the TRANSFER and STOP orders, the active module becomes inactive and its successor becomes active at the conclusion of phase three. Just as in the case of phase two some rules are required to prevent interference of active modules and to provide for cases where there is no nearest A- or P-module along the line of predecessors (the reader is again referred to the 1959 E.J.C.C. paper)[5].

The storage register of each module in the present formulation consists of n+14 bits labelled in the following order:

bit number:	n+14	n+13	...	14	13	12	11	10	9	8	7	6	5	4	3	2	1
label:	y_n	y_{n-1}	$\cdots$	y_0	d_3	d_2	d_1	d_0	i_2	i_1	i_0	s_1	s_2	q_1	q_2	p	a

The function, in the active module, of each bit group has already been discussed.

III

MATHEMATICAL CHARACTERIZATION OF ITERATIVE CIRCUIT COMPUTERS

One purpose of the mathematical characterization summarized here is to define the class of iterative circuit computers precisely enough to allow mathematical deduction to be used in their study. This property of the characterization will be used in later work in an attempt at establishing a theory of adaptive systems (see the next section). At the same time the characterization can be used to generate a wide range of computer prototypes, each with different structural and operational characteristics. Thus, the characterization can also be of help in the design of solid-state computers.

The characterization is made up of the following parts:

1) The positions of the modules are indexed by the elements of a finitely-generated abelian group, A. The particular group chosen determines the "geometry" of the network; for instance, by choosing the appropriate group, the modules can be arranged in a plane, or a torus, or an n-dimensional cube, etc. Thus, for a computer with the modules arranged in a 2-dimensional rectangular grid 1000 modules on a side, A would be the abelian group with two generators a_1, a_2 satisfying the relations

$$1000\ a_1 = e$$
$$1000\ a_2 = e$$

where e is the identity element of the group.

The group, A, is restricted to being a finitely-generated abelian group for several technical reasons. One reason is that the elementary theory of such groups is decidable. When taken with the rest of the definition of iterative circuit computers, this implies that the operation of the computer is effectively defined. Also any such group can be decomposed into a direct product of cyclic subgroups. Thus the elements of the group can be represented uniquely as n-tuples on the basis of certain sets of generators of the group (in other words, the modules are arranged in a "regular" fashion).

2) In order to determine the immediate neighbors of a module we must specify a finite set, $A^\circ = (a_1, \ldots, a_k)$, of elements selected from the

group A. Then the set of immediate neighbors of the module at $\alpha\epsilon\{A\}$ are the modules at $a_i(\alpha) = \alpha+a_i$ for all $a_i \epsilon A^\circ$, where + is the group operation. For example, if we have a module at coordinates (i,j) relative to generators a_1,a_2 and we wish its immediate neighbors to be at coordinates (i+1, j), (i, j+1), (i-1, j), and (i, j-1) then we could choose $A^\circ=\{a_1,a_2,a_1^{-1},a_2^{-1}\}$, where a_i^{-1} is the group inverse of a_i.

3) The state of each module in the computer at each time t must be drawn from a finite set, S, of allowable states. $S=X\boxtimes Y$, the cartesian product of the sets X and Y. X can be any finite set of elements--the elements will be called "storage states"; $Y=\prod_{i=1}^{k} Y_i$, a cartesian product of the sets $Y_i=R=\{a_1,\phi\}\boxtimes \ldots \ldots\boxtimes\{a_k,\phi\}$.

In what follows the notation S_α^t will be used to denote the state of the module at position α at time t. A similar convention will be used for the components of S. Note that the elements of Y can be though of as k by k matrices. The matrix Y_α^t which holds for the module α at time t is called the connection matrix of α at time t. The i^{th} row of Y_α^t, $Y_{\alpha i}^t = (Y_{\alpha i1}^t, Y_{\alpha i2}^t, \ldots, Y_{\alpha ik}^t)$, specifies which of the k immediate neighbors of α are connected through α to the module at $a_i^{-1}(\alpha)$; if $Y_{\alpha ij}^t = a_j$ then the module at $a_j(\alpha)$ is connected through α to the module $a_i^{-1}(\alpha)$ otherwise not.

4) Changes in the k rows $Y_1,\ldots,Y_k$ of the connection matrix Y from one time step to the next, $Y_{\alpha i}^t$ to $Y_{\alpha i}^{t+1}$, are determined by a set of k projections:

$$P_i : S \rightarrow R = \{a_1,\phi\}\boxtimes \ldots \boxtimes\{a_k,\phi\}.$$

The way in which these changes are effected will be described in terms of the transition equations to be given shortly.

5) Change in the storage state from one time step to the next, X_α^t to X_α^{t+1}, is determined by a function

$$f : \prod_{i=1}^{k} S \rightarrow X$$

which will be called the "sub-transition function". Again f can be best explained in terms of the transition equations.

A particular selection for each of the five parts described in 1) through 5), $(A,A^\circ,X,\{P_i\},f)$, determines a particular iterative circuit computer. In addition a function $B : \{A\}\boxtimes\{t\} \rightarrow S$ may be effectively defined for some pairs (α,t); B_α^t gives the input to α at time t, when B is defined for (α,t). Once a particular iterative circuit computer is specified (and its initial state is given), the transition equations together with the function B determine the operation of the computer.

The transition equation for the connection matrix Y_α^t in terms of its elements, $Y_{\alpha ij}^t$, is:

$$Y_{\alpha ij}^{t+1} = \phi \quad , \text{ if } P_{\alpha ij}^t = \phi \qquad \text{("erasure")}$$

$$= Y_{\alpha ij}^t \quad , \text{ if } P_{\alpha ij}^t = a_j \text{ and } Q_{\alpha ij}^t = 0 \qquad \text{("no change")}$$

$$= P_{\alpha ij}^t \quad , \text{ if } P_{\alpha ij}^t = a_j \text{ and } Q_{\alpha ij}^t = 1 \qquad \text{("construction")}$$

where $P_{\alpha ij}^t$ is the j^{th} component of $P_i(S_\alpha^t)$

and $Q_{\alpha ij}^t = \&(q_{a_j(\alpha)j1}^t,\ldots,q_{a_j(\alpha)jk}^t)$

$$q_{\beta ij}^t = 0 \quad , \text{ if } Y_{\beta ij}^t = \phi \text{ and } P_{\beta ij}^t = a_j$$

$$= 1 \quad , \text{ if } P_{\beta ij}^t = \phi$$

$$= \&(q_{a_j(\beta)j1}^t,\ldots,q_{a_j(\beta)jk}^t), \qquad \text{otherwise}$$

defining $\&(c_1,\ldots,c_k) = 1$, if all $c_j=1$

$= 0$, otherwise.

As mentioned in 3) above, row i of the connection matrix Y_α^t can be interpreted as specifying a set of modules $a_j(\alpha)$ connected through α to $a_i^{-1}(\alpha)$; for each such j, row j of the connection matrix $Y_{a_j(\alpha)}$ may specify other modules $a_h a_j(\alpha)$ connected, via $a_j(\alpha)$ and then α, to $a_i^{-1}(\alpha)$; appropriate rows of the matrices $Y_{a_h a_j(\alpha)}$ may specify still others; etc. In other words the matrix Y_β^t tells how information is to be channeled through β to its immediate neighbors, the matrices for these neighbors tell how the information is to be sent on from there, etc. In this way each module serves as the base of what may be a complex branching tree channeling information to it. It will be seen (in the transition equations for X_α^t) that modules belonging to the tree for α pass information to α without a time-step delay.

The transition equation for Y_α^t can now be given the following interpretation: Broadly, $P_{\alpha i}^t$ (ie $P_i(S_\alpha^t)$) specifies changes in row i of the connection matrix Y_α^t while $Q_{\alpha i}^t = (Q_{\alpha i1}^t,\ldots,Q_{\alpha ik}^t)$ prohibits

certain of these changes. More specifically,

$Q^t_{\alpha i j} = 0$ prohibits construction of a new connection from $a_j(\alpha)$ through α to $a_i^{-1}(\alpha)$ at time t. $Q^t_{\alpha i j} = 0$ just in case construction of a new connection is indicated somewhere in the tree for α. Some thought will show that this rule (others, at least superficially more general, could have been chosen) implies the following desirable conditions:

(i) a cycle of connectionswithout delay cannot be formed (operation of modules belonging to such a cycle would in general be indeterminant--consider the analagous case of a set of one-way, non-delay switches arranged in a cycle)

(ii) the tree for any given module α never includes more than a finite number of modules (even if, for theoretical purposes, the group A is infinite).

The transition equation for X^t_α is:

$$X^{t+1}_\alpha = f\,(S'_{a_1(\alpha),1}(t)\,,\ S'_{a_2(\alpha),2}(t)\,,\ \ldots,\ S'_{a_k(\alpha),k}(t))$$

$$S'_{\beta,i}(t) = S^t_\beta\,,\ \text{if } Y^{t+1}_{\beta i} = (\phi,\ldots,\phi)$$

$$\text{and } B^{t+1}_\beta \text{ not defined}$$

$$= B^{t+1}_\beta\,,\ \text{if } Y^{t+1}_{\beta i} = (\phi,\ldots,\phi)$$

$$\text{and } B^{t+1}_\beta \in S$$

$$= f\,(S'_{Y_{\beta i1}(\beta),1}(t)\,,\ \ldots,\ S'_{Y_{\beta ik}(\beta),k}(t)) \otimes Y^{t+1}_\beta,$$

$$\text{if } Y^{t+1}_{\beta i} \neq (\phi,\ldots,\phi)$$

If $Y^{t+1}_{\beta i j} = \phi$ then define $S'_{\phi(\beta),j}$

$$= S_{a_j(\beta)},\ \text{if } B^{t+1}_{a_j(\beta)} \text{ not defined}$$

$$= B^{t+1}_{a_j(\beta)},\ \text{otherwise.}$$

Under interpretation the transition equation for X^t_α specifies the storage state, X^{t+1}_α, in terms of the states at time t, S^t_β, of the modules β belonging to the connection tree for α. Note that, because of the recursive definition of S'(t), f may be iterated several times in the determination X^{t+1}_α--compare this to the determination of the output of a tree of switches without delays.

IV

TOWARD A THEORY OF ADAPTIVE SYSTEMS

As already mentioned, the work reported here is part of a larger effort which has as its goal a theory of adaptive systems. The effort is an individual one and, of course, reflects particular biases of the author. This section will discuss the relation of the present paper to the broader program.

The first step of this program was the description of a computer which could simulate the operation and, in certain respects, the structure of any automaton (growing or fixed). The second step, summarized here in part III, consisted in giving a general and formal description of computers like the one first obtained--the iterative circuit computers. The resulting mathematical characterization represents a broad class of machines, of arbitrary geometries, etc.; by an appropriate choice of $(A,A°,X,\{P_i\},f)$ it is possible to represent directly not only the behavior but also the changing structure and local operation of any given potentially-infinite automaton, tessellation automaton, n-tape Turing machine, or growing logical net. (In this respect note, for instance, that the class of iterative circuit computers properly contains A. Church's class of potentially-infinite automata--any two modules in an iterative circuit computer may eventually become connected so that either affects the other in a single time-step, whereas in a potentially-infinite automaton the corresponding delay, in time-steps, increases with increasing separation and is a constant for any given separation).[3]

For an iterative circuit computer the ideas of sub-program and automaton are, in an important sense, interchangeable. For any automaton, a sub-program can be written which not only has the same behavior but also the same changing structure and local operation. On the other hand, a given sub-program will in general occupy a finite number of modules in the computer and will have its action (or state) determined by bordering modules and the input function B. Thus, as a survey of the characterizing equations will show, an automaton or growing logical net can be constructed which mimics the sub-program.

It is important to note that sub-programs can be set up which, for instance, can shift themselves from one set of modules to another set, i.e. from one position to another. Thus the underlying geometry of the iterative circuit computer (given by A and A°) in effect determines the geometry of a space in which the sub-program is embedded; the transition equations then serve, in a sense, as the laws of this universe. For this reason the sub-programs of a given iterative circuit computer will often be spoken of as "embedded automata". This view of the results of the second step leads directly to the third step.

The central object of the third step is to provide formal apparatus for the implicit definition of automata--definition by means of generators and relations on these generators. Implicit definition of an automaton is analogous to the implicit definition of an algebraic group. In the case of the group, instead of giving an explicit listing of the elements of the group and their interrelations, the group is defined (often quite compactly) in terms of a set of generating elements and relations on products of the generators. In a similar sense an automaton can be implicitly specified by an initial set of elements and a set of growth rules.

The mathematical characterization of iterative circuit computers provides the apparatus needed for a precise formulation of automaton generators. Specifically, the generators will be sub-programs which can be thought of as (relatively) elementary embedded automata having the following properties:

(i) movement--the generators will in general be capable of shifting as a unit from one position to another (as specified by input $B(\alpha,t)$ or the state of adjacent modules--note that motion may be random if the input sequences $B(\alpha,t)$ are random),

(ii) connection--generators will combine under conditions specified internally (within the sub-programs) to form larger sub-programs capable of moving and acting as units,

(iii) production--generators can alter the state of adjacent modules (note that a sufficiently complicated generator could directly duplicate itself).

The generators will act upon other generators or other sub-programs present in the computer ("precursors") by connecting them or breaking them into components. The generators will all be acting simultaneously and if a given generator duplicates itself the duplicate will also in general be active.

So that any possible automaton can be defined in terms of the generators it is necessary to choose the set of generators so that any possible program for the computer can be represented by an appropriate connected set of generators (cf. the process of picking a set of instructions sufficient for a universal computer). Once this is done, the generation of particular automata can be effected by controlling rates of production and connection, movement and contact, the nature of precursors present, etc. That is, the relations on the generators will consist of specifying the initial states and input sequences which control such factors. For any given iterative circuit computer and set of generators, the relations possible can be given an appropriate equational form. Under one approach, the way in which the generators are initially connected and the nature of the precursors in the "environment" are sufficient together to specify the generated automaton. Things become particularly simple if the generators cannot interpenetrate when moving (a kind of "billiard ball physics").

A given program, because of the loops or iterations, is much more compact than the complete sequence of steps in the calculations it controls. In the same sense the connected system of generators which specify a given automaton will be much more compact than the automaton generated. Thus the automaton can have scattered throughout its structure complete implicit descriptions of its structure--such considerations play an important part in the study of self-repairing automata.

In the implicit definition of an automaton certain feedback phenomena play a crucial role. The feedback phenomena can be to some extent isolated by observing at what level a given generator system falls in the following succession of categories (each of which properly includes its successor):

(i) productive systems--the generator system produces other generators or precursors,

(ii) autocatalytic systems--the generator system produces generators or precursors which are used in its construction, i.e. the system produces some of its own components,

(iii) self-duplicating systems--the generator system produces duplicates of itself.

Such considerations lead directly from step three to step four and from work in progress to work which lies in the future.

The central object of step four will be to define the term "adaptive system" for embedded automata. Because of the formal nature of the definition, it then becomes possible to investigate these adaptive systems deductively (and by simulation). The beginnings of such a definition lie in the following consideration: With the help of concepts such as autocatalytic and self-duplicating generator systems it is possible to define such concepts as steady-state equilibria and homeostasis for embedded automata. In fact one can go quite far in this direction obtaining discrete state relaxation processes (Southwell), morphogen standing-wave phenomena (Turing) and so forth.[11, 13] Automata exhibiting these properties will usually have the desirable property that small changes in structure result in small changes in behavior (at least over a certain range). Thus the behavior of a given automaton of this type gives some indication of the behavior of all automata of similar structure ("hill-climbing" techniques become applicable). If the generator system for such an automaton has a hierarchical structure, then a small change in structure produces a small effect in proportion to the "position" of the change in the hierarchy. That is, a generator system may consist of autocatalytic or homeostatic systems, systems of these (which may or may not be autocatalytic or homeostatic), etc.; changes at the upper levels of the hierarchy will generally have a greater effect than those at lower levels. By making changes first at the highest level and then at progressively lower levels of the hierarchy, it should be possible to narrow down rather

quickly to any automaton in this category having some initially prescribed behavior.

Changes in the generated structure result when relations on the generators are altered. The effect of such alterations can perhaps be more clearly seen under the following interpretation. The generation of a particular automaton can be looked at as if all possible generation processes are going on simultaneously but at different rates. Some will be going very slowly (infinitely slowly in the limit) while others will be proceeding very rapidly. The way in which these rates are changed in order to change the generated automaton will have important consequences with respect to the adaptiveness of the overall system (cf. A. L. Samuel's work on changing the weights of a "checker-move tree").[10]

As a final point it should be noted that the environment of an embedded automaton can be made as simple or complex as desired. Since adaptation must be defined in terms of the range of environments in which the automaton is to be embedded, this is an important factor. It has already been noted that the environment may contain other sub-programs (precursors) or in fact other embedded automata. The latter case amounts to an implicit definition of the environment since only the initial state and internal rules of each of the embedded automata need be given. Contrast this with an explicit definition which would require a point-by-point, time-step-by-time-step description of the state of the environment. If the precursors in the environment are relatively elaborate and sophisticated then the adaptation process will look similar to a heuristic learning system (cf. Newell-Shaw-Simon).[8] If precursors are absent or simply generators then the adaptation process will look more like the processes considered by Friedberg.[4] It seems likely that implicit definition of the environment will play an important part in the development of step four.

BIBLIOGRAPHY

1. Burks, Arthur W., "Computation, Behavior, and Structure in Fixed and Growing Automata" University of Michigan Technical Report ONR Contract 1224(21) (1959).
2. Burks, Arthur W. and Hao Wang, "The Logic of Automata" J. Assoc. Computing Mach. 4, 2 & 3, 193-218, 279-297 (1957).
3. Church, Alonzo, "Application of Recursive Arithmatic in the Theory of Computers and Automata" notes from summer conference course in Advanced Theory of the Logical Design of Digital Computers, The University of Michigan (1958).
4. Friedberg, R.M., "A Learning Machine: Part 1" IBM Journal of Research and Development 2, 1,2-13 (1958).
5. Holland, J.H., "A Universal Computer Capable of Executing an Arbitrary Number of Sub-Programs Simultaneously" Proc. 1959 Eastern Joint Computer Conference.
6. Kleene, S. C., "Representation of Events in Nerve Nets and Finite Automata" in Automata Studies Annals of Mathematics Studies no.34, Princeton (1956).
7. Moore, Edward F., "Machine Models of Self-Reproduction" paper (560-52) at October Meeting of the American Mathematical Society, Cambridge, Mass. (1959).
8. Newell, A., J.C. Shaw and H.A. Simon, "Empirical Explorations of the Logic Theory Machine: A Case Study in Heuristics", Report P-951, Rand Corporation (1957).
9. Rabin, M.O., and D. Scott, "Finite Automata and Their Decision Problems" IBM Journal of Research and Development 3, 2,114-125 (1959).
10. Samuel, A.L., "Some Studies in Machine Learning, Using the Game of Checkers" IBM Journal of Research and Development 3, 3,210-229 (1959).
11. Southwell, R.V., Relaxation Methods in Engineering Science; a Treatise on Approximate Computation, Clarendon Press, Oxford (1940).
12. Turing, A.M., "On Computable Numbers, with an Application to the Entscheidungsproblem" Proc. Lond. Math. Soc. (2), 43, 230-265 (1936).
13. Turing, A.M., "The Chemical Basis of Morphogenesis", Phil. Trans. Roy. Soc., ser. B, 237, 37 ff. (1952).
14. Von Neumann, J., The Theory of Automata, unpublished manuscript.

* * * *

This research was supported by the U.S. Army Signal Corps through contract OA-36-039-SC-78057, and by the National Science Foundation through grants G-4790 and G-11046. The paper was written while the author was a member of the Institute of Science and Technology at the University of Michigan.

ON PROGRAMMING A HIGHLY PARALLEL MACHINE TO BE AN INTELLIGENT TECHNICIAN

Allen Newell*
The RAND Corporation
Santa Monica, California

Summary

This paper speculates on how to program a machine that is suitable for microelectronic components to be an intelligent technician. The point of departure is from a class of machines described by J. H. Holland in a concurrent paper entitled, "On Iterative Circuit Computers Constructed of Microelectronic Components and Systems." These machines consist of a regular lattice of active modules, each possessing both processing and memory functions. The goal is a machine with the problem solving capabilities of a smart human technical assistant, and the volume processing capabilities normally associated with digital computers. This goal is chosen because it coincides with many current developments. After discussing the eventual capabilities desired and the most striking features of Holland's machines, the speculation proceeds by considering the basic organization for information processing. This is followed by briefer treatments of the organization for problem solving, supervision, interpretation and production.

Introduction

This is a speculative paper. I have been invited to provide a link in a chain of reasoning stretching from the impending advances in microelectronic components to their social consequences. I take as given a class of machines postulated by J. H. Holland in the preceding paper[1], which I will henceforth call Holland machines. A Holland machine is very different from any current computer, reflecting both potential virtues and vices of microelectronic techniques. On balance, of course, the great increase in speed of action and number of components guarantees a great increase in processing potential. I am to show how this potential can be realized.

A Holland machine is so strange on first contact that the problem seems to be to regain the programming facility attained on current machines in a decade of effort. But magnitude increases in basic capacity should yield equivalent increases in delivered power. The prospect of working hard to reestablish what already exists is not inviting. Hence, my task includes providing some vision commensurate with the new capacities.

The givens and the desired are yet further apart. A Holland machine is an abstract automaton, and I am free-that is, forced--to specify the exact version that will suit my needs. The boundary between the logical designer and the programmer has shifted so that by any normal job description I play both roles. The difference lies in the components available to me. I have not AND's and OR's, but "semi-computers," out of which to construct a machine.

The Vision

An Intelligent Technician

Of the myriad things that will occur with the continued development of information processing machines, I will select just one for consideration--a machine that is an intelligent processor of information. There is nothing very dramatic about such a machine, except perhaps the results that are achievable with it. It is difficult to differentiate this machine from an extremely compliant, fairly bright, human technical assistant, backed up by an impressive computing establishment. The user will converse with such a machine about his problem with the freedom of ordinary technical discourse. The language may not be English, but it might as well be for the freedom and flexibility it will afford. The machine will return answers with a rather breath-taking rapidity, and it is difficult to predict how rapidly a human-plus-machine can advance into a problem. Like all machines (and humans) it will have finite capacities. Ordinary intelligence could pose useful problems that require processing that the machine can do in a reasonable time. Ordinary muddleheadedness could so confuse the machine that it would not know what is desired. Ordinary knavery could fool it into doing the wrong thing. In short, the total product will depend on the joint intelligence and processing capacity of the user-plus-machine. A smart, articulate, well prepared man will

---

* All my thinking on intelligent processes and computer languages has occurred in the context of joint research with J. C. Shaw and H. A. Simon. For the speculative application of these ideas to Holland's machines, I alone must take responsibility, of course. I am indebted to J. H. Holland. He not only provided the basic stuff out of which to fantasize, but he helped me to clarify the underlying concepts and suggested implications of the modes of organization I was trying to achieve.

produce more with the machine than a dull, ambiguous, unprepared man.

All our lives are spent learning to live with other intelligences, and relating to an intelligent machine will seem more familiar than strange. Since we want machines to help us solve problems, the more intelligent we are able to make it, the more unobtrusive it should be in providing this help. Contemplating the more extreme forms of this vision, there is little to describe about the machine except that it gives a great deal of help quickly and with very little pain. Not even this picture of productive harmony is safe, since we will soon accept all the benefits that flow from the new capability and will become aware of the constraints and annoyances that mark the new boundary of limitations. Joining a reasonable level of intelligence with the volume processing characteristics we already associate with the large computers will produce large and dramatic consequences. But these require detailed consideration for each subject matter, a task exemplified by the last paper in the chain by C. W. Churchman.[2]

Why This Vision?

I select this particular vision because I believe it lies along the main path of computer development quite independent of microelectronic advances.

The history of computers is marked by a sequence of innovations, each eliminating some factor limiting the rate of information processing. Most of these limitations were the time it took humans to make decisions or take action. The addition of memory and control to a fast arithmetic device bypassed the human decision time in doing simple numerical procedures. This innovation produced the computer essentially in its present form. The excessive time that it takes for humans to code has given rise to the assemblers, algebraic compilers, list languages, and the like. The wastage from a half-second human tending a microsecond machine is forcing the development of supervisory routines and interrupt systems that will eventually eliminate the operator.

The intelligent technician is simply another step in this development. The time for human invention and accurate specification of memory organization and elementary procedures in most programs is already too long. It will become intolerable with Holland machines made from microelectronic components. We need a "problem-oriented machine" instead of a "machine-oriented machine"--and such a machine, truly conceived, is equivalent to an intelligent assistant.

I do not consider the vision radical. Indeed, the programming and computing world is already on its way to achieving it.

Fundamentals

The Requirements of Intelligent Action

When applied to humans, intelligent action means action that achieves desired ends. There is no reason to modify this usage with respect to machines. An intelligent technician must be able to achieve desired ends. As normally used the term is ambiguous, since the resources given to the intelligence are not stated. This ambiguity is essential, since high intelligence in the real world means the ability to surmount the "givens" if they get in the way of problem solution. It means the ability to achieve desired ends even when defined with surprisingly little information. Intelligence is our word to indicate the fact of successful performance against theoretically inadequate conditions. Intelligence is the resolution of ambiguity by means of an adequate theory.

The notions expressed above trace capricious paths through the concept of intelligence. They can be summarized in four requirements for the behavior of the intelligent technician.

Indefinite Resources. The intelligent technician must be able to solve problems. More importantly, he must be able to work on any problem that can be stated to him. No limit can exist to the resources available for working on a given problem. These need not be all the methods available nor very good ones, but the machine should not quit because it has no more things to try. An intelligent human working on some hard differential equations will try to solve them analytically; then will turn to power series; then to a book on trigonometric series so he can apply these; then to a treatise on differential equations for some new clue (finding this via a reference in his standard text on the subject); then to numerical procedures. On and on it goes. If there is any fixed limit--any sharp restriction to a special class of methods--we will recognize soon enough that the intelligent technician cannot be left on his own, that it is really not intelligent enough.

Indefinite Awareness. A distressing feature of current computers is their instability in the face of trivial errors, a feature often cited in making comparisons between brains and computers. To be intelligent is not to be trapped easily--not to require external help for little things. The machine itself must be aware of its own behavior and of the context in which its work is done. No sharp boundary can exist for which particular features the machine is aware, even though it cannot be aware of everything. This would soon reveal itself as a "quirk" or "blind spot" that limits the intelligence.

Indefinite Language. The machine's capabilities are mirrored accurately in the external language used between it and the human

user. For example, if the language is limited to procedures, then it is not possible to ask the machine to solve problems, such as "Find x such that sin x - x = .5." The model of language required might be labeled the "Hide and Seek" model. Any information the transmitter can hide in the expressions being communicated, the receiver can find, providing the rules of hiding are given in the language. Any restriction to a fixed vocabulary, a fixed grammar, or a fixed technical area will be viewed immediately as a limitation in intelligence.

Indefinite Accumulation. Anything that is reasonably intelligent learns from what it does. Indeed, problem solving involves learning in many ways—e.g., learning about trigonometric series in the example above. Part of the stream of facts, procedures, clues, theories, and so on that pass through an intelligent machine will be accumulated for later use. Again, no definite boundaries can exist on what is selected or how it is filed and indexed. Limitations of this sort will reveal themselves as inappropriate repetitions, and will immediately be seen as a deficiency in intelligence.

Perhaps I am grooming the machine to be a genius, rather than a mere technician. I do not think so. Educated men possess these characteristics to a remarkable degree. Limits will exist, or the machine would be smarter than most men. But these limits will be indefinite and shifting, and will express themselves in a global measure of the machine's intelligence.

It may seem that all the emphasis is on intelligence and none on the volume processing capacities that are also needed. On the contrary, the machine is for high volume work. The intelligence forms the connective tissue that allows the machine to rapidly organize itself to be highly repetitive and efficient.

General Layout of the Machine

Although I cannot provide a complete picture, I will give some considerations about an organization for such a machine. These fall under the headings of information processing, problem solving, supervision, and interpretation and production. These headings are machine oriented and are not coordinate with the requirements of intelligent action. The general layout is shown in Figure 1.

An External Language is used between the machine and those things with which it communicates. This language is completely independent of the internal structure of the machine. Only incidentally will it refer to things and properties inside the machine. The expressions in the external language are taken in bodily and made available in the Input-output area.

There is a single Processing area, structurally homogeneous, but divided into two functional areas: the Interpreter and the Factory. The Interpreter produces the processes indicated by the External Language expressions in the Input-output area. These action processes are constructed in the Factory, and constitute the activities that the machine does in response to the external world.

The processes in the Factory are in the Internal Language, as are the interpretation processes in the Interpreter. In Holland machines it is difficult to distinguish programs from processes--expressions that say what to do from structures that do it. A language expression written in a set of modules may be sufficient to convert these modules into a process that behaves according to the expression. Hence, "internal language," "program" and "process" are used interchangeably.

Besides the Input-output area, there are three stores for information. Each is structurally distinct because of different reading and writing requirements. The Program Store contains the large number of program forms required by the Interpreter and the Factory. The Associative Store contains entities with the properties normally associated with symbols. The Warehouse is a tertiary store that backs up all the other areas. It is a reminder that, even in a machine with 10^{11} components, access time must be traded for space in order to remember enough information.

Holland Machines

A Holland machine consists of a regular lattice of identical modules, as shown in Figure 2. Each module is directly connected with its immediate neighbors in the lattice. These direct physical connections can form paths between distant modules. A module's capacity for paths is limited. Once formed, the paths give immediate access independent of length. Each module has both memory and processing functions, and may be active simultaneously with and independently of other modules. The functions of a module are represented by settings of bit patterns, so that a module may take on any possible character by writing a new bit pattern into it. Operations change the state of a module as a function of the states of other modules connected to it by paths. These include operations for reading and writing information into modules and for building up paths. The machine is basically synchronous.

Within the bounds indicated almost any kind of a system is possible. There is freedom to specify the number of immediate neighbors, the kinds of information held by each module, the operations performed by each module, and the sizes of the paths. For a given amount of basic componentry the more complex the module the more components it will take per module and the fewer modules that will be available. The original papers should be consulted for more detail.

The following features summarize most of the advantages and problems of Holland machines:

Indefinite Parallelism. The most striking feature of Holland machines is their active nature. Many processes can run in parallel, each of arbitrary composition. Our experience is almost entirely with completely serial machines. Machines with a small number of independent processors are just coming into being. Although a telephone system is like a big parallel computer in many ways, it accomplishes unsophisticated functions compared to those needed for general computation. There is little work in the literature on programming systems of this sort. Besides Holland's papers, work by Unger[3] and Selfridge[4] is relevant, although they both focus on pattern recognition.

Industrial organizations are examples of highly parallel systems, and face many problems similar to this one. We can expect to be involved in the same crucial issues of centralization-decentralization, coordination, and division of labor. We can also expect the growing rationalization of management, typified by operations research, to be a prime source of clues for programming these systems.

Local Action. The modules of Holland machines work by local contact along paths. Information is designated by pointing to it, rather than by symbols that refer to it. A major gain recently made in programming was finally to create an entity that behaves in many ways like a linguistic symbol (the address with a list of associated information). No simple correspondent to this exists in Holland machines.

Iterative Structure. The modules in a Holland machine are identical, although different Holland machines can be used for the gross areas of Figure 1. This iterative character, besides being suitable microelectronic production technique, provides a "space" already rich in possibilities in which structures can be created at will by information transfer operations rather than actual construction operations (in the carpenter's sense of the word). However, this implies that most modules will have only a small fraction of their componentry utilized.

Fixed Network of Connections. The multidimensional connections between modules seem at first glance to be a blessing. The virtue fades as soon as arbitrary structures must be processed. The information is invariably incommensurate with the fixed network. "Dead ends," in which all the paths to a module are occupied and no way exists to gain access to the module, are continually a problem. Simple data organizations must be used, even though they prohibit clever ways of encoding particular information.

Information Processing

The first level of organization of the machine is the information processing level. It contains means for building up structures of information and for forming processes to operate on these structures. It involves specifying the Holland machine for the Processing area (the Interpreter and the Factory) and indicating how the problems of memory and program organization will be solved. In the following I draw heavily on the programming experience with the problem solving programs described in the next section and on the list languages[5] constructed to aid in programming them. Properties of the Processing area will be accumulated; then a Holland machine for them will be sketched.

Units and a Principle of Homogeneity

In current machines and coding a large discontinuity exists between the machine level and all structures built up from this level. A subroutine is not like an instruction; a double precision number is not like a number contained in one word. The temptation to create Holland machines similarly is strong--to create neat, powerful, elementary operations for modules and then to build up everything from these in structures that look very different from modules. There would be a preferred size of channel (the one between modules), a preferred size of information (the word in the module), and a preferred set of operations (the operation code of the module).

Contrariwise, the Processing area satisfies a Principle of Homogeneity, according to which a module and a higher unit cannot be distinguished by any of the conventions for dealing with them. The basic structure of the Processing area is the following:

1. The Processing area consists of a stack of planar fields, each field containing spatial units. These are rectangular in shape and vary in size. Their boundaries do not cross each other, but units may exist inside other units.

2. The units are connected by paths. These run in horizontal and vertical segments and may be of any width. Two paths may cross each other at right angles, but otherwise two paths may not occupy the same space.

3. Each unit has registers that hold information in bit patterns. There can be units with any number of registers, and registers with any number of bits.

4. Each unit can accomplish an information process involving reception from some paths, transmission along others, and changes in internal registers.

These properties allow units to be built up from networks of other units in order to accomplish more complex processes. Similarly, units may be analysable into subunits connected by paths. If analysis proceeds far enough, units are reached that are not further analysable. These may be either modules or aggregates of modules that, for reasons of speed and space, have been arranged internally in ways that do not correspond to all the conventions of units.

It is now unimportant where the modular level is, and what operations and registers modules contain. I am now free to specify flow diagrams of units with arbitrary functions connected by paths of arbitrary information content. No assumptions need be made about the module operations, except completeness. For the machine, the problem of organizing itself has been much simplified.

Spatial Operations

The machine needs ways to assemble and manipulate units and paths. These should also be homogeneous and should not require extensive knowledge about the distribution of units in the field. The following additional properties seem appropriate:

5. Units can be moved in any of the six directions (vertical movement between layers is needed). The entire unit moves at once with a velocity of one module per basic cycle (variable or faster speeds are complicated).

6. Movement takes place freely into space not occupied by paths or other units. A moving unit coming into contact with a stationary unit or path sets the new unit in motion in the same direction. Two colliding units stop. A moving unit stops upon contact with the inside boundary of a unit.

7. Paths remain connected to moving units, growing and contracting as required.

8. Units can expand and contract. In expansion the same conventions hold as for a moving unit. It sets other units in motion and stops when it contacts the inside of a boundary. In contraction, the inward moving boundary sets the inside units in motion, and all motion comes to a halt when the subunits are jammed together.

9. Units can be copied. This always occurs in the vertical plane, as shown in Figure 3 (a direct horizontal copy is difficult). The figure shows a horizontal copy obtained by copying-up, moving-over, and moving-down.

These capabilities provide convenient ways to manipulate structures. Movement is controlled by always working inside a higher unit as a sort of corral. Rearrangements occur automatically, since all units tend to move out of the way.

Information Structures

The more complex the programs, the more irregular and dynamic are the structures of information built up of units and paths. The machine must be prepared for network-like affairs, as shown in Figure 4. The left side shows a simple tree; the right side indicates that matters are not always so simple. To put such structures (and their more elaborate cousins) into the rectangular grid of the Processing area, a standard outline form is used (Figure 5). Each unit is allocated a horizontal band. Subunits are indented to the right and put immediately below their superunits. The connecting paths use the space made available to the left. As the tree grows all the units are put in downward motion. Each unit can also grow independently to the right in its band. The right hand limit is given by the boundary of the superunit that contains the entire structure.

Splitting Operations. The simple outline is insufficient for general networks, as an attempt to add the paths for the right side of Figure 4 shows. Already paths can cross paths, but they must also be able to cross units. Splitting the unit is one solution:

10. A unit can be split into two parts, either vertically or horizontally, and these two parts may be moved indefinitely far apart. The space between the parts contains only paths. Elongations of the internal paths of the unit run across the split, so that the unit remains unchanged as far as information flows are concerned. Other paths can run along the split, and so traverse a unit.

The outline conventions and the splitting operation allow the machine to put any kind of information structure into the Processing area. This stylized form is often less compact than other forms, but it avoids solving many problems in memory organization.

Definitional Control

No sharp distinction exists between informational units and processing units. Most data is contained in units that actively transmit the data over paths, rather than waiting passively to be read. Both the Interpreter and the Factory are systems of many interconnected, simultaneously active units, and techniques must be provided for control and coordination.

Experience at the programming level is exclusively with sequential control. Sequencing information is given independent of the content of the processes, usually by the order of instructions. A form of sequential control could be adapted to the present machine. Instead, a more radical, fully asynchronous procedure is used. This is called

definitional control, since it rests on the fact that procedures work from the defined towards the undefined. Although similar to existing asynchronous techniques, I am not familiar with any discussion of it for general procedures. However, some work by Steward[6] is relevant.

Any procedure can be put into the form of a hierarchy of subprocedures: The lowest procedures work on the input data; the next procedures work on the outputs from these; the next higher ones work on the outputs of these; and so on. Figure 7 shows the computation of Ke^{-2K} in this form, as it would occur in the Factory. Iterative, recursive, and conditional processes can be analysed analogously.

Assume that each process is sensitive to whether its inputs are available and to whether its output can be accepted. As long as K remains undefined no computation takes place. As soon as K becomes defined, say by some other process feeding a number into it, then the first multiplier can operate, since both of its inputs are defined. Once its computation is finished and it transmits the product, the exponential process can operate; once this is finished, the final multiplication operates. Finally the result is printed. Each process operates as soon as it can--that is, when the situation is defined for it. Until that time, it simply waits.

Suppose a process at the bottom was feeding numbers into K at a faster rate than the printer could operate. Until the printer accepts a result, the top multiplier does not accept the inputs from K and the exponential; the exponential does not accept the product from the lower multiplier; and it, in turn, does not accept the next value of K. The computation automatically becomes paced by the printer.

This form of control partially rationalizes the construction of procedures. No independent sequencing problem must be solved. A process that is put together correctly according to the flow of information is ready to go, and will sequence itself automatically. Since the machine constructs its own programs, this seems a desirable property.

To obtain definitional control, the following are needed:

11. Every register is either defined or undefined; this can be detected on any path leading into or out of the register. (This is simple enough unless there are multiple transmitters and receivers, in which case things get complicated.) No transmit operation will be executed into a defined register (it's already occupied), and no read operation will be executed from an undefined register.

12. Each unit is either active or inactive. If a unit becomes active all its subunits become simultaneously active also. If it is inactive it will not perform any processing (spatial operations excepted), and all of its registers will be undefined. If it is active, it will perform its processes as soon as all paths involved are admissible (inputs defined and output receivers undefined).

Modules and Mass Operations

I must at least indicate how the numerous asserted properties might be achieved in a Holland machine. The ideas seem independently interesting because they involve mass operations, which affect all modules in a region simultaneously and identically. Little experience exists with such operations[3]; but they seem worthwhile exploring.

Each module consists of two parts: a regular part and a substrate. The regular part involves operation codes and storage registers from which processing and memory functions will be synthesized. The substrate takes care of defining units with their spatial and activity properties. The regular part satisfies the homogeneity principle. The substrate achieves the homogeneity principle for aggregates of modules and has no counterpart at the unit level. The regular part need not be defined because of the homogeneity principle, but the substrate needs discussion.

The substrate consists of a state for each module, some operations for changing state, and some paths interconnecting modules. Every module is either:

1. occupied or unoccupied;
2. active or inactive;
3. repetitive or nonrepetitive;
4. stationary or in motion in one of six directions (four horizontal and two vertical);
5. Open or closed, independently, in each of the four horizontal directions to the transmission of influence.

All the modules in a plane are interconnected so that they must all change state simultaneously. If one module goes from inactive to active, all modules go to the active state. This network of influence, which extends from any module to all other modules, can be broken by modules that are closed. If some modules are surrounded by a boundary of modules, all closed with respect to transmission outward, then changes inside affect everything inside but affect nothing outside the module. A unit is a rectangular set of modules with a boundary that blocks influence from the inside out but transmits influence from the outside in. This last condition makes possible hierarchies of units. Anything that occurs in the larger unit affects the subunits (like moves, executes, stops, etc.), but not vice versa (a subunit can move about in the unit without making the unit move).

A module in motion copies its content into

the next module in the given direction. This leaves the motion state unchanged, and motion continues until something turns one of the modules to stationary. As soon as this happens, all modules within the unit instantly become stationary and the unit stops moving.

A unit is made active by changing any of its modules from inactive to active. All the modules are simultaneously made active, and the unit immediately begins to "push" to execute its process, since those parts which can operate go into immediate operation. Two states of operation are needed, repetitive and non-repetitive. A repetitive module remains active even though it has just executed its operation; it immediately tries again. A non-repetitive module sets itself inactive as soon as the operation is accomplished. This change immediately affects the entire smallest unit that contains the module, but it does not affect the modules outside the unit. Incidentally, this shows that influence must depend on state changes and not states, since otherwise the existence of active modules outside would immediately reactivate a module that had just turned itself off.

Although this development is incomplete, the outlines are clear, including the power of the mass operations.

Symbols and the Associative Store

Both human language behavior and the programming of complex systems point to the need for a stock of symbols for general information processing. Symbols are entities with the following properties:*

1. It is always possible to obtain easily a new symbol, not otherwise being used (within the limits of the total stock).

2. It is possible to produce indefinitely many occurrences of a symbol (symbol tokens) and these can be moved around and placed in structures.

3. Given the occurrence of two symbols, it is possible to determine whether they are the occurrences of the same or different symbols.

4. A symbol is a locus of associations. This means the following operations can always be performed:

a. An associative link can be formed from a symbol S to a symbol S', and the link labeled with the symbol S". Only one link with a given name, S", exists for a given symbol, S.

b. The symbol associated to the symbol S by the link labeled S" can be found. This is the inverse operation to the one above, and forms the sole significance of the "associative link." No association with label S" needs to exist, but if it does it is unique.

c. The link labeled S" for the symbol S can be destroyed.

This set of properties ties the concept of a symbol to the concept of an associative memory. Likewise, these properties give one definition of an associative memory, which is equivalent to being able to form and distinguish arbitrary single-valued functions of one discrete variable. It implies that an adequate concept of symbol and reference can be formed from the concept of a locus of associations.* Without arguing any of these points, this formulation is taken as the requirement for giving a machine symbols.

Although the Processing area contains an elaborate system for processing information, it provides nothing that resembles a symbol. One unit designates another, not by having a symbol that names it, but by directly pointing to it. Although the bit patterns can be copied, transferred and compared, they do not allow associations. They are used entirely as objects and never refer to anything.

A separate Holland machine, the Associative Store, provides symbols. The same difficulties in trying to map a free structure into a rectangular grid arise here. To each symbol there corresponds a structure that encodes an indefinite number of associative links, so that the total store is a completely arbitrary, dynamically changing network.

Figure 8 shows a scheme for the Associative Store. The Holland machine for the Store consists of planes of modules. Each symbol has two adjacent lines of modules in the Store for its associations and an unique bit pattern for its token. Each association takes two modules, one in the lower line and one right above it in the upper line. The lower module is concerned with the symbol that labels the link, and the upper module is concerned with the symbol to which the link is made.

Each module has a single register that holds a token. The horizontal data paths, which transmit tokens, are open, so that all the modules in a line see the same token simultaneously. The horizontal command paths are

---

* I reiterate that this formulation, and several others throughout the paper, come from joint efforts with J. C. Shaw and H. A. Simon.

* This is an old notion with psychologists. However, this associative system is considerably more powerful than any proposed for organisms by psychologists. The essential difference is that here the links themselves can be labeled with symbols and hence can take associations.

directional to the right, so that a command issued by a module is seen simultaneously by all modules to the right but by none to the left. The vertical command paths connect each module to its immediate neighbor above.

Each module is either in the "defined" or "undefined" state. Each can record the token on the data path in the register; transmit the token in the register onto the data path; compare the token in the register with the token on the data path; and receive and transmit commands. Three specific patterns of these actions are required for the associative operations. (The 'link token' means the token of the symbol that labels the link; the 'symbol token' means the token of the symbol to which linkage is made.)

1. Form an associative link: The symbol token is put on the upper data path and the link token is put on the lower data path. Each link module in the defined state compares the link token with the one in its register, and if they are equal commands the symbol module above it to record the symbol token in its register. Each link module in the undefined state records the link token in its register, commands the symbol module above it to record the symbol token in its register, and sets itself to be defined. Further, all modules that took action command all modules to the right that have this link token registered to set themselves undefined. This order overrides any action to set a module defined. The net result is that one and only one module-pair records the association. Many modules may have the information in their registers, but only the leftmost one will be in the defined state. This assures that there cannot be two links with the same label.

2. Find an associated symbol. The link token is put on the lower data path. Each link module in the defined state compares this with the token in its register. If they are the same, it commands the symbol module above it to transmit the token in its register onto its data path as the symbol token. By the nature of the system either no module transmits or exactly one module transmits.

3. Destroy a link. The link token is put on the lower data path. Each link module in the defined state compares this with the token in its register. If they are the same, it sets itself to the undefined state.

The scheme is still incomplete in essential respects. Most important is the connection between the token, which is a bit pattern, and the two lines of the associative store that correspond to its symbol. The mechanism described simply delivers and remembers bit patterns in response to bit patterns transmitted on special data paths. Some way must exist for units in the processing area to take a token and construct a path for themselves to the data paths in the associative store. Once they connect they must be able to command the associative operations and receive bit patterns back along the path in return.

This problem is strikingly similar to that of a telephone system. Connections must be made from a set of subscribers, the units, that are widely scattered throughout the processing area. The connections are only temporary; once formed they persist for variable periods. Finally, only a few subscribers want access to the store at any one time. The token bit pattern, then, is a "telephone number" and the units "dial" the associative store.

Problem Solving

The organization of the machine at the information processing level is now clear enough to allow a brief consideration of higher levels of organization. Most important is an ability to problem solve.

Combinatorial Problems

Recent work on heuristic programs provides considerable information on the processes involved in problem solving. Programs have been written for tasks which are sufficiently complex and difficult to require intelligent action by humans. These tasks include theorem proving in elementary domains, chess and checker playing, musical composition, and some management science problems[7]. All these programs formulate the problem in a common way and attempt to solve the problem by a common approach.

The problems all involve a set of objects and a set of operators for producing new objects from old objects. The goal is to find a sequence of operators that produces an object with certain desirable properties, given some initial objects. For example, in chess the objects are positions; the operators are legal moves; and the goal is to produce a sequence of moves that will result in a winning position. In theorem proving, the objects are theorems; the operators are rules of inference; and the goal is to produce a sequence of inferences that will result in the theorem to be proved.

These problems are combinatorial in nature. They involve the selection of a sequence with certain properties out of the set of all possible sequences. The possible sequences can be represented as an expanding tree: given an initial object, application of the operators in all possible ways yields a set of new objects. Applying the operators in all possible ways to each of these objects yields another (much larger) set of objects two steps removed from the starting point. This can be repeated to generate the tree to any depth by taking all of the objects obtained at depth D and applying the operators in all ways to them to get the

objects at depth D+1. If approximately B applications are possible for each object, there are B branches at each node, and of the order of B^D objects are generated at depth D. Thus the number of paths in the tree goes up exponentially with depth.

If the solution lies at depth D, then of the order of B^D paths must be examined to discover the solution. It is easy to show that immense numbers of paths occur for any real problem. For example, chess has a B of about 30 and a D of about 80, yielding about 10^{120} paths. Since the entire tree cannot be explored, it is necessary to be sophisticated in searching--to look only where the answer is. In hard problems no single item of information tells exactly where the desired object is, and many different bits of information, each of low calibre, must be used to restrict the search to a reasonable number of branches. These odd bits of information and the ways of organizing them are called heuristics. An example from geometry is "Don't try to prove two angles equal unless they appear about equal in a well drawn diagram." This heuristic reduces the average number of branches at each node by rejecting some that would otherwise have to be explored. The effect of heuristics is to reduce the parameters, B and D, in the search formula, B^D, leaving the form of the problem unchanged--search in an exponentially expanding space.

Problem solving by heuristic search is not the universal solvent that dissolves all particular methods. It is the last ditch defense when no special methods are known. The machine has many special methods, from analytic differentiation to Newton's method, and uses them wherever appropriate. It also needs some model for all the other problems that arise. Heuristic search is the best available.

Parallel Search

The problem is to organize the machine for problem solving. In current serial machines a single program, the problem solver, conducts the search. It carries with it an accumulation of information about the maze and its activities in it, and combines this with its more general heuristics to decide which branches to explore. If on the average it takes C time units per node, then the search time is of the order of CB^D.

Being parallel, the machine is not limited to a single problem solving process. With P problem solvers, the time of solution is of the order of $(C/P)B^D$. This speeds up the process by a factor of P, but does not affect the exponential, which governs the growth of the search tree. Coordination problems also exist, since no one problem solver accumulates all the information. Each must contain additional processes for transmitting its information and analysing the information received from the other problem solvers.

Parallelism can be pushed further. Each problem solver can have the task of not only creating the B branch points, but of making B copies of itself and of setting one copy to work independently on each branch. Now every branch point is active, and even if it takes an additional K time units to produce each new problem solver, the total search time to depth D is of the order of (C+BK)D. If this equation were only true, all the world's problems could be solved in a day! It claims that the search time, instead of being exponential with depth, is linear with depth. Thus, at a microsecond per chess position, it would take only a few milliseconds to compute the perfect game. The absurdity of this is manifest, but it is instructive to examine why it cannot hold.

Parallel computation trades space for time. The space required to hold the data of a problem tree also increases exponentially with time (although serial strategies exist that keep the space proportional to D or BD). In the fully parallel case space must be found to put the exponentially increasing number of problem solvers. Since space becomes available at the rate at which units can move apart, the ultimate determiner of the growth of the tree is the velocity of movement. Since this is always limited (to one module per cycle for this machine), the time to search the tree remains exponentially related to depth.* This is simply the Malthusian problem for programs, in which a population of programs reproduces itself geometrically, while its sustenance, space, grows only linearly.

Parallel search is still an effective strategy. By suitably spacing the initial points, extremely rapid exploration can be had of the first part of the tree. The devices described earlier for information processing make the mechanization of these procedures straightforward. Independent problem trees can exist simultaneously in the processing area, each encased in an expanding boundary. The entire set of growing units moves around in the yet larger bounded area that contains them all, utilizing the total space in a reasonable, if not optimal, fashion.

Supervision

Earlier a principle of awareness was stated by which the machine should be able to detect the consequences of its actions and have some ability to correct, modify, or prevent them. This covers reliable computation as well as more subtle things, such as inconsistencies in the user's data.

---

* Even if space expands in N dimensions, the volume cannot grow fast enough to keep up with an exponential demand. However, added dimensions help. If N dimensions are available for expansion, the search time becomes like $B^{D/N}$.

An approach to this problem is to provide reliable containment of consequences and supervision by independent processes that are sufficiently intelligent to handle the errors that occur.* The unit boundaries, impervious to outward flowing influence but transparent to inward flowing influence, are naturally suited to contain consequences. The supervisory processes are units that sit outside the units they supervise, checking and correcting certain classes of errors. Not all errors can be checked, since only a finite amount of effort can be spent on supervision, and there are many errors the machine cannot possibly rectify.

Two aspects of the problem can profitably be discussed further at this level. First, who shall guard the guardians? The simplest answer is to have another supervisor, as shown in Figure 9. Each lower system is contained in a walled cell, and the supervisor outside has paths into it for monitoring it. The problem is to avoid the implied infinite regress. One solution is to adhere to the rule that any unit that acts outside itself must be supervised, but any unit that only takes in information need not be supervised. In Figure 9, assume the action program is for a user and will result in an external response. It must be supervised, so Supervisor #1 must exist. As long as Supervisor #1 takes no action on the action program, it need not be supervised. The moment Supervisor #1 goes into action, Supervisor #2 must come into existence, and so on through an upward recursion of supervisory routines. As units complete their tasks the supervisory units disappear again. If supervisors are constructed to act rarely, say only when errors occur, then the height of the recursion is governed by the successive powers of the error probability and is kept under good control.

The second aspect is the nature of general computation. Current programs normally consist of a single procedure, all of which must be gone through to produce the final result. Contrariwise, most procedures carried out by people are done in the context of a tree of alternatives, where at each step, although one action is taken, others are possible to achieve the same end. Thus, if the computed value of the cosine is wrong, it can be recomputed; if still wrong, it can be looked up in a table; if still wrong, the look-up procedure can be checked; and so on. The strategy is not to correct errors directly, but to bypass them and produce the result a different way. All computations are to be carried out as a problem solving search to discover the solution. The nodes in the search tree are stages in the computation; the branches are the possible computational actions. It makes little difference whether a path to the answer is rejected because it is "a wrong computational step" or because it is "a right computational step with an error."

* An alternative approach, which has been much pursued, is to achieve the reliability at the level of the smallest component by such means as error-correcting codes and multiplexing. This approach tries to solve the problem without recourse to the larger context in which the computation occurs.

Interpretation and Production

There are two conflicting requirements for this machine. For volume processing of data, its programs should be as efficient as possible. But, since these programs come to it from outside, they should be as easily communicated as possible. The external language and the internal language are completely divorced from each other in order to deal with the conflict. The external language is communication oriented; the internal language is production oriented. The intelligence of the machine mediates between them.

The general operation of the machine is as follows: The Input-output area contains expressions in the external language, generated by both outside users and internal processes. The expressions are analysed by a process in the Interpretive area, which builds up a process in the Factory that will take the action appropriate to them. The Program Store provides the parts from which these action processes are constructed, ranging from small parts, like multiplication, to entire programs, like matrix inversion. These parts are forms; they are identical to the final process except for certain variable units and paths, which must be specified to make an actual process. They are moved out of the Program Store along unoccupied planes and copies of them deposited in the Factory. Then the various unspecified units and paths are identified and brought together. For this purpose, the variable parts of the forms carry descriptions of their function. For example, the form for the process in Figure 7 might be xe^{ax} and would look identical to Figure 7 except for an x and an a in place of the K and -2, and an expression xe^{ax}, in place of the print.

Once a complete process is assembled, and the interpreter decides the interpretation is correct, it is executed. The interpreter remains intact during execution to handle difficulties, to remove the action process at the end, and to decide if anything should be salvaged for later use.

The behavior of the machine is a collection of such efforts, each composed of an action process governed by expressions in the external language and mediated by an interpretive process. Some are built up in response to simple external requests and are concluded almost immediately and some involve long

production runs. Others constitute strategies of internal operation, such as the allocation of effort between production and improvement. The rate at which the machine processes information is determined by whatever limit is reached first: production time, Factory space, input-output equipment, interpretive time, and so on.

Productive Efficiency

Little study has been given to the theory of efficient computation.[8] However, certain general features of the machine permit efficient programming.

One source of inefficiency is excessive interpretation. This generally occurs in repetitive processing, where interpretation of the same procedure occurs over and over again. It is avoided by using the first interpretation to set up an efficient procedure, so that subsequent interpretation isn't needed. The division of labor between the Factory and the Interpreter permits this efficiency. A more subtle version of interpretive inefficiency is the use of symbols and other indirect forms of designation. The use of direct paths in the Factory permits the elimination of most of these costs.

The machine operates through the use of standard building blocks, which it puts together in various standardized ways to construct yet bigger building blocks. Often a process so built up can be made more efficient, both spacewise and timewise, by reconstructing it out of more microscopic processing units. One of the continual activities of the machine is to invent more efficient units for processes that are important to it. Independent of immediate demands, it takes program forms from the Program Store, attempts to improve them, and returns them to the Store. The discrete nature of the machine allows the problem of improving a program to be phrased in terms similar to the problem of playing chess or proving theorems.[9]

Interpretation as Problem Solving

The only acceptable dictum for designing the external language is that it cover the full range of expressive devices. It is easy to state some of these. We want the ability to substitute expressions for terms; to add new terms and abbreviations; to express new language conventions; to make ambiguous and vague statements; to state problems as well as procedures; to mix syntax and semantics; and so on. Each of these is needed, not only for communication, but because the machine uses this language for all its internal control and problem solving. Consider the possibility for ambiguity, which may seem an odd property to desire. If the machine cannot state an ambiguous notion, it can never get started on a series of successive approximations to achieve an exact notion. And if the man must be unambiguous in communicating to the machine, he can never get it to help him formulate a difficult computation.

A discussion of the mechanisms required to interpret these features is beyond the bounds of this paper. However, one general mechanism is relevant to all of them--the treatment of interpretation as problem solving. The interpreter in the machine for a given expression is a branching tree of processes. Each node of the tree represents a set of hypotheses about the meaning of the expression (more precisely, about the implication of the expression for current action). Each branch represents an additional hypothesis about the meaning, so that different branches at a node represent alternative interpretations of the expression. The processes at the nodes carry out the analysis of the expression. Each is specialized to the hypotheses at the node, and operationally represents these hypotheses. These processes detect information that would confirm or reject the hypotheses, and if the latter, terminate interpretation along their branch. They find new information that leads to additional hypotheses, and create processes to continue the interpretation within these more restrictive bounds. They also take direct action in building up the action process in the Factory.

Viewed this way, interpretation is the process of discovering by a series of interpretive acts a final action process that is consistent with all the information extracted from the expression. The search goes on in parallel with all alternative interpretations being carried forward. If the expression is clear, then one line emerges cleanly and immediately. If the expression is full of little errors, then many short branches occur at each node, all but one of which prove false. If the expression is ambiguous, more than one line of interpretation continues through to a complete action program, and information from more distant sources must be sought.

To give one simple example of this process, consider the interpretation of the request "Compte Ke^{-2K} for K = 1 and 2." Assuming that nothing is known about the expression, interpretation starts with a standard process. "Compte" is recognized as not corresponding to a word, and the interpretive hypothesis is taken that the user meant "Compute." On this basis a new interpretive process is set up that assumes some formula for numerical computation will follow. When Ke^{-2K} is found this is confirmed. This action is dependent on the first. If the initial word had been "Differentiate," then a form for numerical computation would be wrong. Also, the hypotheses needn't have been confirmed. If the original expression had been "Compte is to mean the same as Compute," then a numerical computation was not desired and an alternative hypothesis is needed, say that "Compte" is being used as a name for itself. Getting the form constitutes a second interpretive act and another interpretive process is created which matches the form to the expression. This results in additional interpretive acts as x is identified with K and a is identified with -2. The final inter-

pretive act occurs when the print process is put in. Since no indication is given for what the machine is to do with the computation, this is a hypothesis that the user wants the information given to him immediately. Its confirmation must wait until after the action process has been executed, and the user reacts to the result.

Although this mechanism does not solve the problems of interpreting the devices mentioned earlier, it seems to be a necessary feature of an intelligent interpreter.

Last Thoughts

Speculation stops here. The problems of making an intelligent technician from a Holland machine form an expanding tree which could be explored both broader and deeper.

Each of the topics touched on in the paper was left incomplete: the nature of the modules that can achieve the properties stipulated; the nature of the telephone exchange; the mechanisms for interpreting the external language; and so on. Major aspects were not even mentioned: the problem of going from an object to its symbol; the problem of organizing the machine's knowledge; the problem of context; and so on. But at this stage of development, with the microelectronic techniques still a little ways distant, it is appropriate to end with a sense of incompleteness and need for further exploration.

References

1. Holland, J. H., "On Iterative Circuit Computers Constructed of Microelectronic Components and Systems," These Proceedings. See also, "A Universal Computer Capable of Executing an Arbitrary Number of Sub-programs Simultaneously," Proceedings of the 1960 Eastern Joint Computer Conference, December, 1959.

2. Churchman, C. W., "On a Potential Customer for an Intelligent Technician," These Proceedings.

3. Unger, S. H., "A Computer Oriented Toward Spatial Problems," Proceedings of the 1958 Western Joint Computer Conference, May, 1958.

4. Selfridge, O. G., "Pandemonium," Proceedings of the Symposium on the Mechanization of Thought Processes, Teddington, England, 1959.

5. McCarthy, J., "Recursive Functions of Symbolic Expressions and their Computation by Machine," Quarterly Progress Report No. 53, Research Laboratory of Electronics, Massachusetts Institute of Technology, April, 1959; or J. C. Shaw, A. Newell, H. A. Simon, and T. O. Ellis, "A Command Structure for Complex Information Processing," Proceedings of the 1958 Western Joint Computer Conference, May, 1958.

6. Steward, D. V., "On an Algebraic Foundation for Constructing Optimum Algorithms," General Electric Technical Report, GEAP 3159, August, 1959.

7. For example, see H. Gelernter, "Realization of a Geometry Proving Theorem Proving Machine," Proceedings of the International Conference on Information Processing, UNESCO, June, 1959 (in press); A. Newell, J. C. Shaw and H. A. Simon, "Chess Playing Programs and the Problem of Complexity," IBM Journal of Research and Development, 2, 4, October, 1958; or A. Samuel, "A Checker Playing Program," IBM Journal of Research and Development, 3, 3, June, 1959.

8. However, see Jeenel, J., "Programs as a Tool for Research in Systems Organization," IBM Journal of Research and Development, 2, 2, April, 1958.

9. Kilburn, T., R. L. Grimsdale and F. H. Summer, "Experiments in Machine Learning and Thinking," Proceedings of the International Conference on Information Processing, UNESCO, June, 1959 (in press).

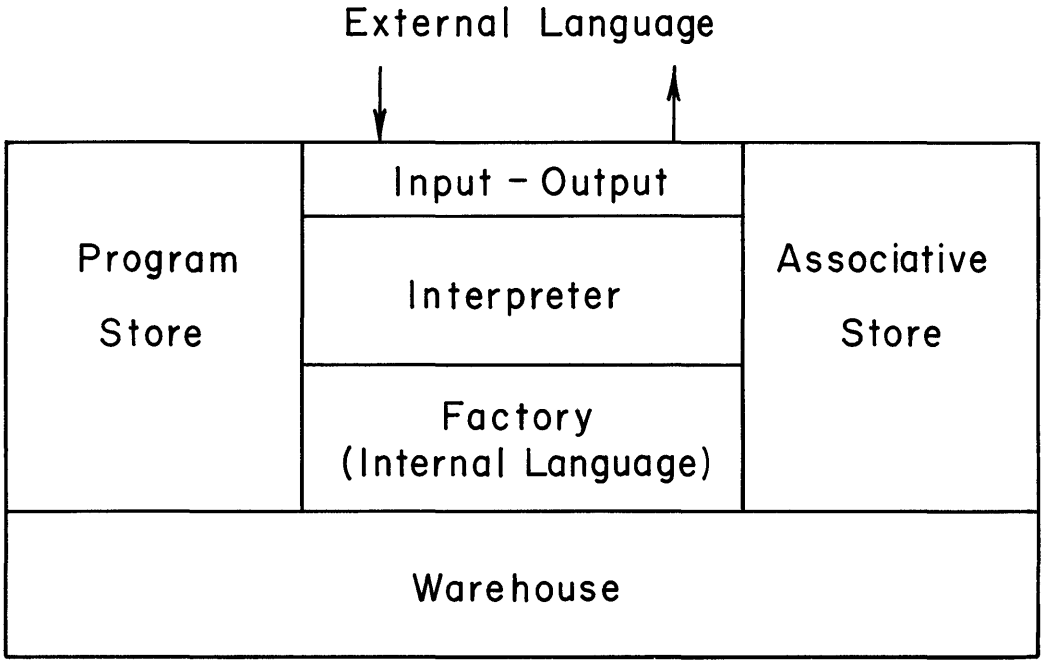


Figure 1 Layout of the Machine

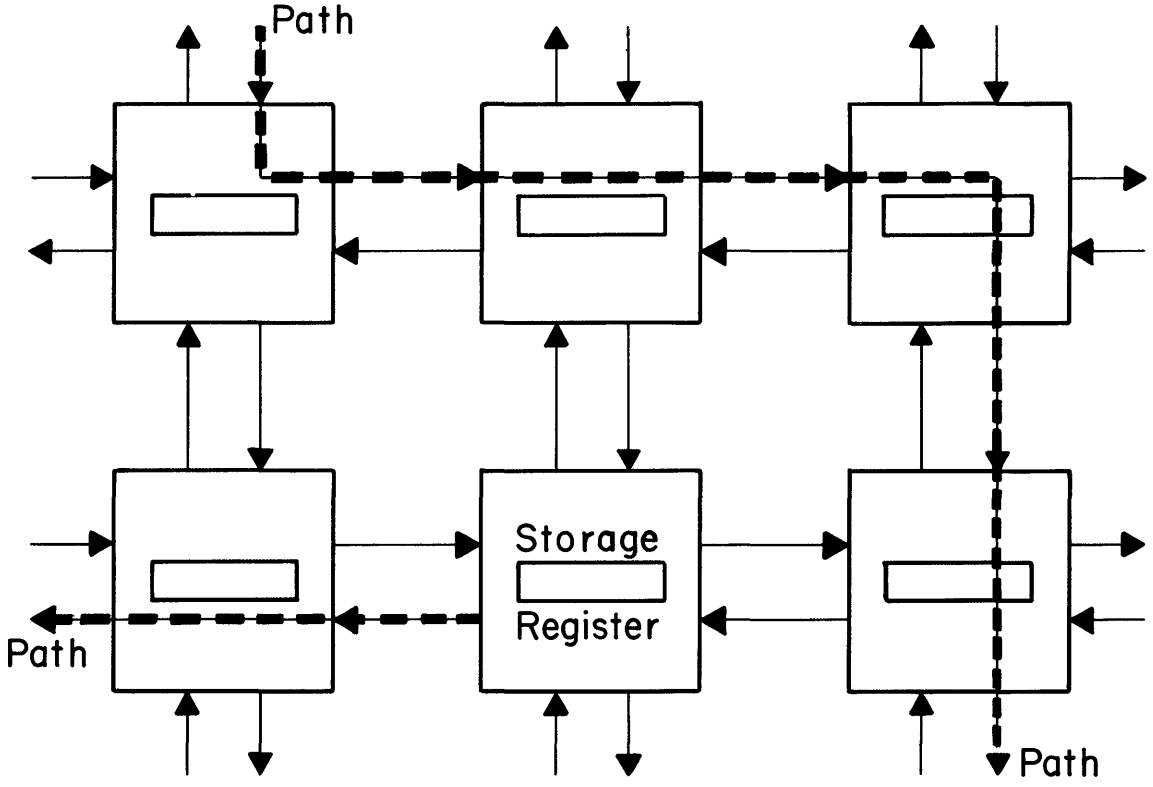


Figure 2 Structure of a Holland Machine

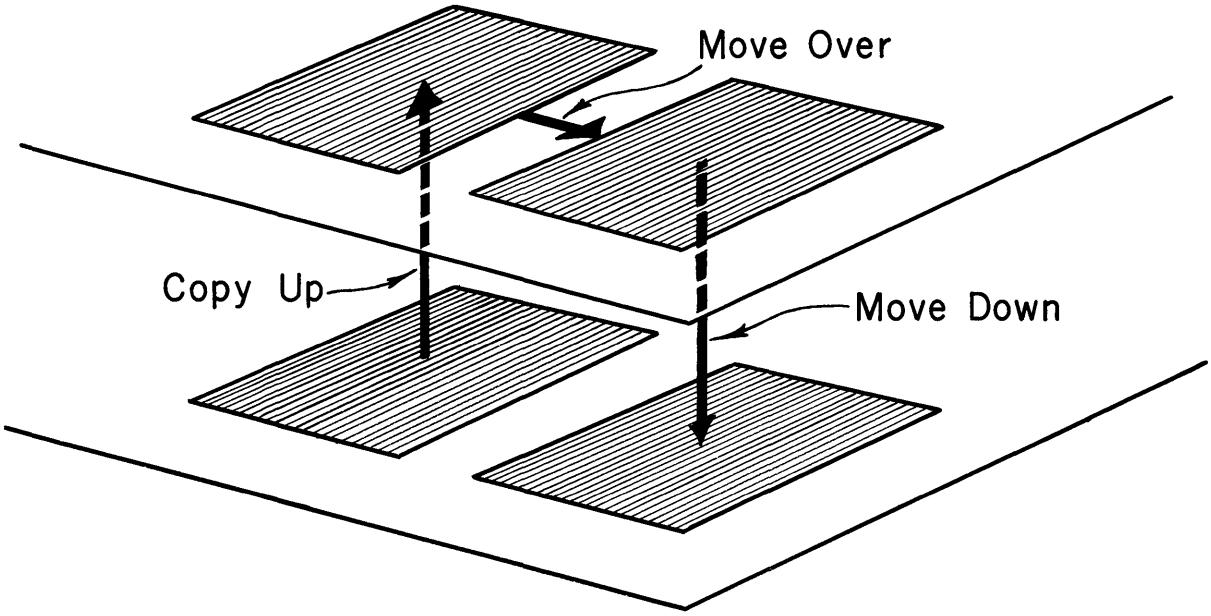


Figure 3 Spatial Operations

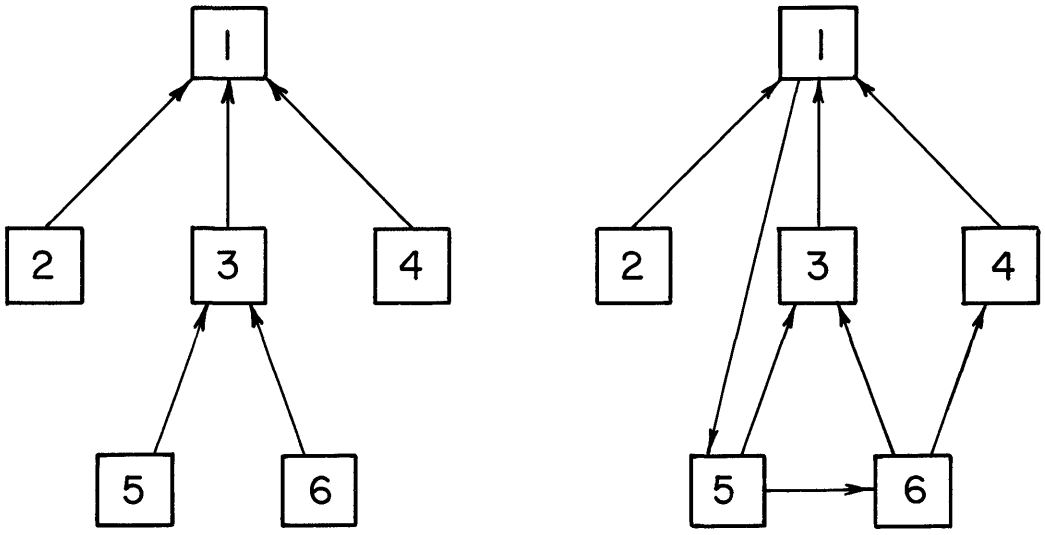


Figure 4 Information Structures

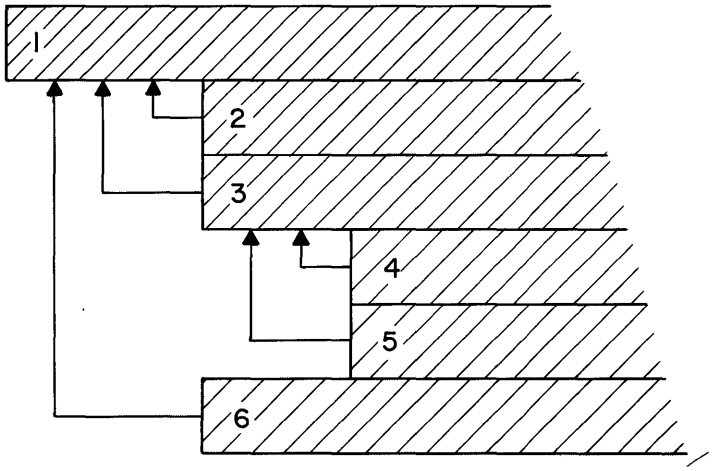


Figure 5 Outline Form for Information Structures

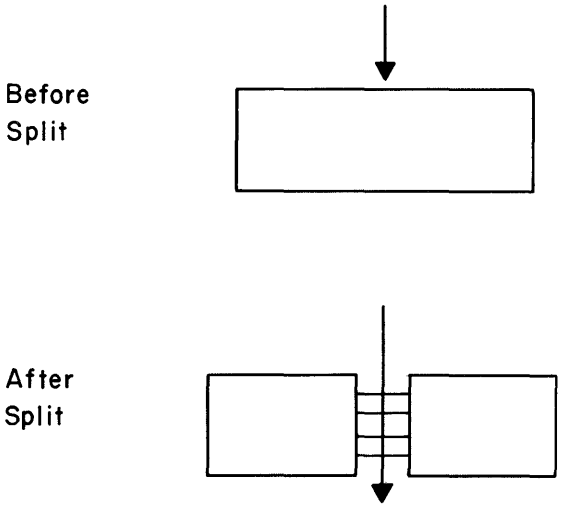


Figure 6 Splitting Operation

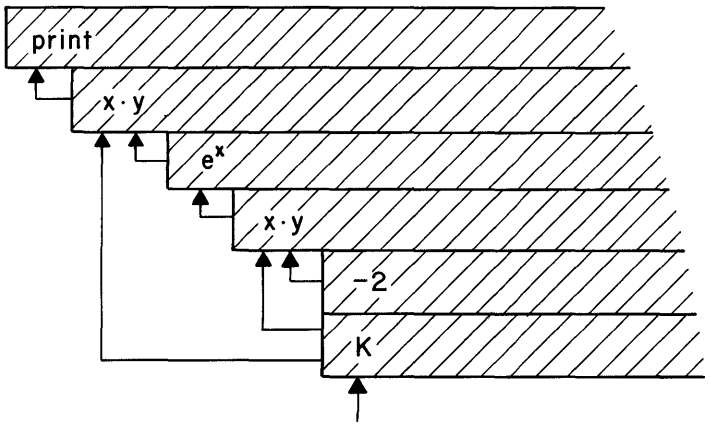


Figure 7 Definitional Control

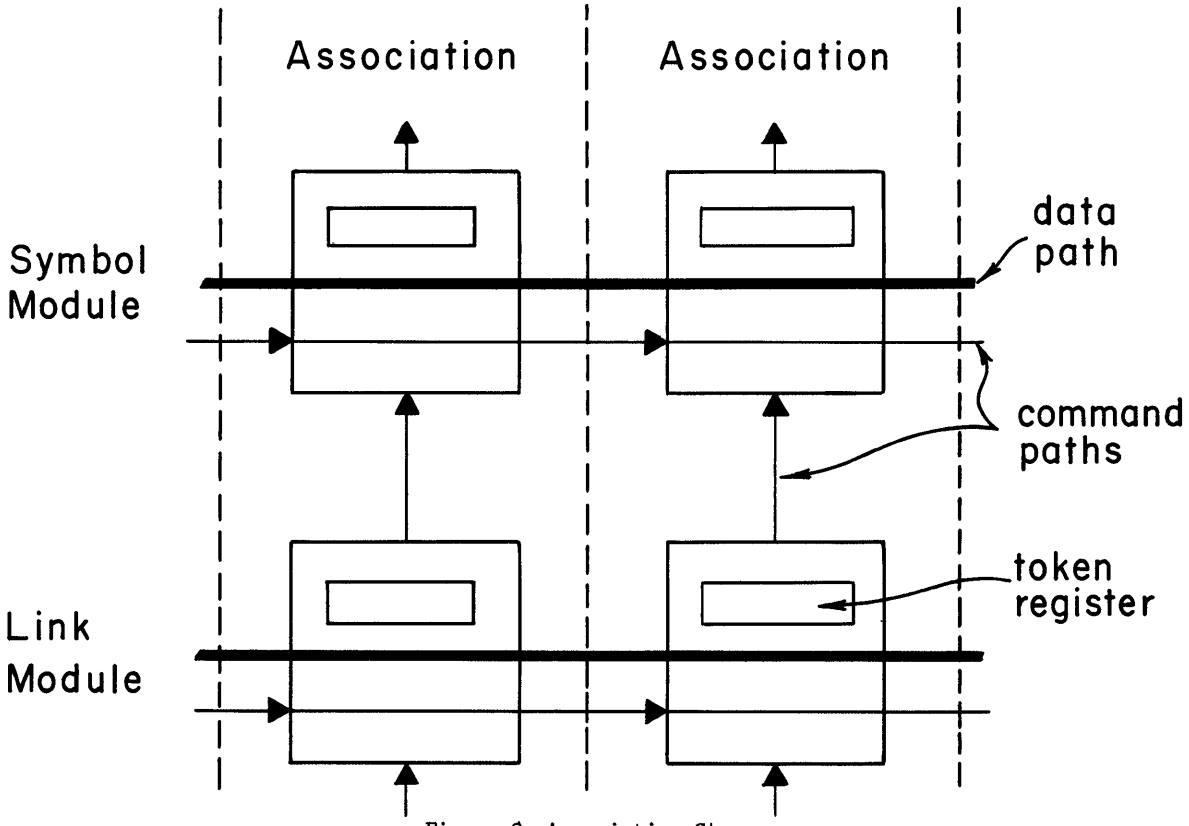


Figure 8 Associative Store

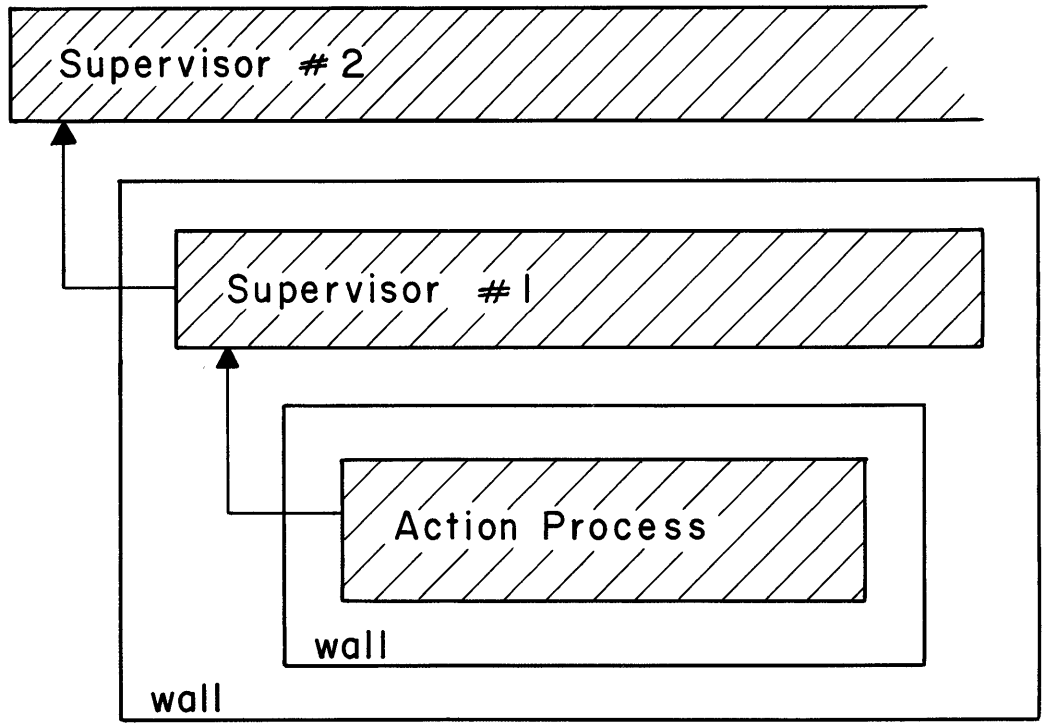


Figure 9 Supervision

ON A POTENTIAL CUSTOMER FOR AN INTELLIGENT TECHNICIAN

C. West Churchman
University of California
Berkeley

The manner in which the other panel members have answered the questions posed to them permits me to consider my own without constraints. I've asked myself, therefore, to name one class of potential customers of an intelligent technician-computer, whose problems are clearly important, clearly in need of computational aids, clearly demanding the kind of intelligence which Newell describes.

Now one ready reply to such a query is in the form of "more and better"; one can scan existing customers and reflect on the ways in which they can more fully and more adequately be served: the physical scientist, the industrial or government data processor or decision maker; the behavioral scientist. But whatever one might say of these consumer needs for more and better computational aids would not, I think, catch the spirit of free speculation which the role assigned to this paper is supposed to capture.

Instead, suppose one begins by ignoring the product for the moment, and asking instead: what is the most difficult problem the human being faces today? The first reply that comes to mind seems also to be the right one: the bargaining problem.

A bargaining problem is one in which the decision to be made must be composed of the decisions of two or more individuals with partially conflicting objectives. It is illustrated in purchasing, voting, labor-management disputes, legislation, treaties, and the present luke cold war. Our ignorance concerning optimal methods of solving bargaining problems is perhaps best illustrated by the fact that in the present international situations all parties are far more interested in research that will make each separately more powerful, than they are in research that will make each a better bargainer.

Suppose we try briefly to break down the components of the bargaining problem, and then ask whether an intelligent technician might or might not be of service. (1) Each bargainer must try to determine the set of alternative decisions which he can make; (2) Each bargainer must try to determine the set of alternative decisions the other parties can make; (3) Each bargainer must try to construct an outcome matrix which predicts the consequence over time of each combination of decisions; (4) Each bargainer must try to estimate the likelihood that specific choices will be made by the other parties.

This list is obvious enough, and is not complete. The other items of the list are more subtle, and consideration of them can momentarily be postponed. If we concentrate, then, on the obvious, it is not difficult to see that in real bargaining situations there is at least a potential use for the kind of computational mind which has been discussed in this panel. The alternative sets are practically never given. Indeed, if we think of the management-labor example, it's clear that year by year the parties learn more and more about the possibilities open to them. If we think in terms of new concepts, labor has learned the walk-out, the general strike, the sit-down, the fringe benefit, the escalator, etc. It is reasonable enough, I think, to ask whether a mind adept at developing new strategic concepts more rapidly than human chess players, might not be equally adept in the area of collective bargaining. It is also reasonable to suppose that in real bargaining situations the parties are only conscious of a very limited part of the outcome matrix, even if this matrix is confined to alternatives of which they are aware. In the recent steel strike, considerable attention was paid to the inflationary spiral by the government and management, even though everyone could sense that this was only part of the description of some of the outcomes. Indeed, it is safe to say that most of us have only a very vague idea of what the parties themselves thought the consequences would be. Here again each party might well benefit from an economic simulator which would start with the economic axioms the party was willing to admit and forecast the conditions that would follow any agreement for suitable time spans. Labor-management bargaining will be with us for a long time, and the research need not be restricted to the period when the actual bargaining takes place. As for the fourth point, - the likehood of the other parties' strategies - this too fits the mentality of the intelligent technician whether or not the other parties are also making use of similar devices.

All I can do here is to pose a suggestion, and not answer it. A number of obvious difficulties in using computers in bargaining would occur to the layman; it would be a matter for bargaining experts to determine whether the costs are properly balanced by the returns. Possibly some controlled experiments of bargainer with and without computer aids would be valuable in this regard. However, instead of loading this argument with the weight of technical details, it seems wiser to complete the broad picture of the suggestion by turning attention to the more subtle issues mentioned earlier. These are the issues of agreement and cooperation. We add to the list already given the following: (5) The parties may decide where they agree and disagree with respect to their determinations of the alter-

natives, the outcome matrix, or the likehood of alternative moves. (6) The parties may determine the alternative methods of settling their dispute. (7) The parties may decide where they agree on alternative settling methods.

These three additions to the list might be described as aspects of "meta-bargaining", for they are concerned with the bargaining of general methods of settling specific bargaining issues. It is not clear of course that a step in this direction is necessarily desirable for all parties concerned. In the labor-management issues, the public presumably wants all parties to do all three of these things. It wants the "facts", which is its vague way of saying it wants to obtain an outcome matrix that all parties will agree upon. It is not misusing terminology, I think, to suggest that morality demands of bargainers that they proceed when necessary to meta-bargaining; such terminology suggests that the rest relevant of humanity is best served if this step is taken. The last step if successful may be the basis of "solving" bargaining problems, for though the parties may not agree individually on what ought to be done in a specific instance, they may agree on a method of settling their specific disagreements. It should be pointed out that in most real bargaining situations we don't know that this last step has failed, because we don't know that the other steps were really carried out. Specifically, we don't know that the parties really disagreed on the facts.

In all this process too the potential of the computer seems real. Specifically, we have learned new concepts of settling disputes: the vote, the fact find board, legislative action, etc. Can we learn any faster, and could a computer technician help speed the process? If we allow this free speculation to proceed to its ultimate limit, could any of this assistance be applied to the bargaining of nations? Would an intelligent technician be of any real value over the years to the U.N.?

One footnote should be added. It will be noticed I'm sure that no where have I mentioned the problem of values, though this is undoubtedly central to the whole bargaining problem. Further, I did not include a pay-off matrix as one of the aspects of the bargaining problem. I did not, because I believe that emphasis on pay-off matrices inverts the bargaining problem unrealistically. If we begin our analysis with the pay-off matrix, we must ask how the parties will settle, given the utilities that accrue to each as a result of a specific set of choices; whereas it seems to me that we should more naturally ask what choices the parties would adopt if they realized the alternatives open to them and the consequences of each alternative. The way in which they would behave as they learned more about the situation is evidence of their values; values are research devices for predicting behavior when the parties are enlightened. Therefore I don't think it's wise for the parties to try to visualize their problem as a game with specific payoffs, because if they really understood the payoffs they'd really understand how each would behave in the bargaining situation where the facts are agreed upon. If the original payoff matrix includes only the utilities of each bargainer under the supposition that he were the sole individual involved, then such a matrix contains very inadequate information concerning the values of the bargainers in a true bargaining situation. Having said this much, I should also remark that my seventh point is a very critical one. Just how critical it is to a behavioristic theory of values can be demonstrated by posing a postulate of this theory: if all the parties to a bargaining process cannot agree on a method of settling disagreements, given that they all agree on the factual outcome matrix, then the values of the interested parties are indeterminable.

REAL-TIME AUTOMOBILE RIDE SIMULATION

Robert H. Kohr
Engineering Mechanics Department
Research Laboratories, General Motors Corporation
Warren, Michigan

Summary

Automobile ride studies have progressed from simple linear analyses to complex nonlinear simulations performed by real-time computers. A complete study of automobile ride must include the road (input) waves, car suspension dynamics, and passenger response to the resulting ride motions. A Ride Simulator capable of carrying two passengers and subjecting them to realistic ride motions offers a new approach to automobile suspension design. The basic simulator, which consists of a magnetic tape input unit, an analog computer for determining car ride motions in real time, and a servo-controlled motion simulator, are described in detail.

Introduction

The study of automobile ride began shortly after the first automobile was built and has continued to the present day. Due to the complexity of the suspension system much of the work on ride has necessarily proceeded on the basis of empirical experience. In general, ride has been improved largely by means of specific cut and try modifications on prototype vehicles under the guidance of expert test driver opinion. Numerous attempts have also been made to analyze the suspension system mathematically but these have been only partially successful. It is the purpose of this paper to define the ride problem, to relate briefly the significant analytical efforts of the past, and to give in detail the use of modern high speed computing equipment in a new attempt to solve it.

Definition of Ride Motions

The automobile considered as a rigid body may experience any of six different types of motion. In the automobile these six motions are denoted by the linear velocities u, v, and w and the angular velocities p, q, and r as shown in Figure 1. The axis system is fixed in the car and moves as the car moves. Because the automobile possesses a plane of symmetry, the symmetric degrees of freedom (u, w, and q) do not couple with the asymmetric degrees of freedom (v, p, and r). Consequently, it is possible to break the problem of automobile dynamics into two separate problems: the first involving motion in the plane of symmetry called longitudinal motion and the second involving motion out of the plane of symmetry called lateral motion.

The fore and aft motion in the plane of symmetry, denoted by u, relates to the acceleration and braking characteristics of the car and is called the "performance" motion. The up-and-down and pitching motions in the plane of symmetry, denoted by w and q, are together considered the primary "ride" motions of the car, while the three remaining motions (yawing, rolling, and sideslipping) are generally associated with steering control and are called the "handling" motions.

Since the "ride" and "handling" motions are largely uncoupled it is possible to study them separately. Consequently a study of "ride" can be begun by studying only two motions, bouncing and pitching.

Historical Development of Ride Analysis

The earliest work of importance in analyzing ride was done in England by Rowell. [1] This work was done at such an early date (1922) that shock absorbers had not come into general use and the only damping in the suspension was due to friction in the leaf springs. Rowell's analytic work in the realm of undamped, free vibration was later extended by Guest[2] and Olley[3]. As is frequently the case, the automobile suspension system is such a nonlinear complex that analytic efforts to understand suspension performance quickly bogged down. It became apparent that computer simulation of some sort would be required to solve the suspension problem. The first analog computer used in suspension analysis was a mechanical differential analyzer built specifically for the task by Schilling and Fuchs in 1941.[4] It solved the suspension problem for only a single degree of freedom but did permit the inclusion of a nonlinear characteristic for the shock absorber. A particularly important use of this computer was the determination of transient vertical acceleration characteristics as experienced by the passengers, and how these transients could be varied by altering the shock absorber characteristics.

A further advance in the analysis of ride was made by Jeska in 1953 when a four degree of freedom model was studied.[5] The motions considered were the vertical motions of the front and rear wheels together with the bounce and pitch of the body. The analysis included an actual road wave, measured by a photographic technique and a transportation lag equal to the car's wheelbase divided by the car's forward velocity. The incorporation of the transport lag was made necessary by the fact that a road disturbance which excites the front end of the car will also excite the rear of the car at a known later time. Good agreement was obtained between the analog simulation and experimental data obtained by driving a car over a known stretch of road.

The General Ride Problem

It can be seen that prior to the beginning of this study the analysis of ride had been concerned primarily with the suspension system. A small amount of work had been done on passenger tolerance to vibration but not in sufficient detail to be useful. There was little or no data on the types of irregularities present in various road surfaces and because of this lack, investigators were using single sine waves or step bumps (curbs) for input disturbances. Viewed in an overall sense the ride problem can be seen to be threefold in nature because it concerns the road, the car, and the passenger. Consequently, a complete analysis of ride must take into consideration all three quantities.

Ride Analysis Through Simulation

Subsequently it became apparent that high speed analog computers could be successfully employed to solve the equations of ride motion in greater complexity than had previously been attempted. Tape recorders had been developed satisfactorily such that if a road wave could once be recorded on magnetic tape, it could then be "played" into the analog computer as an input disturbance to the suspension system. By using two pickup heads the road wave could be determined at two points analogous to the front and rear wheels of the car. There only remained the problem of inserting the passenger into the analysis. It seemed reasonable that the best way to include the passenger in the analysis was to put him in physically. This has been accomplished by constructing a servo-controlled motion simulator which can seat two passengers and which follows the motions determined by the analog computer. An overall schematic view of this system, which is called a Ride Simulator, is shown in Figure 2. It is composed of

(1) A tape recorder to provide road wave inputs into the system. The signal on the tape is picked off at two points simulating the front and rear wheels of the car. The tape speed is variable and by speeding up the tape, the car being simulated is also "speeded up".

(2) An analog computer which takes in the signals from the tape recorder and, from these inputs to the car's suspension, determines what car body motions would result. The analog computer runs in "real time", which means that the ride responses that are generated by the computer occur at the same rate as would the equivalent ride responses of a real car on a real road.

(3) A motion simulator to reproduce the car body motions determined by the computer. The simulator is comprised of a portion of a standard car body and is driven by electro-hydraulic servo-mechanisms whose inputs are output voltages from the analog computer. Two passengers may be accommodated in the motion simulator and the driver may control the speed of the car by pressing the brake or accelerator which sends a signal to a small computer which determines the resulting car speed and which, in turn, sends a speed signal to the tape recorder.

Since the Ride Simulator is comprised of three rather complex components, the following sections of this paper will be devoted to a detailed description of each part.

Tape Recorder

Determination of Road Profiles - As previously noted, a source of real difficulty in simulating automobile suspensions has been the lack of detailed data on road profiles. Numerous efforts have been made in the past to determine road profiles, among them are direct surveying methods and specialized surveying methods using a light beam as a space reference. In general, the shortcomings of these methods are that they are tedious and that they are not adaptable to long stretches of road -- say ten or twenty miles. General Motors is currently exploring another possibility -- that of obtaining the vertical accelerations induced in a towed wheel as it is rolled along the road and then integrating the acceleration twice to obtain displacement. The wheel used in this work, shown in Figure 3, is

essentially a bicycle wheel fitted with a special tire. Also shown in this figure is a standard fifth wheel which is used to generate a speed signal. The wheel is preloaded with a low rate spring to maintain contact with the ground. The wheel-spring combination acts as a low-pass filter with a break frequency of about 60 cps. The accelerometer output is passed through a high-pass filter that has a slope of 18 db per octave and a break frequency of one-half radian per second. This filter is composed of operational amplifiers, as shown in Figure 4, and the double integration is performed within the loop. In Figure 4, S^2X represents the acceleration input and Y is the displacement output. Although there remains considerable work to be done before this system is completely satisfactory, a number of successful short time runs have been made. An oscillograph record of one of these runs is shown in Figure 5. The road profile used in this test was a home-made triangular bump. The acceleration resulting when the wheel rolled over the bump is shown in the center of Figure 5 and the displacement, recovered successfully by a double integration, is shown at the top. The measure of the wheel suspension displacement is used primarily as an event marker on the oscillograph record.

In addition to recording particular sections of various road surfaces, it is also possible to characterize the road surface statistically. Then by use of electronic noise generators and appropriate filters, it is possible to determine ride responses for a known statistical input without recourse to recording of actual roads. Although there is very little data available on highways, the statistical approach, employing power spectral densities, has been satisfactorily used to determine ground loads induced in aircraft when taxiing over rough runways.[6]

Time Delay Generation - Once the road waves have been recorded on magnetic tape, there remains the problem of providing the inputs to the front and rear suspension. For cars with equal front and rear treads, the road input functions for the wheels on each side of the car are identical but displaced with respect to each other by the length of the wheelbase. Previously a number of special purpose road function generators had been built in order to overcome this obstacle. In the case of the Ride Simulator, a general purpose magnetic tape unit was utilized to provide the delay in addition to acting as the input road wave generator.[7] This unit, shown in Figure 6, utilizes a modified commercial tape transport with two magnetic heads. The lower head is used for recording purposes, and both heads are used for reproducing purposes. When the signal is picked off by both the lower and the upper heads there is a delay in the signal of $T = d/V$ seconds, where d is the distance between the heads and V is the tape speed. The tape speed is directly analogous to the car's speed and, similarly, the spacing between the magnetic heads is analogous to the car's wheelbase. Each head will handle two channels of information so that one channel represents the road function under the right side of the car, the other, the left.

The tape recorder employs pulse-width modulation as a recording means. The delay range available is .04 to 1.5 seconds. The tape speed control is continuous over the range 4"/sec. to 40"/sec. This corresponds to car speeds in the range 15 to 150 mph. The unit will accept signals in the frequency range of 0 to 30 cycles per second. Input and output impedances are similar to those normally associated with analog computer amplifiers.

Automobile Simulation

Seven Degree of Freedom Model - The car model simulated in the computer is the seven degree of freedom system shown in Figure 7. This system considers the vertical motions of all four wheels plus the bounce, pitch, and roll of the body. Although the early analysis of ride motions did not include roll effects, the high human sensitivity to roll displacements literally dictates that rolling motions be incorporated in a complete ride study. Only typical car dimensions and parameters are given in this figure to reduce clutter. The simulation of this system involves summing the forces on each wheel and summing the forces and torques that act on the body.

The general procedure is shown in Figure 8. The system is put into motion by the road wave input which enters the tire. The difference between the road wave input and the wheel motion, acting through the tire characteristic, produces the tire force. Motion of the wheel mass results from the difference of the tire force and the suspension force. The suspension force is produced by the difference of wheel motion and body motion acting through the suspension characteristic. Finally, the body motion results from the action of the suspension force upon it. Lagrange's equations were employed to derive the equations of motion of the ride system. These equations are given in detail elsewhere.[8]

In order to determine numerical values of the parameters required in the equations of motion, it was necessary to measure the inertia, spring, and damping characteristics of the body, suspension and tires. For the study a typical

1958 production car was chosen. Fortunately most of the required parameters could be measured on existing test apparatus. For example, the pitching moment of inertia was measured by swinging the car as part of a compound pendulum. Spring rates were determined on a standard test rig that applied force to the suspension and measured the resulting deflections. Shock absorber characteristics were determined by a stroking machine which gave shock absorber force as a function of input velocity. Tire damping was measured by applying a sharp pulse of force to the tire and determining the time the oscillation required to damp out. In all, there are 37 car parameters used in the simulation.[8]

Analog Computer - With the equations of motion and the car parameters in hand, the car ride response was determined on an analog computer. The computer solves the equations of motion by summing the various forces acting on each mass or inertia. Figure 9 shows this procedure for one end of the car whose vertical motion has been denoted by the coordinate X. The forces acting on the mass of this end of the car are summed at the variable gain amplifier which is used as an integrator. The acceleration of this mass is equal to the sum of the input forces divided by the mass under consideration. The variable gain amplifier of Figure 9 performs the three functions of summing the forces, dividing by the mass, and integrating the resulting acceleration into velocity. In Figure 9 this velocity is denoted as the output (dX/dt) of the variable gain integrator. The velocity is then integrated in a second fixed-gain integrator to produce the displacement -X. The displacement signal is then inverted to +X by the inverter. In the computer there are seven circuits like the one shown in Figure 9 together with nonlinear function generators to represent the springs, shock absorbers and tires. The outputs of the computer representing the motion of the front, rear, and roll of the car are fed to the motion simulator. It should be noted that not only the displacement signal X but also the velocity signal dX/dt is fed to the motion simulator. The reason for this will be made clear in a later section.

The analog computer used to solve the equations is a general purpose unit, no special equipment being required. In all, 82 operational amplifiers are required to solve the problem and provide coordinate transformations, and as many as sixteen function generators may be used depending on how many nonlinearities are considered.

Comparison Between Analog and Digital Solutions - An analog computer solution of the linear seven degree of freedom system is given in Figure 10. The corresponding digital check solution is indicated by dots. The input used was a step bump or curb on the left side of the car only. The three traces show the roll response (ϕ), the front end vertical motion (Z_f), and the rear end vertical motion (Z_r). The car parameters are those of a typical production car traveling at 45 miles per hour. This simulation was carried out in real time, and the agreement between the analog and digital computers is quite good.

Motion Simulator

A view of the motion simulator together with the control console which controls it is shown in Figure 11. The body which can carry two passengers is shown without hood and fenders. The weight of the body and frame are supported by hydraulic pressure in the central cylinder. The hydraulic pressure is supplied by an accumulator which also acts as a very low rate spring. Vertical motions at the front and rear of the car are provided by valve-controlled hydraulic cylinders. These servo cylinders exert vertical forces on movable pads which are guided on rails. These pads are connected to the ends of the car frame and, consequently, when the pad moves the car frame moves also. The servo cylinders are located approximately at the centers of percussion in order to prevent the front servo from loading the rear and vice versa. Roll actuation is provided by a rotary actuator mounted on the rear movable pad. It is carried up and down in pitch and bounce and as a result there is no appreciable coupling between pitch, bounce and roll. It is worth emphasizing that the central cylinder supports the entire weight of the car body and passengers, and that the servos are used only to provide dynamic forces to produce motion. The valves which control the flow of oil to the servo-cylinders are actuated by voltages supplied by the computer. A block diagram of one of the servo systems is shown in Figure 12. The input to the servo is the voltage E_1 and the output is X. The servo loop is completed by a voltage E_p supplied by movement of the wiper on the feedback potentiometer. The transfer function of this simplified system, which neglects oil spring effects and numerous nonlinearities, is X/E_1, which is the inches of simulator movement per volt input to the servo. It can be seen that the transfer function can be written as

$$\frac{X}{E_1} = \frac{G}{1 + T_1 S} = \frac{\text{Inches of Displacement}}{\text{Volt Input}}$$

Thus it may be seen that the system will not give good reproduction of output frequencies much higher than $1/T_1$ radians per second. Fortunately

the analog computer provides a way around this frequency limitation. For example, if the system were perfect, the system response to a voltage E_c from the computer would be

$$\frac{X}{E_c} = G$$

If the simulator is to track a voltage E_c it can be shown that the servo input E_1 should be made equal to $E_c + T_1 S E_c$. To prove this set

$$\frac{X}{E_c} = \left(\frac{X}{E_1}\right)\left(\frac{E_1}{E_c}\right) = G$$

But since $X/E_1 = G/(1 + T_1 S)$, it is clear that

$$E_1 = E_c (1 + T_1 S) = E_c + T_1 \left(\frac{d E_c}{dt} \right)$$

Thus it may be seen that if the motion simulator is accurately represented by the simple transfer function $G/(1 + T_1 S)$ that the simulator may be made to track a voltage E_c from the computer provided that the input voltage to the servo is the tracked voltage $E_c + T_1$ times its time derivative. Fortunately this derivative already exists in the computer and it is a simple matter to multiply it by T_1 and feed it to the servo. This action effectively cancels an unwanted pole in the transfer function. The effect of this cancellation compensation is shown in the frequency response curves of Figure 13. It may be seen that the normal servo system has a bandwidth of about 4 cycles per second (the bandwidth is the maximum frequency at which the amplitude ratio is within $\pm$ 3 db of the amplitude ratio at zero frequency). The effect of cancellation compensation is to extend the bandwidth to 7 cycles per second. It is also clear that the transfer function of the system is more involved than the simple formula shown previously. The main performance figures of the Ride Simulator are given in Table I.

	Pitch	Roll
Maximum Acceleration, Deg./sec^2	1,150	500
Maximum Velocity, Deg./sec.	48	240
Maximum Displacement, Degrees	$\pm$ 6	$\pm$ 10
Bandwidth, cps	5	4
Positional Accuracy, Degrees	.025	.10

TABLE I-A: Ride Simulator
Rotational Performance

	Bounce
Maximum Acceleration, In./sec^2	1,250
Maximum Velocity, In./sec.	40
Maximum Displacement, Inches	$\pm$ 5
Bandwidth, cps	5
Positional Accuracy, Inches	.020

TABLE I-B: Ride Simulator
Translational Performance

The bandwidth figures given represent the performance of the servos when cancellation compensation is used.

The simulator performance requirements originated in road test data which determined extremes in each of the variables. The road data was determined by driving an instrumented test car over a number of different bumps at various speeds and determining the peaks of the acceleration, velocity, and displacement transients. From a collection of these curves, an extreme for each variable was obtained. A comparison of the observed maximum of each car motion variable with the maximum of the corresponding Ride Simulator motion variable is shown in Table II.

	Bounce	Pitch	Roll
Maximum Acceleration	540%	640%	125%
Maximum Velocity	130%	340%	850%
Maximum Displacement	83%	120%	110%

TABLE II: Simulator Performance Capability
Relative to Maximum Car Motions

It may be seen that the Simulator has the capability of reproducing the extremes of real car motion with the exception of bounce displacement. This is not considered a major drawback since a five inch bump, which the Simulator can accommodate, represents a violent disturbance to the passengers at almost any road speed.

In addition to providing a means of extending the apparent servo bandwidth, the computer calculates the coordinate transformation required when the center of gravity of the simulated car does not correspond with the fixed pivot of the motion simulator. By means of another coordinate transformation it is possible to provide the sensations of rear seat ride to passengers occupying the front seats of the motion simulator.

Comparison of Overall Response to Real Car Response

The real point of a ride simulator is that the simulator will produce to a high degree of fidelity the motions experienced by a real car on a real road. To prove out the complete system a typical production car whose parameters had been completely determined was instrumented with an integrating accelerometer and then was driven over a triangular bump at 20 miles per hour. The triangular bump used in this work was 60 inches long and 4.8 inches high. Then the Ride Simulator was put through the same procedure, namely:

(a) An identical triangular bump was recorded on magnetic tape and played into the computer by the Simulag.

(b) The computer determined what the car's responses should be and sent motion signals to the motion simulator.

(c) The simulator moved representing the motion of the car over the bump.

To demonstrate the fidelity of the overall simulator, the integrating accelerometer was removed from the car and was installed in the same location in the motion simulator body. The results of this test which compare the response of a typical 1958 production car with a simulated version of the same car are shown in Figure 14. A comparison of accelerations is shown at the top of this figure and a comparison of displacements at the bottom. The agreement is generally good. The oscillations in the acceleration trace for the production car are due to a stiffer rear floor pan where the accelerometer was mounted.

Conclusion

The Ride Simulator has demonstrated its ability to reproduce the on-the-road ride response of real cars. The way is now open to evaluate new suspension systems without recourse to expensive component constructions and to study the passenger and seat combination under closely controlled conditions. The Ride Simulator is an important step forward in ride analysis because it will enable improvements in ride to be made with a minimum of time and effort.

References

1. Rowell, H. S., "Principles of Vehicle Suspension", IAE Proceedings, Vol. 17, pp 455-536 (1922-1923)

2. Guest, J. J., "The Main Free Vibrations of an Autocar", IAE Proceedings, Vol. 25, pp 505-547 (1925-1926)

3. Olley, M., "Independent Wheel Suspension - Its Whys and Wherefores", SAE Journal, Vol. 34, No. 3, pp 73-81, 1934

4. Schilling, R., and Fuchs, H. O., "Modern Passenger Car Ride Characteristics", Trans ASME, Vol. 63, pp A 59-A 66, 1941

5. Jeska, R. D., "A Comparison of Real and Simulated Automobile Suspension Systems", University of Michigan, Willow Run Research Center, UMM-117, February, 1953.

6. Grimes, C. K., "Development of a Method and Instrumentation for Evaluation of Runway Roughness Effects on Military Aircraft", AGARD Report 119, May, 1957.

7. Brenner, M. M., "A New Dead Time Simulator for Electronic Analogue Computers", Proc. National Simulation Council, pp 6.0-6.9, 1957.

8. Kohr, R. H., "Analysis and Simulation of Automobile Ride", SAE Preprint 144A, SAE National Automobile Meeting, March 15, 1960

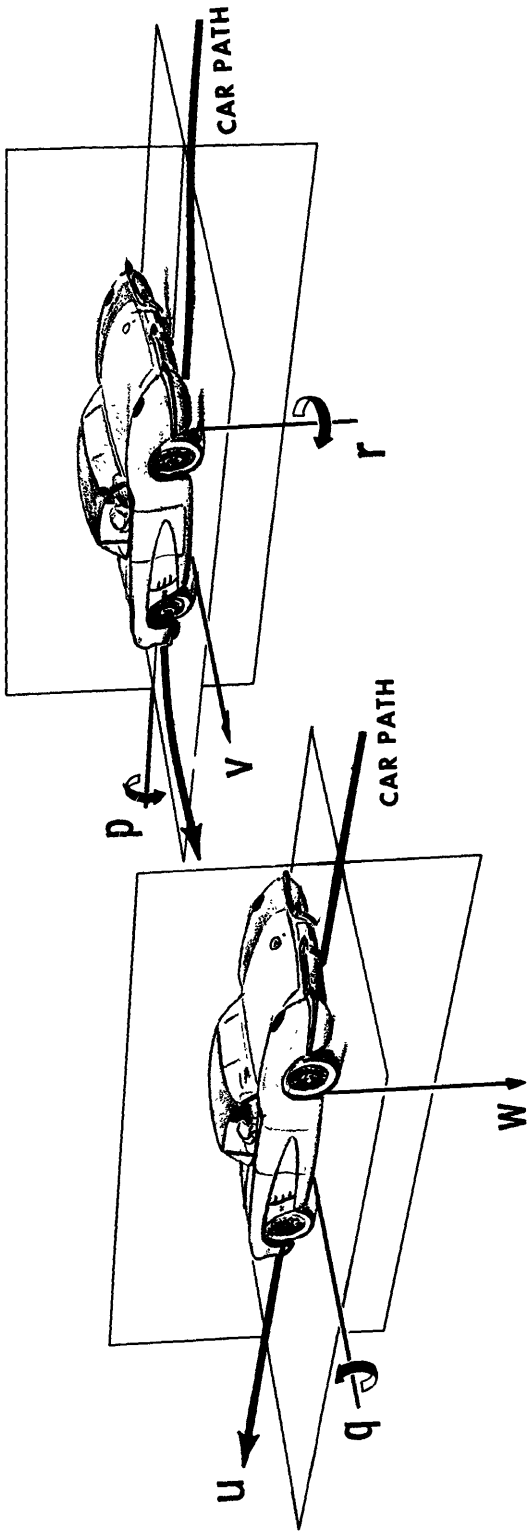


Figure 1
SEPARATION OF LONGITUDINAL AND LATERAL MOTIONS

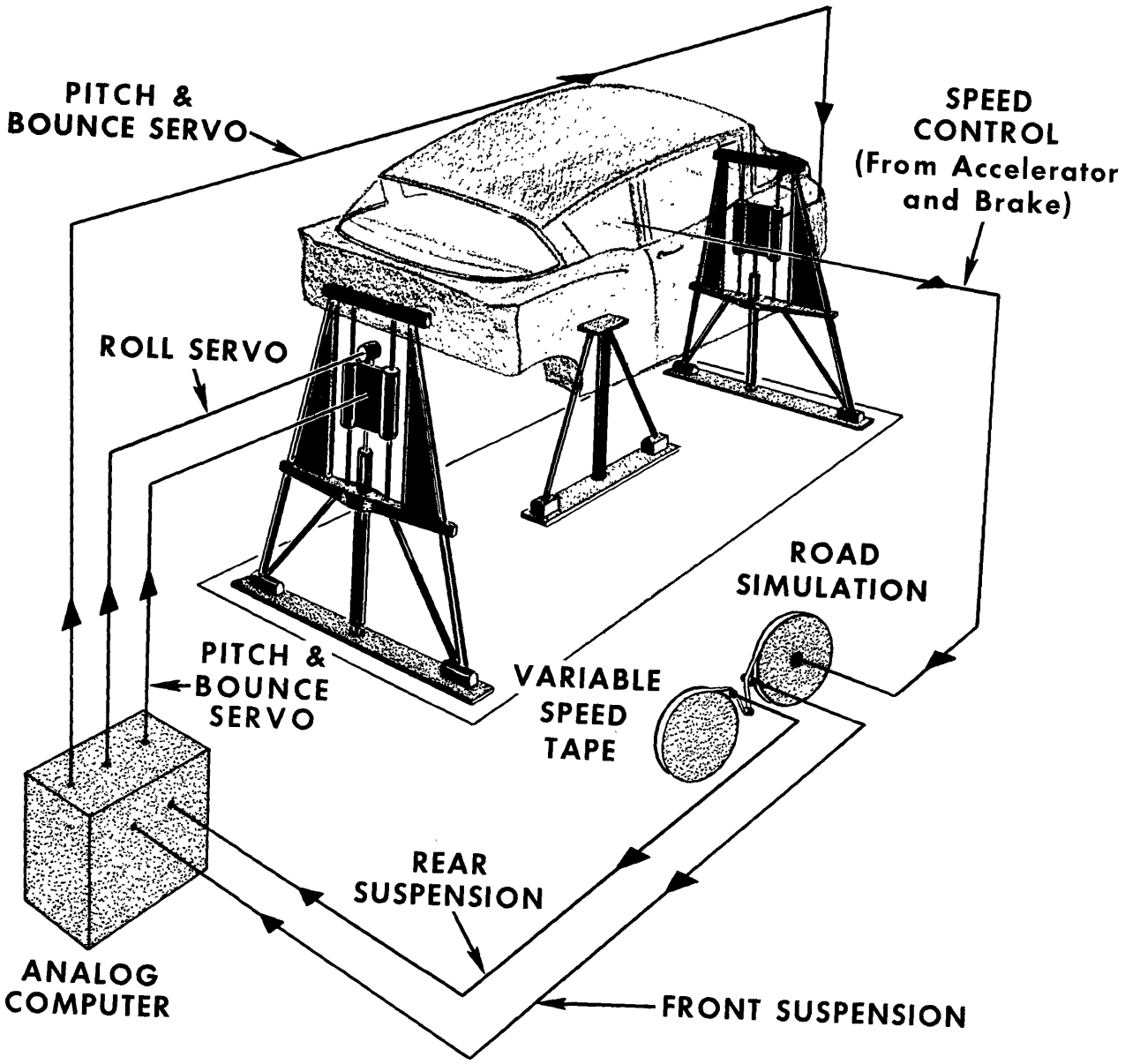


Figure 2
SCHEMATIC VIEW OF THE RIDE SIMULATOR

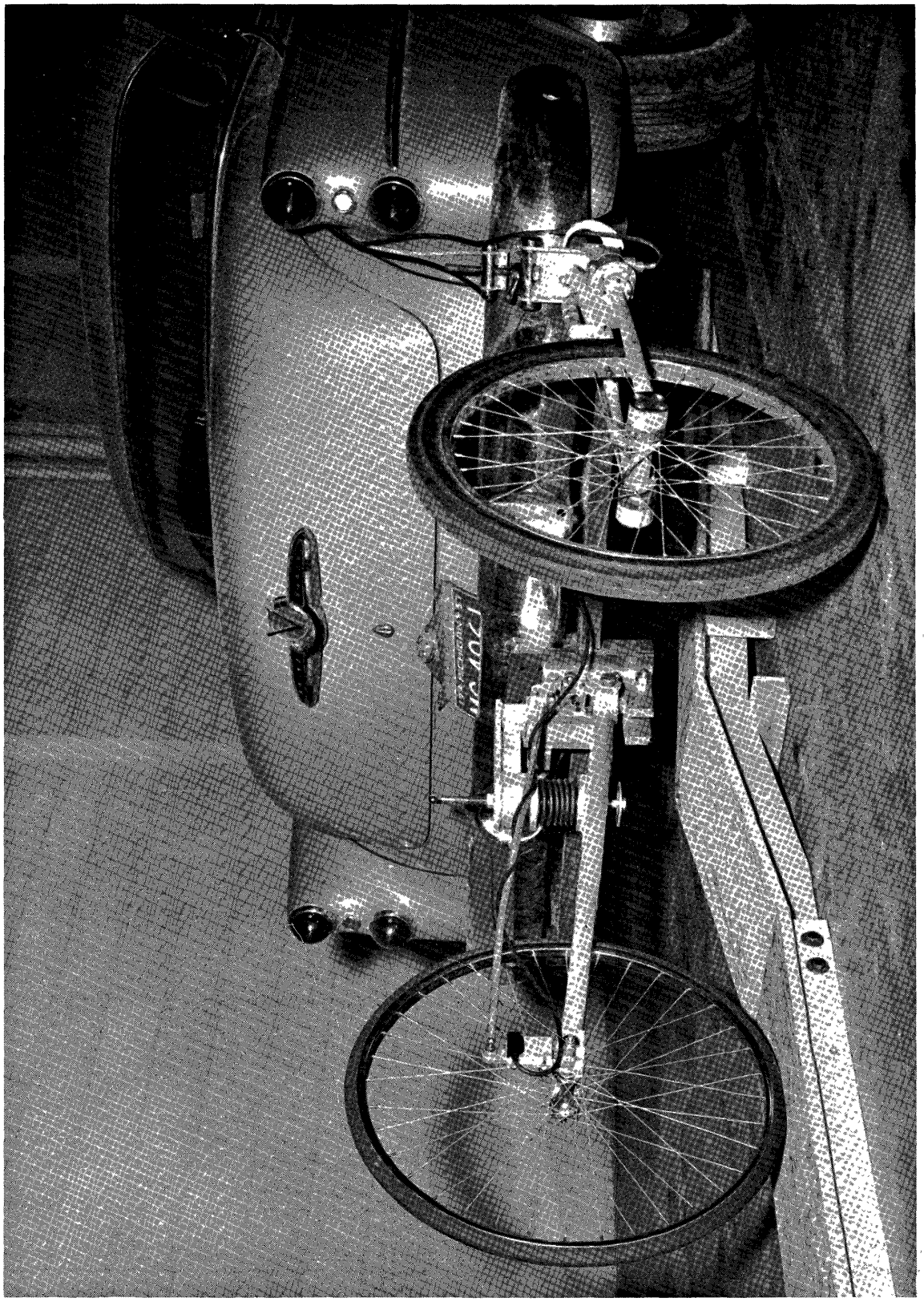
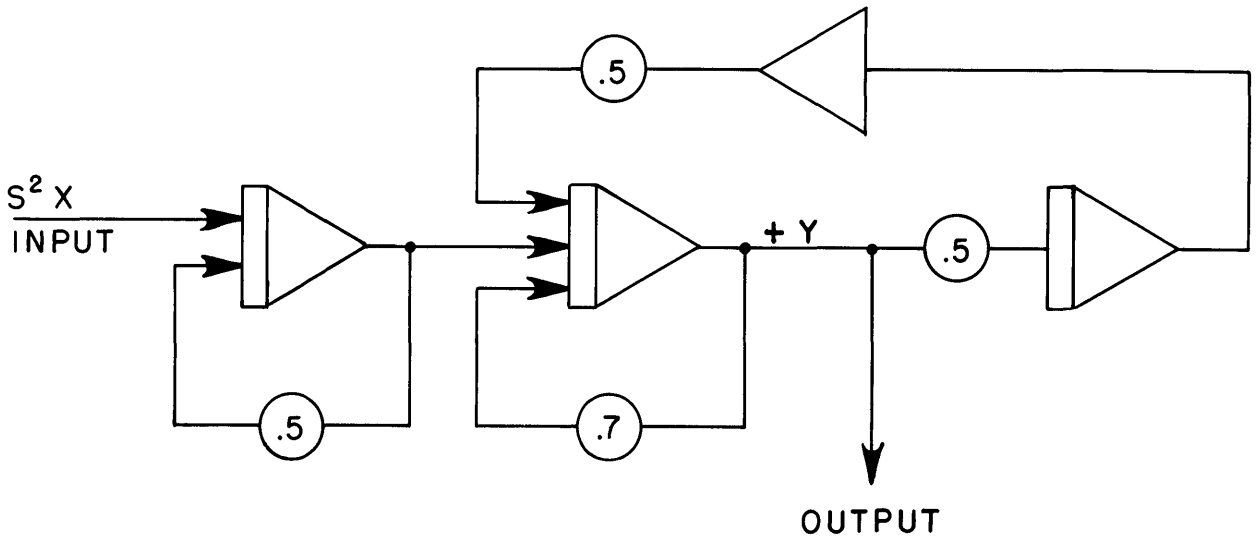


Figure 3
INSTRUMENTED WHEEL USED TO MEASURE ROAD WAVES



$$\frac{Y}{X} = \frac{S^3}{(S+0.5)(S^2+0.7S+0.25)}$$

Figure 4
FILTER-INTEGRATOR CIRCUIT

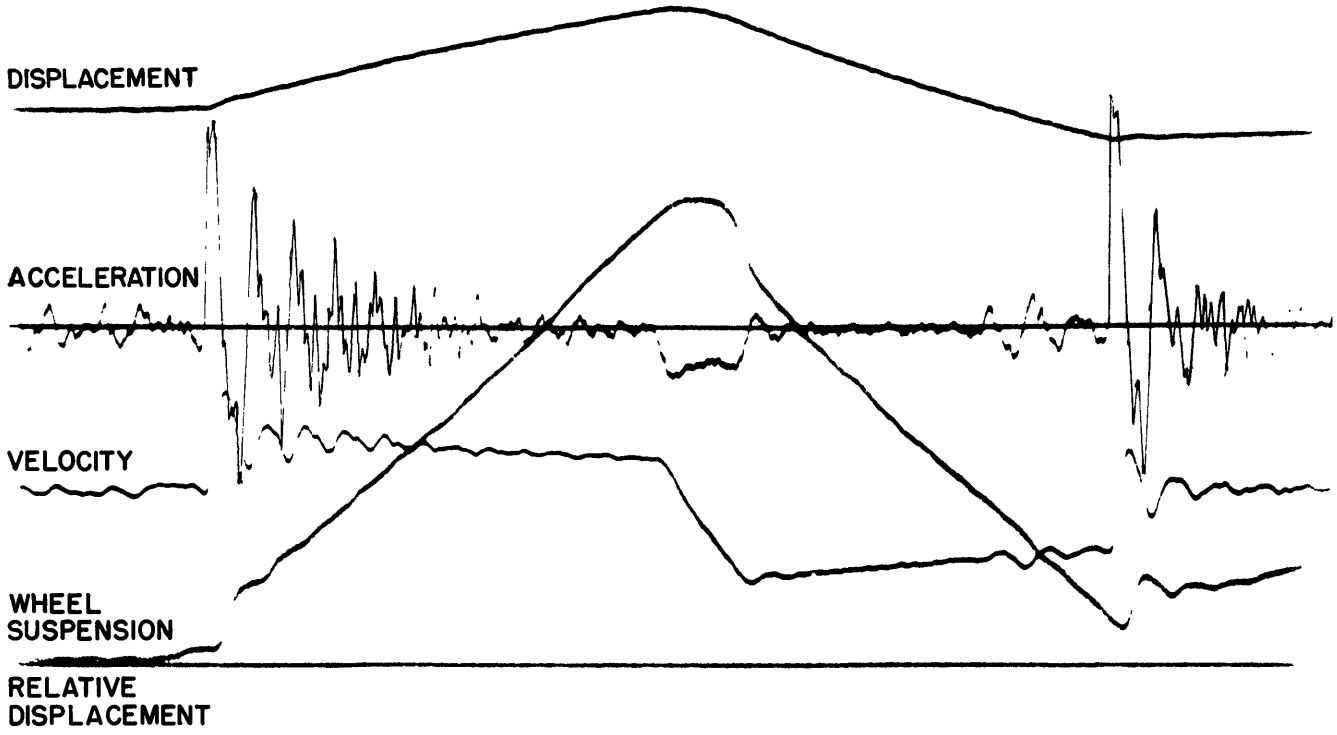


Figure 5
TRIANGULAR ROAD WAVE DETERMINED BY INSTRUMENTED WHEEL

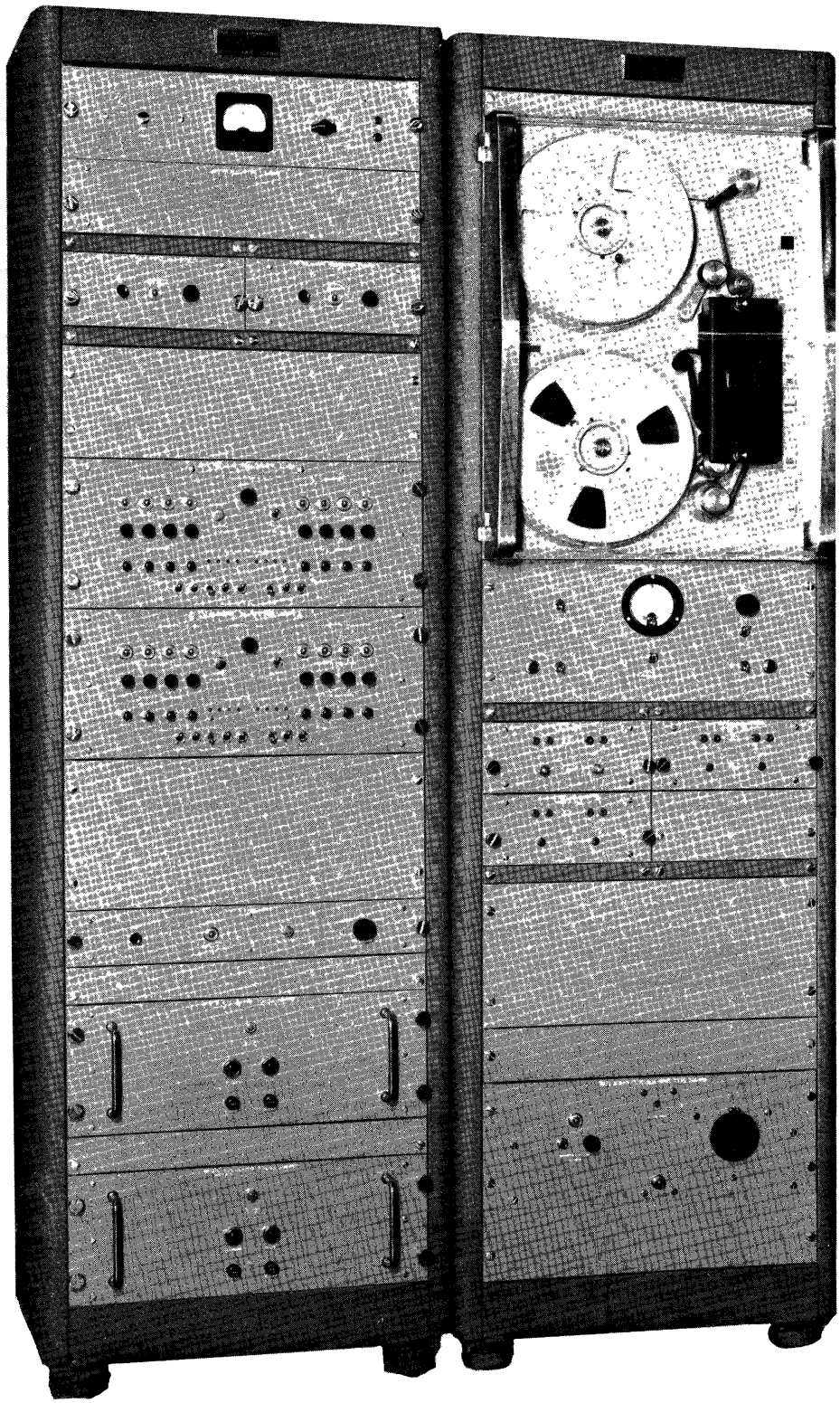


Figure 6
TAPE RECORDER USED FOR ROAD WAVE GENERATION AND TIME DELAY

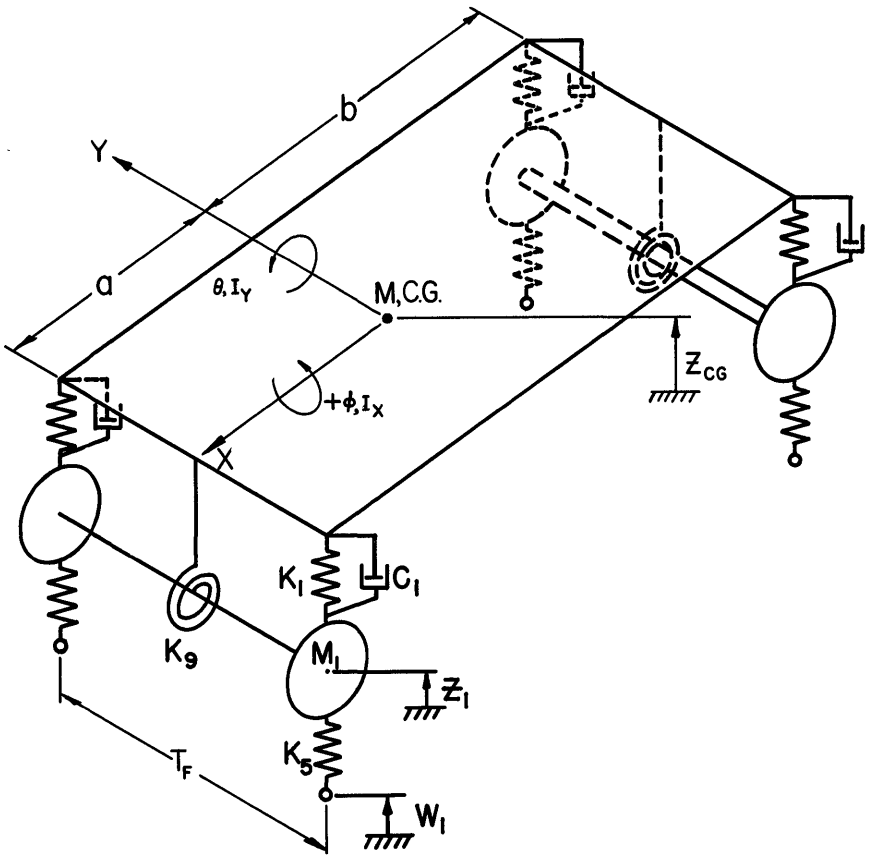


Figure 7
SEVEN DEGREE OF FREEDOM SUSPENSION SYSTEM

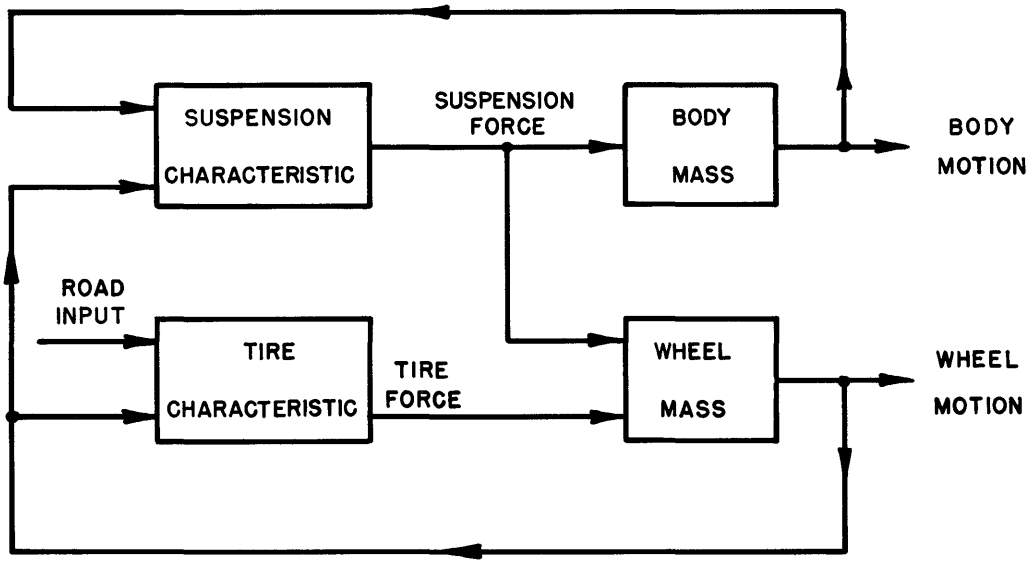


Figure 8
BLOCK DIAGRAM OF SUSPENSION SYSTEM

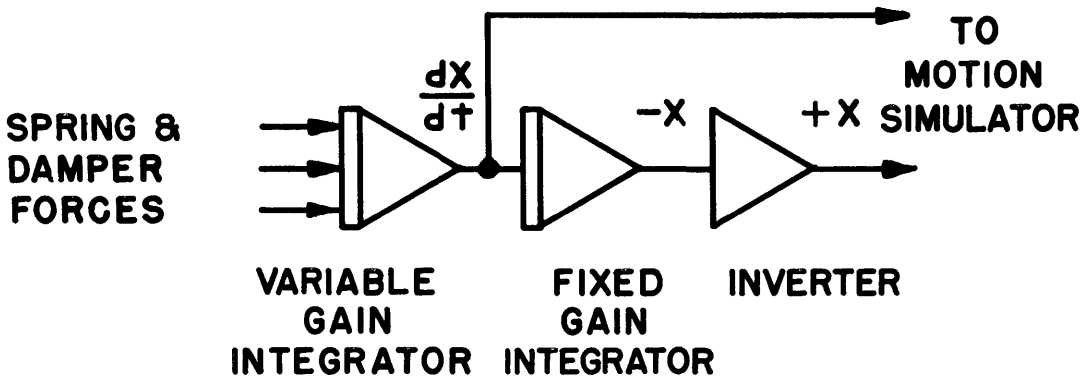


Figure 9
INTEGRATION OF ACCELERATION TO DISPLACEMENT

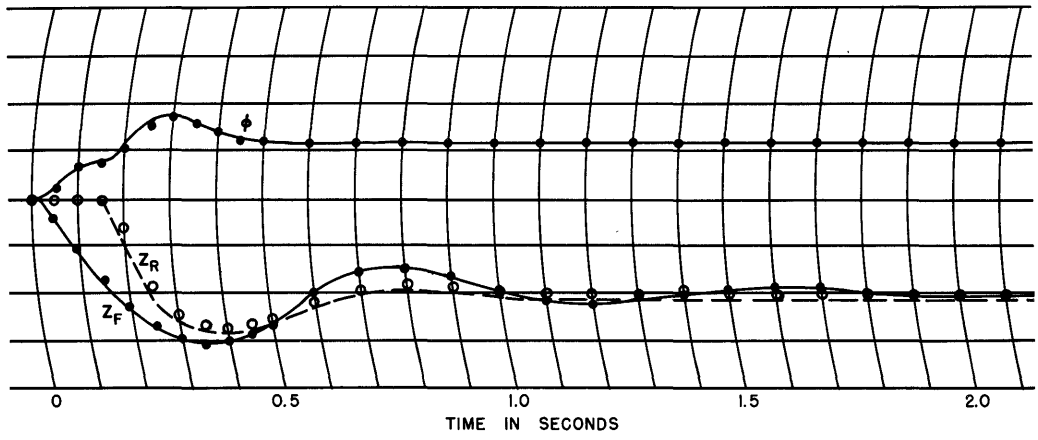


Figure 10
COMPARISON OF ANALOG AND DIGITAL SOLUTIONS

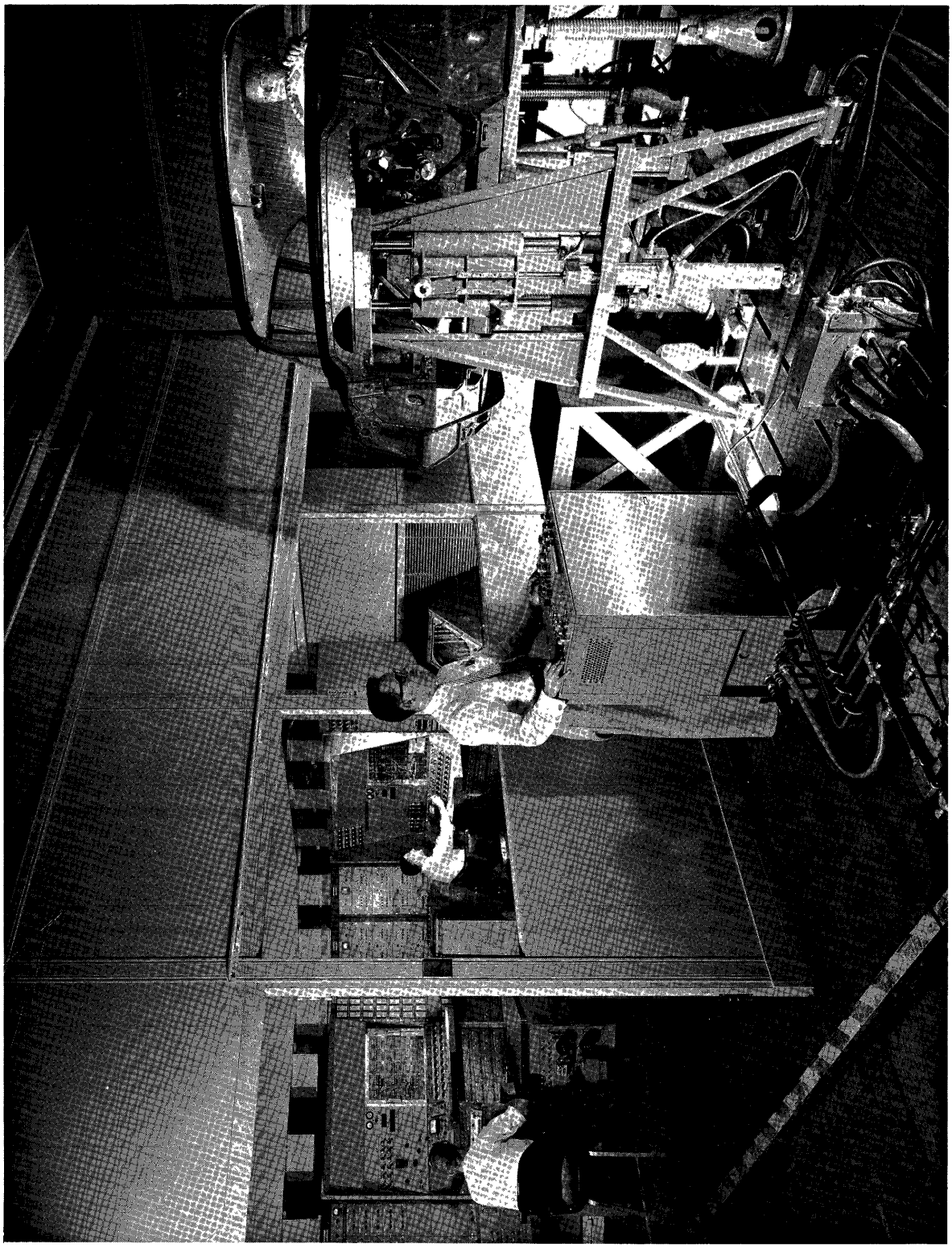


Figure 11
OVERALL VIEW OF RIDE SIMULATOR SHOWING COMPUTER, MOTION SIMULATOR, AND CONTROL CONSOLE

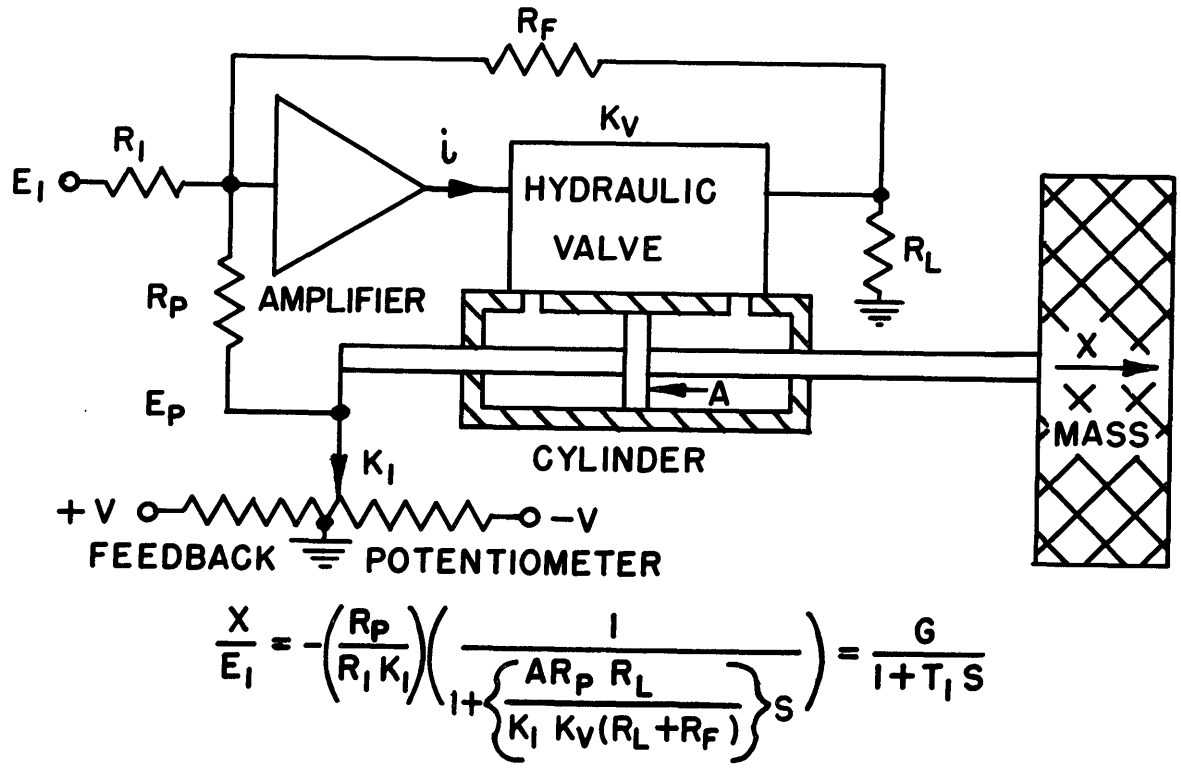


Figure 12
SCHEMATIC DIAGRAM OF SIMULATOR SERVO SYSTEM

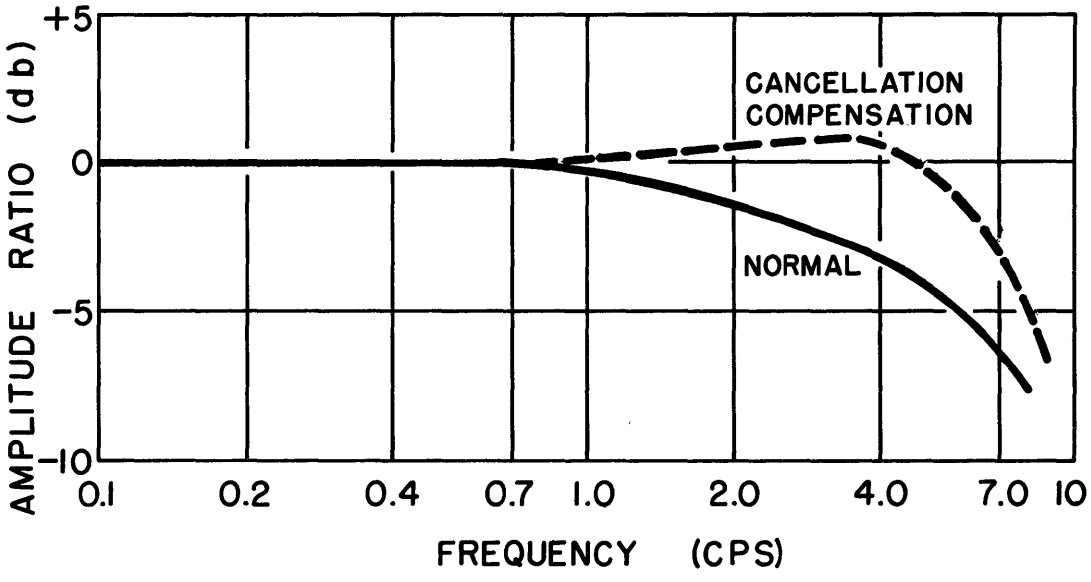


Figure 13
FREQUENCY RESPONSE OF MOTION SIMULATOR SERVO

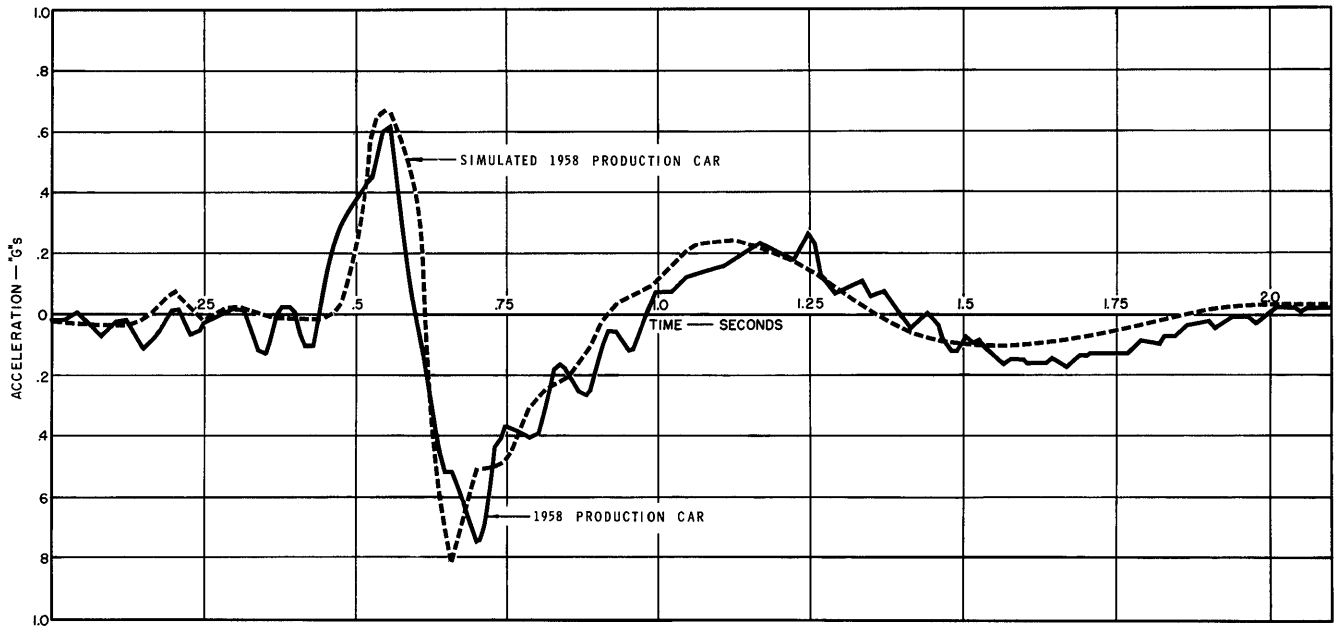


Figure 14A
ACCELERATION COMPARISON OF REAL AND SIMULATED CARS
PASSING OVER TRIANGULAR BUMP

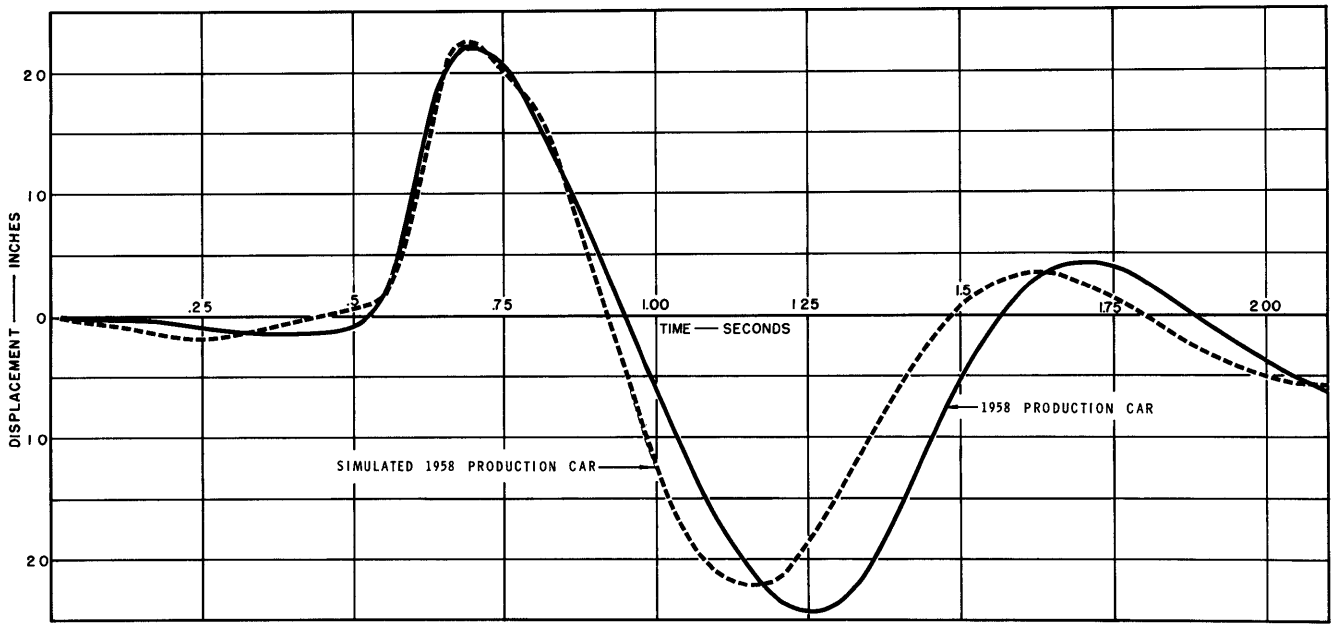


Figure 14B
DISPLACEMENT COMPARISON OF REAL AND SIMULATED CARS
PASSING OVER TRIANGULAR BUMP

ANALOG COMPUTER SERVES AS BOTH SYSTEMS ANALYSIS TOOL AND OPERATOR TRAINING FACILITY FOR ENRICO FERMI ATOMIC POWER PLANT

Samuel N. Irwin
Chief Engineer, Electromechanical Division
Holley Carburetor Co.
Warren, Michigan

Robert R. Kley
Manager, Data Processing and Systems
Analysis Section, Research Department
Holley Carburetor Co.
Warren, Michigan

Summary

An analog simulator with specially designed electromechanical equipment was developed to simulate one of three loops of the Enrico Fermi Atomic Power Plant, including all elements except the steam turbine; i.e., the reactor, intermediate heat exchanger, steam generator, mixing and transport delay phenomena, and steam system hydraulics.

A philosophy of plant control was evolved on the basis of plant operating characteristics, as determined by the simulator and supported by a steady-state digital analysis of the plant. Conceptual control hardware designs were then tested on the simulator and an optimum design was selected and tuned. Actual hardware of the automatic control system was simulated, and this simulation circuitry incorporated in the computer.

Control consoles exactly duplicating the control panel layout in the actual plant were connected to the analog simulator to adapt it for power plant operator training. Trainees thus get the "feel" of plant operating characteristics, learn to meet emergencies, and practice standard operating procedures, using both manual and automatic controls.

The facility has already proved valuable in studying normal and abnormal operating conditions and remains available for any future systems analysis needs; meanwhile it is in continuing service for operator training.

Introduction

The Enrico Fermi Atomic Power Plant on the shore of Lake Erie near Monroe, Michigan, has a sodium-cooled fast breeder reactor as its heat source. Designed by Atomic Power Development Associates, Inc., it is being constructed under the AEC Power Demonstration Reactor Program, but is to be owned and operated by the Power Reactor Development Company, a non-profit organization of 25 industrial and utilities firms.

In the thermal portion of the plant, flowing liquid sodium will carry heat from the reactor to once-through steam generators where superheated steam will be produced. This steam will be delivered to a turbine-generator built and operated by the Detroit Edison Co. Normal power output will be 100 MW (electrical) with steam at 740°F and 600 psia.

Some three years ago a contract was entered into by Power Reactor Development Co. and Holley Carburetor Co. for the study and simulation of an operating power control system for the Fermi plant. Atomic Power Development Associates, Inc. (APDA) was to serve as technical liaison agent for Power Reactor Development Co. (PRDC). Objectives of the program were:

1. To simulate plant performance over a wide power range.
2. To develop a recommended concept for an operating control system.
3. To convert the simulator into a training device for power plant operators.

Because the variables which must be controlled to regulate power production from such a plant are intimately related, with a change at one point causing changes to occur at other points, development of a sensitive and accurate control

system was necessary to establish safe and stable power plant operation. Uniqueness of certain elements of the plant further complicated control planning, and many of the details of control system evolution recorded herein represented pioneering research efforts.

Early planning in the first phase of the project called for parallel digital and analog computer studies to provide a dual approach to plant simulation and determination of adequate control system requirements. Analog simulation proved workable for the entire problem, but digital studies ran into complications of excessive computing time for certain aspects of plant operation and therefore could not be relied on for a complete transient solution. The digital work, however, was useful in supporting and verifying analog results.

Power Plant Description

Figure 1 presents a block diagram of the entire plant. In the power section are the reactor and three parallel heat transmission circuits, delivering reactor thermal energy in the form of superheated steam at 600 psia and 740°F. Each circuit begins with a primary loop of liquid sodium, circulating through the reactor, absorbing and carrying away heat. Since the primary sodium coolant is strongly radioactive, it is not pumped

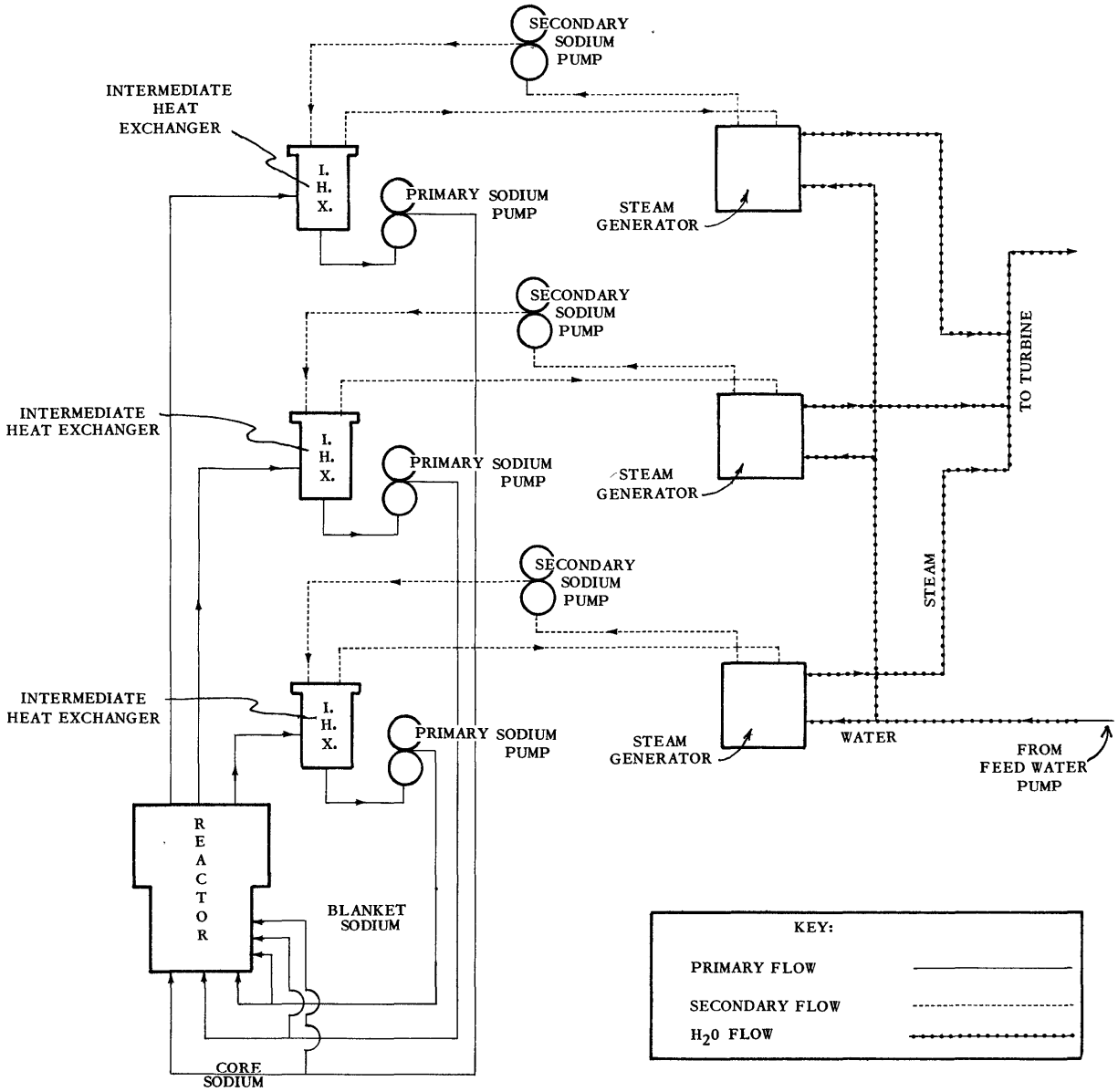


Figure 1. Block Diagram of Enrico Fermi Atomic Power Plant Showing All Three Loops

directly to the steam producing unit, but instead passes through an intermediate heat exchanger where heat is transferred to a secondary, non-radioactive sodium loop. The heated secondary sodium then flows to a steam generator where feedwater, entering at 600 psia and 340°F, is heated, boiled and emerges as superheated steam. The once-through steam generators of the cross-flow configuration represent a fairly new concept in steam power design.

As shown in Figure 1, the combined output of the three steam generators is delivered to a single turbine driving the electric generator. Use of the three steam generators and three related heat supply circuits from the reactor introduces a substantial margin of plant safety, since simultaneous failures in all three circuits are extremely unlikely. Also included in the steam system are steam and feedwater piping and valving, condenser, deaerator, feedwater pumps, pump drives and emergency water supply.

In the early months of the study the physical details of plant layout had not been determined, nor were details available on the actual intermediate heat exchangers and steam generators. However, conceptual designs and data on required performance of components were available based upon an ultimate reactor thermal power of 430 MW. The control system was to be designed for initial plant operation at 300 MW (referred to as 100 percent power) but was to be adaptable to higher power levels.

Purpose of the Operating Control System

This study is specifically concerned with the conceptual design of an operating control system for the power plant. "Operating" is here distinguished from the power limiting or safety system, which has been developed separately to limit large power excursions and to act as a safety-shutdown device in the event of abnormal operation.

The purpose of the operating control system is the maintenance of power plant operating stability, while regulating power in response to external demand. In the reactor this connotes controlled variations in power output to meet changing power demands, and prompt leveling out at each new power level, with any tendency to "hunt" damped out to the greatest extent possible. In the sodium loops temperature rises and drops correspond to power output. Related temperature changes must be controlled to maintain proper phase relationship during power changes to avoid plant instabilities which involve surging of temperature and power output. To the water-steam system, stable plant control calls for regulation of pressure and flow to match power output, while avoiding pressure or flow surges that would cause corresponding fluctuation in steam generator performance and thus reflect back into the sodium system.

Initially a control philosophy had to be derived, determining which of several variables should be selected as controlling, which others they could be made to control, and which could remain passive. Because the Enrico Fermi Power Plant is to a great extent unique, many factors loom large in the control problem that have hitherto been negligible or nonexistent in prior steam and nuclear power development. The complexity of its heat cycle, from the standpoint of system dynamics, its small temperature coefficients, long transport times, and the peculiarities of once-through-type steam generators: these are factors that governed the ultimate choice of control philosophy. Once the philosophy had been established, a control system was to be worked out, and the required operating characteristics of the control components were to be defined.

Preliminary Planning for Plant Simulation

Early plans called for simulation of all three circuits of the plant, at all power levels from source to 150 percent of rated power. Since the complexity of the problem would have made this approach prohibitive in terms of computer cost and maintenance time, the objectives were relaxed wherever possible without seriously compromising the results. First, the simulation was restricted to only one of the three heat transmission circuits, saving nearly two-thirds in simulator costs. The technique is valid since all circuits are identical in operation, and a 3-to-1 compensation is easily inserted wherever the three circuits come together, as in the reactor.

Further, because low-power-level simulation involves voltages approaching the error band of the computer, the stability study was limited to higher levels where error effects would be negligible. With the computer used in the early stages of the study, the minimum level for quantitatively accurate simulation was 25 percent of rated plant power.

In addition, because of the advanced state of steam power technology, it was felt that the water-steam system external to the steam generators need not be simulated, as any significant transient behavior could be predicted when necessary on the basis of existing knowledge.

Components Simulated Individually

Within these limitations, it was decided to simulate the plant as thoroughly as possible, representing all significant components and effects individually, avoiding the "lumping" approach of the transfer function technique, where the identity of an individual component or phenomenon is lost. The individual component approach was required because nonlinearities of the heat system limited the transfer function approach to approximately 5 percent of the power range, while simulation was desired over a much wider power range. The characteristics of each major component of the plant thus were established by the functional interrelationship of its simulated parts rather than by mathematical approximation of the whole.

In this way, a clear physical correlation of components in actual plant and simulator could be maintained, for improved operator comprehension, quick location of simulator faults, and ready adaptation to changes made in the course of plant development. Although it might have been designed to operate much faster than the physical plant, the simulator was built to operate in real time, because of its eventual use as an operator training device.

Identification of significant components and effects warranting simulation, and the determination of the manner and extent to which they should be represented, required considerable preliminary study. Extensive digital computer analysis of the dynamic equations describing individual component operation was necessary to devise workable analogs for setting up the simulator. The digital studies also revealed various undesirable steady-state plant operating conditions which the control system must be designed to preclude.

Phase I Simulation

Following the aforementioned preliminary studies, the first plant simulator was assembled by Holley Carburetor Company during the early months of 1958, using a commercially available general-purpose d-c analog computer, together with specially designed and built supplementary electronic equipment which simulated effects impractical to achieve with standard computer circuitry. Continuously variable voltages, adjustable and fixed resistors, and capacitors in the simulator represent and are analogous to the physical variables and parameters of the plant.

The Phase I simulator represented only one circuit of the power-producing section of the plant --one reactor, one heat exchanger and one steam generator, plus their connecting sodium piping and pumps. Simulations of the individual phenomena and components described in the following paragraphs were interconnected in their proper sequence, following precisely the actual functional relationships of the plant components. Though water-steam dynamics were omitted, a simplified representation of the feedwater pumps permitted a rudimentary simulation of feedwater flow that could be changed either manually or automatically.

Transport Delay

When a system experiences a transient input signal, the system output has some time relationship to the input. This time delay or phase relationship of output to input is an important factor in adapting a control to the system. When the system includes long piping such as is used in the Fermi plant, the piping coolant time (transport delay) between and through components is a significant element in determining the over-all time response of the plant, and simulation of this delay is vital to over-all plant simulation.

Based on the physical analogy of a bucket brigade, an electromechanical system was devised to simulate in a practical and economical fashion the transport time of sodium flow through piping, intermediate heat exchanger, and steam generator. Two delay wheels, one for the primary loop and one for the secondary loop (Figure 2), were constructed using capacitors which represent slugs of sodium flowing through the pipe or element being simulated. Capacitor voltage represents sodium temperature. With the wheel rotating, the time required for a capacitor on the wheel to travel from an inlet to an outlet point is the same as the time required for sodium to travel the length of the physical pipe.

The delay wheel speeds may be varied to simulate different sodium flow rates. Although the basic study was limited to operation at fixed sodium flow rate, the variable flow rate feature permits consideration of cases of abnormal coolant flow, and will allow for higher flow rates if required when simulating plant operation at the higher power levels planned for the Fermi plant after the first few years of operation.

Mixing

Plenums at the inlet and outlet of the reactor, intermediate heat exchanger and steam generator

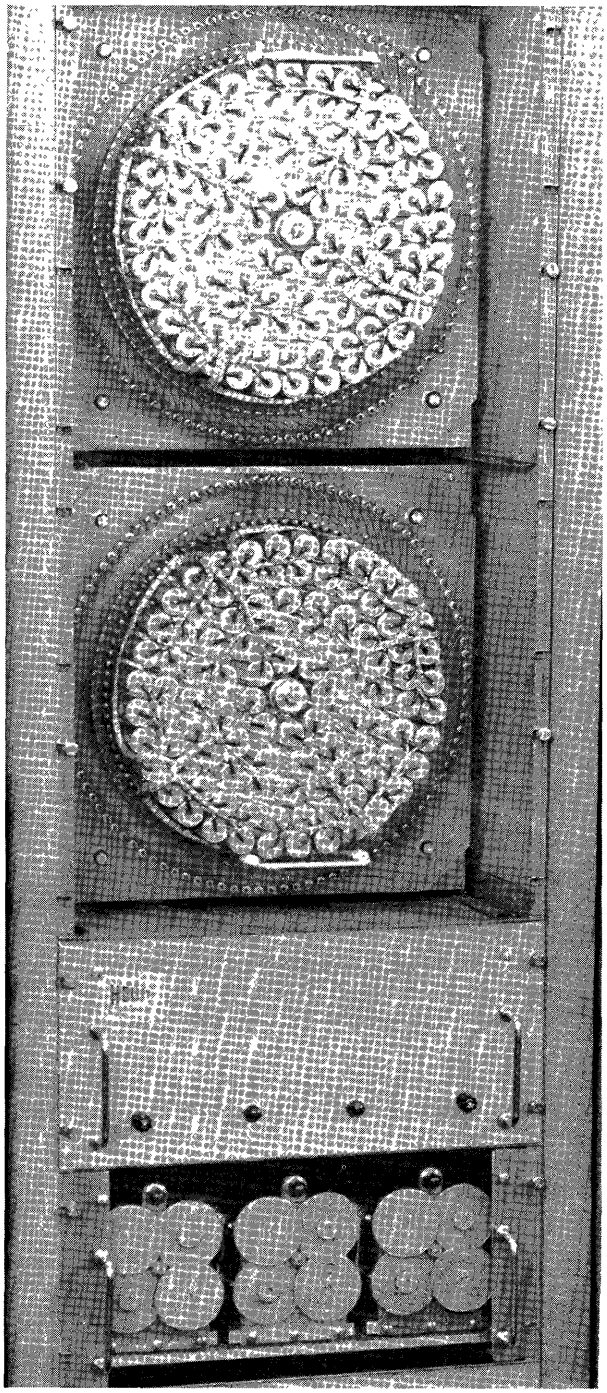


Figure 2. Two delay wheels based on physical analogy of bucket brigade were constructed by Holley personnel. Capacitors represent slugs of soduim flowing through pipe or element being simulated.

introduce a mixing phenomenon. If the temperature of sodium flowing into one of these plenums changes, the outlet temperature will also change. However, because incoming sodium mixes with that already in the chamber, the temperature change at the outlet is more gradual than that at the inlet. The relatively sluggish response of mixing tanks to inlet temperature transients results in a phase shift at the outlet, which complicates the control picture, but also has the beneficial effect of tending to smooth out transients.

Reactor

Power level in the reactor is proportional to the neutron population. Any change in power level depends on changing the position of the reactor control rods and/or temperature within the reactor. Reactor simulation provides for all of the interrelationships between neutron population, control rod position, internal temperatures, and heat extraction rate. The neutron kinetics are represented by a classical six-group simulation. The output of this circuit, proportional to heat produced, goes to separate core and blanket simulations. Although their simulation circuits are basically the same, the electrical constants differ to account for their different sodium flow and heat generation rates. Temperature coefficients from the core, blanket and structure simulations combine with rod position to simulate reactivity, which in turn becomes the input to the neutron kinetics simulations.

Intermediate Heat Exchanger

The intermediate heat exchanger (IHX) simulated in the Phase I study is of counterflow design; the primary or radioactive sodium flows through in one direction while the nonradioactive sodium flows in the opposite direction. The two sodium flows are separated by the walls of the stainless steel pipes which carry the secondary sodium. In the steady state a uniform temperature differential exists from inlet to outlet and heat is transferred very efficiently from primary to secondary sodium. When a temperature transient occurs at either primary or secondary inlet, both outlet temperatures change. The close coupling characteristic of the IHX causes the greater response to a primary inlet temperature transient to appear in the secondary outlet while the greater response to a secondary inlet transient occurs at the primary outlet.

The previously described delay wheels were used as a direct electromechanical simulation of the IHX, in an unusual, simplifying application shown in Figure 3. The wheels rotate in opposite directions, thus simulating the counterflow of sodium in the heat exchanger. As the wheels rotate, charge flows from the primary wheel capacitors to the secondary wheel capacitors via

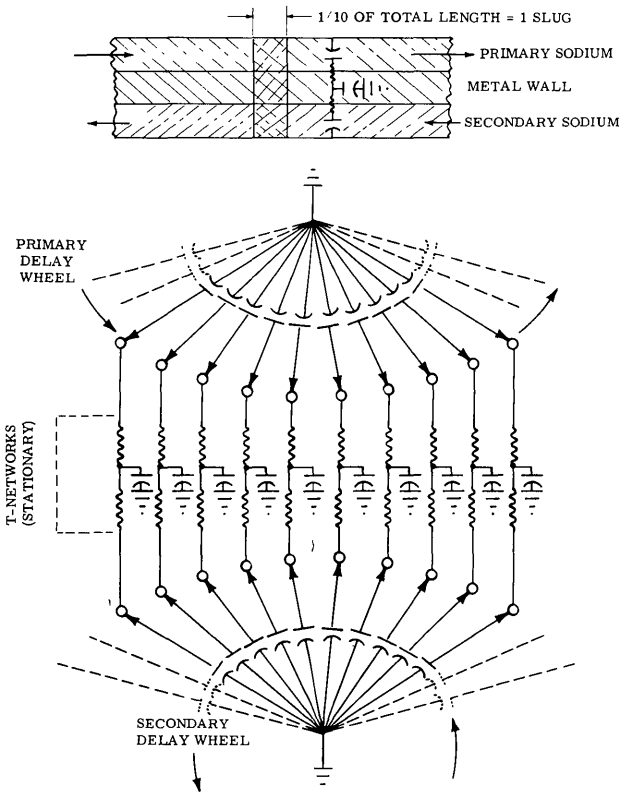


Figure 3. Analog Simulation of Intermediate Heat Exchanger

stationary electrical T-networks simulating heat transfer from primary to secondary sodium. This unique simulation is straightforward, accurate and compact, and eliminates a large number of operational amplifiers that otherwise would be required.

Steam Generator

The once-through steam generator is the most sensitive element in the power section of the plant. Over the entire power range the sodium outlet temperature follows closely any changes in sodium inlet temperature, feedwater temperature, or changes in sodium or water flow rate. The superheated steam outlet temperature also is sensitive to sodium inlet temperature.

Heat carried into the steam generator by the sodium is transferred through a separating metal wall to the H_2O which enters the steam generator as water and leaves as steam. The changing phase of H_2O (from water to low-quality, high-quality and superheated steam) introduces complications into the analysis and simulation of the steam generator because density and heat transfer coefficient of the H_2O change from phase to phase.

In developing a simulation, the steam generator was divided into 14 sections. Heat flow from sodium to H_2O in each section was represented by using a T-network connecting two capacitors (C), simulating heat capacity of sodium and H_2O respectively, as shown in Figure 4. A third capacitor between represented heat capacity of the separating wall, and resistors (R) in the circuit represented the heat transfer coefficients. The sodium side of this circuit was readily simulated by capacitors on the secondary delay wheel already described, but the H_2O side presented a challenge because of the varying characteristics of the H_2O.

Rather than insert a variable resistor and capacitor in the circuit to simulate changing heat transfer characteristics, it was decided to impress a fictitious voltage E (Figure 4) in the circuit, which would allow the same heat flow as variable elements R and C would produce.

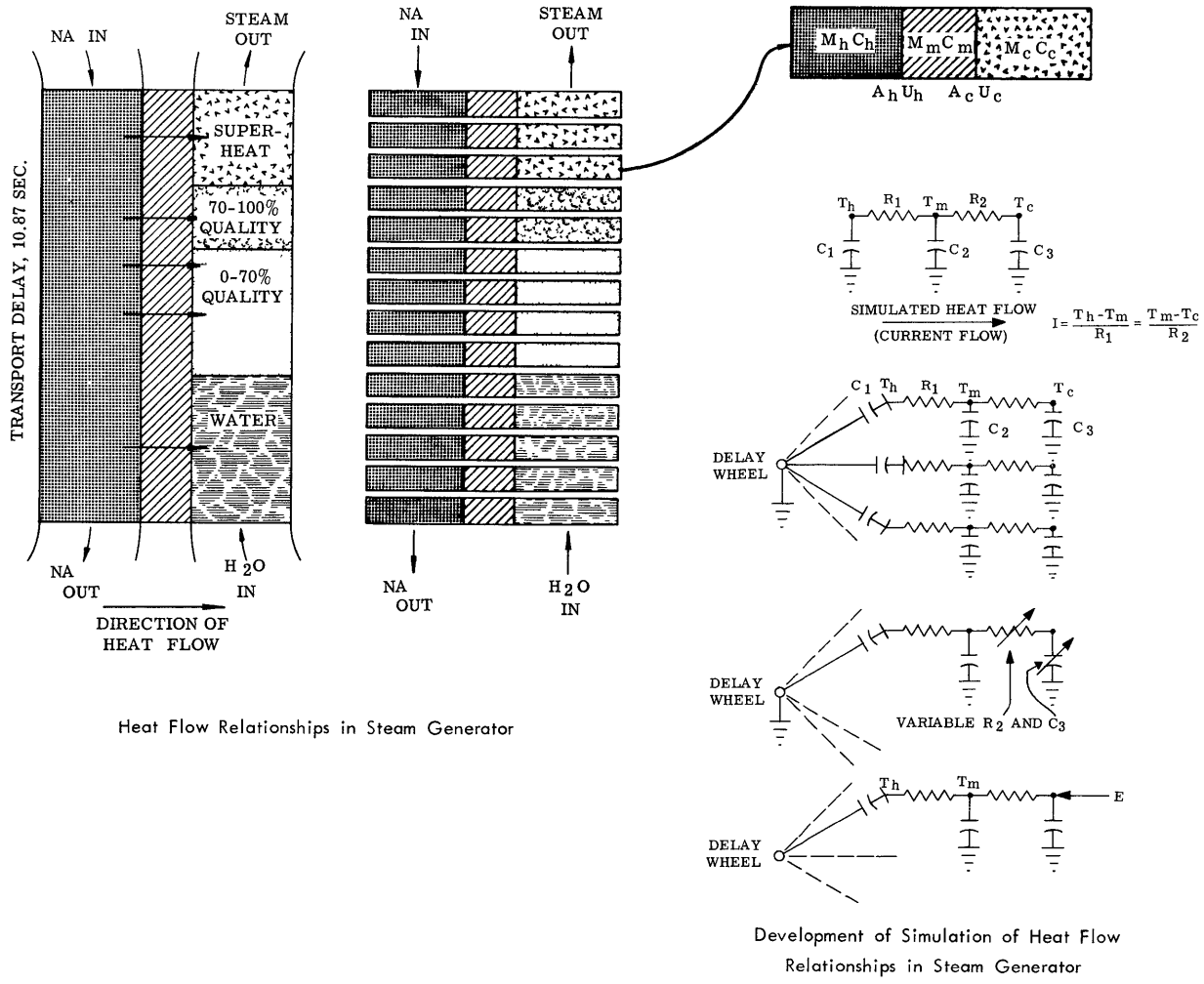
E had to be computed for each section in continuous sequence, and so it was necessary to connect the computation circuit with each T-network in rapid succession using a fast stepping-switch which performs several related functions at each station (section), sweeping continuously from steam generator inlet to outlet, in the opposite direction from the rotation of the delay wheel used to simulate the sodium side. With E so developed, it was possible to use fixed elements (a capacitor and resistor) for H_2O heat capacity and heat transfer coefficient, and yet obtain the correct simulated heat flow conditions.

The H_2O profile obtained by this method of solution is continously displayed on an oscilloscope and provides an excellent means of examining H_2O conditions during performance tests.

Control System Development--PhaseI

After simulations of the individual components described above had been interconnected in proper sequence, the simulator was ready both for testing plant characteristics and for testing and developing control systems for the plant.

Operating within the limits of basic design conditions, and taking into consideration certain ground rules established by APDA, it was decided that control of the plant on the basis of scheduling reactor upper plenum temperature from the demand power setting offered the best promise of maintaining thermal stability of the system, at the same time avoiding undesirable operating conditions.



Heat Flow Relationships in Steam Generator

Development of Simulation of Heat Flow Relationships in Steam Generator

Figure 4. Heat Flow Relationships in the Steam Generator and Development of Their Simulation

With reactor rod control position and feedwater pumping rate as external control mediums, several control systems were studied using the simulator. The system finally adopted embodies rod control in a limited proportional error servomechanism with first derivative output control; feedwater control is a scheduling system with integral error bias. Following optimization of this control system, plant characteristics were studied for a variety of operating conditions, with the optimized control system functioning. Figure 5 presents a block diagram of the control system proposed on the basis of Phase I investigation.

Phase I Results

In addition to considerable work in analyzing the steady-state and dynamic performance of the several plant components, the end products of this Phase I period were the electronic simulator of the plant, and the conceptual design of the operating control system. The Phase I objective had been to simulate only as much of the plant as necessary to include the basic elements directly related to power control, such as the reactor, the sodium loops, the IHX and the steam generator, omitting the section of the plant beyond the steam generator. However, as the study advanced, it became apparent that the counterflow, once-through steam generator has a much faster response than conventional boilers, and that it is in fact the most important and sensitive single element affecting control of the plant. Therefore its simulation should be as accurate as possible, and all factors affecting its performance should be simulated. Thus the entire water-steam hydraulics would

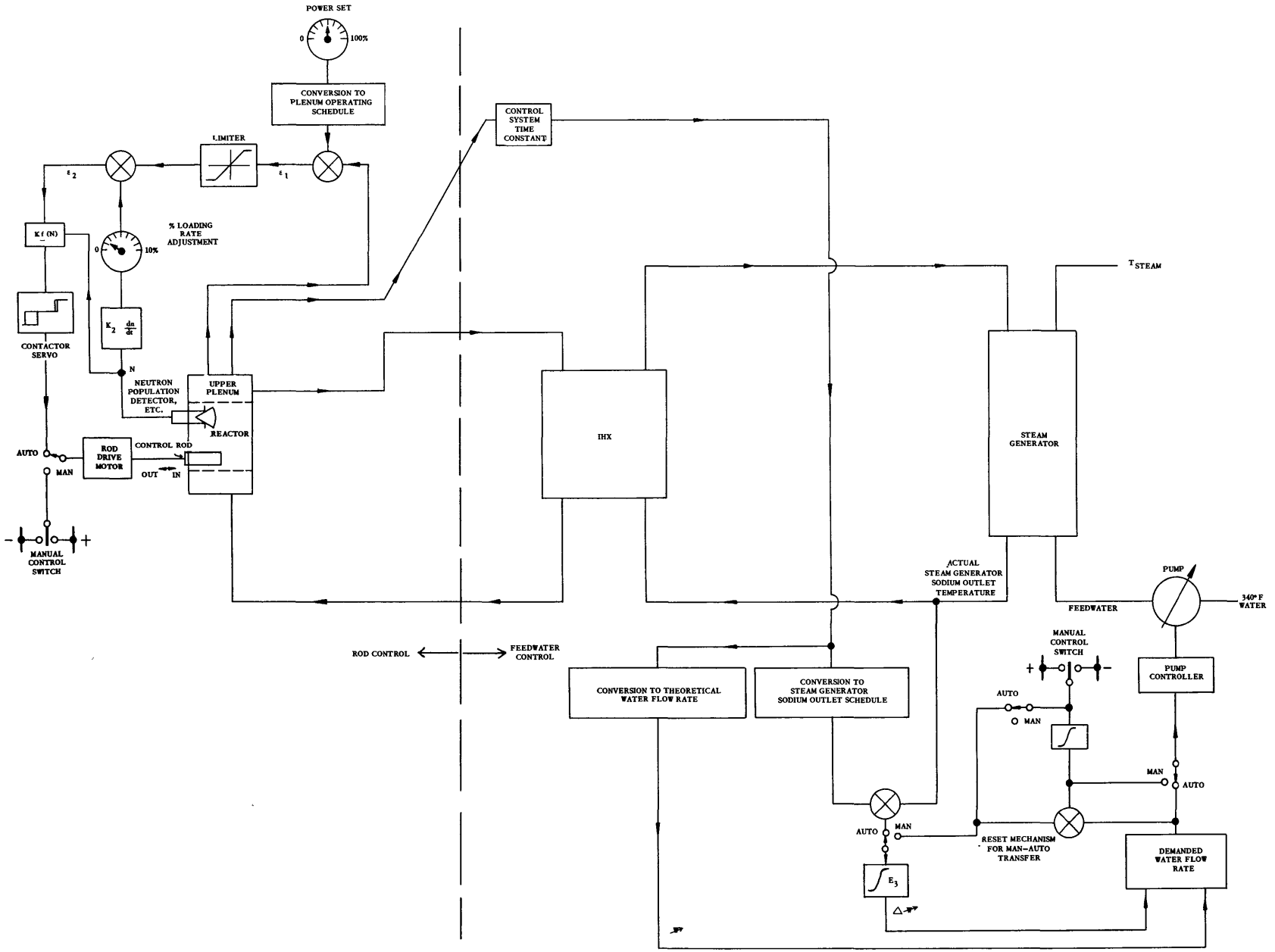


Figure 5. Schematic Diagram of Proposed Control System, Phase I

have to be introduced because of the effect of H_2O pressure and flow rate on the heat transfer characteristics of the steam generator. Further, it had become increasingly desirable to investigate operation at lower power levels than possible with the simulator available at the termination of Phase I. This was primarily due to a new start-up concept, which involved flooded steam generators. Hence an extension and improvement of the control system study was clearly indicated.

Phase II Simulation and Control Studies

Phase II of the control system study was undertaken to expand the simulator's accurate operating range by using an improved basic computer, and also to add an accurate simulation of water-steam dynamics in the power-conversion section of the plant. Other changes in the specially designed supplementary components were introduced to make improvements dictated by the Phase I study. Also the new simulator (Figure 6) was to be developed to serve for training of plant personnel as well as for continuing stability analysis. Since operation is the same in either case with a simulator set up to operate in real time, the training function could be served simply by adding an appropriate training console to the basic simulator.

Further objectives of the Phase II study were simulations of scrams (rapid shutdowns), nuclear start-up, and power start-up with flooded steam generator; a more detailed study of rod control components;and development of feedwater control, represented simply in Phase I, into a complete control subsystem.

Simulator Scaling Changes and Other Improvements

Accuracy of the Phase II simulator is greatly improved over Phase I by the use of higher quality components and a broader voltage range which permits finer scaling. A scaling ratio of 5°F per volt makes the Phase II simulator almost 2 1/2 times as precise as the older one. Low-power-level operation can now be carried on down to 10 percent of rated plant power, well below the level where steam generation will begin in actual

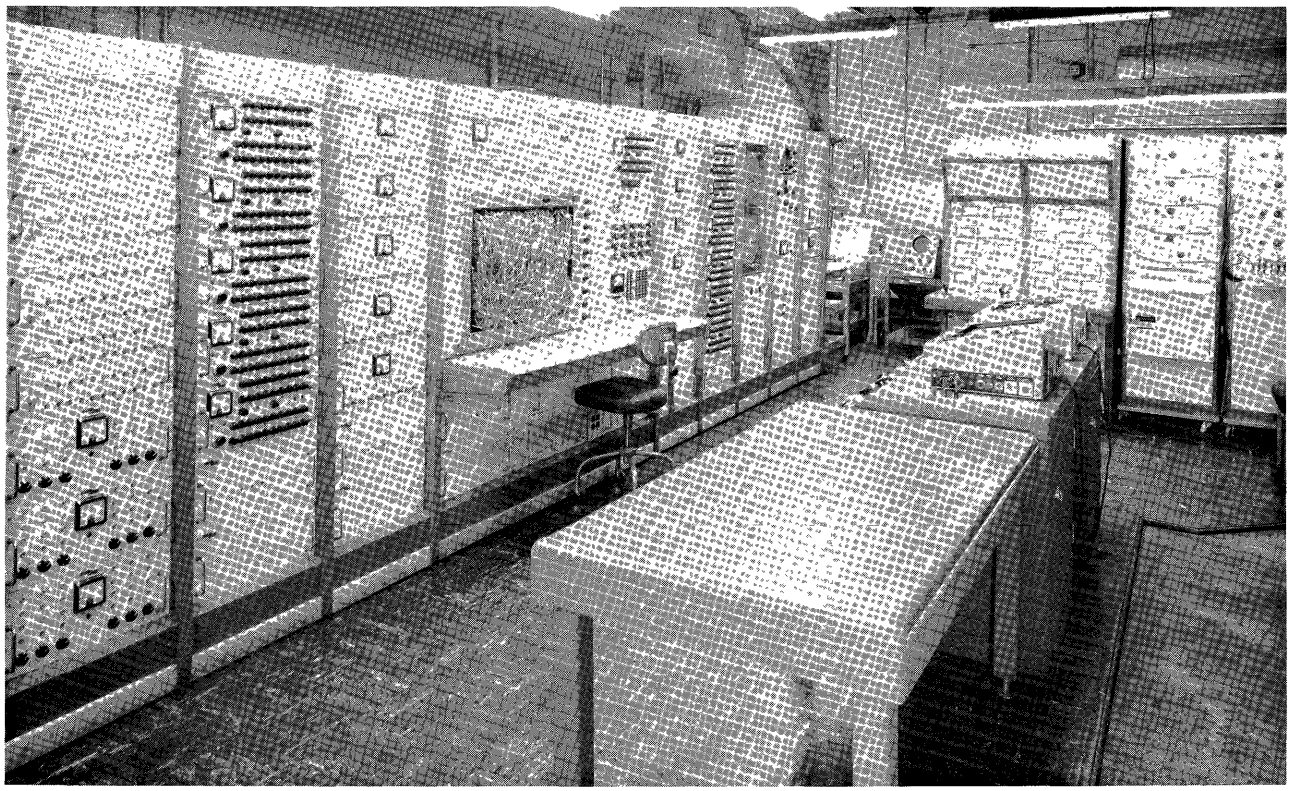


Figure 6. Completed Phase II Simulator with Main Problem Board in Place. Instructor Console at Far End of Room Next to Recording Equipment (right)

plant operation. The upper limit has also been extended by the scaling ratio revision, to as high as 150 percent power operation.

All of the simulated components in the sodium loop benefit from the refined scaling of the simulator. Minor revisions have been made in scaled values throughout the simulator on the basis of data revised by APDA, and in the reactor simulation a circuit has been added to simulate scrams.

While the Phase I study was in progress, Power Reactor Development Company decided to purchase crossflow, counterflow, once-through steam generators instead of the counterflow type originally planned, but the simulation remained basically the same as in Phase I, with the difference in type being accounted for by introducing an equivalency factor determined from digital studies.

Mechanical improvement in the stepping switch, improved electronic components, revised scaling ratio and a different method of calculating heat transfer coefficients all contribute to greater accuracy of the steam generator simulation. However, the major change in Phase II comes from incorporating in the steam generator simulation the effects of variations in pressure and flow rate, respectively, on the relationship of H_2O temperature and heat transfer coefficient with enthalpy, which were not considered in the Phase I study. These pressure and flow inputs to the steam generator simulation are received from the steam-water hydraulics simulation.

Steam System Hydraulics Simulation

A complex representation of the hydraulics and pressure dynamics for feedwater and steam and their related components was undertaken in Phase II. The interplay of valves in the steam system required consideration of short response times of the order of 2 seconds for the turbine throttle valve and steam bypass (dump) valve.

One steam loop as shown schematically in Figure 7, with associated piping and valving, is simulated on the assumption that all three loops operate identically. Included in the simulation are the feedwater pump, feedwater control valve, steam dump (bypass), turbine throttle valve, and auxiliary line for start-up; plus piping friction losses and flow inertia on both water and steam sides, and steam line storage due to compressibility on the steam side. Also included are the extremely important storage effects in the steam generator, in response to changes in flow and pressure. The turbine, condenser and condensate pump are not included.

Considerable pioneering was necessary in devising the simulation of the foregoing elements. Prior to simulating steam system components, extensive analytical studies were necessary to evaluate all variables involved and determine the best approach to their simulation.

Analysis of the steam system was worked out by electrical analogy (Figure 7), using resistors, capacitors, and inductors to represent frictional pressure losses, steam compressibility and fluid inertia, respectively. These analogous electrical elements were then simulated directly without going through the differential equation step, a relatively new technique explained by Larrowe,[1] that saves much time and effort initially and simplifies the task of making changes.

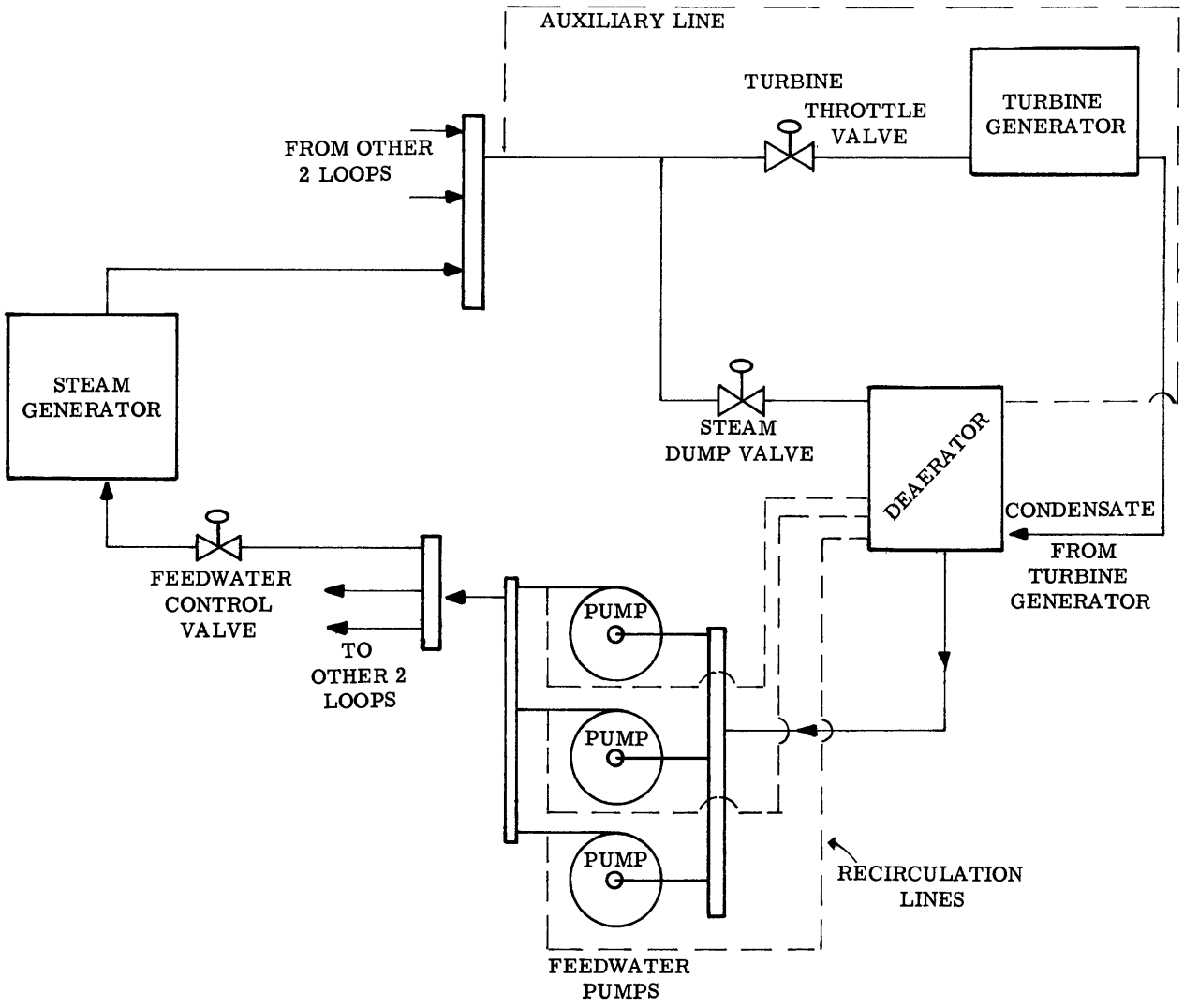
Water storage effects in the steam generator, which account for a predominant part of the transient response of the entire steam system, were a major aspect of the simulation. These storage phenomena cause steam generator output flow rate to lag input flow rate about 15 seconds, which significantly affects the transient characteristics of the system.

Control Studies--Phase II

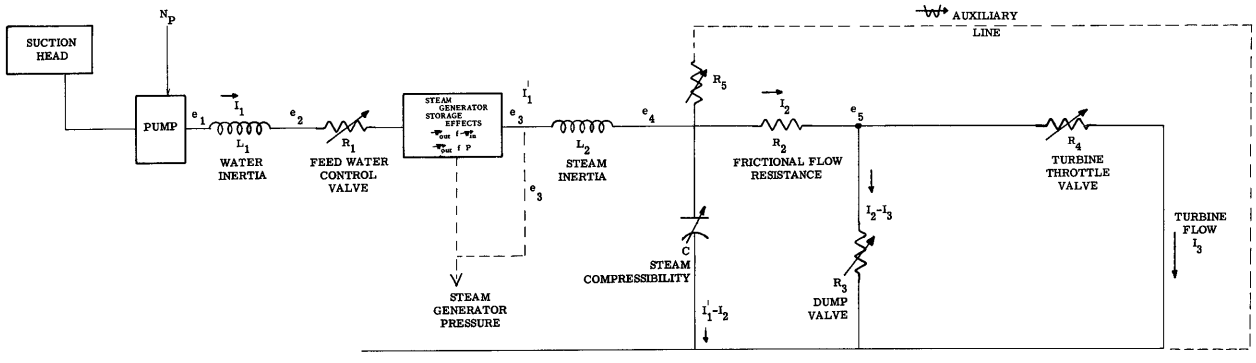
On the basis of the conceptual design developed by Holley in Phase I of this study, the control system supplier designed automatic control system hardware. These control elements were simulated in a more complete form than in Phase I, and stability of the system was investigated with these controls in operation, this time taking into account the important effects of steam-water hydraulics which had been omitted from the Phase I simulation. Results of these studies engendered further refinements in control elements and helped to determine optimum control settings.

As mentioned earlier, the control system for the Fermi plant comprises two divisions: rod control which regulates reactor output in response to power demand; and feedwater control which regulates water flow rate to match power output and to maintain a scheduled sodium temperature out of the steam generator.

Rod Control. Early in Phase II considerable simulation work was done in finalizing the rod



SCHEMATIC DIAGRAM OF STEAM SYSTEM



ELECTRICAL CIRCUIT ANALOGY FOR
STEAM-WATER HYDRAULICS

Figure 7. One Steam Loop Shown Schematically (above) and the Electrical Circuit Analogy Upon Which Simulation Was Based

control system. A large number of computer runs[2*] were programmed to illustrate over-all performance of the system in response to loss of signal and disturbances in reactivity, reactor inlet temperature, reactor coolant flow, and power demand set point.

Feedwater Control. The function of the feedwater control is to regulate feedwater flow rate according to the requirements of reactor power and sodium system, while maintaining steam pressure constant. This control function is performed by manipulation of the feedwater control valve for flow control and the turbine throttle valve and dump valve for pressure control.

The three main divisions of the feedwater control subsystem are an automatic control for the feedwater control valve, a pressure controller for dump valve and turbine throttle valve, and a pump speed control.

Conversion to Operator Training Facility

Following the basic systems analysis work, a control console (Figure 8) was built, duplicating the control panels at the Fermi plant, and connected to the completed analog simulator to convert the entire installation to a training facility for power plant operators. An important factor making it feasible to use this simulator for operator training is that all components of the plant were simulated individually, and therefore it is possible to instrument the training console for indications of input and output temperatures of the various components, or flow rate in different locations in the steam system, just as would be done in the operating control room of the actual plant. As already noted, the simulator was built to operate in real time to facilitate conversion to training purposes.

In the training program now in progress, the simulated plant can be operated with either manual or automatic control, starting from a low level of power (30 MW) and carrying on into very high power ranges (450 MW). Both load and unload type operation, normal shut-downs, and abrupt shut-downs can be tried with either automatic or manual control. The console thus gives operators the feel of plant operation and allows them to develop and practice operating procedures.

The training console, built in the exact size and configuration of the plant control console, shows all instruments and controls just as they will appear in the plant control room. Thus, in addition to items required for operation of the single loop simulation, corresponding items for the other two loops have been dummied on the panel. The operator becomes oriented in relation to appearance and location of all instruments, lights, control levers, etc. This familiarity is vital since he will be isolated from normal plant sounds and these indicators are his only avenues to awareness of plant operation.

Nuclear Start-Up Training and Trouble Simulation

Since the fuel cycle of the Enrico Fermi Atomic Power Plant necessitates frequent reactor

* Details of these performance tests by Holley have been published by others. See Reference 2.

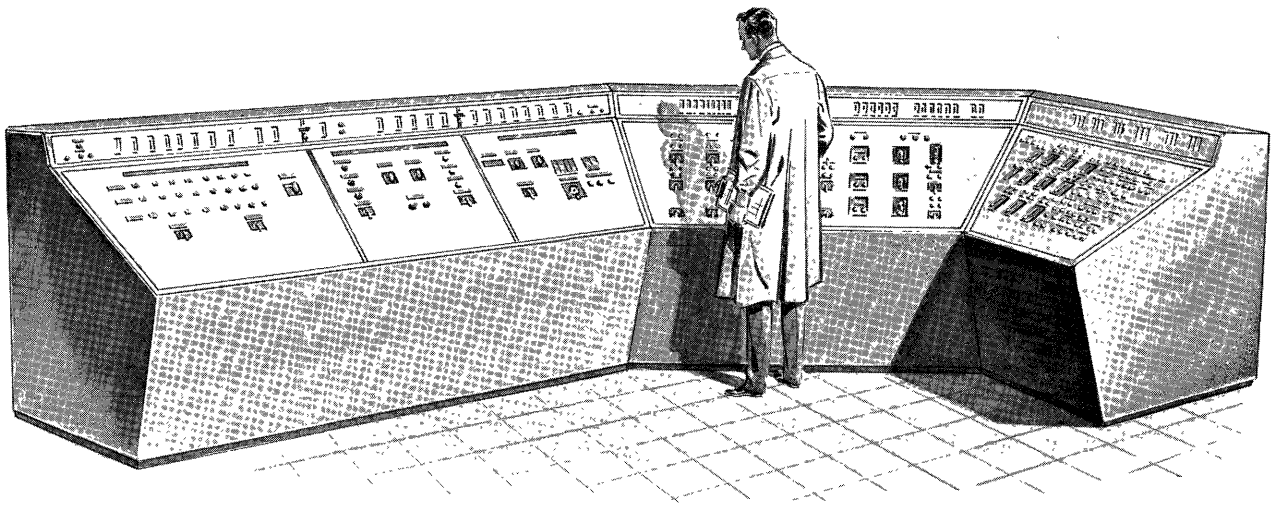


Figure 8. Control Console for the Power Producing Section of the Fermi Plant

start-up from source level, considerable practice in nuclear start-up and its various ramifications is absolutely necessary for plant operators. Using the simulator as a general purpose computer, a new problem board was set up to simulate the physical behavior of the reactor alone from source level--1 watt power--through six decades up to 1 MW.

The simulation was based on standard equations describing the reactor on a one-point basis. Following the method outlined by Franz and Simcic, [3] the equations were transformed to expressions containing the logarithm of N (neutron population) rather than N, because the range of values for N could not be simulated within the usable voltage range of the computer.

A trouble simulation board has also been developed, whereby an instructor can deliberately introduce plant malfunctions into the operating problem without knowledge of the trainees.

Simulator Capability

The complete one-loop simulation of the plant is now available as a systems analysis tool for any further studies required, and serves at the same time as a training device which familiarizes plant operators with both manual and automatic operation. Both systems analysis console and operator training console are connected at all times, and the type of operation desired may be selected at will.

The simulator is accurate to 1 percent, and with the main power problem board in place operating conditions over a range of 30 to 450 MW power can be simulated. With the nuclear start-up patchboard installed, operation from 1 watt to 1 MW can be simulated.

The entire program has thus been able to meet the three basic requirements set forth at the beginning of the project. The facility in its present form will be transferred to the reactor site on or about May 15, 1960, and will become a vital part of the on-site training facilities.

References

1. Vernon Larrowe, "Direct Simulation Bypasses Mathematics," Control Engineering, Nov. 1954, pp. 25-31.

2. "Design of the Reactor Normal Operating Control System for the Enrico Fermi Power Plant with Analog Computer Verification," Research Report No. 1191, Project J.O. 1960BC, Apr. 15, 1959, with addendum of June 23, 1959, Bailey Meter Co., Cleveland, Ohio.

3. J.P. Franz and N.F. Simcic, "Nuclear Reactor Start-Up Simulation," I.R.E. Transactions on Nuclear Science, Mar. 1957, pp. 11-14.

For complete details of this project write for a copy of Report 59-12 with addenda to:

Robert Kley or Samuel Irwin
Holley Carburetor Co.
11955 E. Nine Mile Road
Warren, Michigan

"ANATRAN" -- FIRST STEP IN BREEDING THE "DIGINALOG"

L. A. Ohlinger
Chief of Computing and Datamation Center
Norair Division, Northrop Corporation
Hawthorne, California

Summary

A planned and controlled integration of digital and analog computers could be effected by industry if the final general purpose "diginalog" were produced in steps commensurate with the developing state of the art. The first step, described herein, is the development of an analog symbolic language and digital compiler such that the engineer can write his problems in quasi-mathematical and familiar language which the digital computer can then translate into a digital input function generation, a set of calculated pot settings, and a complete analog flow diagram. Subsequent steps in developing the diginalog are also described.

"The Hiatus Closed" - A 3 Act Play

The Plot

Kipling could well have been speaking of digital and analog computers when he said: "....never the twain shall meet". The gulf that has existed between digital and analog computing people has varied all the way from one of complete disdain to one of open antagonism between the two sects. Only rarely has some prophet raised his voice in supplication for a union of these two remote members of the machina sapiens family. Still more rarely has the voice crying in the wilderness resulted in any actual cooperative effort or in any union of hardware between the general purpose medium or large scale digital machines and the general purpose analog equipment.

Certainly, there has been little concerted effort toward bringing together these two powerful engineering tools in a planned union that will take advantage of the best features of each class of computer and that will lead ultimately to a single still more powerful machine - - the "Diginalog". True, the digital and analog machines live in two different worlds, they speak two different languages, and the only family resemblance is that they are both largely electronic and do computing. However, each has its place in the general scheme of things and can do some things better than the other.

Therefore, it is time to end the artificial segregation and take steps leading toward a marriage of these strangers with the hope of cross-breeding them selectively for their dominant characteristics and gradually eliminating the objectionable recessives. Although several generations of computers may be necessary to achieve the desired paragon, now is the time to take the necessary steps toward the union and cross-pollination.

The Characters Involved

The principal characters in today's plot are Don Digital and Ann Analog, strangers to each other and only distantly related. Don is an aggressive young machine who is developing rapidly into a powerful force in his community because of his many fine characteristics. He has a quick mind, faster than any human being we know, and he is extremely accurate. He can read and write at fantastic rates, do lots of arithmetic rapidly, and rarely makes an error because of his clever ways of checking himself. However, at the time of our story he can handle only one thing at a time and has not learned how to do integration very rapidly.

Don's talents have been extended to many fields of science, engineering, and industry where his dominant characteristics are invaluable. He is equally at home performing complex engineering and scientific computations, or in processing tremendous masses of business and industrial data such as for Finance or Manufacturing.

In other words, Don's is an exact mathematical mind and, even though he reads and writes all alphabetic and algebraic characters as well as many punctuational and mathematical symbols, he thinks of all such symbols in the language of numbers and performs his amazing feats in this language. His is the language of numbers.

Ann, on the other hand, is the more retiring type, and lacks Don's aggressiveness. She has always been the physical type and, accordingly, has learned to understand only the physical things in life. As might be guessed, she knows what to do with curves. She is a clever mimic and can simulate other physical things very cleverly but she can express her thoughts only in physical form, and converts these thoughts to numbers only very crudely and with some difficulty. Hers is the language of physical quantities and curves.

This does not mean that her mind is not facile, because actually she can handle a number of varying ideas simultaneously and, with almost feminine intuition, arrive at an answer. Her mind is agile too, and she can arrive at a conclusion based on a number of varying facts almost instantaneously, although the answer may be somewhat approximate.

Accordingly, Ann's forte is obtaining approximate answers instantly, performing complex jobs like integration instantly, and coming up with an answer to what happens when a lot of inputs are varying simultaneously or, as the engineers and scientists say, when there are several "degrees of freedom".

The Play Unfolds (in synoptic form)

Prologue. Ann and Don have talked only on rare occasions because of the language barriers and with not too much success. It is difficult to carry on any kind of a courtship or even an intelligent conversation with a third person translating each statement and answer. In addition, it makes the courtship a long drawn out affair when each must sit idle and wait for the translator to convert one's statements into a language understandable to the other.

However, Dan Cupid will not be outdone and so, with the aid of the author, a new element is injected into the plot -- a new means of communication by which Don can speak to Ann. These, then, are the characters that we hope to bring together today, at least in the courtship stage, through a new proposed language we have named "Anatran".

Act I -- The Courtship. During the past few years, Don Digital has been learning some new methods of translating the thoughts and symbols of homo sapiens into the only language Don understands so that he can take over part of this translation that used to be so frustrating when it was necessary to depend upon slow, inaccurate humans. At this time, he is capable of accepting an almost completely human language and converting it, at his own rapid and accurate rate, to his own dialect so that he can handle the problem at his own fantastic speed and accuracy.

However, there is one drawback -- there must be some relationship between human language and the slightly pidginized computerese that is the present communication link between man and machine. For example, Don, in his 704 form, accepts Fortran; in his Univac form he accepts Mathmatic; and so on.

Now, a new computerese titled "Anatran" has been conceived and is being studied so that Don can dialect, translate it into a sort of love letter -- the analog flow diagram and set-up data that Ann's attendants can use directly for Ann. This includes her pot settings, her wiring flow diagram, the number of pots and linear and non-linear elements she needs, and even an easy means for the attendants to check that Don has not said anything inadvertent in his missive.

At this point in Act I, Don's attendants, who have been studying analogese, have developed a simple and easy means of producing the missive that will convey all this information to Ann's attendants for her consumption. Having developed the input and output of the new language, Anatran, Don's attendants are now looking into the actual writing of the vocabulary and rules of syntax. (See the author's footnotes and comments for details of the actual missive.)

It should be noted that, in the approved manner of courtship, Don is the aggressive one and not only takes the lead in communicating with Ann, but also controls Ann's actions.

Act II -- The Engagement. Inpatient and chafing at the slowness of Ann's attendants in translating Don's missive for Ann, the idea is conceived of adding to Don's missive another communication medium in the form of a paper tape or magnetic tape on which all of the pot settings are listed.

With the help of an automatic pot setter, all of Ann's pots are set automatically in a matter of minutes instead of the hours formerly consumed by her attendants. Thus a physical bond is established in which Don not only directs Ann's actions but actually furnishes to her a part of his own handiwork that enables her to perform her own marvelous feats more rapidly.

Act III -- The Marriage. There still exists one weak link in the union between Don and Ann -- the slow, laborious task of wiring Ann's patchboards from the instructions in Don's missive. With the assistance of a new technique of electronic switching and, perhaps, the help of some relays, means are found whereby the patchboard wiring by Ann's attendants can be eliminated. However, instead of relaying the instructions from this new automatic patchboard wiring system by means of the same paper or magnetic tapes, it is time for the actual marriage of these two, who have now learned to communicate and work together toward joint goals in life.

Accordingly, a physical bond is made so that Don can send directly to Ann all of the instructions that will not only set her pots automatically but also will perform all internal wiring automatically so that Ann is ready to perform her appointed tasks quickly and efficiently with little lost time between jobs. As before, masterful Don "wears the pants" in the family and tells Ann what to do, because he is the stable and reliable one that can be depended upon to produce the same results time after time, whereas Ann, in true femine fashion, feels that an approximate answer is good enough as long as she continues to handle the jobs for which she is best suited.

It is an ideal union because Don is still free to do his tasks as he pleases without interference from Ann; he is free to call upon as much help from Ann as he needs to do those parts of his job for which she is best suited; and with it all, he can direct Ann and assist her in setting up and performing those jobs on which she is best suited to work alone. Would that all families in homo sapiens might emulate this marital status!

Epilogue. The marriage of Don Digital and Ann Analog has brought together finally these two classes of machina sapiens into a unified system that exploits the best features of each of the individual members but it still is a union of two existing systems and is not a single system designed specifically to exploit only the best features of both systems and eliminate those features that are less desirable or superfluous.

Thus, the next generation will be the offspring

that achieves this goal -- the machine designed to perform either digital or analog operations or both in the most efficient manner possible. This is the "Diginalog".

Footnotes and Comments

The play is ended and our flight of fancy is over. However, the allegory contained therein is not one based on imaginary conditions or wishful thinking. It is founded upon actual facts coupled to new concepts that the author and his people at Northrop have developed.

Retrospection

Evolutionary Relationships. Probably one of the principal reasons that digital and analog computers have remained so far apart has been the historical development of these two classes of computing equipment. Analogs, as the name implies, started not as computers but as physical systems designed to simulate by analogy other physical systems whose characteristics and behavior under various conditions of operation were to be predicted. In this role, they served the engineering and scientific world very well, but their invasion into the mathematical world always was limited by their very low accuracy and precision. Only with the advent of electromechanical and electronic equipment did the application of analog computers to the solution of engineering and scientific problems become of significant value.

Meanwhile, digital computers, even in their earliest crude forms, have always had the advantage of accuracy but they have been comparatively slow. With the many advances in electromechanics and electronics, digital computers have advanced in speed and capability to the point that their uses are becoming increasingly widespread. However, there still have been two different classes of digital computers developed -- the calculators and the data processors, as indicated in Table I. The differences shown there in their characteristics resulted in differences in logical design and equipment.

Fortunately, the continued use of calculators and data processors has led to the changes in characteristics indicated in Table II so that there no longer is any reason for the next generation of computers to have two different types of equipment for the two types of functions. We have now reached that stage in the state of the art in which the next generation of computers and data processors will have one and the same machine performing both types of functions with equal facility.

Earlier Marriages - Future Problem-Oriented Languages (POL's). Because of these different lines of development, the jurisdiction over the two classes of digital computers and over the analog computers has almost always been separate and unrelated, with each type of equipment under the control of a different group in the organization. Thus, there was little opportunity or incentive to bring these different machines together in any attempt to improve the efficiency of solving problems by exploiting only the best features of each class of computer in a joint solution. In a few cases, small analog-to-digital (and vice versa) converters permitted a limited amount of intercommunication by digitizing analog information or interpolating discrete digital information into continuous form. Convair with its application of DATRAC is an example of interposing a translator.

Ramo-Wooldridge with its large converter that connected the 1103A and the analog equipment, is another example of the form earlier attempts have taken in interconnecting physically these two classes of equipment. However, the problems of such an approach to digital/analog operation are fairly obvious when neither equipment dominates the picture. Program interruption on the digital machine when called for by the analog is not always a desirable mode of operation nor is it usually economical to let either class of equipment stand idle, waiting for the other to be available.

The new concepts proposed herein would not be possible were it not for the advent of the newer types of internally programmed electronic data processing machines (EDPM) that operate at high speeds and, particularly, those that permit the performance in parallel of several functions simultaneously. It would not have been possible either were it not for the ingenious development of the new POL's and their digital translators that permit these languages to be converted for and accepted by a digital computer that speaks in a different dialect. The fast memory and high speed of the new EDPM make it possible for this equipment to accept POL's that express the problem in quasi-mathematical or engineering form and translate these into machine language for solution on that particular digital computer.

Northrop's Experience

Fortunately, at Northrop, the basic philosophy of computing and data processing has taken a different turn from the segregation described above. Located in the heart of the new Engineering, Science, and Research Building is the Computing and Datamation Center referred to as COMPADAC for simplicity. It includes a large analog installation with precision EAI equipment, an IBM Type 704 EDPM, the necessary EAM equipment for support of the 704, and miscellaneous other equipment. (By December 1960, the 704 will be replaced by an IBM Type 7090 or a TRANSAC S-2000.) Its functions and responsibilities run the entire gamut of computing and data processing including digital and analog calculation, analog simulation and model testing, engineering and flight test data reduction, business data processing, reconnaissance/intelligence data handling, and even a moderate amount of research and development. Because these various functions have been centralized under one authority, it has been possible to enforce a greater integration of the various types of functions with a resulting improvement in efficiency and capability.

For example, every former analog analyst has

been thoroughly trained in digital programming, and every digital programmer has been trained in the programming and use of analog equipment so that there exists no language or specialization barrier between the people who are responsible for these two classes of computing equipment. The only problem, then, is one of having the programmer decide which equipment to use for the solution of a particular problem and how best to get the answers rapidly and economically.

This has not always been so, and the author can remember the time when the rivalry between these two factions was great. It even crept into the lunch hours when digital and analog programmers played chess and proudly announced to the author as he went by, "See! analog vs. digital!". However, through the forced integration of these two factions by the Chief of COMPADAC, this rivalry has been eliminated and the effort now is toward better use of the equipment, either independently or jointly.

The latest development along this line has been the concept of a series of progressive steps leading toward a better integration of the two classes of equipment than has been achieved in the past by plain converters. This new concept will ultimately result in the "Diginalog" (which will combine the best features of each class of equipment without the superfluous or objectionable features), but it will not spring up full blown. Rather, it will develop as a series of logical steps commensurate with the current and changing state of the art.

On To Diginalog

Anatran - The First Step. The first of these steps (which is the basis for this paper) is the development of a problem-oriented language that will permit translation on a digital computer of the complete input information for an analog computer. This new language has been termed "Anatran" (ANAlog TRANslation).

The first step in developing Anatran was to determine the feasibility of producing an acceptable output from Anatran on existing equipment -- a wiring flow diagram for the analog computer such that one of the programmers could wire a patchboard for the analog computer directly without further translation or interpretation. The conventional flow diagram produced by humans for the translation of the problem statement into patchboard wiring is typically a symbolic wiring diagram involving circles, triangles, rectangles, etc., as indicated in Fig. 3. To produce such a diagram on current EDPM would involve a number of technical problems such as the introduction of new print characters on the output printer and the storage of the complete diagram with interconnecting lines while selecting individual and unrelated areas for printing one line at a time, as most digital output printers do.

Accordingly, a new type of wiring flow diagram was developed based upon the dc operational amplifier or the non-linear element as the fundamental unit. In this new flow diagram format, the printout format is arranged to include the inputs and outputs of each such basic element as a complete entity with that element, as indicated in Fig. 4. This will become clearer as we follow the details of the new type flow diagram and compare it with the earlier form shown in Fig. 3.

Concurrent with the development of the new format, it was found possible to include on this wiring flow diagram a complete list of all potentiometer settings in numerical order so that the diagram can also be used for manually setting the pots on the analog equipment. In addition, by including the origin of all inputs and the destination of all outputs, this same diagram becomes the means whereby the programmer or analyst can check the accuracy of the formalized flow diagram prepared by the digital computer. This will become clearer as we follow the diagram in detail. Finally, the diagram also provides a notation of the total number of potentiometers and linear and non-linear elements required by the problem.

A Typical Problem. Now let us look at a typical problem to be solved on an analog computer -- that of determining the stability of an aircraft in steady horizontal flight, with and without an autopilot, as the tail is struck by a vertical gust of short duration. The equations are shown in Fig. 1 together with the coefficients and other data on the input parameters in Fig. 2.

Normally, this problem would be programmed either by the originating engineer (if he were one of those trained by COMPADAC) or by one of COMPADAC's programmers. From the mathematical statement of the problem, a flow diagram would be constructed showing the complete step-by-step solution of the problem and closing all loops. All potentiometer settings, scale factors, voltage levels, and initial conditions would be determined and listed so that the potentiometers and other equipment could be set up for proper solution of the problem. A patchboard would be wired, checked, and put into a control console in the equipment. When all of the setup had been checked, the machine would be turned on for solution and the results recorded on six-channel strip chart recorders, changing input conditions as desired.

Now let us refer again to Figs. 3 and 4. The flow diagram in Fig. 3 is the manual product that will normally result from the engineer's or programmer's work except for the fact that the equations, pot numbers, amplifier numbers, and output destinations would not be shown normally on the diagram. The sophisticated programmer does not need such data on the diagram and they are shown here for the reader's convenience in following, in detail, the comparison between Figs. 3 and 4. The numbers within the circles are always gain factors while the numbers within the triangles are the voltage levels for the amplifiers and represent the number of volts per unit of the quantity produced by the amplifier.

Fig. 4 is the corresponding wiring flow diagram, listing of pot settings, check sheet, and

notation of the number of pots and functional computing components required, as it will be produced by the proposed Anatran. This will be the final product of the 704 (or other EDPM) output printer. Once familiarity with the format has been realized, no headings are really necessary for the columns since their content is almost self-defining. However, it is no problem for Anatran to produce the necessary headings over the various columns and over the various groups of columns that constitute a functional field on the diagram. Pre-printed paper is unnecessary and would be impractical if the analog problems were mixed in with the digital problems handled by the EDPM.

The similarity between Figs. 3 and 4 may not be apparent on first observation but a little explanation should make the principles of Fig. 4 perfectly clear. We know this because we have tried out the new format on two different programmers who were unacquainted with its development and who had to translate its contents into patchboard wiring with only a brief explanation of the rules of format. One of these had originally been trained in digital programming and the other in analog, although each does both classes of work now. Each of these men found the format easy to grasp and simple to apply accurately.

Let us follow the flow diagram and tabular data in Fig. 4, explaining the ground rules as we go. The first six columns list in sequence the inputs to any amplifier or non-linear element in the form of a name for the quantity depicted, its source, and the various data connected with potentiometer settings and the input. The next two columns indicate the voltage level and type of analog component involved, together with its identifying number. In this case, the numbers are assigned serially while the letters are assigned according to the function of the analog component according to the code listed at the bottom of Fig. 4. The last two columns indicate the output of the analog component by name and destination.

The diagram is built up one component at a time, with the component name or identification letter and number listed oposite the last input to that component as indicated in Fig. 4. Following the complete input and output story of any element, a line is skipped to indicate the change to the next element and the new inputs are listed.

Notice that where no potentiometers are involved, such data is omitted. The components in Figs. 3 and 4 have been lettered and numbered identically for exact comparison, and the reader can trace through the diagram shown in Fig. 4 by comparison with the corresponding steps in Fig. 3.

The reader will observe in Fig. 4 that the potentiometer numbers and settings are listed simply and clearly in columns 3 and 4 so that the diagram constitutes a list for the potsetting operation. He should also observe that the "component count" table at the bottom of Fig. 4 indicates the number of elements required to solve that particular problem.

The wiring can be accomplished by patching cords from the output destinations in the last column to the corresponding input information in the input field. Notice, too, that this arrangement of a tabular form of flow diagram permits easy checking of the solution by the programmer to make sure that all loops are closed and all inputs and outputs included.

In order to extend these same principles developed in this simpler form of stylized flow diagram to other analog components, comparable flow diagrams were developed for other problems involving servo-multipliers, resolvers, ungrounded pots, and diode limiters. The problem statements, and corresponding manual diagram and EDPM-produced flow diagram are shown in subsequent Figs. 5 and 7. As the reader compares the corresponding diagrams, the extension and application of the same principles described before will become self-evident.

Having determined that it is feasible to produce on existing EDPM (without modifying that equipment) a suitable flow diagram, listing of pot settings, easy checking diagrams, and a notation of the number of components required, the next step is the development of the vocabulary and syntax required to translate engineering and quasi-mathematical statements such as shown in Figs. 1 and 5 into the output that has been shown in Figs. 4 and 7. The development of the Anatran translator program itself is our next goal. This is the "courtship" of our allegory.

Mechanization. The next step in automatizing the setup of general purpose analog equipment by use of a digital computer requires no new digital equipment and is also commensurate with the state of the art. Currently, the manual setting of potentiometers (which is a laborious task that is time consuming and easily subject to errors) is being replaced at Northrop by an automatic potsetting device employing punched paper tape as its input information source. As the digital computer computes the potsettings, scale factors, gains, and voltage levels, and develops the end product shown in Figs. 4 and 7, it is possible for it to produce punched cards or punched paper tape that contain just the potsetting information. Cards can be converted into punched paper tape (or even magnetic tape) format so that the tape can be fed to the automatic potsetting device and thereby eliminate the manual step of setting pots. This is the "engagement" referred to in our allegory.

Means already have been developed for eliminating the analog patchboard wiring by the use of automatic crossbar switching, but such equipment is very large and expensive and difficult to justify. The author and his people have ideas about the possibility of electronic switching devices in matrix arrangement (with or without relays) that will perform the same functions as patchboard wiring more rapidly and less expensively, but this will be the subject of a later paper. This is the "marriage" referred to in our allegory.

When this marriage is consummated, it will be

possible for the digital computer to determine the entire course of the solution of a problem on both digital and analog equipment. If the problem is one for which the digital machine is best suited, it can solve the problem without recourse to the analog, but it can, meanwhile, set up the analog equipment to solve problems that are best suited to analog solution while it is doing its own work. Thus, both kinds of equipment will be kept busy under the control of the digital machine. Problems that the analog portion of the united equipment might solve would include "quick and dirty" solutions that do not warrant greater accuracy and precision, parametric studies that involve many cases, and multiple degree of freedom problems.

However, in those cases in which the problem calls for integration or some of the other functions peculiarly suited to analog equipment, the digital machine can perform its own part of the solution and call upon the analog to do the integration or other function when and if required.

Extrapolation

One can extrapolate the principles of this united digital/analog computer into the "Diginalog" in which the logic of the machine is such that only the best features of each class of computing equipment are exploited and all other superfluous or redundant features are eliminated.

Of course, with crossbar switching, vacuum tube or relay switching, or matrix electronic switching, it is possible to retain the individual identity of the digital and analog machines as described in the "engagement" stage above, by utilizing punched paper tape or magnetic tape output from the 704 as the input information for controlling the automatic patchboard wiring device. Such a technique would postpone the "marriage" if the latter were uneconomical compared to the cost of the added manual step in moving tapes from the EDPM to the analog machine.

Likewise, the digital differential analyzer (DDA) can perform some of the desirable tasks of the conventional analog with dc operational amplifiers while improving the accuracy and repeatability of the solution. Therefore, it might be equally desirable to develop the interconnection first by use and later by hardware of EDPM and DDA equipment much as has been described in the text above. The principles of developing the interrelation will be the same.

TABLE I

THE FIELD OF COMPUTERS
Prior to 1959-1960

CLASS → / ITEM ↓	Calculators	Data Handlers	Simulators and Model Testers	Controls
Amount of Data Handled	Small	Large	Varies	Varies
Amount of Calculating	Large	Small	Varies	Varies
Usual Type	Digital and Analog	Digital	Analog	Analog and Digital
Usual Size	Small to Large	Large	Small to Large	Small to Large

TABLE II

THE FIELD OF COMPUTERS
After 1959-1960

CLASS → / ITEM ↓	Calculators	Data Handlers	Simulators and Model Testers	Controls
Amount of Data Handled	Large	Large	Varies	Varies
Amount of Calculating	Large	Large	Varies	Varies
Usual Type	Digital and Analog	Digital	Analog	Analog or Digital
Usual Size	Small to Large	Large	Small to Large	Small to Large

(For Interpretation Of These Tables, Refer To Text)

Longitudinal Motion Of An Aircraft With

And Without Control By An Autopilot

Equations Of Motion

$$\dot{u} = -Au - Bw - B\theta$$

$$\dot{w} = -Du - Ew + G\dot{\theta} - H\delta$$

$$\ddot{\theta} = -Ju - Kw - L\dot{w} - M\dot{\theta} - N\delta + F_1$$ (This is the gust disturbance of forcing function)

Autopilot Equation

$$\delta = +P\theta + R\dot{\theta}$$

Load Factor Equation

$$\Delta n = S\dot{\theta} - T\dot{w}$$

Altitude Equation

$$\dot{h} = -V\theta + Yw$$

Definition Of Symbols

u	=	Forward velocity of the aircraft	δ	=	Total control surface deflection
$\dot{u}$	=	Forward acceleration of aircraft	h	=	Altitude of the aircraft
w	=	Vertical velocity of the aircraft	n	=	Load factor
$\dot{w}$	=	Vertical acceleration of aircraft	F_1	=	An input disturbance to pitch acceleration
θ	=	Pitch angle of the aircraft			

Notation

Capital letters A, B, C,.....Y represent constants that will be determined and put into the equations for each set of conditions in the problem.

The single dot over a symbol represents the rate of change of the quantity represented by that symbol; e.g., $\dot{u}$ = rate of change of the velocity u.

The double dot over a symbol represents the rate of change of the quantity represented by the single dotted symbol; e.g., $\ddot{\theta}$ = rate of change of the "rate of change" of the pitch angle θ.

MATHEMATICAL STATEMENT

OF A TYPICAL PROBLEM

FIG. 1

Pot Set Scaling With Positive Coefficients

COEFFICIENT		FACTORS		POTENTIOMETER	
Symbol	Value	Scale (Voltage)	Gain (Resistor)	Identity	Setting
A	.0059	1	0.1	1	.0590
B	.0366	1	0.1	2	.3660
C	32.20	100	0.5	3	.6440
D	.1030	1	1	4	.1030
E	.4480	1	1	5	.4480
G	536.0	100	10	6	.5360
H	27.50	100	1	7	.2750
J	.000450	0.01	0.1	8	.4500
K	.0122	0.01	5	9	.2400
L	.0000037	0.01	0.05	10	.0074
M	.3500	1	5	11	.0700
N	1.270	1	2	12	.6350
P	1.500	1	10	13	.1500
R	1.500	1	10	14	.1500
S	17.64	100	0.2	15	.8320
T	0.031	1	0.1	16	.3110
V	536.0	1000	1	17	.5360

SET UP OF TYPICAL ANALOG PROBLEM

FIG. 2

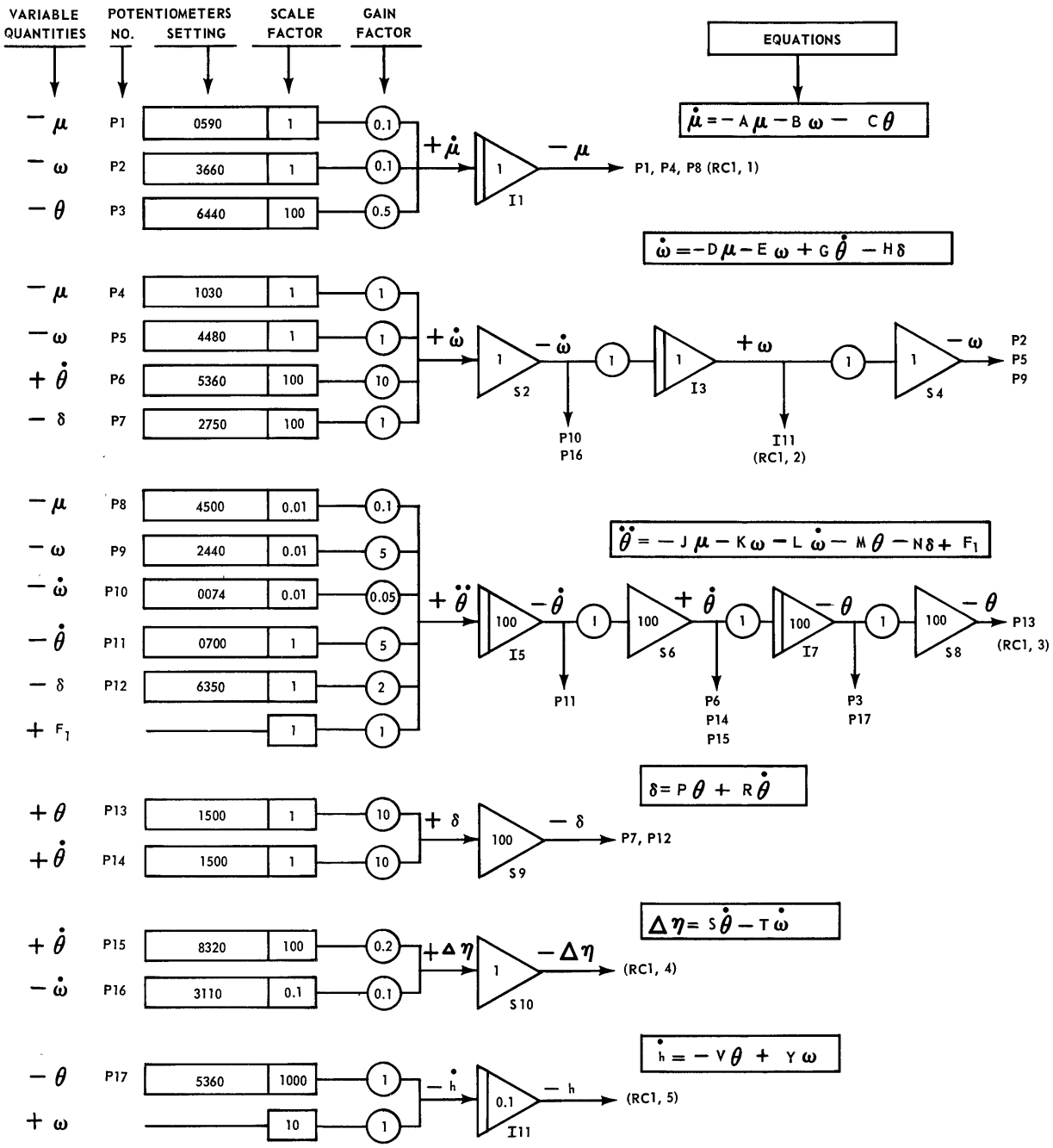


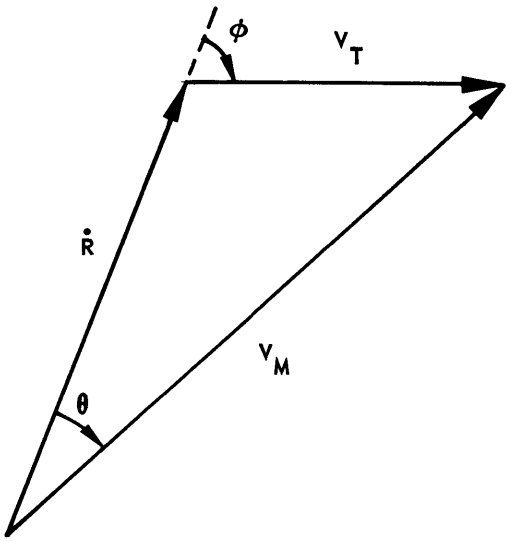
Figure 3
ANALOG BLOCK DIAGRAM (BY EQUATION, MANUALLY PRODUCED)

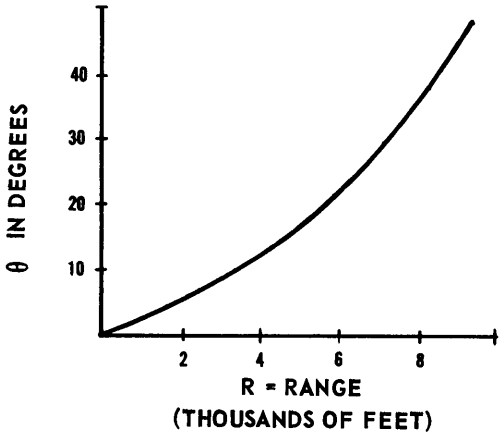
INPUT						AMPLIFIER OR NON-LINEAR ELEMENTS		OUTPUT	
QUANTITY	SOURCE	POTENTIOMETER NO.	POTENTIOMETER SETTING	GAIN	SCALE FACTOR	VOLTAGE LEVEL	ELEMENT AND NO.	QUANTITY	DESTINATION
-U	I 1	1	0590	0.1	1				
-W	S 4	2	3660	0.1	1				
-THETA	I 7	3	6440	0.5	100	1	I 1	-U	P1,P4,P8,(PC1,1)
-U	I 1	4	1030	1	1				
-W	S 4	5	4480	1	1				
+THETADOT	S 6	6	5360	10	100				
-DELTA	S 9	7	2750	1	100	1	S 2	-WDOT	I3,P10,P16
-WDOT	S 2			1		1	I 3	+W	S4,I11,(PC1,2)
+W	I 3			1		1	S 4	-W	P2,P5,P9
-U	I 1	8	4500	0.1	0.01				
-W	S 4	9	2440	5	0.01				
-WDOT	S 2	10	0074	0.05	0.01				
-THETADOT	I 5	11	0700	5	1				
-DELTA	S 9	12	6350	2	1				
F1				1	1	100	I 5	-THETADOT	P11,S6
-THETADOT	I 5			1		100	S 6	+THETADOT	P6,I7,P14,P15
+THETADOT	S 6			1		100	I 7	-THETA	P3,S8,P17
-THETA	I 7			1		100	S 8	+THETA	P13,(PC1,3)
+THETA	S 8	13	1500	10	1				
+THETADOT	S 6	14	1500	10	1	100	S 9	-DELTA	P7,P12
+THETADOT	S 6	15	8320	0.2	100				
-WDOT	S 2	16	3110	0.1	1	1	S 10	-DELTAN	(PC1,4)
-THETA	I 7	17	5360	1	1000				
+W	I 3			0.1	10	0.1	I 11	+H	(PC1,5)

POTENTIOMETERS	AMPLIFIERS	MULTIPLIERS	RESOLVERS	RECORDERS
17	11	0	0	1

LEGEND:
- An = High Gain Amplifier (n=no.)
- In = Integrator (n=no.)
- Mn = Servo Multiplier (n=no.)
- Sn = Summer or Sign Reverser (n=no.)
- NLPnt = Non-Linear Potentiometer (n=no.,t=tap(T=top,C=center,B=bottom))
- Pn = Potentiometer (n=no.)
- (RCn,c) = Recorder (n=no.,c=channel)
- Rn,c = Resolver (n=no.,c=output channel (1 or 2))

Figure 4
ANATRAN OUTPUT
(Wiring block diagram, list of pot settings, checking diagram and component count. Produced on 704 or other EDPM output printer.)





V_T = TARGET VELOCITY

V_M = MISSILE VELOCITY

θ = LEAD ANGLE

$\dot{R}$ = RANGE RATE

ϕ = TARGET ANGLE WITH RESPECT TO RANGE

EQUATIONS:

$$\dot{R} = V_T \cos\phi - V_M \cos\theta$$

$$\dot{\phi} = (V_M \sin\theta - V_T \sin\phi)/R$$

CASE 1:

V_T = 500 FT/SEC

V_M = 700 FT/SEC

R_o = 4000 FT

ϕ_o = 0.5 RADIAN

VOLTAGE LEVELS:

V_T	0.1v/UNIT
V_M	0.1v/UNIT
R	0.01v/UNIT
$\dot{R}$	0.1v/UNIT
θ	0.5v/DEGREES
ϕ	100v/RADIAN

Figure 5
A COLLISION COURSE SIMULATION PROBLEM

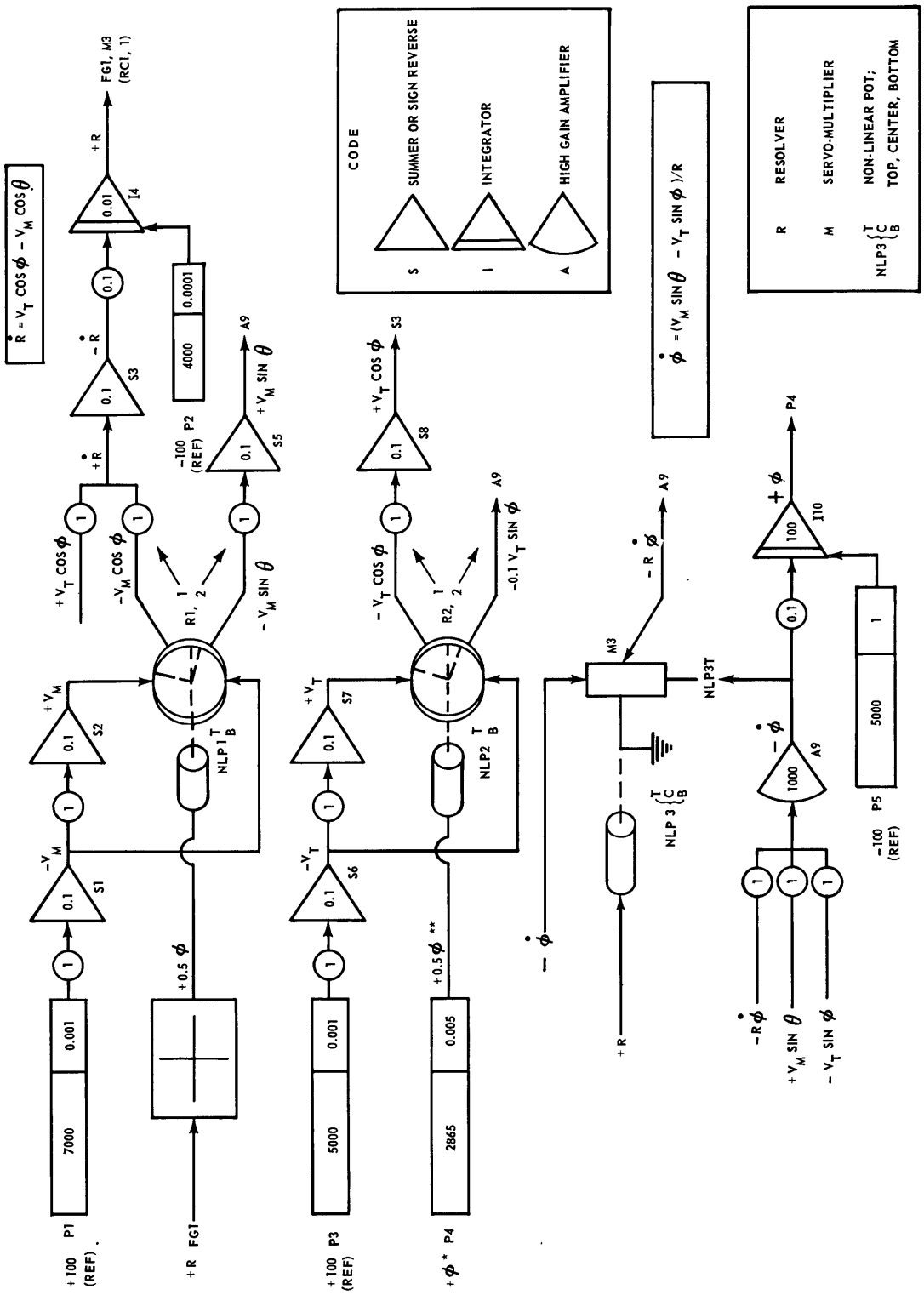


Figure 6
ANALOG BLOCK DIAGRAM - MANUALLY PRODUCED

INPUT						AMPLIFIER OR NON-LINEAR ELEMENTS		OUTPUT	
QUANTITY	SOURCE	POTENTIOMETER NO.	POTENTIOMETER SETTING	GAIN	SCALE FACTOR	VOLTAGE LEVEL	ELEMENT AND NO.	QUANTITY	DESTINATION
+100	REFVOLT	1	7000	1	0.001	0.1	S 1	-VSUBM	NLP1B,S2
-VSUBM	S 1			1		0.1	S 2	+VSUBM	NLP1T
+0.5THETA	FG 1								
+VSUBM	S 2	NLP1T					R 1,1	-VMCOSTHETA	S3
-VSUBM	S 1	NLP1B					R 1,2	-VMSINTHETA	S5
-VMCOSTHETA	P 1,1			1					
+VTCOSPHI	R 2,1			1		0.1	S 3	-RDOT	I4
-RDOT	S 3			0.1					
-100	REFVOLT	2	4000	IC	0.0001	0.01	I 4	+P	FG1,M3 (RC1,1)
-VMSINTHETA	P 1,2			1		0.1	S 5	+VMSINTHETA	A9
+100	REFVOLT	3	5000	1	0.001	0.1	S 6	-VSUBT	NLP2B,S7
-VSUBT	S 6			1		0.1	S 7	+VSUBT	NLP2T
+PHI	I 10	4	2865		0.005		R 2,1	-VTCOSPHI	S8
							R 2,2	-VTSINPHI	A9
-VTCOSPHI				1		0.1	S 8	+VTCOSPHI	S3
+R	I 4								
-PHIDOT	A 9	NLP3T							
+0		NLP3C							
OPEN		NLP3B					M 3	-RPHIDOT	A9
+VMSINTHETA	S 5			1					
-VTSINPHI	R 2,2			1					
-RPHIDOT	M 3			1		1000.	A 9	-PHIDOT	NLP3T,I10
-PHIDOT	A 9			0.1					
-100	REFVOLT	5	5000	IC	1	100.	I 10	+PHI	P4

POTENTIOMETERS	AMPLIFIERS	MULTIPLIERS	RESOLVERS	RECORDERS
5	10	1	2	1

LEGEND:
- An = High Gain Amplifier (n=no.)
- In = Integrator (n=no.)
- Mn = Servo Multiplier (n=no.)
- Sn = Summer or Sign Reverser (n=no.)
- NLPnt = Non-Linear Potentiometer (n=no.,t=tap(T=top,C=center,B=bottom))
- Pn = Potentiometer (n=no.)
- (RCn,c) = Recorder (n=no.,c=channel)
- Rn,c = Resolver (n=no.,c=output channel (1 or 2))

Figure 7
ANATRAN OUTPUT
(Wiring block diagram, list of pot settings, checking diagram and component count. Produced on 704 or other EDPM output printer.)

(Compare with Fig. 6)

MAN-TO-MACHINE COMMUNICATION AND AUTOMATIC CODE TRANSLATION

A. W. Holt and W. J. Turanski*

Summary

The ACT system is a programmed adjunct to a general purpose computing machine (currently considered for the MOBIDIC)whose purpose is to facilitate the initial encoding and subsequent application of specific "code-to-code translation" procedures. The phrase "code-to-code translation" is primarily meant to cover the conversion of problem oriented pseudo-codes into machine code equivalents. The ACT system provides for the "housing" of many distinct translation procedures - carrying algebraic, data-processing, simulation languages, etc., into any of a variety of computer codes - within the bounds of a single controlling system.

The programmed components of the ACT system include:

1. An allocation interpreter - AI
2. A "core library" of basic translation functions
3. A general translation library whose content varies with use of the system.

The allocation interpreter permits the formation of large translation programs out of large (and small) pre-stored sub-programs. Without greatly impeding computation speed, AI determines internal and external storage allocations for all space-taking portions of information as part of problem computation. This facility is shown to be essential to the construction of ACT, and perhaps widely useful in other types of programs.

The core library contains translation functions which play a fundamental role in a wide range of code-to-code translations. Included are many functions which usually play an important part in machine assembly programs. Other functions provide for general library handling, code reordering, etc.

Also associated with ACT is a writing convention called "canonical form" which governs the presentation of data (i.e., programs in source code submitted for translation). Canonical form, while flexible enough to permit highly diverse problem codes, still standardizes all those signals which the basic ACT components decode.

The ideas presented herein were developed at the University of Pennsylvania under contract to the U.S. Army Signal Corps in connection with the FIELDATA family of data-processing equipment. Although reference is made to the FIELDATA family in the presentation below, the ideas should be applicable to other families of equipment. Because of the prevailing tendency towards modular design in computing systems, such families are rapidly becoming more numerous. [1,2,3]

General Considerations

The Problem

The ACT system, is designed to reduce the man-to-machine communication problem in operating the Army FIELDATA system. The fundamental assumptions with regard to this communication problem are:

1. that the FIELDATA system employs a variety of different digital computers;
2. that the computational problems to be solved with these machines belong to many different "problem areas" - e.g., trajectory computation, estimation of statistical parameters, information retrieval, language translation; etc.
3. that the exigencies of time and circumstance may make it necessary to run any of the intended range of problems on almost any of the intended range of machines;
4. that the overwhelming majority of Army personnel concerned with these problems will not be computer programming experts for any (let alone all) of the intended computers; and
5. that the efficiency of the entire FIELDATA system hinges primarily on the speed of problem solution - i.e., the total "lapsed time" from problem statement till solution delivery. Therefore, the speed with which a problem can be correctly encoded for computer solution may matter a good deal more than the

* The work described in this paper was done for the U.S. Army Signal Research and Development Laboratory, Fort Monmouth, New Jersey, as part of a "multiple task" project at the University of Pennsylvania. The aurhors were engaged in this work by sub-contract of Remington Rand Univac, Philadelphia, Pa., to the "Common Programming Language" task of the above project.

efficiency of the machine program with which the computation is finally performed.

Existing Techniques

The man-to-machine communication problem has been widely recognized and seriously attacked in many of its special instances. Most of the present day "automatic programming systems" are the result of manufacturers' attempts to narrow the gap between "the problem", which exists in the minds of technical specialists, and "the program" which "causes" the computer to compute a solution. The invention and construction of an automatic programming system normally involves two major steps:

1. The invention of a special problem code (often referred to as a "pseudo-code") which:
   (a) is compatible with already established habits of expression within some technical specialty, and
   (b) is sufficiently formal and complete to permit automatic conversion - or "translation" - into a computer-coded program.
2. The construction of a special translating routine which accepts as input data problems represented in problem code and produces as output corresponding machine programs.

Most commonly, the translating routine is designed to accomplish its entire task before computer solution of the stated problem begins. Thus, bringing a problem to solution with the help of an automatic programming system is typically a two-stage process - translation followed by computation. (See figure 1).

Automatic programming systems which function as in Figure 1 depend upon three codes: S, V, T. They are usually produced by manufacturers as adjuncts to particular computers.

Thus, "Computer V" and "Computer T" of Figure 1 are almost always one and the same. If, for example, the manufacturer is putting a "scientific machine" on the market, he will usually design an algebraic-mathematical pseudo-code to go with it, and he will prepare the translating routine to produce programs for itself on that computer.

Past experience with the building of automatic programming systems has brought to light several facts which have a strong bearing on the FIELDATA man-to-machine communication problem:

1. To produce an automatic programming system (S,V,T) normally requires major programming effort - measured in tens of man-years.
2. Such systems are usually very difficult to modify in response to changes in the specifications for S, V or T. This difficulty arises because the translating routine, within which the specifications for S, V, and T are "frozen", is usually a large computer program from which it is impossible to 'factor' the components, S,V,T.
3. As a consequence of the immediately preceding point, it is further true that a translating routine which has been programmed for the triple (S_1, V_1, T_1), does not give a substantial head start for producing the routine for (S_1, V_1, T_2) or for (S_2, V_1, T_1).

FIELDATA Requirements

The requirements for FIELDATA computing, as set forth at the beginning of this introduction, make it clear that:

1. The operation of the FIELDATA computing facility will require a large variety of source codes available for easy use in a rich variety of technical specialties.
2. A given source code will have to be translatable into any one of several target codes.

Some more detailed discussion of both points follows.

Problem Code Flexibility

It is a well known fact that each technical field has associated technical "jargon" - special vocabulary as well as special forms of expression - which belongs to it and it alone. This specialization of language resembles the specialization of computers for particular purposes. In general, special computer problem codes for statisticians, physicists, psychologists, business administrators, operations researchers, etc., have not been developed because the art of automatic programming is young and the building of new systems very expensive. If the cost of translating routines were not an issue, it would be desirable to provide each technical field with coding conventions which most nearly suit its established habits of expression.

Quite apart from making provisions for presently fixed habits of expression, the creator of a translating routine must also note that habits of expression change with time and circumstance. As the constellations of problems to be solved undergo change, so do the related language habits. Old words are put to new uses; new words are invented; abbreviations are created to refer to oft-repeated combinations; previously disjoint sets of notations must be combined, etc. Thus, expression forms change in response to changing needs. In many important cases the expressions lead to computable problems. The speed with which answers appear depends heavily on the amount and kind of hand "recoding" (or "encoding") of the original problem statement before the computer can "take over" and ultimately execute the proper sequence of computer steps to achieve a problem solution.

Computers - Variety and Compatability

No matter how similar two computers may seem, they always turn out to be distressingly different in relation to some problems. Consider, for example, two computers, C and 2C, which differ from each other in no respects other than that 2C has twice the internal storage capacity of C. For all those problems easily accomplished

by a single run on C, Computers C and 2C are fully equivalent. For those problems which involve more than one run on C - or perhaps the use of overlays - Computer 2C is equivalent to Computer C in only the sense that those originally programmed for C will also run on 2C. Thus, for such problems, a major task of conversion from one computer to another may be encountered if

1. the problem was originally programmed for 2C, or
2. the problem was originally programmed for C but must be made to run efficiently on 2C.

So-called "compatibility" between machines (in the sense of very similar or identical instruction codes) does not guarantee ease of program conversion from one computer to another. Only small problems will convert readily, while big problems will offer difficulties not much less formidable than when "incompatible" machines are considered. Although the FIELDATA computers are being designed with mutual compatibility in mind, significant differences between one translating routine (S, V, T_1) and another (S, V, T_2) must be expected.

The need for many source codes and the inclusion of many machines (and many configurations for each machine) in the FIELDATA system makes the normal automatic programming approach economically unfeasible. To go from s source codes to m machines patently requires s.m translation routines - a prohibitively large number when the effort for each such translating routine is considered. Even worse, if one had gone to the labor of constructing these s.m translating routines, one would still be faced with an unmanageable chore in keeping them up-to-date in response to changing requirements.*

Code-to-Code Translation

The study of many problem oriented source codes and their translation into computer code reveals that various translation processes have many important common elements. Often commonness between translation processes can be found by examining their respective constituent parts. For example the translation of S_1 to T_1 - briefly (S_1, T_1) - and the translation (S_2, T_2) may both employ the same assembly system for the terminal portion of the translation. Sorting is a frequent common sub-process; at a finer level of analysis, table look-up, list forming, subdividing a "sentence" into "words", economizing storage locations, etc., have frequent occurrence.

Another kind of commonness frequently found between two translation processes is the employment of a common control sequence with differences in regard to what is controlled. This is illustrated by generator systems (such as FLOWMATIC) in which the individual generators are properly regarded as sub-translators subject to an over-all controlling scheme. Two quite different pseudo-codes might be handled by the same controlling scheme with differences only in the library of generators being controlled.

These examples indicate that code-to-code translation is coherent area of computation for which programming aids may be prepared. These programming aids (whose nature and form will be discussed below) will:

1. Reduce by large factors the effort required to produce an operating program for the translation (S, V, T).
2. Similarly reduce the effort required to modify existing translation programs in response to changed specification for S, V, or T.

The ACT system is designed to be a programming aid such as discussed above. Once constructed, it should have a far reaching effect on the manner and effectiveness of FIELDATA utilization. In working on some particular class of problems - or even just one large problem - programmers will find it convenient to refine their tools of program expression as they go - by modifying or adding to some previously available set of conventions. Thus code or "language" development will go hand in hand with problem solution, just as is normally the case in technical or scientific work without computers. Furthermore the ACT system should greatly facilitate the re-translation of a particular source code, originally translated to one machine but now required for another.

The Act System

The ACT System is a collection of programmed devices which loosely fit together to help in the implementation of specific translations.**

Also associated with the ACT System are two sets of writing conventions; one set governing the presentation of translation procedures - to be called "ACT translation procedures" - and the other to govern the presentation of data (i.e., source-coded programs submitted for translation.) This is analogous to computer systems, for which data and programming conventions must always be specified. A complete description of ACT will therefore include:

* This is almost exactly the problem to which a proposal named UNCOL (sponsored by SHARE and described in ACM Communications, Vol. I Nos. 8, 9) is addressed. In our opinion the UNCOL proposal, while aimed in a valuable direction, is practically unworkable. For more details, see section on General Translation Library below, also a letter from George H. Mealy, Switching Systems Dept., Bell Tel. Lab., Inc., to the members of the UNCOL committee, dated Feb. 3, 1959.

** Some of these components have importance and usefulness above and beyond their function within ACT. This is exactly comparable to parts of a computer (such as memory units or tape transports) which can be integrated into many different systems.

1. A description of major components.
2. Writing Conventions governing ACT translation procedures.
3. Writing Conventions governing programs submitted for translation.

The programmed components of the ACT system include:

1. An allocation interpreter - AI
2. A "core library" of basic translation functions
3. A general translation library whose content varies with use of the system.

Allocation Interpreter

The allocation interpreter permits the formation of large translation programs out of large (and small) pre-stored sub-programs. Unlike assembly programs, it accomplishes its function dynamically - i.e., during the actual running of translation programs. This mode of operation is dictated by two important characteristics of translation programs: (a) frequent use of data dependent storages (b) the possibility of analyzing such programs into parts some of which are large sub-programs (unlike mathematical subroutines).

Although AI participates at run-time, it does not cause any substantial reduction in computation speed. Almost all program instructions are performed by the actual computer without interpretive interference. Only at infrequent intervals in program running time, when there is a change in requirements for internal or external storage space, is there a jump out of AI for bookkeeping and allocation activities, followed by a return of control to the running program. in this respect AI behaves in a manner analogous to that of contingency routines.

The special jump-return instructions which interrupt the main computation for AI intervention might be conveniently viewed as new macro-instructions, added to the previously available stock of computer commands. These macro-instructions are entirely concerned with space allocation, and programs written with these instructions in mind have new formal properties. Correspondingly, a computer-plus-AI on which such programs are intended to run may be conveniently regarded as an "Extended Machine" - EM. Figure 2 is based on this conception. The central cross represents the extension.

Associated with EM is EM-code in which programs for EM must be stored. This code differs from ordinary machine code in several respects.

1. The program is formally subdivided into program "phases", each phase headed by a phase description. The phase description lists "sequences" of "segments" of information belonging to the phase and requiring space (internal or external) when the phase operates.
2. The phases of a program relate to each other through special calls (interpreted by AI). When a call for a new phase is encountered, the calling phase is suspended and the called phase initiated. When the called phase terminates it may either be held for re-use or dropped, depending on the manner of call.
3. Aside from special AI macro instructions having to do with space allocation (including new phase call) EM code contains ordinary computer instructions with sequence-and-segment-relative addresses. These addresses are converted by a program loader into fixed form when the phase is first encountered during program operation. If a phase is held for re-use and then re-activated, it is returned to memory to the same set of locations to which it was originally assigned. Between uses of the same phase AI will have taken care not to assign any newly developed information which this phase requires to locations causing conflict with old information already belonging to the phase.

Although phases of EM-coded programs will normally be functional units (corresponding to a box on some flow-chart description), the converse need not be expected. The only use which is made by AI of the knowledge that a program is subdivided into phases with particular interconnections is to temporarily dump some information in favor of other information which is now required and which could not otherwise be accomodated. Therefore functional units which, in their coded form, occupy very little space in the various computer media are not worth treating as separate phases. The bookkeeping involved in their handling is not worth the space return which they can bring.

The power of EM may be said to derive from the following fact. In the case of EM programs the total space requirements of the program (for external and internal space) have been broken down into space requirements for individual program phases (where different phases may overlap by requiring the same information). At any one moment in computation time only one phase is operating and hence only its space requirements must be satisfied. Any parts of other phases which do not overlap with the active phase can be dumped and later restored if necessary. But the greater the amount of dumping and later restoring the slower the program operates. If, therefore, there is enough memory space to permit the assignment of phases side-by-side the program will run correspondingly faster. In this way EM is able automatically to trade time for space depending on the total available storage of the computer and the sizes of various data lists which may have grown in course of computation.

The Core Library

The core library consists of a collection of sub-programs - called translation procedures - which have near-universal importance in translating pseudocodes into machine codes. (Many of these functions derive from a single fact about the relation of source to target: namely that the information in the source code is packaged by time of generation, while in the target code

it must be packaged by time of use).

The writing conventions - called "Canonical Form", abbreviated "CF" - governing source-coded programs submitted for translation, are entirely related to the core library. All of the signals for which CF provides are decoded by one or several procedures in the core library. Conversely, all those translation procedures which only deal with signals specified by CF belong in the core library. The following is a partial list of functions fulfilled by the core library procedures.

Coding Item Handling - Each CF item of pseudo-code (such as a FORTRAN formula, an assembly instruction, a FLOWMATIC sentence, etc.) has a formal structure which provides for standard item features such as: (a) an item identification number (b) a symbolic item name (c) item type indications. In addition, after some preliminary editing, the body of the item may be recognized to consist of a sequence of symbols. The core library provides CF item handling functions which allow specific translation processes to gain access to constituent portions of items by means of "item dissecting" commands. Similarly, if a translation procedure produces CF coding items as output, they may be assembled from their constituents by use of "item assembling" commands.

Patch Interpretation - CF coding items may be generated (by programmer or generating routine) in some order wholly different from the order in which they will be next required. Item rearrangement may be coded into the original item sequence by use of the item identification numbers as well as by use of special "patch items". Core library functions will cause item re-ordering, thus interpreting the "patch" intentions expressed in the original item sequence.

Symbol Interpretation - By "symbol" is meant the smallest meaningful constituent of a coding item. Thus, the computer instruction 'CLA B 542' is composed of three symbols. Symbols may be of various types from the point of view of what interpretation they require during translation. Some symbols, for example, are absolute and require no interpretation; other symbols are 'dummy' and require replacement by specified expressions; yet other symbols refer to named coding items - i.e., are "addresses" - and will require address interpretation. The core library contains a variety of procedures to effect symbol interpretation.

Type Interpretation - CF coding items may contain marks to indicate their specific type. Thus, for example, a FORTRAN DO-statement and a dimension statement might be distinguished as to type; or a statement to be interpreted by FORTRAN as opposed to a statement to be interpreted by SAP may be distinguished by type. In general, type indications always mean that the items thus categorized, are to be respectively delivered to different translation procedures for treatment. Core procedures are used to deliver CF coded items to appropriate translating routines in proper priority sequence, according to type indication.

Call Interpretation - One of the frequent mechanisms employed in code-to-code translation is the adjoining of one or more batches of coding items, taken from a library of such "batches", to the coding items which already belong to the source coded routine being interpreted. A good example of this is the selection of subroutines from a library. After the subroutines have been picked up and (with some adjustments) adjoined to the routine undergoing translation, this latter routine has been 'expanded' by the inclusion of new coding items. Special items in the source coded routine, called "call items" tell what particular batches are to be selected from the library for inclusion.

In the case of subroutine selection, call interpretation may be carried out through a hierarchy of levels. The subroutines selected for adjoining to the calling routine may themselves contain further call items requiring (for their interpretation) the selection of yet other library routines for inclusion. Thus the original source coded routine contains "first level" calls; the subroutines thus selected may contain "second level" calls, and so forth, for nth level calls. [4]

Call interpretation may be usefully applied to a variety of cases which have nothing to do with subroutine selection. For example, a routine coded in FLOWMATIC might be said to "call" on file descriptions from a library. The interpretation of such calls has much in common with subroutine selection. To go one step further: if there were a number of different files which all involved the same item description, then there might even be "second level" calls - the routines calling on file descriptions, and the file descriptions calling on item descriptions. Core procedures are used to carry out hierarchic fetching and adjusting of library information.

The General Translation Library

The contents of the general translation library are governed by the particular source codes and target codes for which translation procedures are desired. When stored in the translation library, these procedures are represented in EM code - hence contain special macro-instructions having to do with allocation. They also contain macro-instructions referring to translation procedures stored in the core library. In general, the time at which various core functions are brought into play in course of a translation is dependent upon features of source and target codes beyond those which are governed by canonical form. Hence the highest level control of a translation procedure always resides in a program belonging to the general translation library. With ACT, as presently conceived, there must be a distinct high-level controlling program for each source-to-target pair (S, T) desired. Ofcourse these controlling programs will, in general, be very short, the bulk of the work being encoded in sub-programs

belonging to general or core libraries.*

One class of translation procedures expected in the library deserve special comment. In course of many code-to-code translations one reaches a penultimate code form which may be roughly described as "assembly code" and a final step in translation called "assembly". There is considerable variation between present-day translation systems (such as FORTRAN, FLOWMATIC, IT, etc.) in regard to the complexity of this final assembly phase. In the FLOWMATIC compiler for example, only the most rudimentary assembly phase, consisting of nothing more than relative address interpretation is employed,[5] while FORTRAN translation produces SAP assembly code with symbolic addressing, library reference ability, and many other features.

In the development of ACT a special analysis was undertaken of the terminal translation steps to determine whether a widely applicable 'end game' could be prepared for inclusion in many different translation procedures. This analysis resulted in a set of specifications for assembly - system and code - which includes many features normally included in such systems but also some noteworthy additions. (Description of these lies beyond the scope of this paper.) Assembly procedures conforming to these specifications make extensive use of the core functions previously described as well as some additional ones. Since, however, such procedures always depend, in part, on the target computer, they are themselves represented in the general translation library.

The success of the ACT system conception hinges critically on the feasibility of general translation library building, and we will now try to show that past experience with code-to-code translation justifies the expectations on which the over-all design of ACT is based.

It is well known that, in normal code translation procedures, the source code is made to pass through some number of intermediate forms until, through final processing it becomes target code.

(1) $C_S \rightarrow C_{I1} \rightarrow C_{I2} \rightarrow C_{I3}$ ……$C_{A1} \rightarrow C_T$

In (1) the subscripts, I1, I2, etc., are meant to suggest successive Intermediate forms, while A1 suggests Assembly form (A2, A3, etc., being possible intermediate forms through which the code passes during assembly). In FORTRAN translation, for example, the initial formula statements pass through several intermediate forms until they reach "triple" form;[6] they are then translated further into SAP assembly code, etc. Generally speaking, as the code progresses from form to form, it becomes, step-wise, less and less conditioned by the specifications of the source code. Thus, for example, in FLOWMATIC, by the time the code has reached Op. file (1) form, it is largely independent of the detailed rules for FLOWMATIC statement writing. One could substitute many new conventions for FLOWMATIC sentence presentation, and correspondingly only alter the action of the so-called "glossaries", which (together with some general controlling code) determine the first intermediate code form. To put this another way: many details of FLOWMATIC code form are interpreted and done with in the first stages $C_S \rightarrow C_{I1}$. All such details could therefore be altered without requiring any revision in the compiler, save a change in the early phases.

Exactly parallel observations apply to the terminal stages of code translation with regard to the details of the target code. Again choosing FLOWMATIC as an example: one can alter the target computer (hence the target code) with little or no effect on the early stages of translation which are source-code determined.**

Figure 3 shows a rough break-down of code-translation into stages which are successively more target-code determined and less source-code determined.*** The division into gross stages as in Figure 3 is neither unique nor optimal. Its sole purpose is to show how the ACT translation library may be used to good advantage. Suppose for example that a certain translation (S_i, T_i) is coded with exactly six intermediate stages, shown in Figure 3. The over all ACT program would read:

(2) Do stage 1
Do stage 2
⋮
Do stage 6

This program would be stored in the ACT library together with all six sub-programs which are called upon. Now suppose that one wishes to produce a translation for (S_j, T_i) where S_j differs from S_i only in its rules for operation and operand presentation (such as parenthesis notation for algebra replaced by Polish notation; or the rules for file and item descriptions changed from "explicit" to "pictorial" form, etc.). The new translation may be obtained by writing a new controlling program

(3) Do stage 1'
Do stage 2
⋮
Do stage 6

* It is projected that a special component of ACT can be prepared, to be called the ACT "translation control" program. Highest level control of translation would then always be vested in this program, and the total number of individual translation procedures required to handle a large class of (S,T) would be greatly reduced. Further study is required to test the feasibility of this.

** This observation is the basis for the so-called AIMACO system which translates FLOWMATIC code into 1103 target code.

*** The fact that some computer effects enter the translation procedure almost from the beginning is what makes the UNCOL proposal - namely to accomplish all translations in two stages, the first source determined and the second target determined - unfeasible.

and also coding a new sub-program corresponding to 1'. If, on the other hand, one wishes to produce a translation for (S_i, T_j), one will have to consider how radically T_j differs from T_j . The more radical the difference, the greater the number of intermediate stages which must be replaced.

Consider next, the problem of producing a translation for (S_k, T_i) where S_k is a source code which allows parts to be written in S_i and other parts in S_j with "type" labels to show which sections are written in accordance with which conventions. Suppose further that both translations (S_i, T_i) as well as (S_j, T_i) were already available, and both coded in six stages as in Figure 3. The new control program would cause some number of initial stages for S_i to be performed, and also some number of initial stages for S_j . As soon as both streams of coding have achieved a common form, they are, from there on in, submitted to the same terminal stages. How many stages would be involved before the two streams converge is a function of how profoundly the two codes, S_i and S_j, differ in their features as analyzed in Figure 3.

These greatly over-simplified examples are meant to show that, because of the partial "factoring" of source from target effects which one may realize in code-to-code translations, one will have ample opportunity for the re-use of major translation sub-programs to be stored in the ACT translation library. As already discussed previously, the extended machine base seems indispensible for the combining of such major sub-portions into entire operative programs.

Final Summary

The primary aim of the ACT system is to reduce to a minimum the labor required to implement a given translation (S_i,V_j, T_i). The reduction of labor is achieved through:

1. Storing translational functions of almost universal applicability in the core library and providing for their incorporation in new translation programs.
2. Making it possible for the translation library to grow in such a way that new programs (S_j, T_j) may use, as sub-programs, previously coded procedures already stored in the translation library.

Regarding item (2) above, its realization is entirely dependent on EM as the fundamental base on which ACT is built. The effictiveness of item (1), on the other hand, is dependent on the analysis of coding functions which leads to the definition of canonical form.

(Many of the representational facilities which CF provides will not be directly employed in programmer-written source codes, but will appear in intermediate code-forms which are the result of early editing and other translations.)

Figure 4, is a schematic summary of the three main portions of ACT. Each box in the figure is horizontally divided into three sections. The middle section tells by what functional requirements the object is determined; the lower section tells by what coding forms the object is determined. ('EM-code' refers to the extended machine. In regard to the triplet (S,V,T) the machine used - extended or unextended - is V. Because of the nature of EM, the ACT system can be transferred from one V to another without recoding if the two machines differ only in configuration but not in order structure.)

Bibliography

1. A. W. Holt and W. J. Turanski, Final Report of Common Programming Language Task, Part II, "Automatic Code Translation System", USASRDL, Fort Monmouth, N.J., Report no. AD59URl, July 31, 1959

2. A. W. Holt and W. J. Turanski, First Quarterly Report of Common Programming Language Task, Part II, January 15, 1960

3. A. W. Holt and W. J. Turanski, Second Quarterly Report of Common Programming Language Task, Part II, March 15, 1960

4. "Univac Generalized Programming", Remington Rand Univac, RRU 21, 1957, New York, N.Y.

5. "X-1 Assembly System", ADP, Remington Rand Univac, Philadelphia, Pa.

6. Peter Sheridan, "The FORTRAN Symbolic Translator", ACM Communications, Vol. II, No. 2, p. 9.

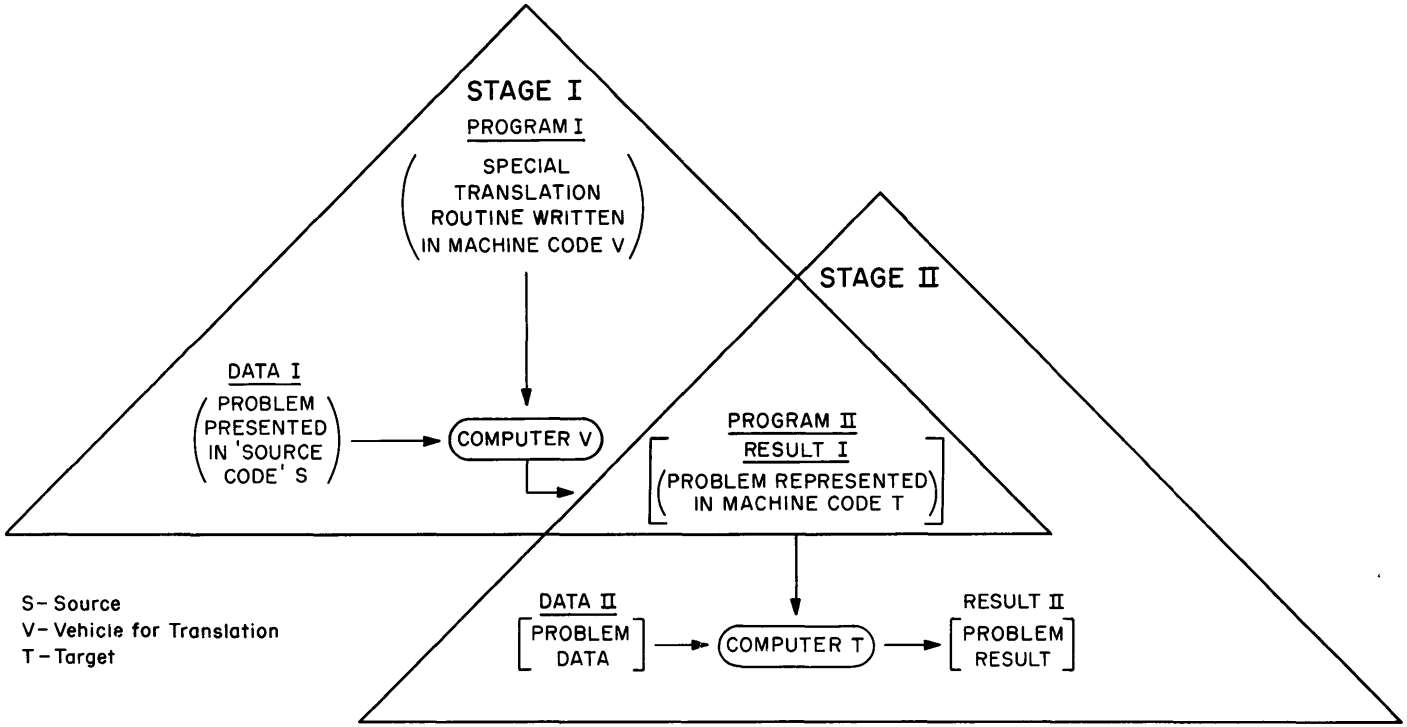


Figure 1
TRANSLATION AND COMPUTATION

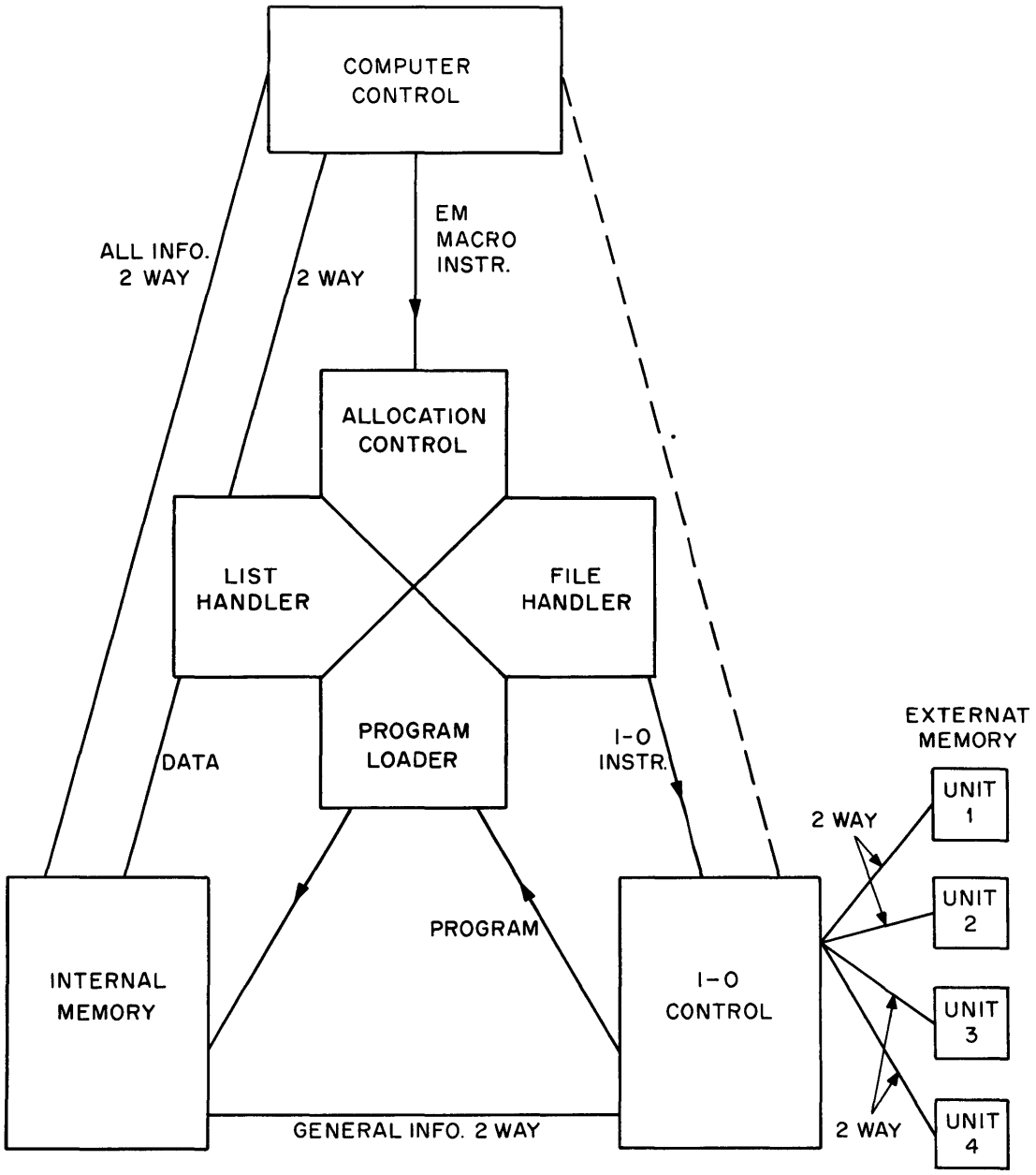


Figure 2
EXTENDED MACHINE

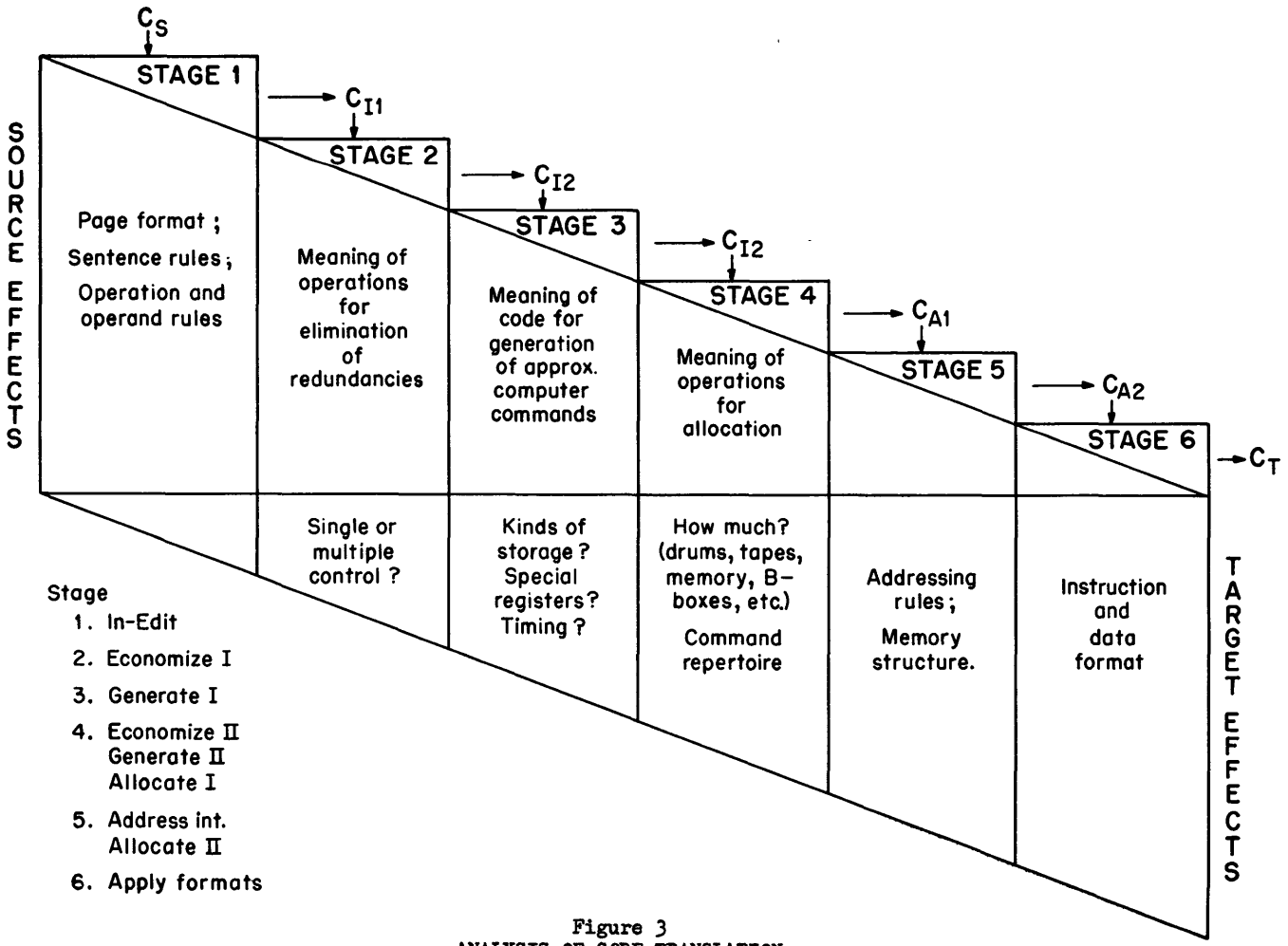


Figure 3
ANALYSIS OF CODE TRANSLATION

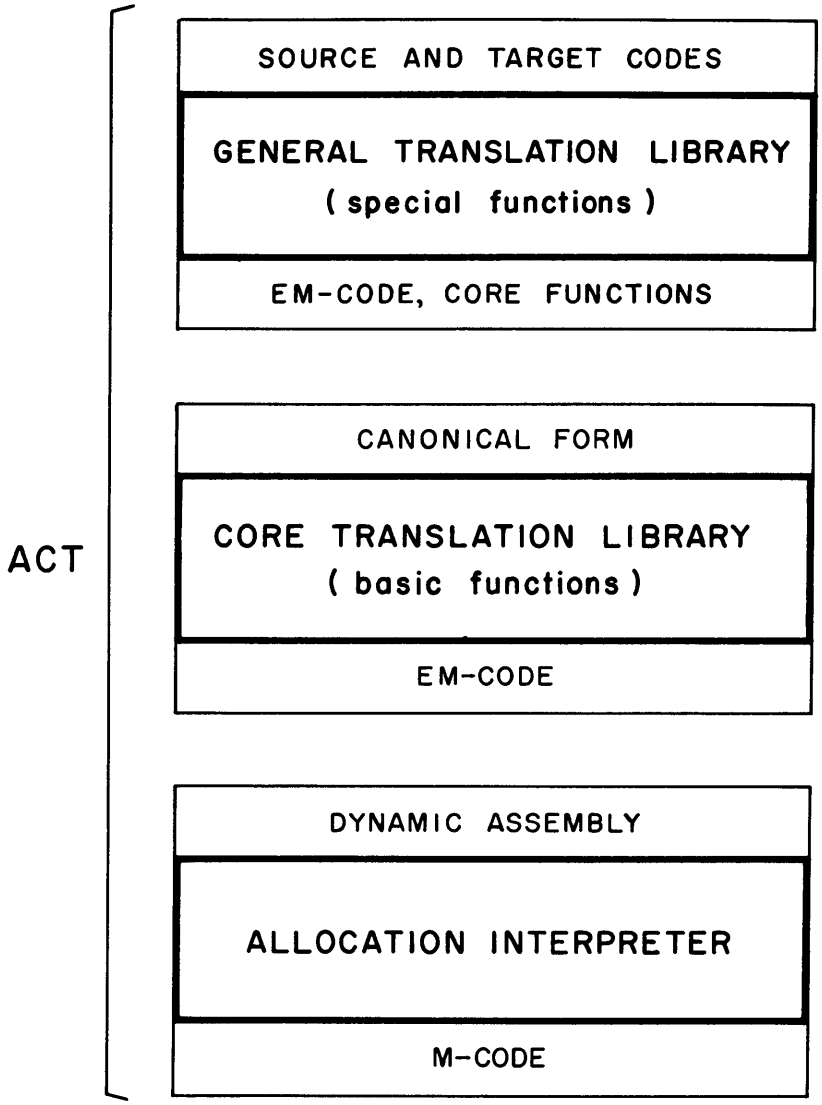


Figure 4
THE ACT SYSTEM

THE COMPUTER OPERATION LANGUAGE

George F. Ryckman
General Motors Research Laboratories
Warren, Michigan

Summary

The many machine oriented and problem oriented languages existing and being planned today give rise to more complex problems in operating a computer. The development of computer operating systems have materially aided the problem of getting a program or series of programs on and off the computer efficiently. The variety of language processors have created a new operating system function - that of interpreting a language describing the processing to be performed for a given job. In particular the language is used to call a given processor, perscribe pertinent options and other control information. As such it is termed a computer operation language or operator oriented language. Not only does this language serve to free the human operator of many clerical tasks but it also provides for more consistent and error-free operation of the computer.

Introduction

In recent years the computing profession has recognized two classes of computer languages. The first of these is termed the machine oriented language. Characteristic of this language is that it is tied quite closely to the hardware itself. That is, coding in this language is either directly interpretable by the computer or an assembly process produces a language which is directly interpretable by the hardware. The assembly process then is working on a one-to-one translation basis. Examples of machine oriented languages are basic coding, relative coding, and probably the most used, symbolic coding.

A second computer language is called a problem oriented language. It is characterized by having few ties to the computer hardware. Rather it is aimed at problems in specific areas of mathematics, engineering, physics, accounting, and so on. Characteristic also of the problem oriented language is the compiler or generator which examines the source statements and generates strings of basic machine code. The compiler is not only doing all of the basic machine coding, but in many cases it is also doing some of the analysis and even furnishing numerical techniques. One of the more important differences between the machine oriented language and the problem oriented language is that it requires fewer pencil marks to state a problem in the problem oriented language. This in turn means less effort, less time, and less cost involved in preparing a problem in a problem oriented language. Still there is a need for the machine oriented language for those problems which are not conveniently expressed in any of the presently prepared problem oriented languages. Examples of problem oriented languages are Flowmatic, Fortran, and the data processing generators. So much for the machine oriented language and problem oriented language.

The Computer Operation Language

I should like to introduce a third class of computer language. It is one which is known but probably not as universally recognized as the first two. We call this the computer operation language. Like the problem oriented language, the computer operation language has few ties to the actual hardware. Unlike the problem oriented language, it is aimed at informing the computer of the major steps to be taken in processing a job. The information supplied to the computer is much like the operator instructions written for the machine operator. Relatively unimportant is the distinguishing characteristic that the computer operation language is processed by an interpreter. This of course is not essential, and, in fact, the language could just as well be processed by a compiler.

Turning our attention for the moment to some history in the development of the computer operation language, we began experiments in this area in the summer of 1957. At that time we were processing something over one hundred jobs per day on our computer with the aid of an operating system. Our only coding system at the time was a symbolic assembly program. Anticipating a wide use of Fortran which had just become available, we began the design and checkout of an operating system for the Fortran compiler. One of our first requirements was that the programmer have the ability to compile a program in Fortran and then immediately execute it all on one machine pass. We also wanted him to be able to easily control these options. As a result we made our first attempts at a computer operation language. The Fortran operating system had an interpreter built into it which recognized such simple terms as compile, execute, and others. As more features became available in Fortran, their terms were added to the interpreter's dictionary or vocabulary. Fewer than twenty-five verbs and nouns comprise the vocabulary today. However, there is unlimited expansion room.

Environment of the Computer Operation Language

Recognition of the computer operation language is significantly important in the evolution of computer languages. The user has a powerful command of the computer. In our experience the language has been enthusiastically accepted by the programmer. First, because it is easy to use and second, because it gives him consistent results in the processing of his problem on the computer. From the standpoint of the computer operating system this language is an input, informing the system of the actions to be taken for each job. A few words about operating systems is in order at this point. Charged with the major responsibility of operating the computer continuously and smoothly, the operating system shows up most favorably when there are a large number of jobs to process in a day. The author recognizes that operating systems were originally justified on the basis of such a machine load. It takes effort in terms of man-hours to devise these systems, to expand them and to maintain them. On the other hand, our payoff was great enough in terms of savings both on the part of the user and also in more efficient machine operation to justify the expenditure. Some users of present-day computers may find they are unwarranted in developing an operating system on the basis that they are running a relatively few problems. It should be pointed out that long-running problems on today's computers will be short-running on the next generation of computers. But more important are the benefits accorded the user through other functions of the operating system, e.g., more powerful and consistent control of the computer via the computer operation language.

Developments in Planning

I should like to explore what might be the future of the computer operation language. At the time of this writing we had just completed the incorporation of a new coding system in our operating system. The Surge compiler is a problem oriented language processor which is designed to aid in the preparation of data processing programs. It will sort a file of information, update it, and prepare reports from a file. In addition to adding it to our master tape, we have also increased the vocabulary of our computer operation language. The computer now understands a call for Surge along with execution of the resulting program if so desired. Many of the options available under Fortran processing are also available under Surge processing, and accordingly are understood by the computer. The effort of incorporating Surge amounted to approximately one man-month - an insignificant part of which was the expansion of the computer operation language vocabulary. It was not surprising to learn again, as we had with Fortran, that the Surge coding system was considerably enhanced just by virtue of the fact that it was more readily usable. The user prepares the Surge statements as well as the processing statements which are all incorporated in one deck of cards and sent to the computer room for complete processing.

It would be well at this time to bring out another important advantage of the language in that it has the inherent ability to merge parts of a program which have been coded in different languages. As an example, a program which had been partly coded in symbolic language, partly in Fortran, and partly in Surge could be quite conveniently combined, loaded into memory, and executed as a single job. The merging can be done at any level at which the languages are in fact compatible, such as at the symbolic language level. This advantage of the language should not be underestimated.

In the near future we intend to add still another coding system to the package. The Dyana coding system, which was reported at an earlier Joint Computer Conference, will be added with the necessary extensions to the vocabulary. In this case the words: DYANA, EXECUTE, will trigger the following actions on the part of the computer. First the DYANA processor will be retrieved from the system master tape, and it will commence processing the DYANA source statements and as a result will produce Fortran statements. Given the Fortran statements, the system then understands that it is to bring in the Fortran processor; and as a result of this operation, the object program will be produced. In response to the word EXECUTE then, the system will load the resulting object program into memory and transfer control to it. The DYANA object program will get its input from the same source which contained the DYANA source statements, and it records its outputs on the appropriate media. At the conclusion of execution, the DYANA object program releases control to the system to bring in the next job. Thus through the computer operation language the programmer has controlled a sequence of processes which do not require any human intervention.

Potential of the Language

Given a large capacity memory device with reasonably good access, it will also be possible to store a number of production programs in the computer. Through the use of the computer operation language, the programmer may then call any one of these production programs, supply it with appropriate data, and produce the necessary results.

Future developments of the computer operation language may well include some conditional statements, boolean expressions, declarative statements, and others which are

presently found in problem oriented languages. The macro concept can be used to good advantage in specifying a sequence of processes. In one of my earlier examples, I mentioned a call DYANA. This was interpreted by the computer to mean call the DYANA processor and when it has completed its compilation. call the Fortran processor. In a very elementary sense then, the command DYANA is a macro.

Figure 1 shows the potential power which might be built into the computer operation language. The first statement is a simple call for a processor Dyana. Next the programmer has specified the action to be taken if a machine error occurred on the first try at a Dyana processing. He wants to go back and recall Dyana. The second statement calls a report generator. Here again, if a machine error occurs on the first try he wants to go back and recall the report generator. Assuming these two processes are successfully completed, he then wishes to set a debugging mode followed by an execution of the object programs resulting from the Dyana and report generator processors. During the execution of this program, if something causes an interrupt, he wants tapes A and B to be saved. If on the other hand an error occurs, then he would like to get a dump of all core variables and also tape A. Following the completion of all processing he wants to exit to the next job. At the bottom of the illustration is a macro which will perform the same functions as all of the statements 1 through 5. In this case a simple CALL ABLE.

One of the points which becomes quite clear in this discussion is the necessity for processors such as SAP, FORTRAN, SURGE, etc., to behave as subroutines to the operating system. They must be subroutines in the sense that they are informed by the operating system of the machine configuration available to them for each execution. As an example a processor must be told where it will find its input(s), where to record its outputs, what primary and secondary storages are assigned to it and even where the processor itself is stored. This information may be thought of as parameters in a calling sequence at the time the operating system calls the processor. The operating system is of course aware of the needs of a given processor and proceeds to call that processor if the required portion of the machine configuration is available. Alternate sets of needs should be allowed which will allow the processor to operate more or less efficiently on the computer.

The major reason for subroutinizing processors is solely to allow compatible operation of more than one processor on the computer. Two or more processors which make a priori assumptions concerning input, output or other storage use probably preclude compatible operation. A simple case is that of two processors both of which assume they theyselves occupy the first position of a given tape. The programmer effort required to eliminate these incompatibilities is not inconsiderable. It should be noted that some large processors have already been written as subroutines such as the SHARE Operating System for the 709 computer. A second reason for subroutinizing processors is to allow a program to be written in more than one language, as stated earlier. The saving of each segment of the program as processed is another function of the operating system as directed by computer operation language statements. In summarizing this point processors must be written as subroutines before the full potential of the computer operation language can be readily realized.

Conclusion

Although the computer operation language has been used in many of today's computer systems, it has never been extracted and recognized as being a powerful concept. Once it has been recognized, it will be much more fully exploited. The SHARE organization's General Standards Committee is currently considering the impact of the computer operation language on future computer systems. The language is a key to the sound and successful operation of the computer from remote stations. It will one day take its place with the machine oriented language and the problem oriented language.

A Program Written in a Computer Operation Language

Statement Number	Statement
1	CALL DYANA IF (ERROR FIRST TRY), GO TO 1
2	CALL REPORT GEN IF (ERROR FIRST TRY), GO TO 2
	SET DEBUG MODE EXECUTE IF INTERRUPT, GO TO 3 IF ERROR, GO TO 4 GO TO 5
3	SAVE TAPES A, B GO TO 5
4	DUMP CORE VARIABLES TAPE A
5	EXIT TO NEXT JOB

MACRO FOR STATEMENTS 1-5

CALL ABLE

Figure 1

A New Approach to the Programming Problem

William Orchard-Hays
C-E-I-R, Inc.
Arlington, Virginia.

The computing and data processing field as we know it today, that is, employing high-speed EDP equipment, although but a few years old, has grown accustomed to major advances, changes and reworkings of systems in quick succession. However, the increasing complexity of programming and of applications, together with the rapid appearance of several powerful hardware systems, has now reached such proportions as to cause considerable uneasiness and confusion. It appears sometimes that the effective lead time for preparing for a new machine has not only gone to zero but become negative. As to problems, it has become a cliche' that they are always wanted "yesterday".

It would be hard to deny that the programming profession has brought some of this on itself although the total problem is much broader than what is usually considered programming. It appears sometimes that the equipment users have not kept pace with the equipment makers in imagination, sensible distribution of effort and just plain pride of workmanship. Yet, a realistic evaluation of the work done on large machines in the past three or four years will elicit admiration from anyone capable of understanding it. But mingled with the understanding admiration is likely to be a trace of fear that man cannot yet effectively utilize the powerful machines which he is building. Some of the most ambitious, thoroughly planned and well funded projects have been little short of fiascoes. And machines five times as powerful are on the threshold, with other machines twenty times as powerful as that following.

A brief look backward is sometimes helpful. In earlier days of machine computing, say as far back as 1952-3, there was a keen interest in getting jobs done and a lively anticipation of doing these jobs better on the equipment soon to be available. The dark ages of the 602 and 604 had given way to the transitional CPC with occasional excursions on the EDVAC, SWAC or UNIVAC. People were steeped in the care and precision of EAM procedures, still believed in the doctrine of economy of space and time, and still accepted a challenge to do some operation in a better way. Machines were looked on as tools, and it was assumed that people would be required to make effective use of them -- people who understood them. It was also accepted that computing departments existed to serve some higher purpose and that it was the function of the programmer or machine specialist to provide means to make machines useful to those with problems.

It would be hard to say that these notions have ever been deliberately scrapped but, somehow, they no longer seem to describe current attitudes. Perhaps it is just that the problems have gotten harder faster than we have gotten smarter. We can hardly be accused of idleness. Programs have proliferated in ever increasing sizes until today one person can scarcely be aware of all the routines available for one computer, let alone understand their use. At the same time, the number of new jobs presented which require original programming has grown at least as fast. This in itself, of course, is encouraging; we do not wish to work ourselves out of a job. But neither, I think, do we wish to attempt to re-define every field of technology. I am aware that much lip service, and much real effort, is devoted to providing better programming aids and "automatic" systems, presumably for those who have a need to exploit machines but who have neither time nor inclination to become professional programmers. Few would claim an outstanding success here. Perhaps the trouble is that we are attempting to bridge an ever-widening gap with the same double step that was effective in the days of the CPC, namely the general programmer and the system programmer.

The standard solution to the problem of "getting on the air" with a big project has been to proliferate ever larger programming groups - by hiring, training, renting, pirating or any other means including dumping a bunch of people in to sink or swim. This method is becoming increasingly ineffective. A large number of people doing a mediocre job only adds to management's problems. This is not to say that the people are mediocre. Rather they often lack direction.

However, that there is a definite limit to the number of potentially top-flight professional programmers - dedicated, intensely interested, highly skilled and capable of creative work - is apparent to anyone in the industry. This limit is indeed small, I would guess in the hundreds.

This problem has not gone unnoticed, of course. Besides a number of elaborate processors and operating systems, there has been talk of a universal problem-oriented language, of a universal machine-oriented language, and even of some kind of a universal language for both purposes. People evidently take these seriously since there are many active committees in existence. But just what meaning do such phrases have?

Let us look at the problem-oriented language first. Before we can begin, we must assume that it is somehow possible to foresee every kind of term that will ever be used and every kind of action that will ever be taken in defining and solving a problem. We can, you may say, assume synonyms and isomorphisms so that it is not necessary to pre-define every actual combination of characters and symbols. But this assumption implies a recognition and/or memory capability which far exceeds anything we now know outside of the human brain. Moreover, there is the deeper implication that all classes of quantities and operations are now known. To say that the operation of defining an operation will be permitted is only evading the issue since this implies that we know how many innovated operations will interact with all existing operations. Given any defined language, it is not very difficult to show that there are reasonable problems which may require instructions, i.e. notions, outside the language. Hence no problem language can be universal.

Similar arguments can be made regarding a machine-oriented language. Consider a simple example. One of the tasks to which such a language must be adequate is the translation of any program to a form usable on a machine and, at the same time, responsive to the needs of the problem formulator. Suppose this person is studying improved methods of handling his problems and wishes to determine how much information is required in translating a program. Then he must be able to write a program which outputs information about every action of the processor. Hence the processor must accept modifications to and interruptions of itself. But then it is no longer merely the processor of the original language. Hence he must be using a more extensive language.

There are other and more powerful arguments against the feasibility of a universal machine-oriented language. This is betrayed by the fact that such supposed languages have a bad habit of turning into rather specific problem-oriented languages where the problem is how to describe a machine, whereas what is desired is the ability to use a machine. Yet the notion of a universal machine-oriented language system does have meaning, it seems to me, in a somewhat different sense.

In order to develop this argument, it is necessary to consider a few basic principles. I will restrict myself to four, all of which have to do with the dynamic nature of computing.

1. The introduction of new ideas and new applications is an inherent part of EDP technology, both in the manufacture and in the use of equipment. Some of these new ideas and applications have unanticipated but far-reaching logical implications.

It seems to me that a failure to recognize this principle has caused us to get farther and farther behind. We are forever coming up with excellent solutions to problems which we muddled through a year previously, while at the same time trying to cope with current problems which have new requirements. Furthermore, we should certainly be aware by now that every major procedure, when perfected, is looked on by the analyst as an available technique or operation and that he will seek new degrees of complexity in his next project. No fruitful line of development is ever complete and very few computer program systems will remain static if they are found useful. This is verified by many years of consistent experience in nearly all installations. Compatibility is a greatly overrated virtue. It is desired most urgently when there is a need to reprogram, say for a new machine, a horrendous complex of codes which we have lost control of and no longer understand. We don't really want the compatibility, which stifles advancing the art, but rather we fear to expose the muddle into which our own work has gotten.

2. Since the handling of information involves the generation and manipulation of information about information, the growth of data handling requirements is self-compounding unless carefully constrained.

This principle is insidious, but true. It might be called the Parkinson's Law of data processing. It is interesting to note that human beings, in doing clerical work or calculations, impose the necessary constraints in the form of what we call common sense and judgement. The dullest sort of person is able to exhibit these qualities in far greater measure than we often admit, but the brightest genius has difficulty in precisely defining them. Which leads to the third principle.

3. Nothing can be mechanized which cannot be precisely defined. This is as true of second, third and higher orders of control as of primary mechanization.

I have come to the conclusion that many people who ought to know better don't really believe this principle. Sometimes I try to ignore it myself, always to my eventual grief. It is too tempting to solve easy problems, pushing the hard ones back with a sort of mental note that "we'll do something" when the time comes. If pressed we may mumble something about statistical methods, which sounds impressive. Now I would not want to appear to minimize the worth of faith, self-confidence, and drive. But it is foolish to undertake a piece of work without having first estimated the effort required. In data processing, we have sometimes been trapped into doing this because of a failure to understand the true complexity of what we are asking for. Thus complex, far-reaching and important decisions and plans are left to those who are unqualified to make them, technically, administratively or policywise.

The inability to precisely define all aspects of a technology does not bother us too greatly in most endeavors. A car still requires a driver, a plane a pilot. We think of an assembly line as highly mechanized but it works only because many people are working on it and making little spot judgements all the time, sometimes big ones. But who can work at the rate of 40,000 additions per second, even in a supervisory capacity? Of course, the very essence of the stored-program concept is to allow complete pre-planning, which amounts to canned supervision, of the whole sequence of operations required by some process. There are two catches to this. The first is our second principle, i.e. the bookkeeping soon outweighs the data processing operations and becomes a bigger data processing problem than the original one. The second catch is that it is impossible to preplan the proper action for every eventuality; in fact, it is questionable whether the term "proper" has any meaning in this context except as dictated by particular policy. All this is simply a roundabout way of saying that human beings cannot be taken out of the daily activities for which they are responsible and which exist in the first place by their decision. The introduction of new machinery - no matter how revolutionary, powerful or potentially useful - does not change this basic fact of life. Rather than relieving people of effort or requiring less in the way of disciplined endeavors, this new equipment is more demanding if its use is to be profitable.

Let us admit boldly, or humbly if you prefer, that there is a lot of confusion in data processing work today. Let us further admit that there is a great deal of difficulty and uncertainty in the projects to which DP equipment is often applied. If there weren't, most of us would be out of work. Now while it may be possible to cut down on the confusion, it appears realistic to me to assume that the problem difficulties will increase, certainly not decrease. These problems are not the exclusive domain of any one discipline. Certainly they do not all belong to the field of programming. Programming has its share of tough problems and also its share of skill and helpful techniques to be contributed. What makes the programmer invaluable is that he knows how to make a computer perform. What makes him inadequate is that he doesn't always have the script. After all, problems originate in people's minds and techniques of operation reside in other people's minds. We need to remember, and to believe, that the really valuable residue of a completed job consists of the ideas, insights and techniques recorded in human brains, in short the experience gained. This leads to the fourth principle.

4. A wide variety of skills are involved in present-day EDP applications thus multiplying the problems of communication.

Communication evidently means different things to different people -- which seems odd. But one must recognize that vast efforts are underway to increase communication between systems, sub-systems, or just plain code and systems. It is not clear to me that communication between two inanimate objects can exist but I admit my knowledge of psychology is limited. At any rate, I am more concerned with communication between humans and by humans with machines.

Now there are at least three important conditions for communication:

(1) The parties concerned must have rapport in the area under discussion;

(2) the quantity and rate of information sent at one time must be within the capacity of the receiver to assimilate;

(3) the language and medium used must be known and compatible to both parties.

A lack of any of these can be compensated for by sufficient determination, repetition, patience and time. In EDP, however, efficiency must be given great weight. We cannot always afford to compensate for poor communication.

It might be asked how one can gain rapport with a machine. The best way is to build into the programs the kind of receptors and output devices that one would like. But this option is usually available only to a limited number of people, namely, expert system programmers. But once the user is one step removed from the machine, as say a production programmer, the communication is mainly between people. Furthermore, the system programmer now has a new responsibility, viz., to provide techniques which can be responsive to the user without major overhauls in handwritten code. Repeat, responsive.

Turning to the other end of the spectrum, how can the language and working papers of the physicist, economist, engineer, or operations researcher be compatible with the language and experience of the professional programmer. Except in isolated cases of fortuitous backgrounds, the answer is they can't. The truth is that there is a whole spectrum of languages from the problem formulator to the machine operator, with only slight intersections. I dislike to call these language "levels". It is not a question, at least not anymore, of one being above or below another. The language of the engineer is every bit as specialized as that of the programmer. Generalizations can be made in both directions but this may or may not help to get a job "on the air".

Now the machines are not the exclusive domain of any one group, any more than the problems, except in an administrative sense. The problem formulator is just as anxious, and probably more needful, to exercise control and introduce changes from his viewpoint and with his language as is the system programmer. I think we all recognize the fact that human beings still must work out together the roles they will each play in getting a job done. That is, that the preliminary discussions, planning, mutual training, etc. are going to be with us for a long, long time. What we apparently have failed to recognize is that the machine itself can be used in such a way that it can help us remember, control, modify and use the fruits of all this labor much more efficiently.

What should the system programmer be doing? He should not be building systems for himself. He should be providing standard parts and routines with which systems can be built. Of course, his consultation will be needed but the whole system should be literally built by the whole spectrum of people who are going to rely on it. How can the programmer anticipate the kind of system and languages that the next project will require? He must be prepared to build any definable system or at least to supervise its building. That is his specialty. The same thing is true of language. What we need is not a universal language but the ability to define and utilize any language that suits our purposes.

Consider a moment. Systems of 10,000 - 15,000 instructions are routine, 25,000 - 50,000 instructions are not uncommon. Some plans now underway will likely require 50,000 - 75,000 instructions. Where do we stop? Some of these systems are supposed to be all things to all people but on closer examination it turns out that they generally do about one thing well. Maybe they compile algebraic statements, or maybe they assemble and execute handwritten programs or maybe they perform certain standard D.P. functions. But they are only straight-line cuts out into a whole field to be harvested. Furthermore, they are difficult to understand and require a staff of specialists to service each one. Far from facilitating communication, they vastly complicate it.

Suppose then we begin by trying to extract from all systems those parts which are common. We will, without question, have to establish certain rules and conventions, for example, the card form for instructions to a monitor (any monitor) and the first and last words of each tape record, etc. In other words, we have to establish standards. But these standards must not be restrictive to the user. Their only purpose is to allow standardized parts from which any system can be started. We must not forget that knowledgeable people must be integrated with systems if versatility is to be maintained.

In addition to a package of standard parts, there is something more needed. This is a standard philosophy of using them. There is a somewhat subtle distinction here. We do not want to make all systems be just alike, otherwise we may as well program a standard system. The idea is to have standard techniques to cause a system to grow in whatever way is needed at the time. The word "grow" is not used lightly. Certain characteristics of an organism must be provided. To me, the notion of a universal computer-oriented language has meaning only in this dynamic sense.

There is one further requirement. The method must not lead to unnecessarily cumbersome systems, that is, this must not be inherent. With the present state of the art, at least, this will require some rather stringent conventions regarding symbology. Hopefully, they can be relaxed as experience is gained. I do not see this as a great problem, however. People are pretty good at picking up rules of this kind and at getting used to cryptic notations. They may grumble but their real problems usually lie elsewhere. If they understand the structure of the system and its attendant language, the actual encoding is not too important.

What are some of the features that might be expected to be available with such a method? I think the following is a minimum list.

(1) A core of highly polished utility subroutines, a standard skeleton monitor, and a basic assembler, all capable of expansion in well-defined ways and maintained by a single small group (per installation) of highly skilled professional programmers.

(2) The capability of defining hierarchial sets of macro-operations tailored to the job and understood to the depth required by the various echelons or categories of professional people involved. Since they would be involved in defining them, they should easily understand them.

(3) The ability to make changes at any level provided that at least the next lower level were understood, and to add levels as need dictates.

(4) The ability to debug in detail at object time without detailed pre-planning.

(5) Complete revision of a system by revising definitions and restarting from the basic system, with a minimum of elapsed time.

(6) Partially automatic documentation of a system since higher levels of operations will require fewer lines of programming.

(7) Ability to experiment with systems as such, and to extract production systems at will.

(8) Alleviation of the training problem by spreading the load now imposed on programming over a wider variety of skills.

Let me re-emphasize that such a technique as just outlined is going to be very demanding of the people using it. Responsibility is a necessary counterpart to responsiveness. The ability to exercise higher level control implies the ability to make bigger, more disastrous errors. With proper safeguards, however, this is somewhat compensated for by the ability to quickly reconstruct a previous situation.

To sum up the foregoing very briefly, no one can become expert in many fields simultaneously. Programming must provide access to machines for others, via the programmer but not by the programmer on every occasion. To expect to find individuals who can understand all the goals, restraints, techniques and policies of a large project and still be capable of translating these into a mechanized system is unrealistic. To hope that a system can be defined by one group and turned over to another for implementing is just as unrealistic. Everyone must play his proper role from the outset and communication must exist between members of a team, at the least by pairs. After all, systems exist to serve some human purpose, not to be self-sufficient automatons. The trick for making the hardware a useful member of a team is to construct systems which are self-adaptive in a machine oriented sense, but externally responsive in a problem oriented sense. The know-how to accomplish this exists.

As will be seen, there are implications of so-called self-organizing systems here. I believe that the reduced cost per computation, promised by the bigger, faster and more expensive machines around the corner, can only be achieved in practice by some such innovation. This is

not, however, a new idea, as suggested at the outset. It is simply the old idea of breaking down a technology grown complex into simpler pieces understandable separately by a wider cross-section of the population. The skills so highly developed and so cherished by the expert programmer are as needed and as important as ever. But his field has also become so specialized that he must concentrate on particular areas to be effective. He must use his skill to open up new facilities to others so that they too can work more effectively. Only by a multi-link chain of people between problem and machine can the growth of EDP be kept on a workable basis such as gave it its initial impetus.

A LINE-DRAWING PATTERN RECOGNIZER

Leon D. Harmon
Bell Telephone Laboratories, Incorporated
Murray Hill, New Jersey

Summary

A machine is described which recognizes line drawings of circles, triangles, squares, pentagons, and hexagons. This identification is independent of rotation and, within limits, of size, precision of drawing, or positioning. The system will also distinguish and count separate objects up to six with limited independence of the size, shape and position of each object.

This "Gestalt" recognition is accomplished by using a dilating circular scan. The virtue of such scanning is that object-size changes translate into time-of-arrival changes while object rotation preserves topological relationships. The n-gon detection depends on side counting, and the object counter uses the same logic in conjunction with a detection of discontinuities.

Thirty-two small photocells arrayed in a circle are mechanically puckered across the input plane. Light signals derived from opaque line-drawings are converted into binary representation in a special thyratron register. Time sequencing of logical operations and output displays is relay controlled. The actual decision logic uses a novel combination of optical and electronic devices in a simple but effective manner.

It is also shown that this circular scan technique is applicable to automatic detection of letters and numbers over a wide variety of styles, and several examples are given.

Introduction

One of the very interesting classes of problems in pattern recognition is that of the detection and classification of two-dimensional visual images such as written language and line drawings. Research of this nature is interesting because it may elucidate some of the problems of human and machine pattern recognition; it may be useful as well since it can lead to devices such as automatic reading machines. This paper discusses some of the problems encountered in such visual pattern recognition and describes a new approach to machine recognition of printed and handwritten letters and numbers.

In the past, machine detection of two-dimensional visual patterns has been generally restricted to carefully constructed and positioned alpha-numeric characters. Ideally however, pattern recognition should be independent of character size, rotation, and within limits, position, style, and noise. Such general or "Gestalt" recognition takes advantage of a set of characteristics defined to be invariant under these transformations, and can be usefully applied to many detection problems.

If we want a machine to make identifications which agree with those a human observer might make, the rules for classification of "essential features" must be similar for each. The definitions of letters, numbers, triangles, or any other symbols are quite arbitrary and are made in terms of the structure and limitations of the observer's mechanisms. Hence it is necessary to be quite careful when establishing the "essential feature" requirements for any classification. These essential features may be quite diffuse and exist only within ill-defined limits.

Recognition of visual patterns is most interesting and valuable when it is not restricted to a small class of inputs. We should like to understand how an A or a triangle may be detected irrespective of position, rotation, dilation, or of variations in style. A really useful reading machine will be one which can recognize the symbol 5 whether it is continuous or in two parts and whether it has been printed by a machine or scrawled by a child.

There are in existence a number of systems and proposals for obtaining limited recognition. IBM[1] has developed computer programs for reading a variety of alpha-numeric characters. Intelligent Machines Research[2] and Solartron[3] have machines for sale which read numbers and letters printed in one carefully controlled type font. Stanford Research Institute[4] and others have produced magnetic ink check readers.

Most of these systems have similar design concepts, based on inspection of sequences of pulses generated by scanning with a slit aperture moving past a symbol. Segments of numbers and letters are detected by matching pulse sequences for best fits to stored matrices of segments. The number, size, and position of these elements determines recognition of a character.

These devices have limited utility because of their critical dependence on the style of type which can be reliably read and their sensitivity to format and to tilt and size changes. They generally come into being to serve one specific purpose, that of high speed reliable recognition of one fixed symbol style. The IBM system is built around their output printer type font, the IMR scheme for credit card reading uses a letter and number style especially created for this purpose, and the SRI magnetic-ink marks must be carefully held to close size and position tolerances.

The principles that these systems use to achieve recognition are not very general or easily extendable. The desire for automatic reading of income tax forms, train car markings, postal addresses, bank checks, toll tickets and the like will have to be met by machines using much more powerful algorithms than those presently available. Considerable effort is currently being put into finding more general and flexible systems for pattern recognition, (5-10) but there are many difficult problems yet to be solved adequately.

One new approach which attempts to get at some of the problems of achieving more general recognition will now be described. Following that is the description and circuit details of a simple pattern-recognition machine based on that system.

Principles of Operation

It is possible to inspect a simple line drawing in a way which yields similar transformations for geometrically similar figures, independent of their absolute size or rotation. This "Gestalt" recognition is accomplished by using a dilating circular scan. The virtue of such scanning is that object-size changes translate into time-of-arrival changes while object rotation preserves topological relationships.

Consider an array of picture elements arranged in c concentric rings, having r elements in each ring. If each ring is inspected sequentially, say from center outward, the resultant c sequences of r signals can be shown to have the desired transformation properties. A logically equivalent implicit scan is possible with unidirectional propagation along radii and with unit delays inserted between adjacent elements on each radius. This type of signal processing has interesting analogies to physiological structures.

Suppose that an equilateral triangle is centered on such an array. The first intersections of the scanning interrogation ring with the figure will be three simultaneous "hits" on the mid-points of the triangle's sides. As the scan continues its outward sweep, points on each side of the initial intersections are progressively touched. The last parts of the triangle which are inspected are the vertices, those points at the greatest radial distances from the center.

Thus the outward puckering scan produces three radially equidistant epicenters of activity. An accumulating register of r places, representing the r elements of each ring in turn, will display three single active cells initially, equally spaced along the length of the register. There will then be a spread of activated cells in each direction from each of the initial locations. Finally the apices of the equilateral triangle will cause the last three cells of the register to be activated. Again these are equally spaced, representing the 120° radial symmetry of the vertices.

A similar but smaller triangle would have generated similar signals earlier in the sweep cycle; conversely for a larger presentation. Rotational changes are represented as lateral shifts in the contents of the register. Thus the detection of three simultaneous, equidistant, and uniformly expanding waves of activity is sufficient to "recognize" an equilateral triangle independent of size or rotation. (Bounds on size exist, of course. The lower limit is fixed by the resolution of the array of elements while the upper bound is determined by the field seen by the array.)

An isosceles triangle will generate simply detectable time and position asymmetries. Squares and rectangles are differentiable in a similar manner; their unique common property in this system is, of course, a count of four. A convex polygon of any number of sides can be detected, given sufficient resolution; circles are represented by a simultaneous filling of all places in the inspection register.

Let us consider now how a system can be developed to make use of such scanning. Suppose for example that the input transducer consists of 32 concentric rings of 64 photocells each, as in the illustration below.

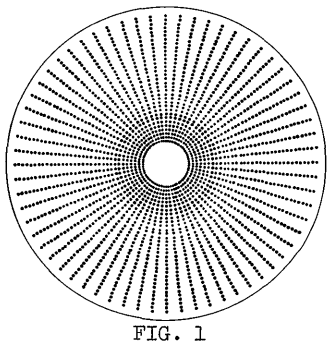


FIG. 1

Let a sweep cycle comprise the simultaneous gating of all 64 cells in each ring, one ring at a time, into 64 corresponding amplifier channels. Thus for an outward puckering scan, the innermost ring is first interrogated, then the next, etc., until the gating out of the contents of the outermost ring completes a sweep cycle. Fig. 2 shows, in block diagram, one system for using this information.

The beginning of each sweep initiates a storage operation in each of the 64 function generator locations. A linear accumulation (e.g., current) continues in each location until stopped by the first black element encountered by the expanding sweep. The value of each function at the end of sweep may be read out as a voltage level which is proportional to the radial distance of the corresponding image point from the center of the array.

At the end of a sweep, a comparator makes peak detections on the 64 generated functions. This operation is performed by local differencing over several adjacent locations. A simple count of such peaks is sufficient to identify an n-gon, while a measurement of the inter-peak distances (given by the addresses of the radii on which the peaks occur) establishes criteria for separation of isosceles and equilateral or of square and rectangle.

It can be shown that the alphabet of line drawings recognizable by this system is not restricted to a few polygons. Many figures have unique "invariants" if they are interrogated with an expanding circular scan whose origin is near the center of gravity of the figure.

A modest number of more complex comparator operations produces much more sophisticated recognition. Detection of concave and convex segments distributed in time over the 64 channels leads to identification of numerals, for example. A series of gedanken experiments on many examples of hand-printed digits has shown that very modest extensions of logic considerably enlarge the classes of recognizable images. Fig. 3 shows several examples of numerals which have been "recognized" in these experiments.

Each of the four numerals was drawn freehand and roughly centered on a layout of the photo-receptors. The rectangular grids are obtained by unfolding the circular array after making a cut from the center to the top of that array. Thus points along the top of the rectangle represent points on the inner circle of the photocell matrix while the bottom line contains the points forming the periphery of the circular array. The left and right vertical edges of the rectangle were coincident in the original layout, being the two edges of the cut we made.

This simple transformation makes it considerably easier to follow the scanning events and to discover useful detection criteria. The expanding sweep circle is seen to be represented by a horizontal line 64 cells wide moving down in time through 32 positions. Any given horizontal line may be thought of as the contents of a 64 place register at any epoch. Each activated receptor location on the circular array is shown as a dot in the rectangular representation.

It can be readily seen that size changes would appear as vertical (time) shifts while rotational transformations cause lateral displacements of the pattern. Since the left and right edges of the rectangle are actually contiguous in the original array, lateral displacements would simply wrap around the rectangular grid, disappearing off one vertical edge and reappearing at the other.

A set of tests for the examples shown in Fig. 3 will now be given. These tests are for purposes of illustrating principles and should not be considered the most complete or effective rules which can be found. We shall invoke two simplifications; the numerals will be centered and there will be no rotation.

An ascending function will mean a mostly continuous mostly monotonic decrease in values of $\underline{c}$ as $\underline{r}$ increases. The first 2 shown would be said to have an ascending function from r = 0 to r = 6; conversely the first 5 has two descending functions between r = 0 and r = 6. In a similar manner we may describe horizontal segments, concave or convex functions, etc. We test for the following:

1. An ascending function for $0 \leq r \leq 8$. This describes the curved section of the upper right portion of most 2's.

2. An ascending function for $24 \leq r \leq 32$. This describes that portion of the tail of a 2 lying to the right of center.

3. A partially horizontal or ascending function for $32 \leq r \leq 40$. The presence of at least a short segment which is level or rising during this interval indicates the left end of the tail or the bottom of the down-stroke for most 2's and is independent of whether or not a loop is present.

4. A two-valued function for $0 \leq r \leq 8$ and $60 \leq r \leq 64$ establishes the presence of the top bar and top of the curved portion of a 5.

5. A descending function for $16 \leq r \leq 40$ describes the tail of a typical 5.

The limits on $\underline{r}$ need not be taken as absolute. If our tests are based on relative $\underline{r}$ values, then the results of the tests can be independent of rotation of the sample numerals.

Rules based on the five tests listed above are sufficient to obtain unique "recognition" of the samples given and of a variety of other examples of these two numerals. The first three tests concern properties of the number 2 which are relatively invariant over a population of about twenty samples which have been checked. Similar rules have been successfully applied to all of the ten digits for a number of hand-printed samples. The extension to letters of the alphabet while more complicated, does not, in principle, appear difficult.

A Model to Demonstrate Principles

A simplified model has been built to recognize line drawings of circles, triangles, squares, pentagons, and hexagons. This identification is independent of rotation and, within limits, of size, precision of drawing, or positioning. The system will also distinguish and count separate objects up to six with limited independence of the size, shape and position of each object.

The present device simply detects and counts the segments intersected during one expanding sweep cycle. It is the largest count which determines the correct number for identification. In general, this maximum count may occur only briefly during part of a cycle, or it may be generated by several different segments encountered sequentially during the cycle. For any non-concentric placement of the figure with respect to the sweep center, one or more sides will be encountered earlier than the others. The relationships are depicted for the case of a square in the illustrations below.

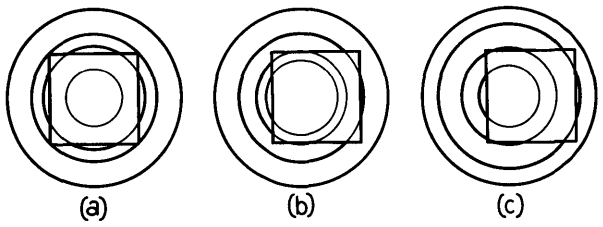


FIG. 4

In A, the innermost circle represents the minimum sweep diameter where no part of the figure has been encountered. The next circle out, corresponding to a time later in the sweep cycle, indicates the four segments cut from a well-centered figure, and the succeeding circle has just encountered the vertices. Finally the sweep has passed out of the square; all portions of the inspection circle have encountered the figure.

A slightly non-concentric placement, as in B, results in first one side of the square intersected, then three, perhaps four for a moment, then one again. This is in contrast to the case A where if any sides were contacted at any instant, all four were. In C where the decentering is excessive, there are never more than three sides intersected at any radius of the expanding sweep. It is required then to detect, store, and sum the number of separate sides encountered during one sweep cycle. This number can never be more than n for any position of an n-gon, and except for accurate centering of regular figures it is generally less than n at any instant.

Let the expanding circular sweep be quantized into a sufficient number of elements, and suppose that those elements which have not passed through the figure are given the value of 0; those which have encountered the figure are called 1. The system then must count the transitions between strings of ones and zeros around the circle. This count thus indicates the number of sides in the figure.

We can distinguish separate objects in the following manner. Notice that for any closed figure lying within the swept area and enclosing the center of sweep, all ones will eventually be generated. This is not the case for separate objects placed in the inspection field as in the illustration below.

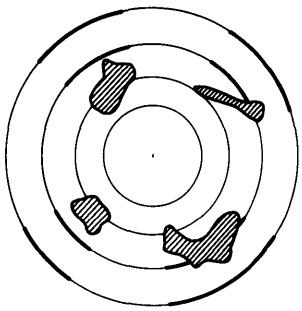


FIG. 5

Here there will be four discrete strings of ones and zeros but not all zeros will have been changed to ones by the end of sweep. This simple fact is used to provide discrimination of separate objects from continuous (or nearly continuous) line drawings.

Notice that the count does not depend on the size or shape of the objects, but only on continuity of form, i.e. on unbroken strings of ones.

The block diagram for this system is shown in Fig. 6. In this version, 32 rather than 64 quantized positions are used in each ring. Also, successive positions of the expanding sweep provide continuous signals rather than the 32 discrete signals described earlier in the general model.

The requisite scanning and resultant signals which provide input for the machine is obtained from 32 small photocells which are mechanically puckered across the input plane. As successive portions of the line drawing are encountered, a thyratron register records this information. At the completion of a sweep cycle, this stored data is examined for one-zero strings, i.e. - a series of ones followed by a series of zeros.

Conventional means for determining these binary sequences in a register involve shifting the contents of the register or otherwise sequentially commutating the digits. The present system avoids this complication by a simple parallel operation. Note that for a count of the numbers of strings of ones and zeros, the essential information is contained in the transitions between strings, i.e., the one-zero or zero-one locations in the binary number. For a closed ring of digits, as in the register representing the state of the 32 photocells, the numbers of strings of ones and zeros are equal. Consequently there will be 2n transitions for n strings of ones. Since the number of strings of ones (or zeros) corresponds to the number of sides of the inspected figure, a triangle is represented by 6 transitions, a square by 8, etc.

Parallel inspection of the thyratron register is obtained by connecting a small neon lamp (NE-2) between each pair of adjacent thyratron plates. As photocells pass beneath portions of the figure being scanned, they fire their respective thyratrons, which thus change state from zeros to ones. Those thyratrons yet unfired because their photocells have not been darkened, remain as zeros. At the junctions of ones and zeros, the neon lamps light, since there is sufficient potential difference between the plates of fired and unfired thyratrons. As can be seen in the illustration below, n sides produce 2n discontinuities and thereby 2n lighted neons.

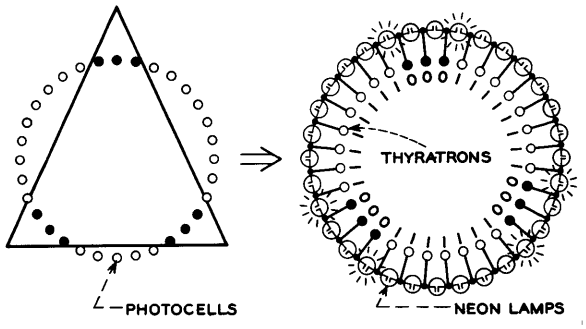


FIG. 7

The lighted lamps are "counted" in the maximum-count detector circuit (see Fig. 6), which sums the light output of these lamps and produces a voltage proportional to the number of sides of the figure. This part of the system consists of an enclosure with the 32 neon lamps arrayed in it together with a photocell. The signal from this cell is amplified, peak detected, and stored in a capacitor.

Since the peak detector indicates only the maximum number of neons lit at any instant, there will be fewer than the correct number of sides indicated in some cases. Consider Figures B and C of Fig. 4. Unless the object and scan centers are coincident, there may not be n intersections at any one time; rather the n sides of the object may be encountered successively during the sweep. One simple solution to this problem is to "save sides", i.e. to prevent the n strings of ones from flowing into each other as the vertices of the figure are encountered. This is accomplished by inhibiting that thyratron controlled by the last photocell passing through a vertex. It is locked up by adjacent thyratrons (thus preserving the sequence ....101...) and consequently the neon lamps on either side remain lighted for the duration of the scan.

In the final step a level detector classifies the stored signal and lights up the corresponding display. Since the amplitude of the signal is proportional to the number of sides in the figure inspected, the level detector is set to report a hexagon for the highest signal, a pentagon for the next lower, and so on through square, triangle, circle, and no-figure.

This detector circuit comprises five thyratrons, each of which has a preset firing threshold corresponding to the signal it is expected to classify. The stored, peak-detected signal is applied in common to all five stages. A common holdoff bias is then removed from one thyratron at a time, starting with the hexagon (highest level) detector and proceeding monotonically to the circle (lowest level) detector. When that stage is encountered where the signal is sufficient to fire the thyratron, recognition occurs.

This machine can also count up to 6 small objects placed at the inspection station. Only if a closed figure is inspected will all photocells be shadowed, whereas if discrete objects are presented, only a few cells, those passing beneath the objects, will have responded by the end of a sweep cycle. The distinction is easily made by monitoring the total current of the 32 thyratrons.

Notice that the count does not depend on the size or shape of the objects, but only on continuity of form, i.e. on unbroken strings of ones. The detection scheme used for the present device will erroneously count as one object any two or more which lie on a single radius, even though they are separated (at different radial distances). A more elaborate scheme can easily be made to count such objects separately.

A working model of this system, built to demonstrate principles, is shown in Fig. 8. A pentagon figure is on the horizontal platen, the central portion of which is the inspection station. The figure has been inspected, and the responding display appears in the small vertical panel behind it. Beneath the platen may be seen the mechanism which moves the scanning photocells in radial paths and which drives a rotary switch for controlling the sequence of events in a scanning cycle. At the very top of the apparatus is the lamp (within the pyramidal shade) which transilluminates the figure for scanning.

Fig. 9 shows four objects which have been counted, and the answering display.

This machine serves to illustrate the feasibility of using a circular scan to identify or count simple objects independently of rotation and, within reasonable limits, of dilation or translation. It is relatively uncomplicated, despite the detailed appendix which follows.

Appendix

In this section, some of the essential features of the pattern recognizer are described in greater detail. A complete schematic of the system is shown in Figure 10.

Scanning and Control Mechanism

The scanning photocells are actuated by a motor-driven mechanism which comes into play when identification of a figure is requested. The same motor drives a rotary tap switch which programs the sequences and durations of voltage applications to different parts of the circuit. A complete scanning cycle lasts about 1 second.

The photocells are mounted on rods which are part of a mechanism that is very much like an umbrella rib assembly. The cells are attached to the free ends of the rods and face upward. As the mechanism is operated, the cells, forming a circle of variable radius, move along radial paths. In a cycle, the cell assembly is first brought in to a minimum radius of about 1.5 inches, is then expanded at a linear rate of 4 inches per second to a maximum radius of about 5 inches.

The rotary tap switch has two sections of 32 contacts each, one section being used to close control relays and the other section to open them. These relays enable and disable different sections of the circuit by switching appropriate voltages. The sliding contactor arrangement in Fig. 10 represents the operation of these relays.

Not shown in this sequence chart is a relay which operates when the button is momentarily pressed to request identification. The relay locks itself up and turns on the overhead light and the driving motor.

Register

In this section of the circuit, the one signals derived from the lines of the figure by the photocells are used to fire thyratrons, thus registering these events for subsequent use. Each of the 32 photocells triggers a thyratron as it passes under a line of the figure to be identified.

At the start of a cycle, the 32 thyratrons are disabled (no plate voltage) so that the scanning mechanism can come up to speed. When it does, relay 1 applies 150 volts to the plate bus and the active scan cycle begins. As a photocell passes under a line of the figure, it generates a positive pip, large enough to overcome the bias on the thyratron control grid, causing firing.

The 32 thyratrons form a closed ring, tied plate-to-plate by small neon lamps shown as L7, L8, L9, etc. When one thyratron is conducting, its plate potential falls, while that of its neighbors remains high, and so the two lamps on either side will light up. If a neighbor is fired, the lamp between the fired thyratrons, having little or no potential difference across it, goes out. However, another lamp goes on between the newly fired thyratron and its unfired neighbor. As more thyratrons are fired, lamps are progressively turned on and off, the lit one at any moment locating the boundary of fired and unfired stages. Thus as the sweep expands, we continue to have a pair of lit neons for each line segment of the figure as is shown in Fig. 7. We see then that as the cycle proceeds, two lamps will light at the first points of contact with each side of the figure, then separate and run out toward the vertices.

If the figure to be identified should be displaced relative to the circle of cells so that one or more vertices are completely scanned before others, the correct number of lamps (2n) will not be on simultaneously. Since the photocell in the lamp housing reports only what it sees at any instant, it will report too low a number.

The circuit of 32 thyratrons has a "side-saving" feature which makes it unnecessary to center the figure precisely on the sweep center. We can save sides in the following manner: if the thyratron controlled by the last photocell to sweep through a vertex can be prevented from firing, the neons on either side of it will remain on even after the circle of photocells has dilated past this corner. If this action takes place at all vertices, the lighted neons denoting sides will all remain on. The photocell in the lamp housing will then see 2n lit lamps corresponding to all sides scanned, regardless of the temporal sequence.

This feature is realized by using the plate voltage swings of fired thyratrons to inhibit firing of adjacent stages. A resistor network couples each plate to the screens of both neighbors. If a tube is flanked by either one or two unfired stages, its screen is kept at ground by the associated diode which is being held in conduction. This condition allows normal operation of the stage. However, if both neighbors are fired, the screen of the tube between them will be driven sufficiently negative to inhibit firing, the clamping diode releasing for this condition. Thus this stage (corresponding to a vertex) is never fired, and the two sides have been saved.

The 1μF capacitor, 4.7 K resistor, and diode associated with each screen prevent inadvertent lockout. When both flanking stages fire, a slight delay is introduced ($\tau \approx 4.5$ MS). This gives the flanked stage an opportunity to fire even if the request occurs late due to minor electrical or mechanical uncertainties. This delay is long enough to accommodate all such variations encountered, but is short compared to the normal rate of progression of cells through a vertex, thus permitting lockout to occur for the desired conditions.

A decentering of the inspected figure by fifteen percent or so will usually cause no error. This lockout action is unreliable, however, when the figure is so positioned that two photocells pass through a corner at the same time. The thyratrons controlled by these cells will both fire, and the associated neons will go out

immediately. In practice, the number of errors due to this positioning is extremely low.

Maximum-Count Detector

Having translated the number of sides n in a figure into 2n lighted neons in the register circuit, we next convert this information into an electrical analog.

The 32 neon lamps are in a light-tight housing, together with a photocell. Since this cell has a nonlinear input-output characteristic near the origin, a bias source of light (two grain-of-wheat incadescent lamps) is used to move the operating range into the linear region. The signal from this photocell is amplified by a chopper-stabilized operational amplifier and is peak detected and stored by a diode and capacitor. The charge on the capacitor is then proportional to the number of sides in the figure.

Display Selection

In this final step of the recognition cycle, the electrical analog of an n-gon is classified and then identified by a lighted display.

This part of the system uses five thyratrons each with a relay in its plate circuit, to operate the appropriate display. The thyratrons have different triggering thresholds, arranged in a graduated series. The signal previously stored on the capacitor is presented to all five control grids simultaneously, via a common connection, but the firing thresholds are reached sequentially.

The screen grids of these tubes are connected to biasing networks through diodes. Each screen is set to a different bias in a stepped series. While the signal is being stored on the capacitor, the 5 thyratrons are inhibited, since their screens are held deeply negative by relay 2 (see sequence diagram) until the active scan is completed. This relay then removes the -33 volt supply from the screen bus, and the screens rise toward the different biases determined by their networks. Capacitors from these screens to ground are part of RC delay networks, which permit the screens to reach their preset biases sequentially at intervals of about 5 milliseconds.

Now let us take an example. A signal of +35 volts, a typical value for a square, appears on the storage capacitor, and hence at the control grids of all 5 thyratrons. None can fire, however, until the screen bus is released from the -33 volt supply which is holding all stages completely cut off. When this inhibition is removed, the lower thyratron in the schematic (corresponding to a hexagon display) rises almost immediately to its preset bias, there being no delay built into this stage. With the bias selected, this tube will not fire with the +35 volt signal on its control grid. Five milliseconds later, the "pentagon" thyratron reaches its preset bias, which is also large enough to prevent firing for this signal. Another five milliseconds later, the "square" thyratron reaches its threshold, which is such that signals over 30 volts will cause firing. Consequently it fires, its plate relay closes and lights up the display of a square. At the same time the storage capacitor is discharged through grid conduction in the fired thyratron before the remaining stages (triangle and circle) reach their thresholds, and so no other stages can fire.

Object Count

When discrete, opaque objects on the platen are scanned by the 32 photocells, thyratrons controlled by intercepting cells are fired, while the rest are not. At the juncture of fired and unfired thyratrons, neons will light up, two for each object. Since two neons also light up for each side of a polygon, the processing of the signal is the same for both figures and objects, except for the display.

If a figure is being scanned, all thyratrons are fired, except the n inhibited by the sidesaver; if objects are being scanned, only a few thyratrons are fired. The difference is detected by a relay (6) in series with the plate supply for the 32 thyratrons. This relay is adjusted to operate when about 24 of the 32 thyratrons are fired. In turn another relay (7) is operated to switch between the object display and the figure display.

Relay 7 also has another function. In the identification of circles, should the figure be precisely concentric with the scan circle, all 32 thyratrons will fire essentially simultaneously. Consequently, there will be no lamps turned on and no signal generated. In this case, relay 7 (which is energized because $\geq$ 24 thyratrons were fired) brings the storage capacitor to 0 volts (from -33V). This is sufficient to fire subsequently the circle thyratron.

I acknowledge with pleasure the design of the mechanical pucker by C. F. Mattke and the coaxing of the electronics to perfection by S. E. Michaels.

References

(1) Greanias. E. C., Hoppel, C. J., Kloomok, M., Osborne, J. S., - Design of Logic for Recognition of Printed Characters by Simulation - IBM Journal, January, 1957.

(2) Shepard, D. H., Bargh, P. F., Heasley, C. C. Jr., - A Reliable Character Sensing System for Documents Prepared on Conventional Business Devices - IRE Westcon Convention Record, Part 4, 1957.

(3) Solarton Electronics Group, Ltd., Surrey, England.

(4) Eldredge, K. R., Kamphoefner, F. J., Wendt, P. H., - Automatic Input for Business Data Processing System - Proc. Eastern Joint Computer Conference, December 1956.

(5) Bledsoe, W. W. and Browning, I., - Pattern Recognition and Reading by Machine - Proc. Eastern Joint Computer Conference, December 1959, (in press).

(6) Bomba, J. S., - Alpha-numeric Character Recognition Using Local Operations - Proc. Eastern Joint Computer Conference, December 1959 (in press).

(7) Grimsdale, R. L., Sumner, F. H., Tunis, C. J., Kilburn, T., - A System for the Automatic Recognition of Patterns - Proc. Inst. Elect. Eng., December 1958.

(8) Sherman, H., - A Quasi-Topological Method for Recognition of Line Patterns - Proc. Int. Conf. on Information Processing, Paris, June 15-20, 1959, (in press).

(9) Unger, S. H., - Pattern Detection and Recognition - Proc. I.R.E., Vol. 47, No. 10, October 1959.

(10) Sprick, W. - Character Reader - U. S. Patent 2,838,602, June 10, 1958.

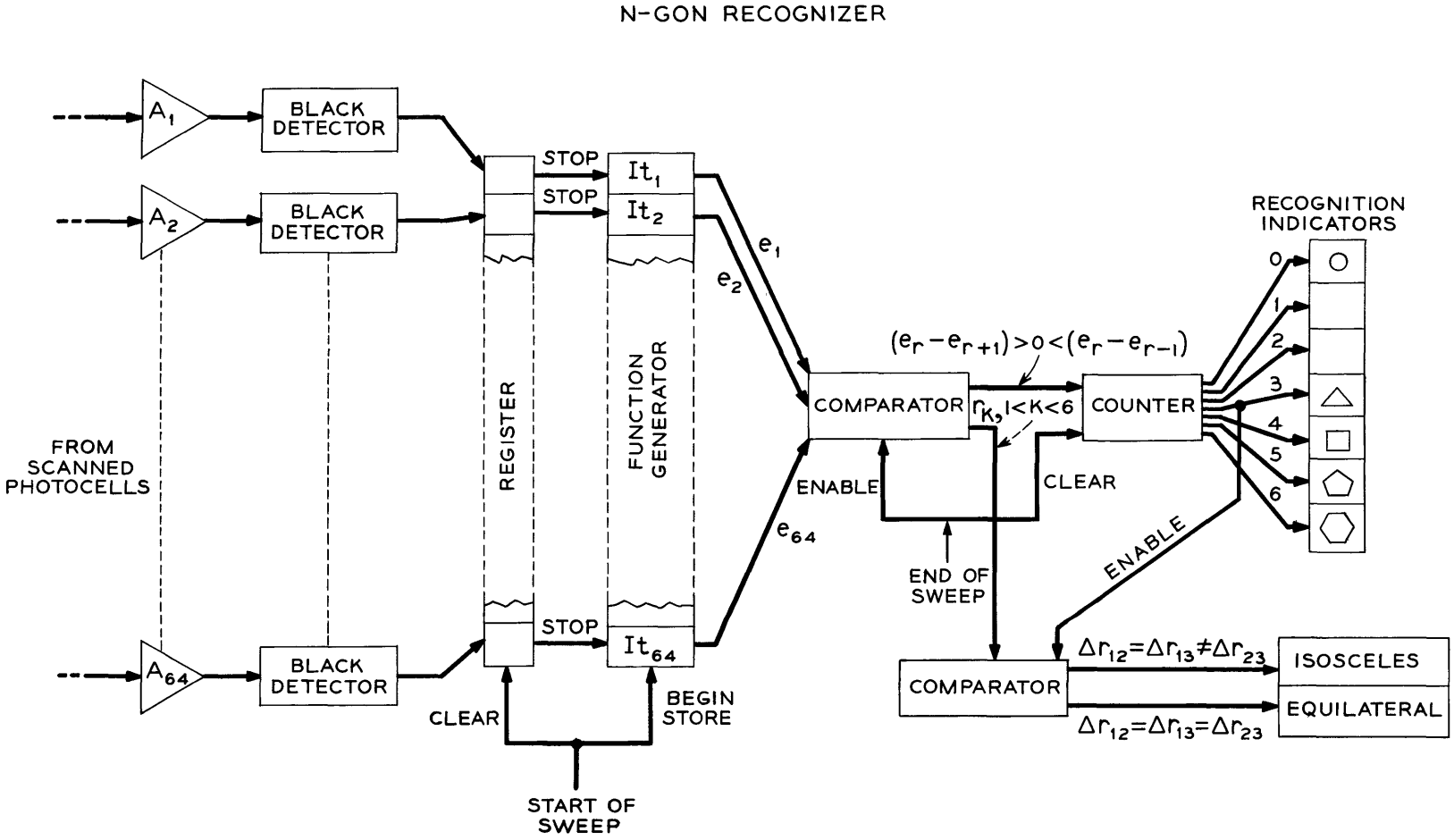


FIG. 2

Block Diagram of N-Gon Recognition System

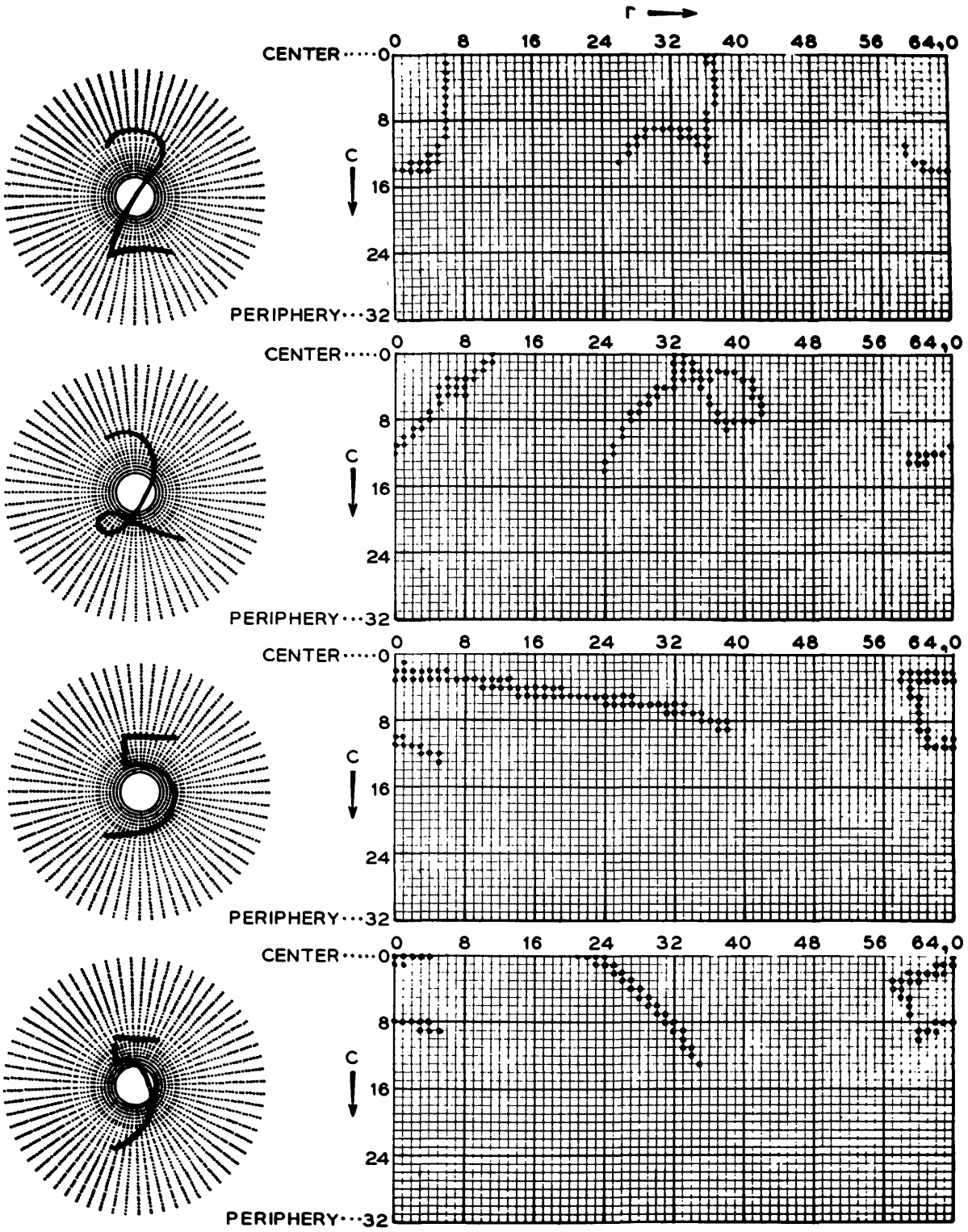


FIG. 3

Invariants in Numerals

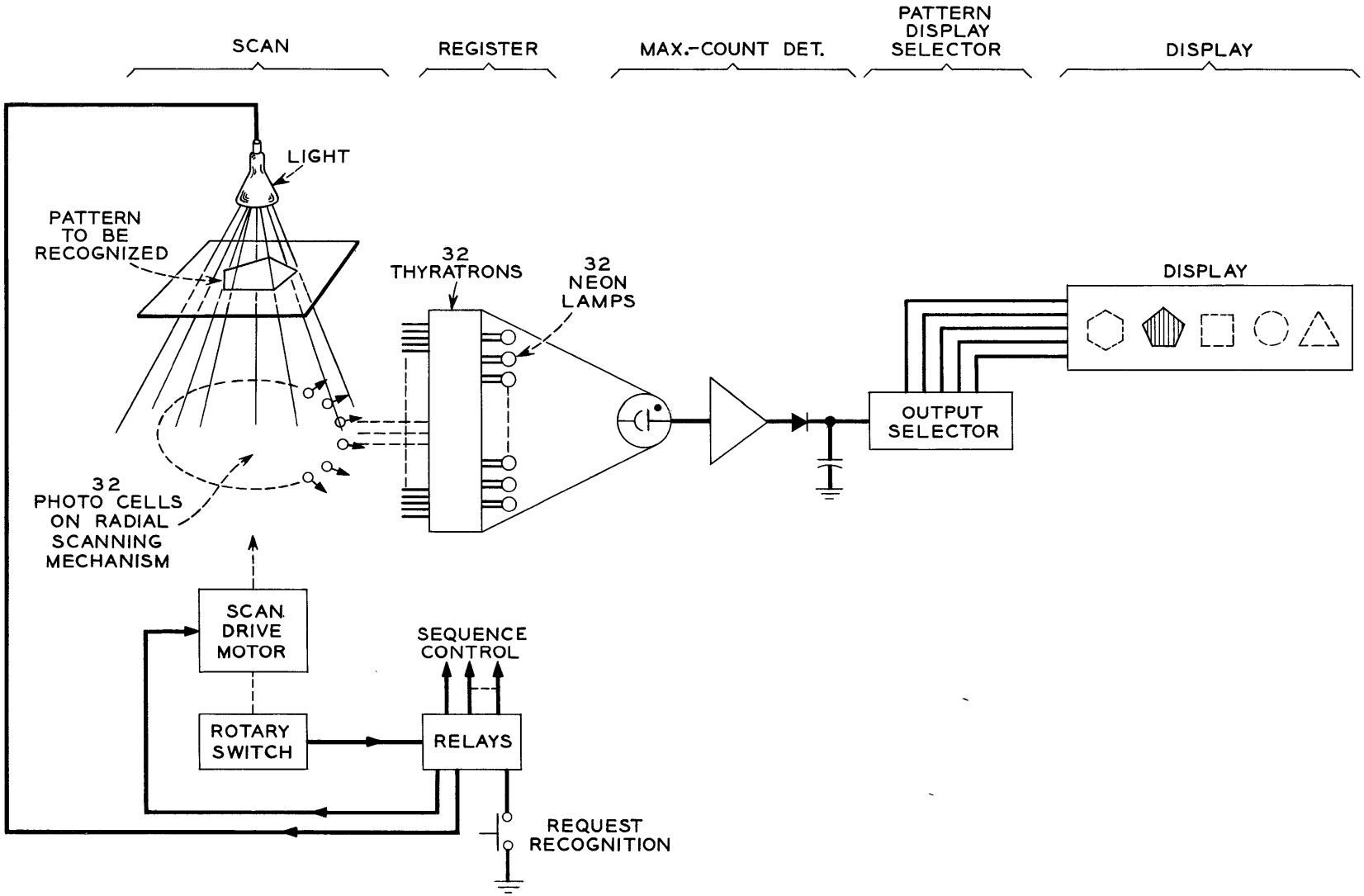


FIG. 6

Block Diagram of Recognition System as Constructed

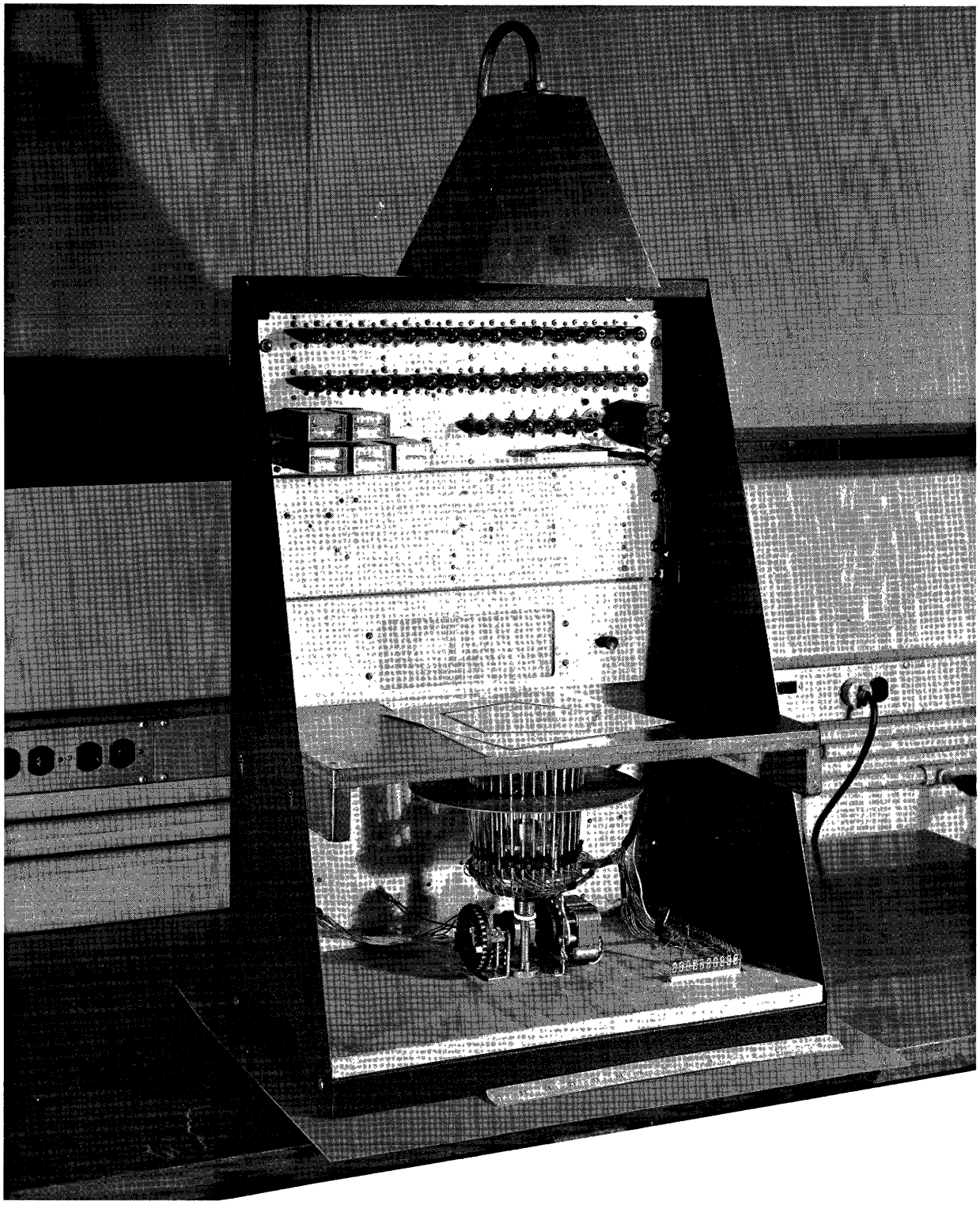


FIG. 8

N-Gon Recognizer, Showing Mechanical Scanner and Output Indicator

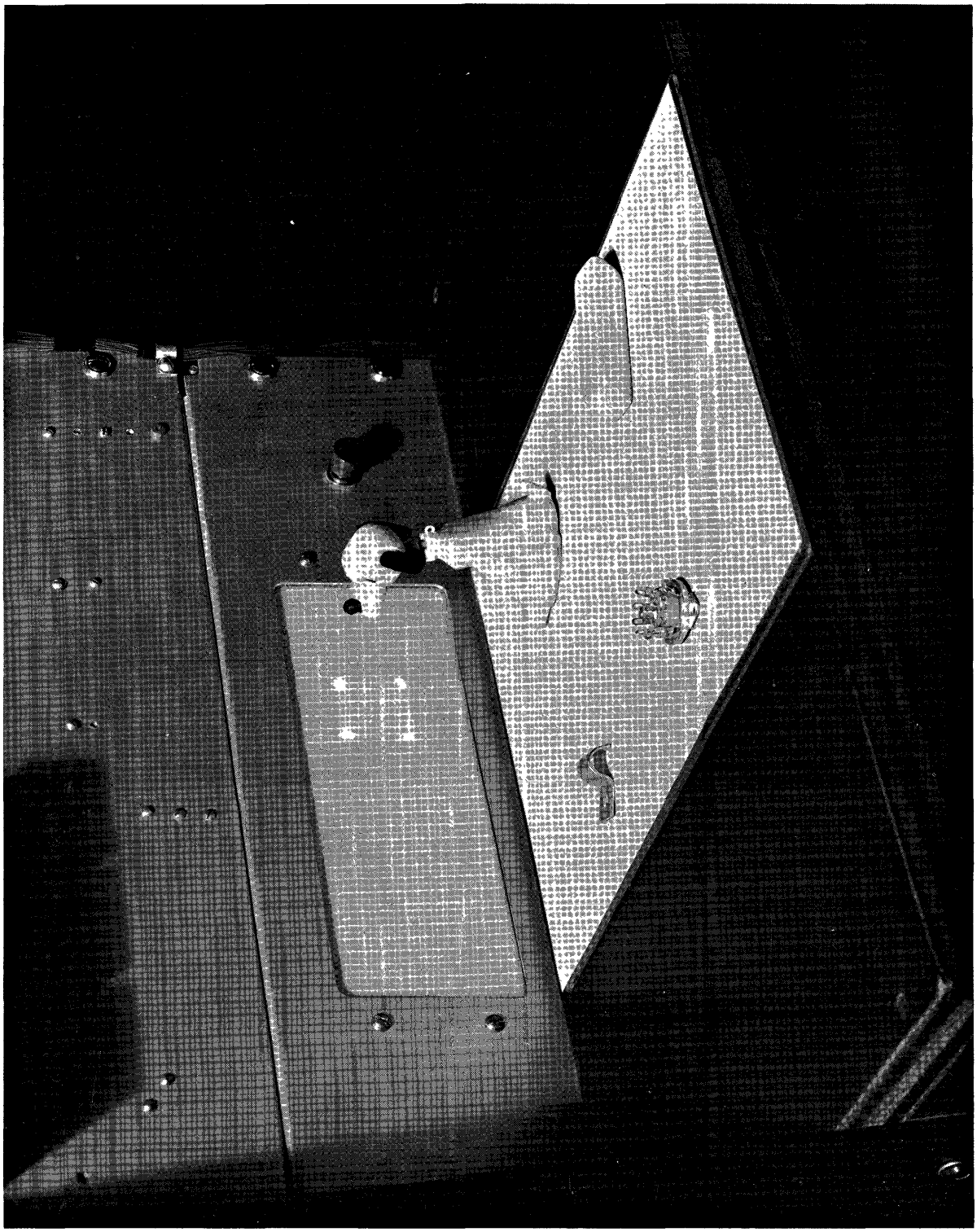


FIG. 9

Four Separate Objects Have Been Counted

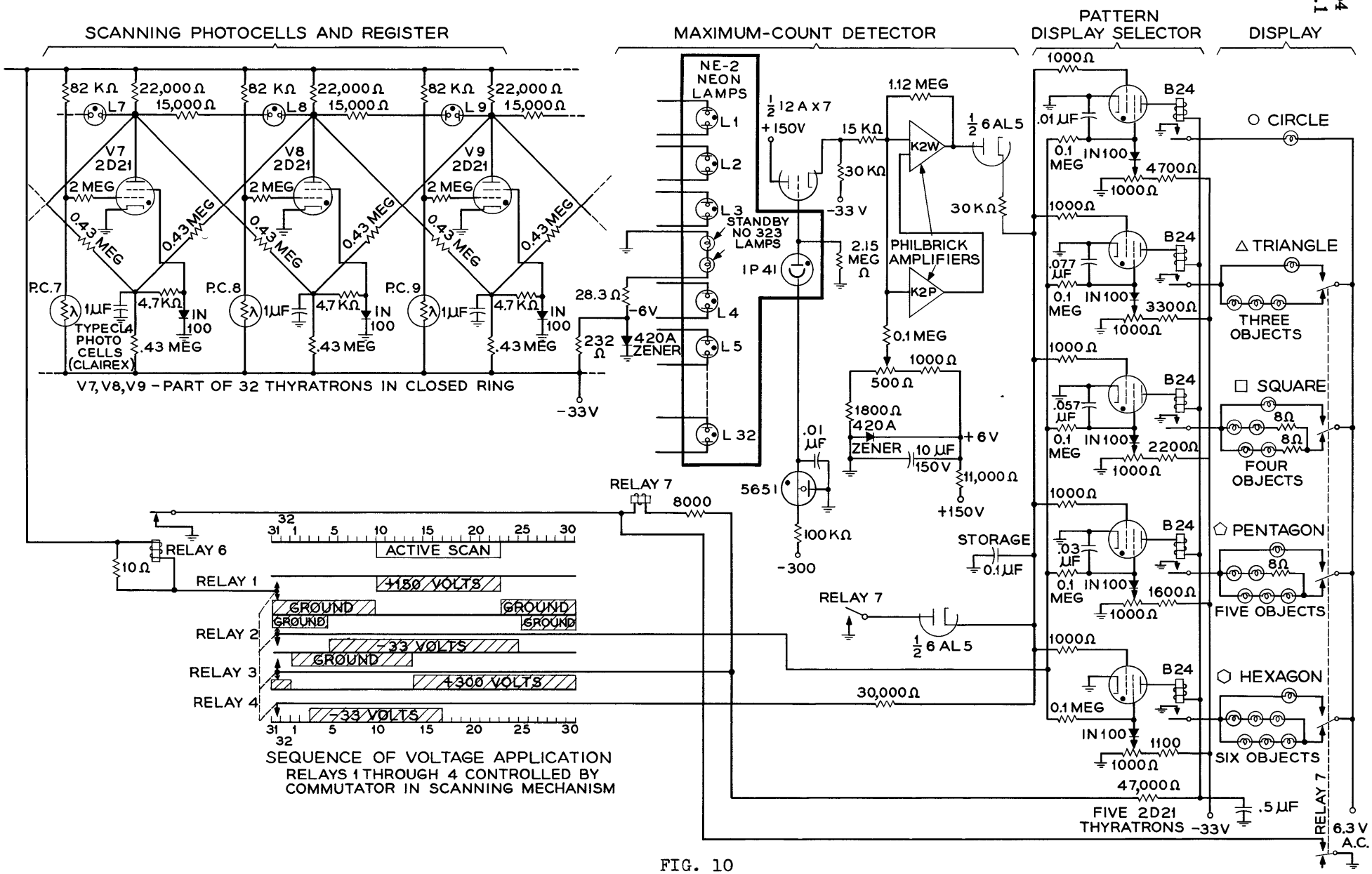


FIG. 10

Schematic of N-Gon Recognizer

RCA's Automatic Store and Forward Message Switching System

by
T. L. Genetta, H. P. Guerber and A. S. Rettig

Electronic Data Processing Division
Industrial Electronic Products
Radio Corporation of America
Camden, New Jersey

INTRODUCTION

AutoData is an automatic, fully transistorized store and forward message switching system. It is designed for use in common user communication networks in which message traffic in digital form may be relayed throughout the world. When the message switching system is used, many advantages may be obtained, some of which are:

- Economic utilization of communication facilities
- Automatic transfer of messages through the network
- Effective utilization of communication facilities operating at different data rates, with different codes, and with different coordination procedures
- High degree of message protection all with,
- Building block expansibility to provide increased service as required.

The design features of the system are presented by first describing the configuration of a communication network, then explaining the basic switching center functions, outlining the main system design considerations, and finally reviewing the flow of messages through a typical switching center.

THE COMMUNICATION NETWORK

Figure 1 shows the elements of a typical communication network using AutoData Message Switching Centers. Messages originate and terminate at subscriber terminal stations (T_1 through T_n) that are connected to the centers by tributary channels, or at Electronic Data Processing Equipment (EDPE) stations which may be local or remote from the centers. The switching centers are interconnected with trunk channels. All communication channels and their associated line terminal equipments are of types currently available in the communication industry.

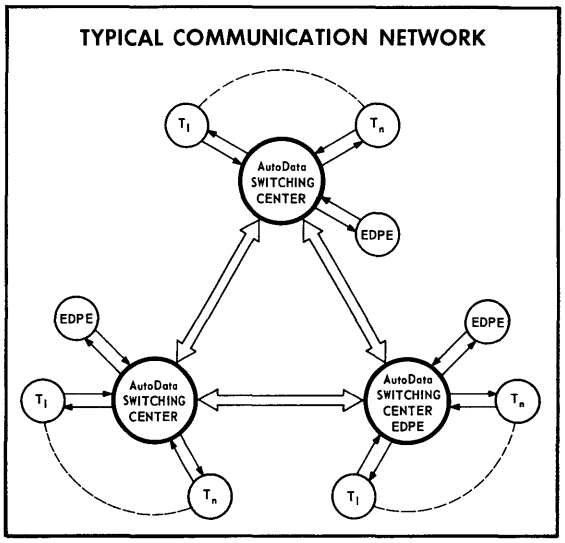


Figure 1

The message traffic handled by the communication equipment discussed here is always in digital form. Each message consists of three parts: header, text, and ending (See Figure 2). The header contains all the information needed to route the message. The text comprises the intelligence to be transmitted. The ending discreetly marks the termination of the message. The header and ending must consist of characters of information organized in a manner such that they can be interpreted by the switching center. The text may consist of plain text characters, encrypted characters, or bit-stream traffic organized into checkable pseudocharacters.

Messages are interchanged through the network by sending them first to the nearest switching center. They are automatically relayed among the switching centers and finally distributed to the destinations listed in the headers.

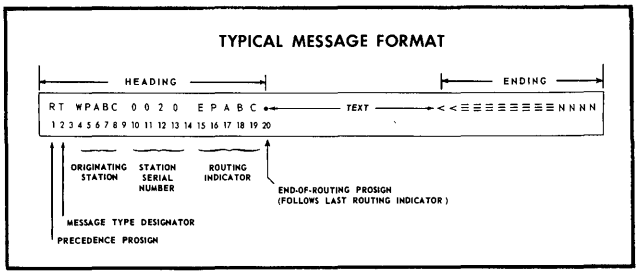


Figure 2

A message may be relayed among different types of communication equipment, such as teletype, punched card transceivers, as well as the new high-speed digital data transmission devices which will be available soon. These equipments have widely varying characteristics. The data transfer rates will vary from around 30 to 3000 bauds. Data characters will be represented in a variety of codes varying from 5 to 8 elements, with different redundancy features. The control procedure used on the various channels will differ. Some will provide closed-loop control while others provide no return path for signalling; some will be fully automatic while others are manual. The modes of transmission will include several types of keying: For example, DC keying, tone keying, frequency-shift keying, and phase-shift keying and will be over wire line, microwave, or radio circuits. The AutoData equipment provides compatibility between these many types of communication devices.

SWITCHING CENTER FUNCTIONS

A simplified functional diagram of a Message Switching Center is shown in Figure 3.

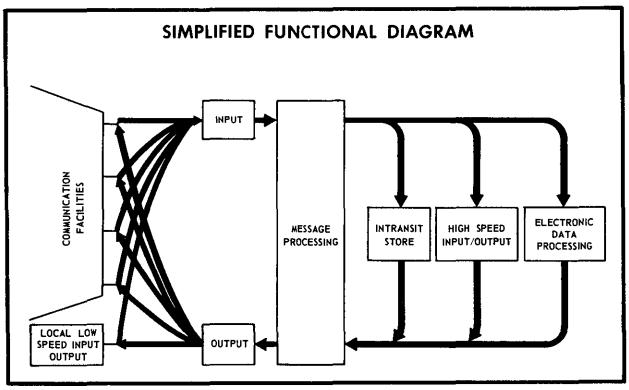


Figure 3

The Input function receives message characters from the various incoming channels and accumulates them into blocks conditioned for processing. This input function includes character and block checks and exercises control over the transfer of messages over the incoming communication channels. These accuracy checks together with the ability to repeat blocks allows errors in transmission to be corrected automatically.

The Message Processing function interprets the heading of each message and directs it to the appropriate outgoing channel. As the blocks of a message are received from the Input, they are transferred to the Intransit Store; only when the message is complete does it become a candidate for transmission. As each outgoing channel becomes available, the Message Processing function transfers to the Output the oldest message in the highest precedence category for that output channel. A stored program data processor is used for this function. This provides great logical power, a high degree of flexibility and very fast message handling speeds within the center.

The Intransit Store function provides a reservoir for messages when their outgoing channels are busy. It is this function that provides the "store and forward" capability of the switching center. It permits any subscriber to send his message without waiting for a through connection. It permits one incoming message to be delivered to any number of addressees. It allows messages to be transmitted out in sequence by priority as well as by age. It further allows top priority traffic to automatically interrupt lower priority messages and provides the means for repeating the interrupted messages.

The Output function coordinates the transfer of outbound traffic on each tributary and trunk channel. Storage capability is provided for each channel to maintain continuous transmission. Repeats of traffic blocks are provided by the output function as requested by the next switching center or tributary equipment.

Messages to and from local EDPE and other High Speed Input-Output devices have access to the switching center by way of the Message Processor. Local Low Speed Input-Output equipment is handled like any other tributary channel.

DESIGN FEATURES

Some of the more important design features of the message switching system are illustrated in Figure 4. These features are fully automatic

except in a few cases where the subscriber terminal equipment does not permit.

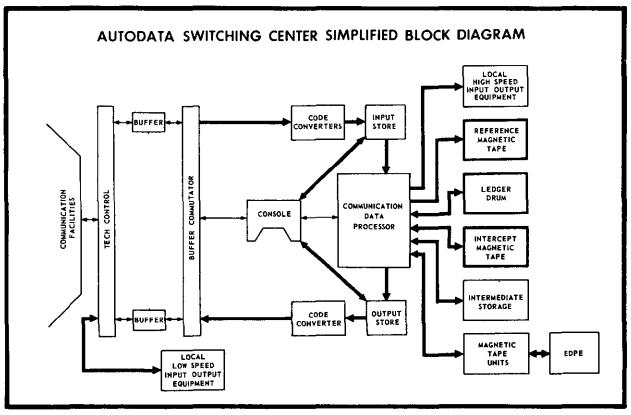


Figure 4

The communication channels are shown terminating in an equipment complex called Technical Control. Technical Control contains switching facilities which provide flexibility in the connection of the various types of buffer units to the corresponding types of communication channels. In some installations this switching capability is manual and in others it is semiautomatic. A further refinement provides for patching one communication channel directly to another. This allows any subscriber to be connected directly to any other subscriber in the network; a direct connection of this type is convenient in cases where large volumes of traffic are to be transmitted to a single destination. Needless to say, this capability can be exercised only when both terminals and all the intervening communication channels are compatible.

Technical control also includes test equipment for monitoring and maintaining the various communication channels.

The Input function of Figure 5 is composed of the Buffers, Buffer Commutator, Code Converters and the Input Store.

The Buffers provide independent termination for the various communication channels. A family of buffers is available for use with the various types of communication terminal equipment. This permits different data rates, different coding and different control signalling to be used on the various channels. The present system is designed to handle data rates from 30 to 3000 bauds on each channel with a maximum of 50 incoming channels. The number of channels may be increased in groups of 25 if some of the channels operate at the lower baud rates.

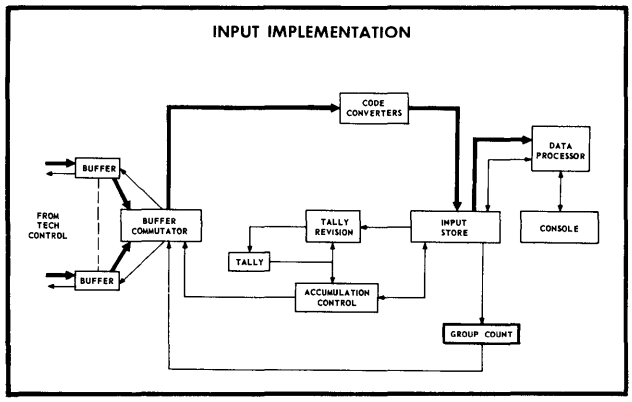


Figure 5

The Buffer Commutator causes all the buffers to be examined every 1-1/2 milliseconds. This speed is fast enough to require storage of only one character in each buffer.

A family of Code Converters is available, one for each code conversion required. Each converter is shared by all the channels using its assigned code, converting characters in that code to characters in the common language code of the center. Some channels, such as trunk channels may use the common language code. Characters from these channels, of course, bypass the code converters.

The Input Store consists of a magnetic core high-speed memory with storage areas assigned to the individual incoming channels. The exact storage location for each incoming character is obtained from tally information also stored in the high-speed memory. The tally system also provides the basis for channel coordination.

The philosophy of channel coordination used here is one of control by the receiver. As each block of characters is received, control characters are sent over the return channel either requesting the next block of characters or requesting a rerun of the same block. In this manner, no block is accepted unless it passes all the checks provided on that channel. Message numbering is available to provide protection on those channels operating without feedback control.

The Communication Data Processor (Figure 6) incorporates a wide variety of advanced programming and logic features:

- Basic Memory consists of 16,000 56-bit

words accessible on a 1-1/2 μs cycle
- Provision is made for overlapping operation of two memory units
- Instructions are of variable length with provision for index registers for relative addressing, and address substitution to achieve direct addressing
- Data tagging is provided for automatic identification of messages as well as for data separation
- Simultaneous operation is provided. The number of routines that can take place simultaneously is not fixed, but is a function of the relative speed of any associated devices involved and the complexity of the individual routines. Such simultaneous operation is particularly useful for file searches and general "housekeeping" as well as for servicing on-line input/output devices.

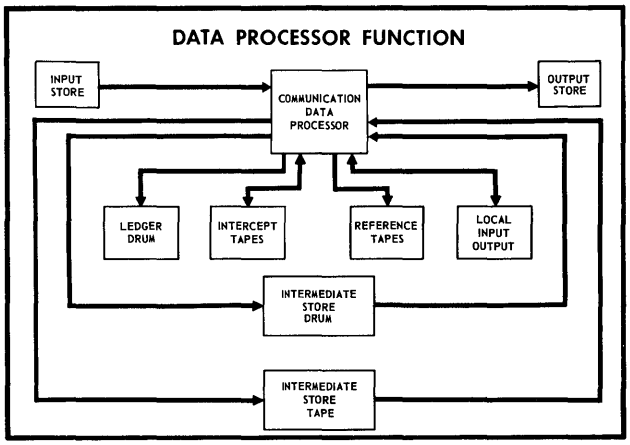


Figure 6

As many of the required system operating features are included in the CDP as possible. Some of the system requirements that are programmed are:

- Routing of messages according to single, multiple, and collective routing addresses
- Transmission by priority and age
- Interruption for top priority traffic
- Alarms for messages not processed out within a time period specified by priority
- Intercept on magnetic tape of selected traffic
- Editing and format conversion
- Collection and display of statistics
- Automatic routing by both routing address and priority.

These are some of the special functions provided by the Communication Data Processor in addition to its primary function of transferring traffic blocks from the input store to the intermediate store and later to the output.

The Output function (Figure 7) comprises the Output Store, Code Converters, Buffer Commutator, and Output Buffers. These elements are the direct counterpart of the corresponding input elements.

Messages to and from magnetic tapes associated with local subscriber Electronic Data Processing Equipment, are accommodated through one of a series of Tape Station Converter Units. These units provide the needed signal and code conversion.

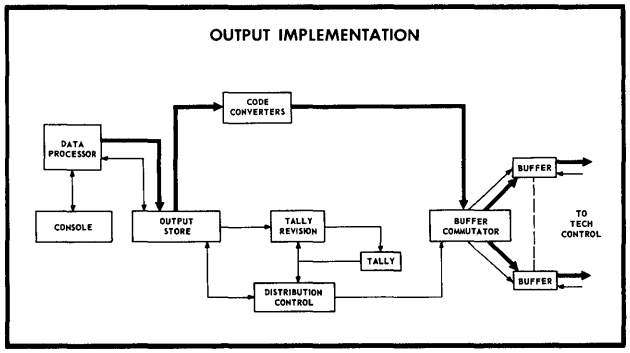


Figure 7

MESSAGE PROTECTION

Figure 8 illustrates the message protection and duality features in greater detail. Journal and Reference Magnetic Tapes are maintained under control of the Communication Data Processor. The Reference Tape contains a copy of every message received by the center. Each message is tagged to make recovery easy. The Journal Tape contains an abbreviated record of each message that has been relayed by the center. These records include a discreet message tag corresponding to the tag in the Reference Tape, time of processing in and out, channels and channel sequence number, if used.

The Ledger Balance subsystem maintains a file on a magnetic drum of all messages currently passing through the center. Each message is logged as it enters and as it leaves. This file is continuously searched for messages that have been in the center too long. Any overdue messages found are directed to the operator for expediting.

CONTINUITY OF OPERATION

This equipment is intended for communication

use and therefore requires much higher standards of reliability than are currently available in data processing equipment. Although transistor circuitry of proven reliability is being used in the construction of the switching center equipment, an occasional failure can be expected.

Elements of the system that are common to all channels are duplicated to provide back-up capability. One Communication Data Processor has enough capacity to handle all the switching center channels and normally carries the entire traffic load. A second processor is provided which is kept in idling status--exercising and performing subordinate tasks for the operating unit. Both processors have access to the high speed memories of the other, as well as all input/output stores, magnetic drums and magnetic tapes. The operating Communication Data Processor continuously updates address lists in the idling machine so it is always ready to assume the traffic load.

No automatic back up is provided for those elements whose failure would effect only a single channel or a small group of channels. The buffers fall in this category.

Figure 9 is a photograph of a scale model of a 50-channel AutoData Message Switching System. This equipment is presently being constructed by RCA for use by the military.

This system will provide interconnection between existing communication networks that are presently considered incompatible and will provide communication between computers on a fully automatic basis.

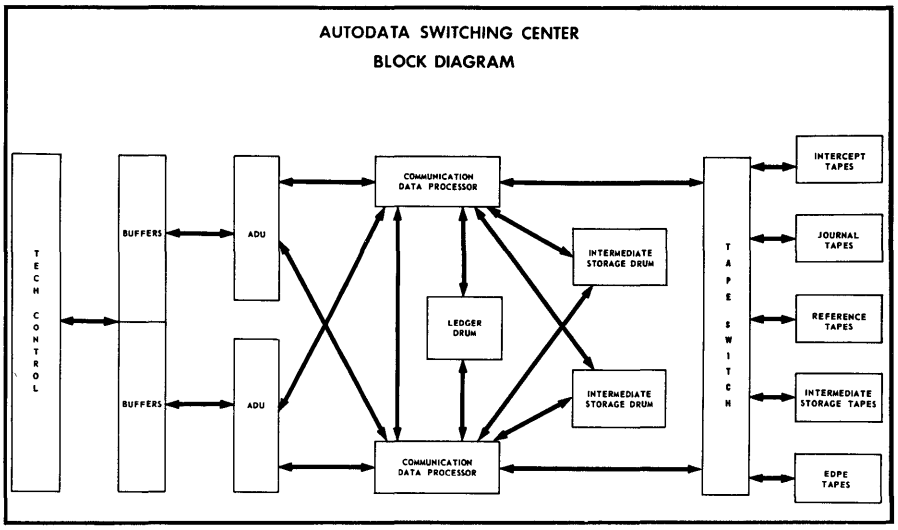


Figure 8

MODEL OF AUTOMATIC STORE & FORWARD MESSAGE SWITCHING SYSTEM

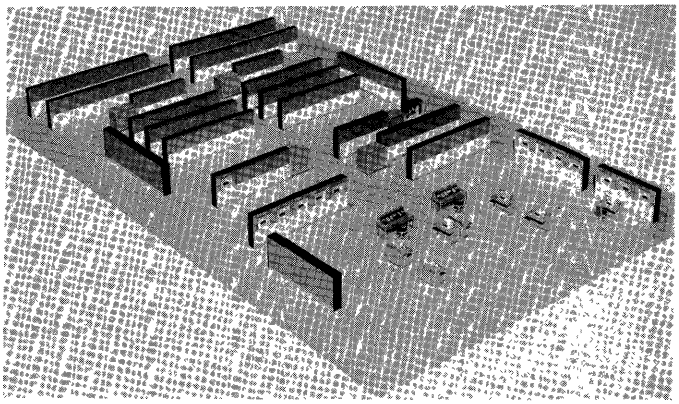


Figure 9

PRODUCTION OF MAGAZINE LABELS BY THE VIDEOGRAPH PROCESS

By

B. H. Klyce
Technical Coordinator
Subscription Service Division
Time Incorporated
Chicago, Illinois

J. J. Stone
Manager, Electronics Engineering
and Research Department
A. B. Dick Company
Chicago, Illinois

Abstract

System considerations point out the feasibility of using high density master file tapes in the preparation of subscriber labels for magazines, thus eliminating the necessity of using the central computer complex for the preparation of edited tapes. The operation is completely automatic, and operators are required primarily to feed raw materials and file tapes into the machine and to remove the finished products. Production of labels at the rate of 131,000 per hour is accomplished by means of an electrostatic process in which the image is formed directly on a special paper. Several safety features insure reliable operation of the equipment.

Introduction

TIME Inc. maintains an active file of approximately 10,000,000 subscribers to four of its domestic magazines - TIME, LIFE, FORTUNE and SPORTS ILLUSTRATED. A file of this nature is a dynamic one for there is constant activity in the form of changes of address, new subscriptions, renewals, inquiries, etc. Since 1946, the subscriptions have been processed with a semi-mechanical punched card system. Beginning in the fall of 1960, the punched card system will be "phased out" and a magnetic tape system will be installed, bringing a new degree of machine processing to subscription service.

The installation of the new equipment is the result of nine years of intensive study of magnetic storage and processing techniques. As late as 1955, it would have cost considerably more to process subscriptions by magnetic means than by punched card means, but as equipment was improved, the cost kept decreasing until it became obvious in 1957 that a practical magnetic system could be designed.

Several computers were available. Feasibility studies were made and several proposals from computer manufacturers were evaluated with help from the Armour Research Foundation during the early part of 1958. Each computer had its advantages and disadvantages but all of them had one thing in common--no satisfactory solution to the problem of printing labels. While 10,000,000 small magazine labels can be printed on standard tabulating type computer output printers, this solution is a compromise at best and is very expensive.

The label printing operation itself is an ideal application for special purpose equipment. The work load is almost constant from week to week, the form of the label shown in Fig. 1 is constant for long periods of time, and the paper on which the label is printed seldom changes. Therefore, rather than bend the system around to fit existing equipment which was designed for another purpose, separate proposals were solicited for new equipment designed primarily to print labels.

This equipment, of course, must cooperate with other elements in the overall system in order to achieve optimum results. The basic source of information for the label printing job is the master file of subscriber data. Activity to the file in the form of new subscriptions, renewals, changes of address, etc., are accumulated on a weekly basis and the file is updated just before the labels are to be printed. This insures that the labels are as "fresh" as possible. The arrangement of the file is geo-alpha, each record being identified by a code that is derived from the subscribers name and address. The code is constructed with the city-state portion major, followed by last name, first initial, street name, and house number.

Optimum file maintenance procedures dictate that certain inactive subscriber records be retained in the file for several months, including recently expired subscriptions and the old address in a change of address activity. An analysis of the list shows that on the average there is about one inactive record for every three active records.

On magnetic tape the records will appear as follows (Fig. 2):

(a) They will be grouped in blocks of somewhere between 10 and 20 records per block with 14 being the most likely choice in order to achieve an approximate balance between tape time and compute time.

(b) In order to conserve tape and thereby to reduce process time, the city-zone-state portion of a subscriber's name and address, and the code for that city, will not be carried in the individual record but in a header record that appears at the beginning of each new city-zone-state grouping.

Header record and subscriber record formats are similar (Fig. 2b and 2c) and conform to the requirements of the IBM 7070 computer on which the file will be updated. Statistical information which is relatively constant is placed at the beginning of the record while, variable address information is placed at the end, allowing maximum conservation of tape. In other words, the record is terminated with the end of the address information and short addresses mean short records, long addresses mean long records. Specifically:

(1) The record always begins with a delta.

(2) The next character, called the indicative character, identifies the record as active, inactive, or header, as well as a special record classification called Suppress Header which is used for subscriber records that contain special city-state information, thereby, making the header unnecessary when printing labels.

(3) A second delta signals the beginning of the alphabetical portion of the record.

(4) After the second delta, the first 16 digits are part of the code that identifies the subscriber record. (The next four digits are ignored.)

(5) The name line begins with the 21st digit. The first address line begins with the 41st digit. The second address line begins with the 61st digit.

(6) The record may end with a record mark symbol in any multiple of five character positions after the twentieth position, or with no record mark if the record is a maximum length record.

(7) The first 10 digits following the second delta of the header record are part of the code line.

(8) The city-state line always begins with the 11th character.

Because of the importance of correctly identifying the header record, several checks are available in addition to the indicative character.

(1) The first character following the second delta of the subscriber record will always be the initial letter of one of the months of the year, and the first character following the second delta of the header record will never be one of these eight characters.

(2) The header information following the second delta will always be exactly 30 characters plus a record mark word, no more and no less.

In addition to the format characteristics listed above, there is available a programmed signal in the form of a special record called an "End of Roll" record. This record appears to the printer as an active record and the information contained in it, roll identification, number of subscribers in the roll, etc., is printed as though it were a label. It's distinguishing characteristic is a series of five Z's following the second delta. This is a signal to the printer to stop printing on that roll and to start a new roll. As a check on the performance of the printer, a count is kept of the active records going through the printer and this count is printed beside the computer count shown in the "End of Roll" record.

This arrangement of records is advantageous from the file maintenance standpoint, and it also makes feasible the printing of labels directly from the master file. The manner in which the problems of printing control and editing were solved is described in the following sections.

General Description of Label Printer System

A. B. Dick Company undertook the task of developing and constructing a label printer system meeting TIME Inc.'s requirements. From the outset this system was conceived to be a production machine fulfilling a vital role in the daily production of magazine address labels. As such, it has been designed to operate directly from the magnetic tape master file records without the necessity of using valuable computer time for the preparation of a pre-edited printing tape. It produces and delivers completed address label rolls ready for use by the magazine printers in their standard, label-affixing machines.

The printer operates at a basic rate of 6100 characters per second. This rate increases to approximately 7500 characters per second during the key line operation. These printing rates permit 36 labels per second to be produced and, with the automatic paper processing devices incorporated, permit the machine to produce five million labels per week during a standard 40 hour shift. Thus, two machines adequately meet the printing load faced by TIME Inc. with either machine capable of turn-

ing out the entire production if operated more than a single shift. Each machine provides standby service for the other.

To achieve this function requires two major units as shown in Fig. 3. The first, or lefthand one, of these units, 1) reads the magnetic tape records, 2) edits the information read, 3) stores the printing information until used by the printer, and 4) formats the printing data before transmission to the printer. This unit is called the Tape Reader-Buffer Unit, or TRBU.

The second unit, the one on the right, converts the digital codes received into printing signals and produces the magazine address labels. This unit is the Videograph Address Label Printer. This unit operates continuously with provisions for automatically changing paper supply rolls and for cutting off completed address label rolls and switching to a new spindle in response to input control pulses. These cutoff and transfer operations proceed without operator intervention and without machine stoppage.

The Videograph Process

Before proceeding with a description of the address label printer consider briefly the printing process itself. Fig. 4 and 5 diagrammatically illustrate this.

As shown in Fig. 4, a specially constructed cathode ray tube produces an electrostatic image on the treated surface of a moving paper web. The face of this electrostatic printing tube (or EPT) consists of a matrix array of small, closely-packed wires extending from the inner vacuum through the glass to the outside. By directing the electron beam to a point on this array and turning it on, a current is caused to flow through the wires selected to the surface of the paper, producing a charged area on the paper that is capable of later development. Thus, through control of the deflection signals, and suitable intensity modulation of the electron beam, any desired electrostatic image may be produced within the boundaries of the wire matrix array.

This printing process requires two characteristics of the paper. First, the surface must be capable of accepting and retaining an electric charge. This is accomplished by coating the surface with a thin layer of dielectric material. Second, the base, or body of the paper must be conductive to create a suitable ground plane near the imaged surface. This is accomplished by impregnating the paper with suitable materials. The resulting paper, therefore, does not depend upon photoconductive or other unusual properties, and is, incidentally, easily and economically produced.

Following the printing process, the latent image is developed as shown in Fig. 5 by dusting with a mixture of iron powder and a black toner dust.

The toner particles, in such a mixture, adhere to the negatively charged regions on the paper surface and are conveyed with the paper to the fixing station. The iron particles serve to impart a charge to the toner causing them to more readily adhere to the image. The iron becomes, through the charge interchange, oppositely charged and is repelled by the imaged areas. The iron, therefore, serves only as a carrier for the toner and is not used up in the process of development.

Following development, application of heat to the developed image causes the thermoplastic toner particles to fuse and attach themselves to the paper surface. The image is thus made permanent and will not smudge as a result of later handling.

Character Generation

In order to use the electrostatic printing tube for the printing of alphanumeric characters, some means must be provided for the conversion of digitally coded information into video-type signals. Many approaches were possible due to the flexibility of the printer tube.

The means selected utilizes a monoscope tube such as shown in Fig. 6. This tube is basically a cathode ray tube with an electrostatic deflection system. At the screen end of the tube a circular aluminum disc replaces the phospher as the target for the electron beam. Immediately in front of the target is a somewhat more positive electrode for the collection of the secondary electrons emitted from the target.

Operation of the monoscope depends upon the secondary emission characteristics of the target surface. When the beam strikes the aluminum surface of the target, a secondary emission ratio of approximately two results. This causes an electron current to flow from the external circuitry into the target. This condition would be reversed should the surface be covered with a carbon-based ink having a secondary emission ratio of somewhat less than one.

The target for the monoscope used contains a square array of 64 characters on its front surface. Each character is etched into the aluminum surface. The remaining surface is covered with a carbon-based ink to form the background. As the primary beam scans a single character in a raster-like manner, the resultant current into the external circuit represents a video signal describing the character. By scanning the printing tube with an identical raster and intensity modulating the beam with the video thus produced, an electrostatic image of the character on the monoscope may be created on the videograph paper.

Character selection signals originate in a highly stable circuit generating one out of eight voltage levels

for both the horizontal and the vertical deflection plates. An input, six-bit digital code, representing the desired character, controls the selection circuits in such a manner that the first three bits of this code select the vertical position, and the second three bits select the horizontal position of the character to be scanned. By this means, the positioning of the characters on the monoscope target decodes the input and eliminates the need for further logical circuitry.

Since no cutouts or stencils of the letters are involved, and since the plates may be easily produced from original artwork by photo-etching techniques, almost any type font may be chosen. In fact the type font may be varied by switching tubes to meet special printing needs in the future.

Electronic System of Printer

Using the printing process and the character generator described above, the electronic system of the Videograph Label Printer produces an electrostatic image of the labels on a continuously moving web of paper. The operation of this system is shown in the block diagram of Fig. 7.

Digital signals from the TRBU trigger the character generator once for each input character. In this unit the selection circuits select the character to be printed and the raster-scan circuit causes this character to be scanned. The same raster signals are transmitted to the vertical deflection coil of the printer tube and through the horizontal deflection circuitry to the horizontal coil. In addition, an unblanking signal and the resultant video signal modulate the beam of the printing tube so as to produce the latent image of the scanned character.

The raster signals used for the character scan consist of a 200 kc vertical sine wave that is continuously applied to both the monoscope and the printer tube. During the generation of a character, a 100 microsecond horizontal sawtooth signal produces the slower horizontal scan in the monoscope and controls the horizontal movement of the beam in the printer tube. The resultant horizontal signal for the printer tube, therefore, consists of a stepwise signal combined with a sawtooth. The stepwise component provides the character by character separation, and the sawtooth controls the writing of individual characters. The horizontal circuitary is reset for the beginning of each line by the pulse supplied from the TRBU.

The First Line Pulse modifies the horizontal deflection circuitry so as to compress the key line as shown in Fig. 1. This compression occurs as a result of reducing the magnitudes of the horizontal sawtooth and step signals in the horizontal deflection circuitry. The compression circuits are reset by the next "Beginning of Line" pulse from the TRBU.

The motion of the paper controls the printing of labels so that each label will be accurately centered between holes. The paper control circuitry, therefore, produces a label demand pulse and transmits this pulse to the TRBU to initiate the printing of each new label. At the end of each label roll, as determined by information from the master file tapes, an "End of Roll" pulse is sent to the printer from the TRBU. This causes an interruption in the transmission of the label demand pulses for 2.5 seconds in order to create a blank strip of paper between label rolls. At an appropriately later time the web is cut and caused to start rewinding on a separate spindle.

Paper Processing System

The mechanical paper processing system consists of three basic modules. These are shown in Fig. 8. From left to right these are 1) the Unwind module, 2) the processing module and 3) the rewind module.

The unwind module contains two supply rolls of paper, a flying splice mechanism and a supply elevator. The upper supply roll normally feeds paper into the printer while the lower roll is in standby. When the upper roll is depleted, as detected by a feeler arm and cam, the splicer stops the flow of paper. During the splice operation, the supply elevator supplies paper to the printer so that actual printing does not stop. After the paper motion stops at the splicer, the old web is cut off and the leading edge of the new web attached to the trailing edge of the old. The web is then released and the new roll commences to provide the system. The entire unwind mechanism then revolves, bringing the new roll to the top and placing the empty spindle on the bottom for loading of a new roll of paper for standby.

In the central, or processing, module, the paper passes in front of the EPT where the electrostatic image is created in its left surface. It then proceeds through the developer and around the fuser unit. From this point it proceeds to the rewind module. The operations of the printer tube have already been described. The developer unit conveys the iron powder-toner mix to the paper surface to develop the image. The fuser uses rf to create a corona discharge to the paper surface causing a very rapid surface heating for the fusing operation.

In the rewind module, there is a primary paper drive-punch unit, a rewind elevator and a six spindle rewind assembly. The primary drive-punch unit uses hardened dies in the drive drum meshing with soft steel punches on an adjacent drum. These elements of a rotary punch are on one inch centers and create the holes in the label strips as shown in Fig. 1. In addition, this unit generates pulses in synchronization with the hole punching operation to initiate the printing of each label. Since the distance from this drive unit back to the printer tube remains fixed, this exactly locks the

position of the printed labels with respect to the holes punched.

The paper web normally travels over the left most spindle of the rewind station and is wound on the lower left spindle as shown. At the end of a roll, a cutting knife opposite the left most spindle cuts the web and a roller, together with a vacuum in the spindle, causes the rewinding to commence on the left most spindle. The mechanism then rotates 60° counter-clockwise, returning the system to its original state. The completed roll is conveyed to the lower right position where it can be removed by the operator and a new core placed on the spindle.

Operation of the Tape-Reader-Buffer Unit

From the preceding it may be seen that the printer operation proceeds in a most conventional manner but at a very high rate of speed. It could just as easily be recording data directly from the computer as an "on line" device or could be printing the text of the congressional record. However, the TRBU converts this conventional printer to a TIME Inc. printer system. This unit contains the editing, logic, and buffer circuits necessary to permit this printer to be driven directly from the master file records produced by the computer.

With the tape organization described previously, the TRBU's input control circuitry can, as shown in the block diagram of Fig. 9, edit the incoming information from the tape. Information read a block at a time from tape 1 in the figure, passes sequentially through S1 and the two character storage to S2. While in this limited storage, the characters are examined by the input logic control circuits in order to check the parity of the characters as read and to control the setting of the switch, S2, at the input to the main buffer storage unit.

Switch S2 normally is in the lower position so that no information enters the buffer. When the first delta symbol is received from the tape, the input logic circuit examines the numerical digit immediately following it. If the record is a non-printing subscriber record, the control circuitry leaves S2 in the open position throughout the remainder of that record. If the record is that of an active subscriber, or a header, S2 connects with the two character storage input at the second delta symbol in the record and enters into storage the second delta symbol together with the portion of the record following it. S2 immediately returns to the open position at the instant the next delta symbol, signifying the start of the next record, is detected by the input logic circuitry.

A parity error detected by the input logic circuitry is counted and added to the total count of such errors stored in the 144 character buffer unit. In addition, S2 and the special symbol generator inserts the code for a question mark for the character misread. Normally the detection of a parity error does not stop the machine since a few such errors will still permit delivery of the magazine. A label is a very redundant item, especially when coupled with the postman's memory. Such an error will stop the machine, however, when its frequency exceeds a present amount, or when it occurs at a critical point where confusion is likely in the handling of the header.

The 2184 character main storage buffer has a minimum capacity of approximately 26 combinations of header and subscriber records. When the number of records in storage drops below approximately 12, the input tape drive control causes the tape unit in use to read the next block. By this means the main buffer is kept loaded and the tape reading rate is under the control of the printer and its operations. At the end of the first tape, S1 automatically switches to Tape 2 and operation continues as before while the operator reloads Tape 1 with a fresh tape.

The time required to read a block of information from the input tape is approximately 50 milliseconds. The time to empty the main buffer by printing operations is approximately 300 milliseconds. Thus, approximately 6 blocks of information could be read from the input tape before the buffer would be emptied by the printer. As a result, there could be a series of up to 84 consecutive inactive records without causing the printer to print a blank label because of insufficent input information. However, no useful information is lost as a result. Operational studies of the master files at TIME Inc. indicate that such a series of inactive records will almost never occur.

At the output of the 2184 character buffer, S3 and S4, under the control of the output logic circuitry, determine the sequence of characters coupled to the printer. For the printing of a subscriber address, the data denoting the specific subscriber constitutes the subscriber record and is obtained for the printing operations directly from the main buffer. For this the switch S3 is positioned in its center position. The last seven characters of the key line and the last address line are associated with the city-zone-state and are part of the header record as stored in the 144 character buffer in the block diagram. For the printing of these characters, S3 and S4 are in their lower-most positions. This causes the output of the smaller buffer to be delivered to the printer and at the same time to circulate back into the input of the buffer so that the same header information will be available for the printing of the next label. The two spaces and the "Beginning of Town" marks are supplied from the special symbol generator with S3 in its uppermost position. The "Beginning of Town" marks are only placed on the first label in each new city-zone-state series.

The delivery of data to the printer commences with the receipt of a "Label Demand" pulse from the

printer and continues at the character writing rate of 6100 characters per second until the label is completed. Completion of a label is normally determined in one of three possible ways. These are:

(1) detection of a delta symbol out of the main buffer.

(2) detection of a record mark out of the main buffer.

(3) completion of a full label character count by counters in the output logical circuitry.

The occurrence of any one of these causes the output to switch to the 144 character circulating buffer for the printing of the city-zone-state line.

In the event that other than a delta symbol terminates the printing, the main buffer advances to the next delta symbol. The digit immediately following this delta is then examined by the output logic circuitry to determine the type of this record. If this record is a header record, S4 switches to its upper position and the new header is rapidly transferred to the circulating buffer to replace the old. If this record is a subscriber record, the operations stop until receipt of the next "Label Demand" pulse from the printer. If this record is an "End-of-Roll" summary record, the output logic circuitry causes the next record, a summary for the label roll just completed, to be printed upon receipt of the next "Label Demand" pulse and also:

(1) causes the error and label count registers in the 144 character buffer to be printed with the summary record, and

(2) sends an "End-of-Roll" pulse to the printer following the printing of the summary record.

The other signals shown in Fig. 9 are for the purpose of controlling the printer's operation in the manner previously described. It may be mentioned at this point that the clock controlling the printing rate is part of the output logic circuitry of the TRBU and for this reason the printer is under control of the TRBU once printing commences.

Summary

Little has been said of the built-in checks in both the TRBU and in the printer to assure the operator that the entire system has and is working properly. Space does not permit a discussion of these features. Suffice it to mention that they do exist and will sound alarms or stop the machine depending upon the severity of the malfunction. This system differs, however, from many other systems in regard to the treatment of errors. It is essential to the welfare of the magazine that each subscriber obtain his regular copy. It is not necessarily essential that every label be perfect to obtain this result. Thus, certain errors are permitted in the operation of the system and a count is maintained of the number of such occurrences. The system stops only if the number of such events becomes too numerous, or until the operators feel that the system should be serviced.

The principle advantages of the Videograph printing and character generation processes are their speed and flexibility. Neither the printing tube nor the character generator operate near their limits in producing the 36 labels per second. Each could function at twice the present rate although little would be gained in TIME Inc.'s application by doing so. Their label printing requirements can be met, together with anticipated growth, with two machines operating on a single shift and a portion of a second shift.

The TRBU makes possible the operation of the label printer direct from the information contained in the master file records. This eliminates the need for using computer time in the preparation of edited tapes for printing purposes. This saving is an operating cost saving and its magnitude more than offsets the additional cost of the TRBU.

The flexibility of the system and the process makes possible the compression of the key line, quick changes in type font or input digital coding, easy reversal of the printed image for heat transfer applications, and a minimum of circuitry capable of rapid adaptation to future changes and applications. Thus, the system developed amply meets the requirements set forth by TIME Inc. and, in addition, provides them with a system capable of expansion and modification to meet their future needs.

Acknowledgements

As with many modern developments, the Videograph process and, specifically, the Videograph Label Printer System described in this paper, has been the result of a cooperative effort on the part of many companies and individuals. In particular, mention should be made of the contributions of Stanford Research Institute in the development of the process and many of the circuits under the sponsorship of A. B. Dick Co. and Telemeter Magnetics, Inc. for the design and construction of the TRBU.

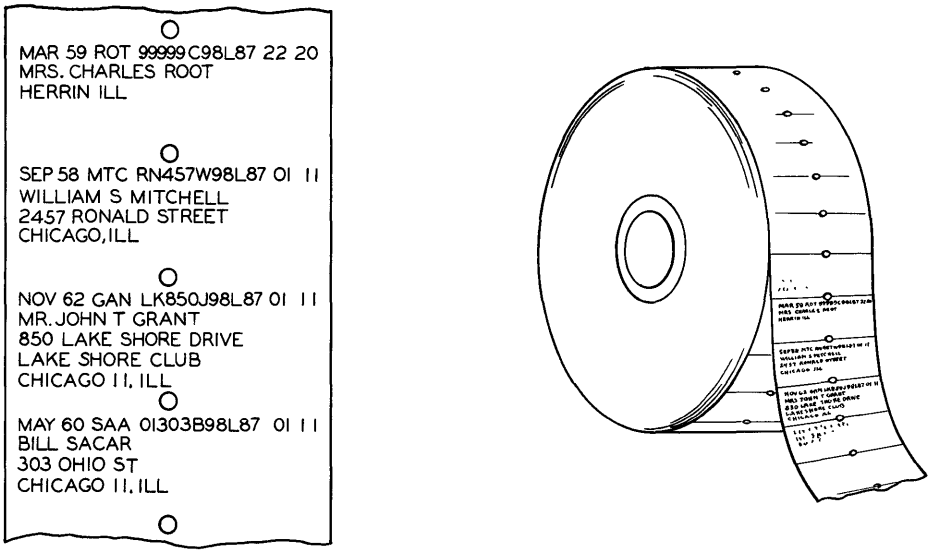


Figure 1
FORMAT OF MAGAZINE LABELS

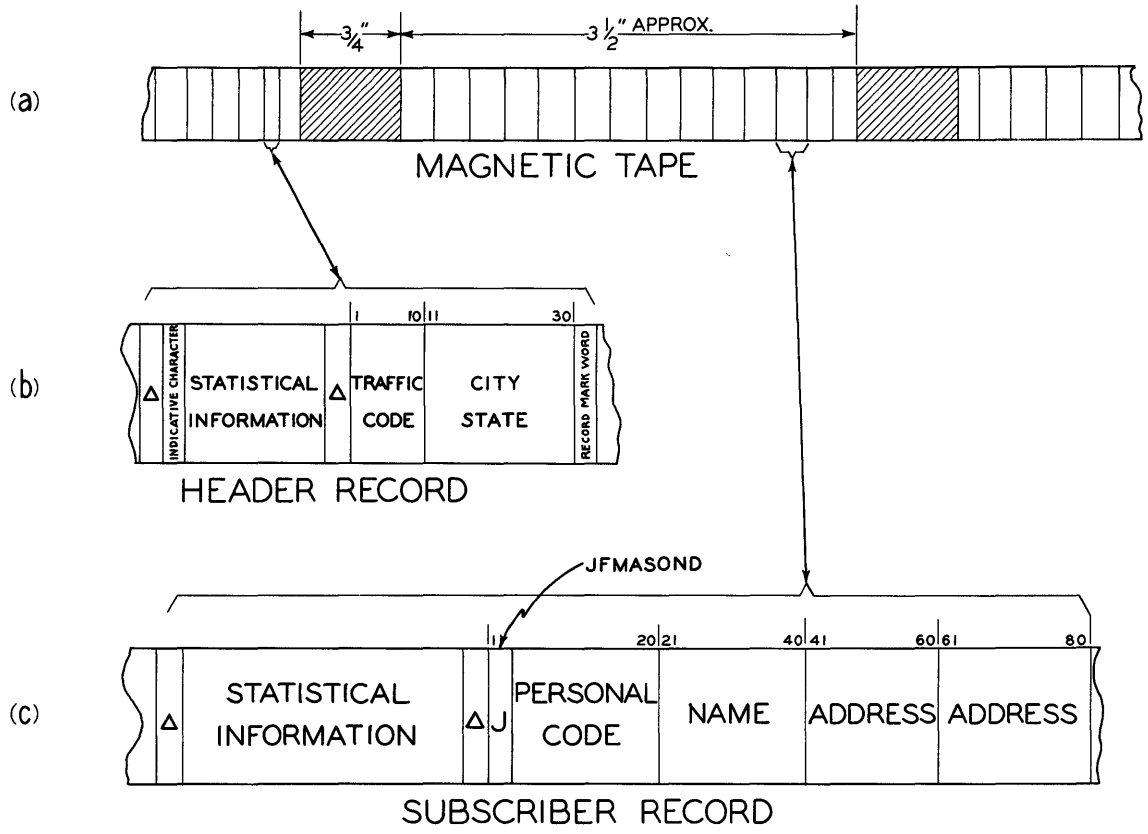


Figure 2
MAGNETIC TAPE & RECORD ORGANIZATION

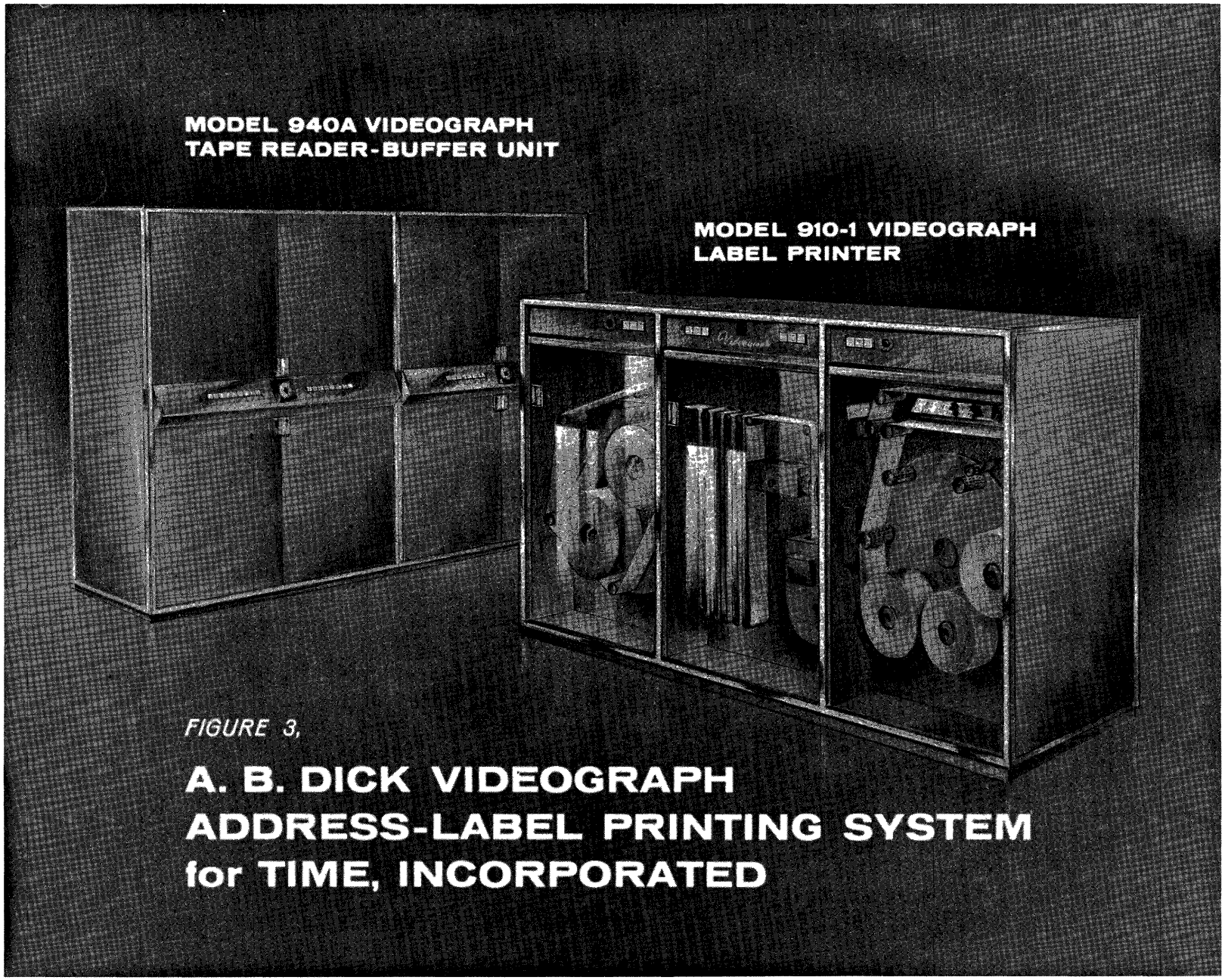


FIGURE 3,

A. B. DICK VIDEOGRAPH ADDRESS-LABEL PRINTING SYSTEM for TIME, INCORPORATED

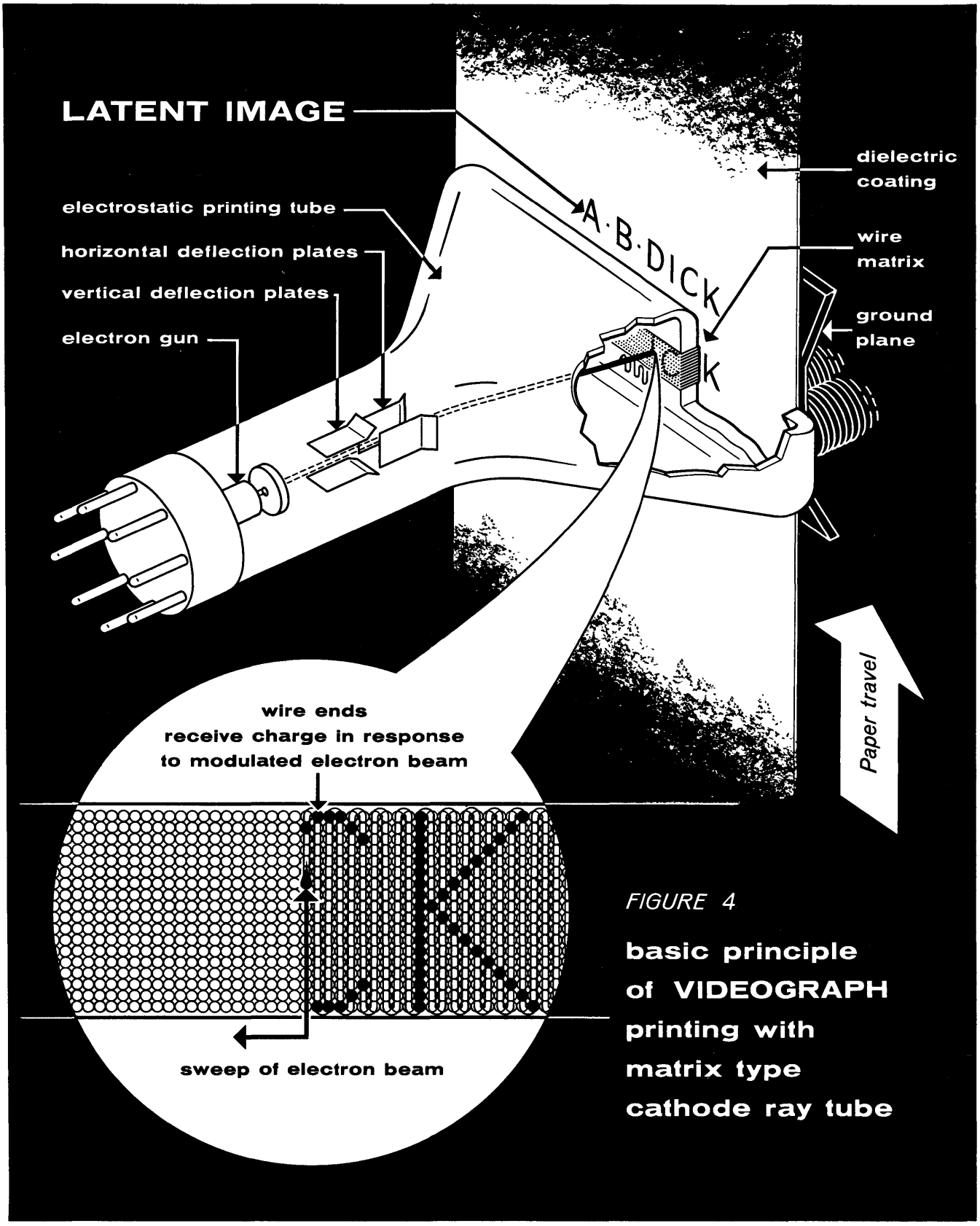
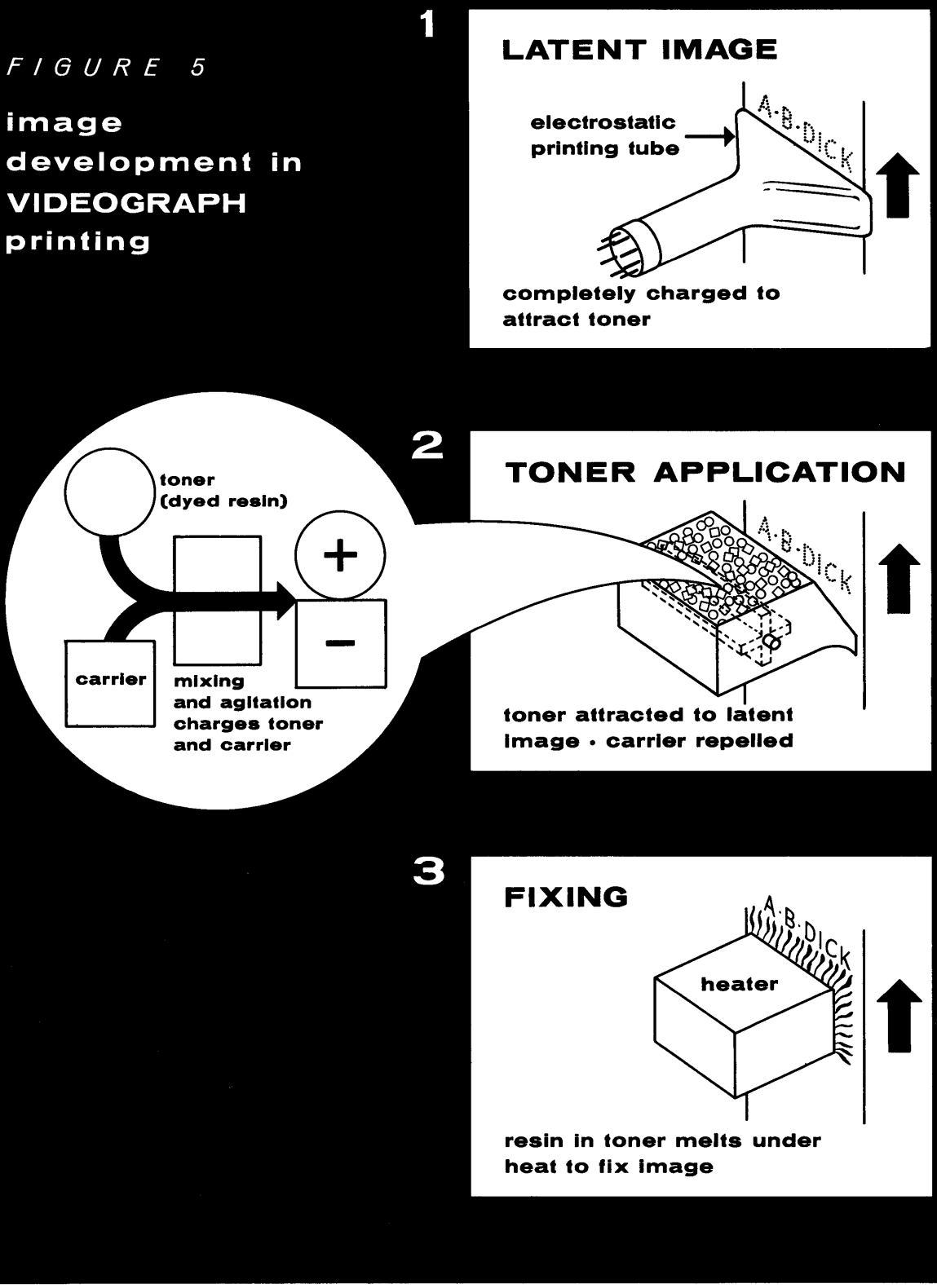


FIGURE 4

basic principle of VIDEOGRAPH printing with matrix type cathode ray tube

FIGURE 5

image development in VIDEOGRAPH printing



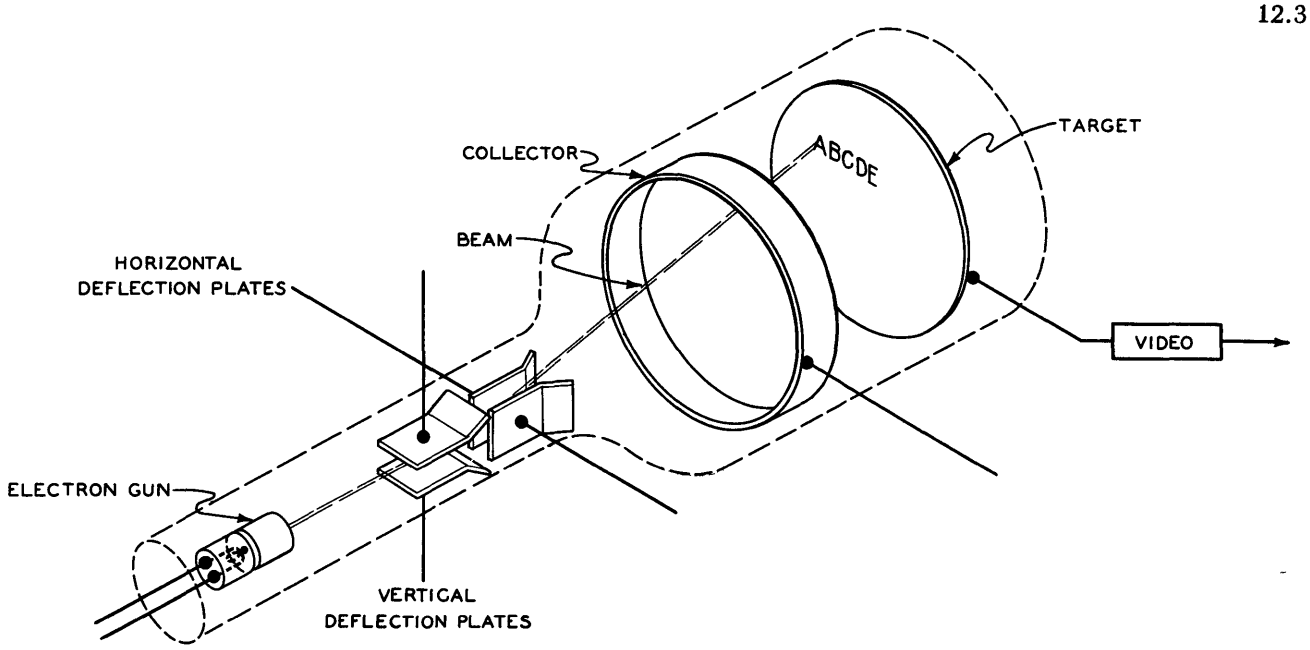


Figure 6
VIDEOGRAPH CHARACTER GENERATOR TUBE

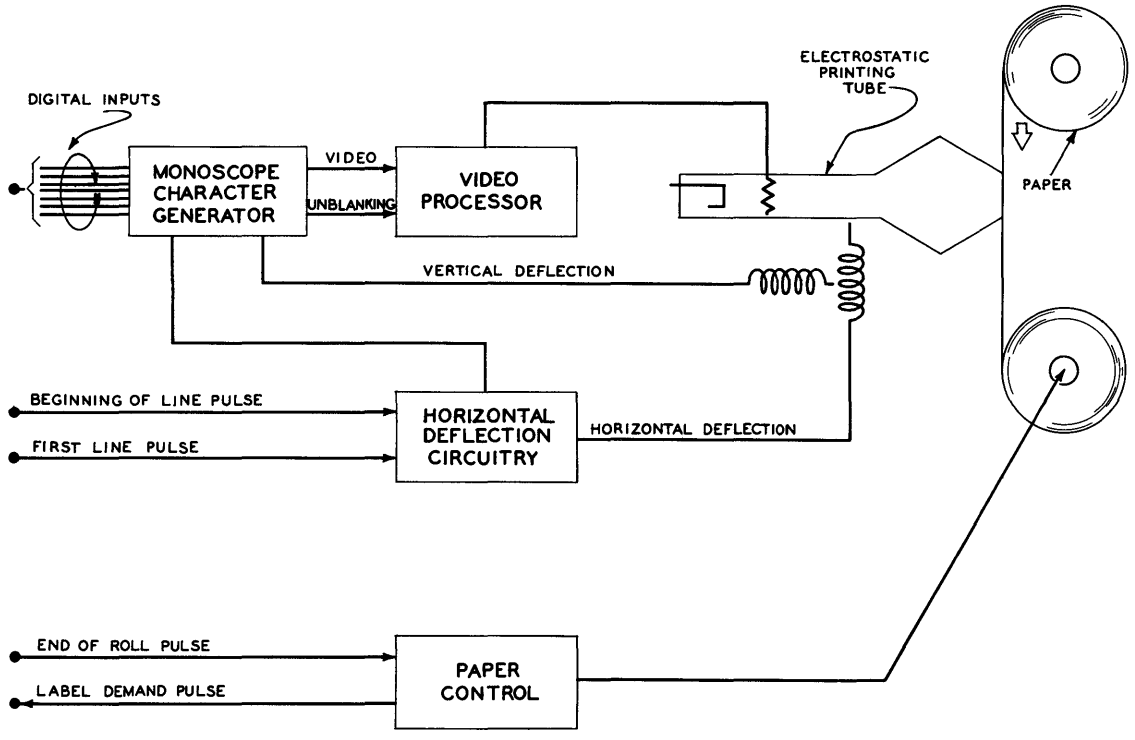


Figure 7
VIDEOGRAPH LABEL PRINTER-ELECTRONIC BLOCK DIAGRAM

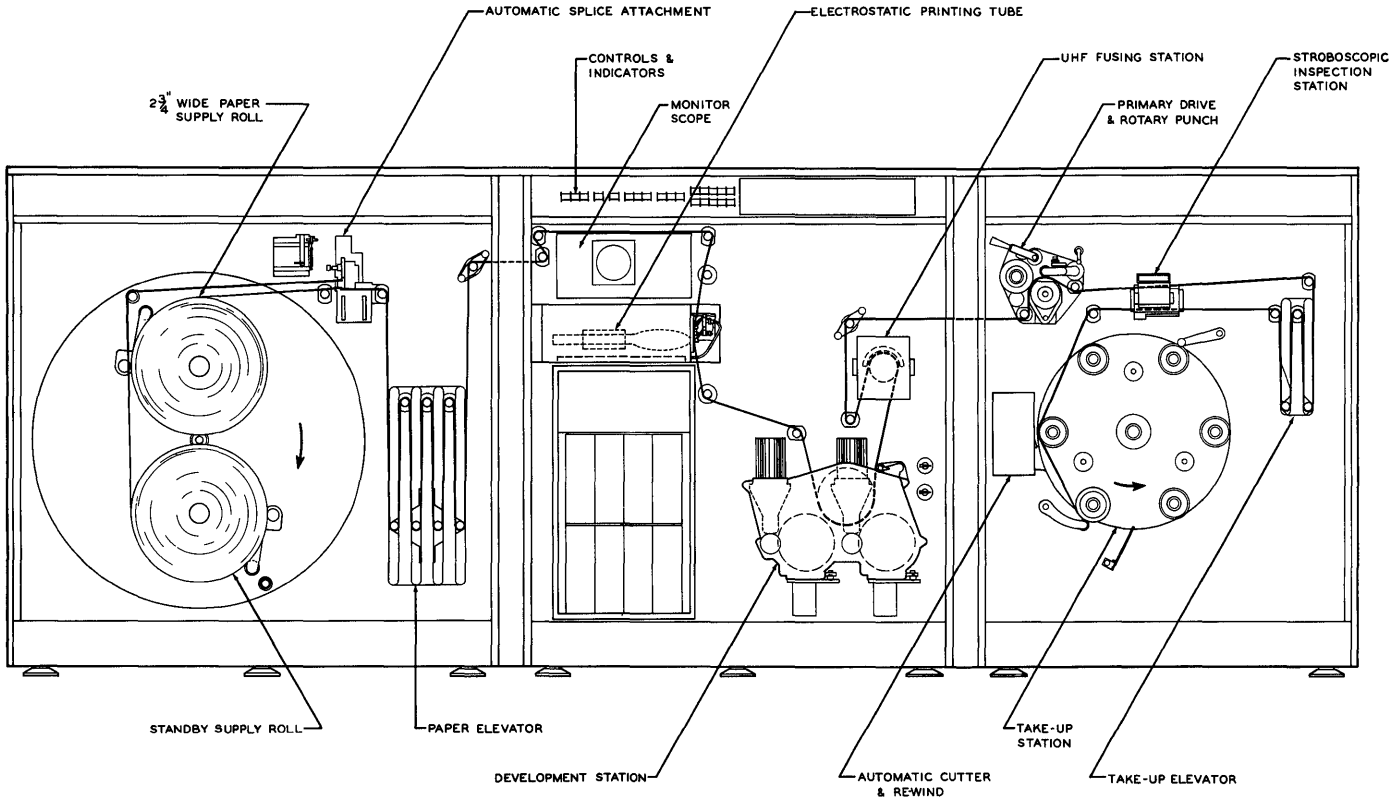


Figure 8

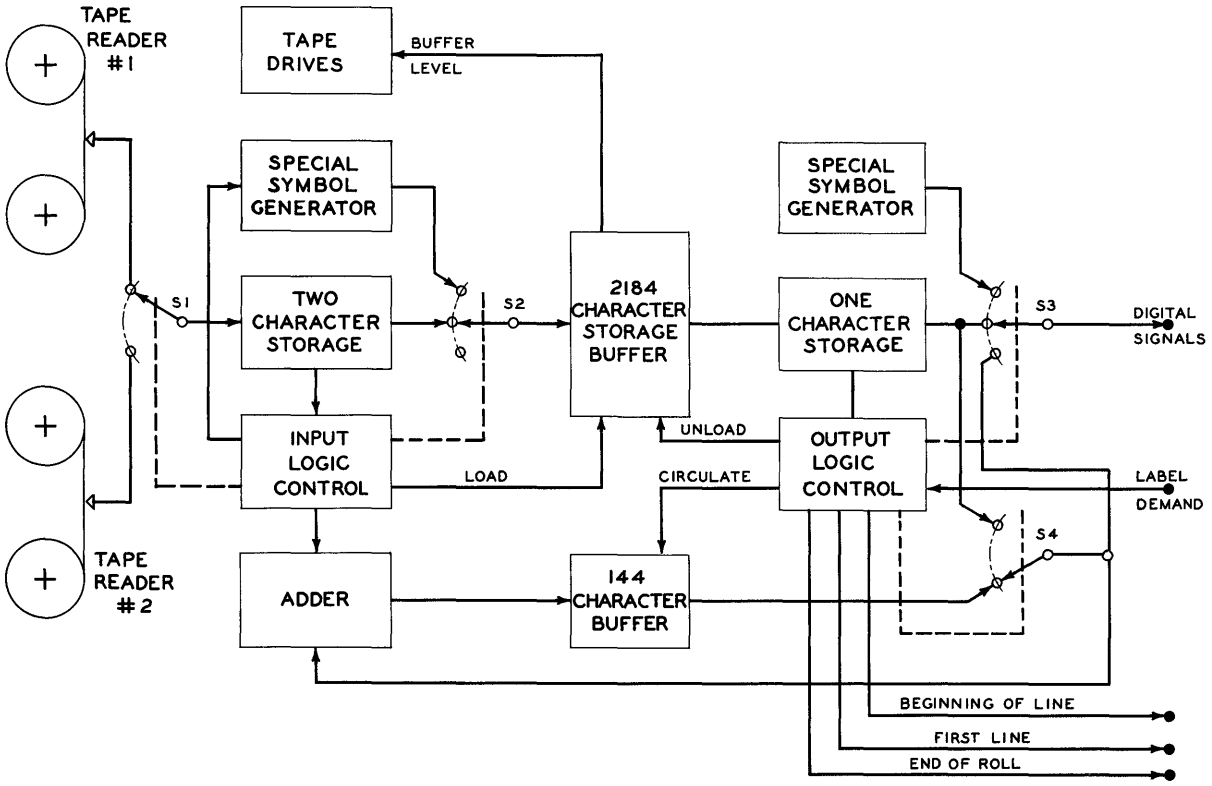


Figure 9
TAPE READER & BUFFER UNIT-BLOCK DIAGRAM