Course Outline - Day 1

- Overview

1. Introduction
2. Physical characteristics
3. Bus phases
4. Messages

- SCSI Commands

1. Commands for all devices
2. Commands for disk device
3. Command for tape device
4. Command for processor device

Outline -- Day 2

- SCSI 1 SCSI 2 Comparison

1. Physical characteristics
2. Asynchronous Event Notification
3. Contingent allegiance
4. Command queueing
5. Disk caching

- SCSI Devices

1. SCSI ICs
2. SCSI disks
3. SCSI tapes
4. SCSI adapters

- Comparing SCSI against IPI -- Guest speaker

Course Outline -- Day 3

- SCSI Features for Micro- and Mini-computers

1. Auto-configuration
2. Virtual disk
3. Write cache
4. RAID -- NCR ADP-92-01 SCSI Disk Array

- Opportunities

1. Operating system
2. Interconnect
3. Controller
4. Device

- Guest Speaker

Overview -- Introduction

- History

  FBA -- Fixed Block Architecture, IBM 1971

  SASI -- Shugart Associate System Interface, 1978

  SCSI -- Small Computer System Interface, 1982

  CCS -- Common Command Set, 1986

  SCSI 2, 1989

- Peripheral Device Bus

  Common electronic interface

  Peer to peer

  Initiator/target protocols

  Multiple hosts and devices

  Logical block address, hiding device geometry

  Inquiry command allowing auto-configuration

Specifications

- SCSI-1: ANSI X3.131- 1986
- CCS: X3T9.2/85-82, 1986
- SCSI-2: ANSI X3.131 - 1990

Overview -- Physical

- A cable -- 50 pin, shielded or non-shielded
- B cable -- 68 pin, shielded or non-shielded
- P cable -- TBD, 16 devices
- Single ended

  Odd numbered pins -- ground

  Even numbered pins -- 9 data signals, 9 control signals, 1 terminal power -- additional 5 ground pins, 1 reserved.

  SCSI 2, 18 or 36 data signals

- Differential

  Pair of wires for each data or control signal

  Pin 21 -- DIFFSENS

Physical -- Cable

- Length

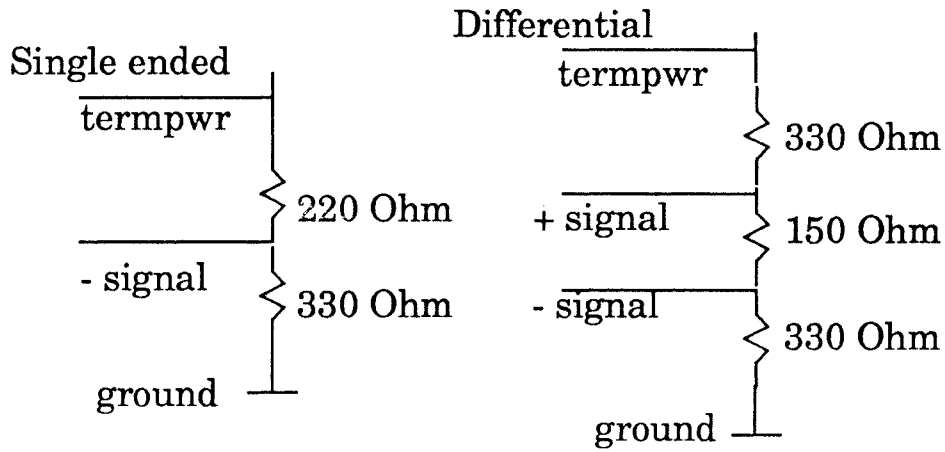
  Single ended -- 6 meters or 25 feet

  Differential -- 25 meters or 80 feet

- Terminal power

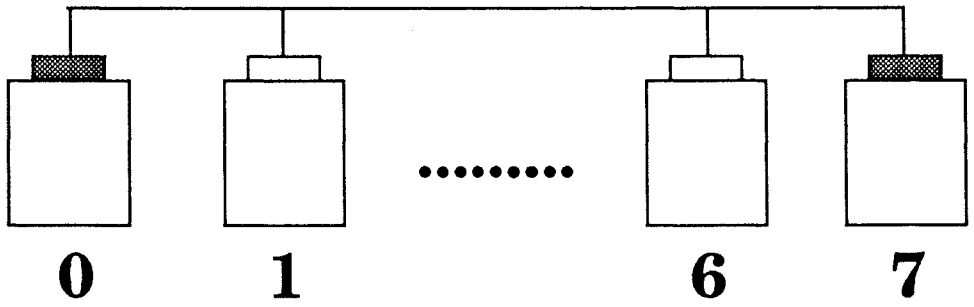
  Any device can supply terminal power

  (SCSI 2 -- Initiator must supply terminal power)



Overview -- Physical

- Up to 8 devices
- Either a target or an initiator



Overview -- Physical

- Control Signals

  RST -- Reset, bidirectional

  BSY -- Busy, bidirectional

  SEL -- Select, bidirectional

  ATN -- Attention, initiator to target

  ACK -- Acknowledge, initiator to target

  REQ -- Request, target to initiator

  CD -- Command/Data, target to initiator

  IO -- Input/Output, target to initiator

  MSG -- Message, target to initiator

- Data Signals -- D0 to D7 and PAR

  8 data signals + parity, bidirectional. 18 signals for B-cable.

  Most devices generate parity and optionally check parity

  Parity is required in SCSI 2.

Overview -- Configuration

- Eight devices on the bus
  - Each device has a unique ID, 0 to 7 -- 7 has the highest priority
  - Each device has up to eight LUNs (1 controller 8 devices)
  - Initiator -- initiates operations
  - Target -- services initiator commands
  - Anyone can be both an initiator and a target
- Single host -- single or multiple targets
  - Dumb host --has no ID, up to 8 target, no disconnection
  - Smart host -- has an ID, up to 7 targets
- Multiple hosts -- single or multiple targets
  - Sharing target(s) among hosts
  - Host to host communication

Overview -- Bus Phases

- Control Signals BSY, SEL, CD, IO, MSG

1. Bus states

   Bus Free -- BSY SEL = 00

   Arbitration -- BSY SEL = 10

   Selection -- BSY SEL = 01, IO=0. (Reselection, when IO=1)

   Bus busy -- BSY SEL = 10

2. Bus phases (BSY = 1) -- target controls bus phases

   MSG_IN (111) -- Target sends message to initiator

   CMD_OUT (100) -- Initiator sends command

   DATA_IN (010) -- Initiator reads data

   DATA_OUT (000) -- Initiator writes data

   STATUS_IN (110) -- Target returns command complete status

   MSG_OUT (101) -- Initiator sends message to target

   (Initiator to start message out -- Raises ATN and wait)

Overview -- Bus Phases

- Arbitration -- gaining control of the bus
- Selection

1. Initiator starts selection

   Arbitrating the bus prioritized by SCSI IDs on data signals.

   Selecting by asserting SEL and two ID bits on data bus -- own and target IDs. (In SASI, only the target ID is used.)

   Use ATN signal to let target enter message out phase.

   Use IDENTIFY message to pass LUN number.

2. Target Starts Reselection

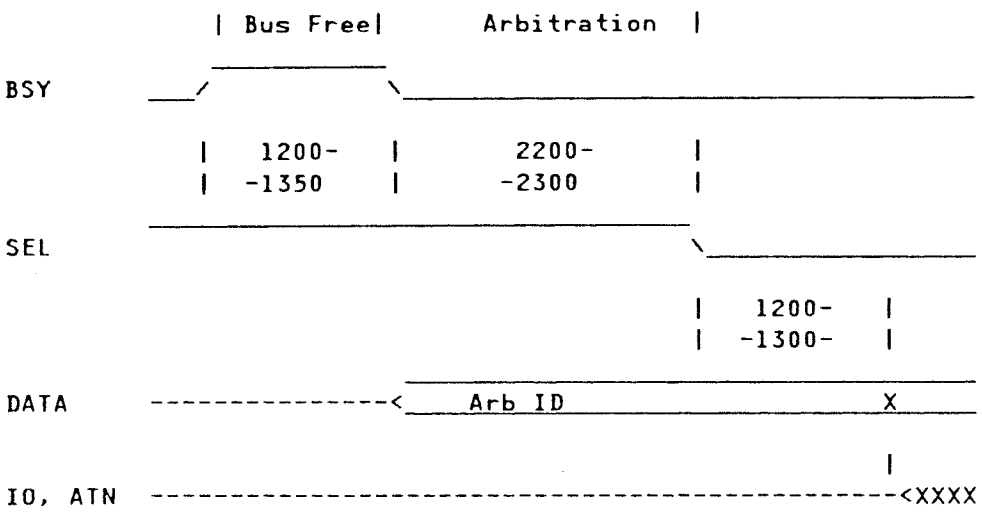
   Same as initiator selection except IO signal is set.

- Information Transfer

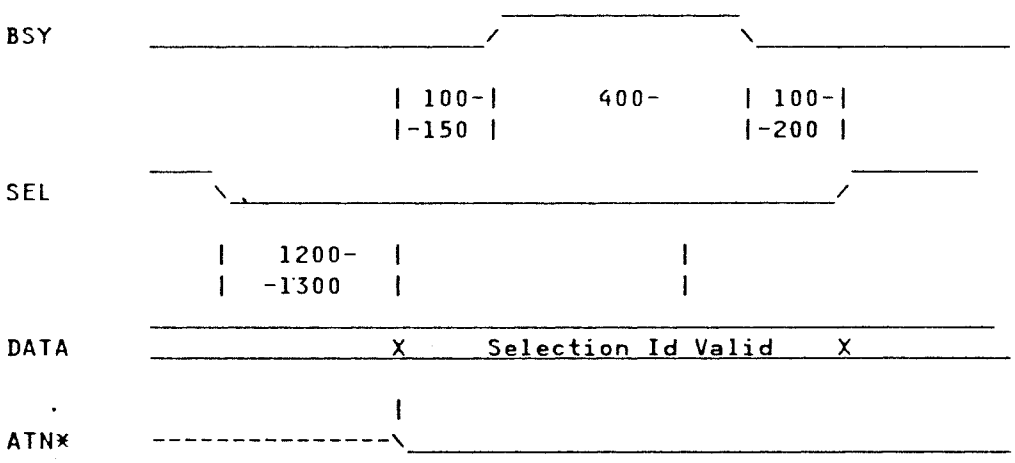
  MSG_OUT, MSG_IN, CMD_OUT, DATA_OUT, DATA_IN, STATUS_IN

- Command, message, data transfer -- REQ, ACK

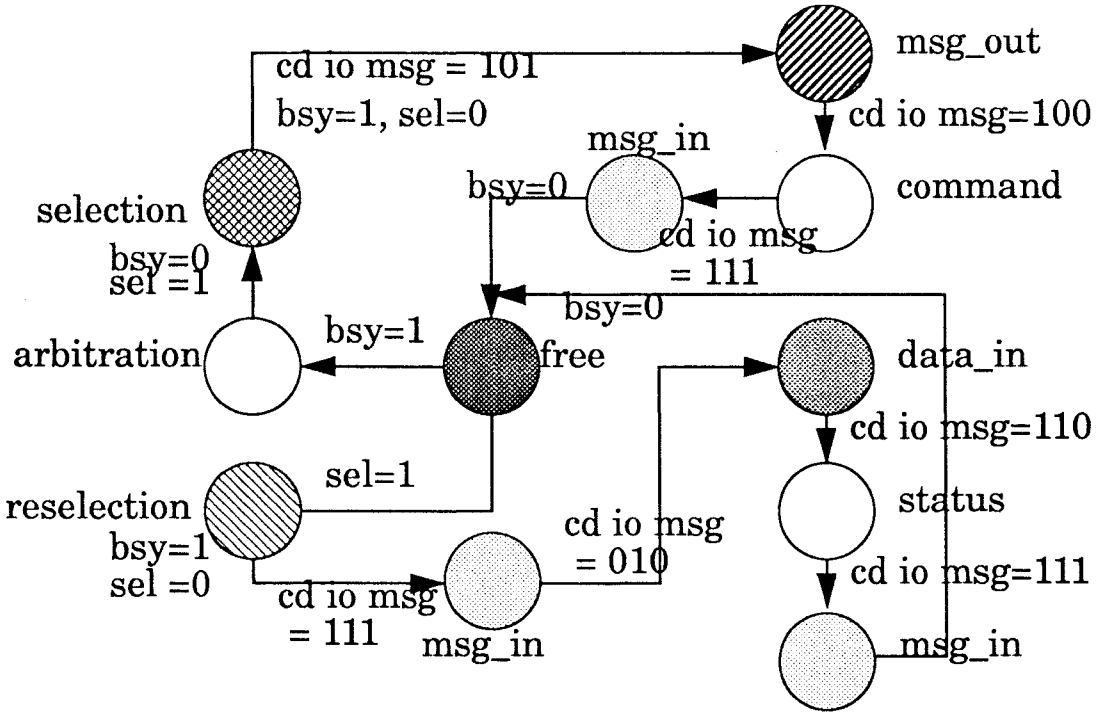
10.2.19 SCSI Arbitration



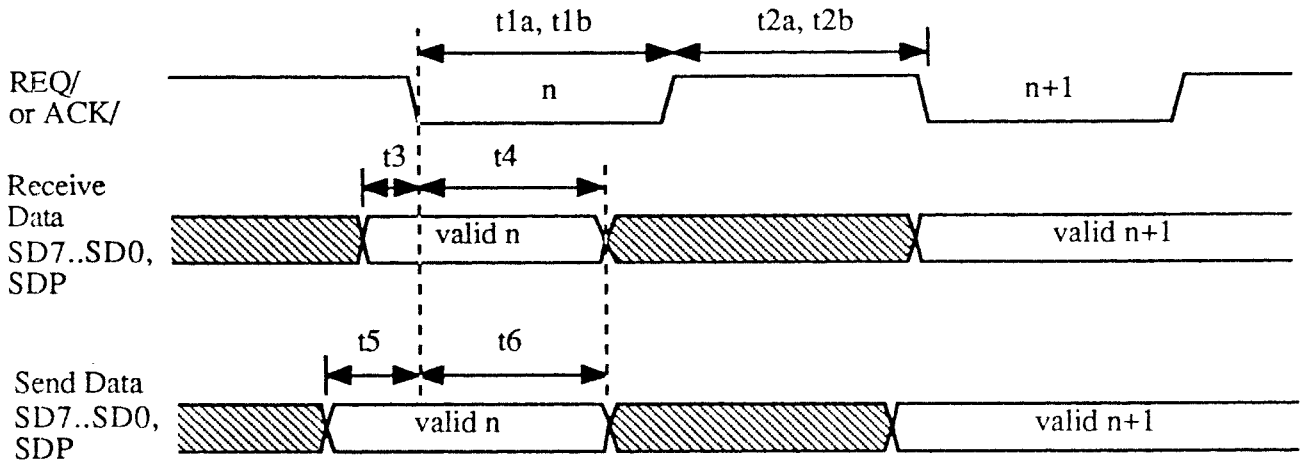
10.2.20 SCSI Selection (Initiator)



Overview -- Bus Phases



Initiator and Target Synchronous Transfers

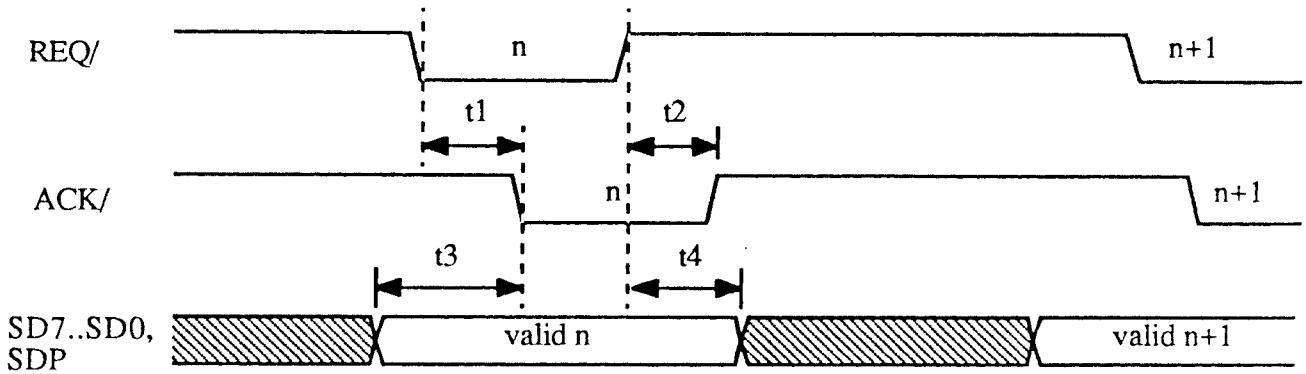


Initiator and Target Synchronous Transfers				
Symbol	Description	Min.	Max.	Units
t1a, t1b	REQ/ or ACK/ assertion pulse width	90	-	ns
t2a, t2b	REQ/ or ACK/ deassertion pulse width	90	-	ns
t3	Receive data setup to REQ/ or ACK/ asserted	0	-	ns
t4	Receive data hold from REQ/ or ACK/ asserted	45	-	ns
t5	Send data setup to REQ/ or ACK/ asserted	55	-	ns
t6	Send data hold from REQ/ or ACK/ asserted	100	-	ns

Fast Initiator and Target Synchronous Transfers				
Symbol	Description	Min.	Max.	Units
t1a	Receive data REQ/ or ACK/ assertion pulse width	30	-	ns
t1b	Send data REQ/ or ACK/ assertion pulse width	80	-	ns
t2a	Receive data REQ/ or ACK/ deassertion pulse width	30	-	ns
t2b	Send data REQ/ or ACK/ deassertion pulse width	80	-	ns
t3	Receive data setup to REQ/ or ACK/ asserted	0	-	ns
t4	Receive data hold from REQ/ or ACK/ asserted	18	-	ns
t5	Send data setup to REQ/ or ACK/ asserted	25	-	ns
t6	Send data hold from REQ/ or ACK/ asserted	35	-	ns

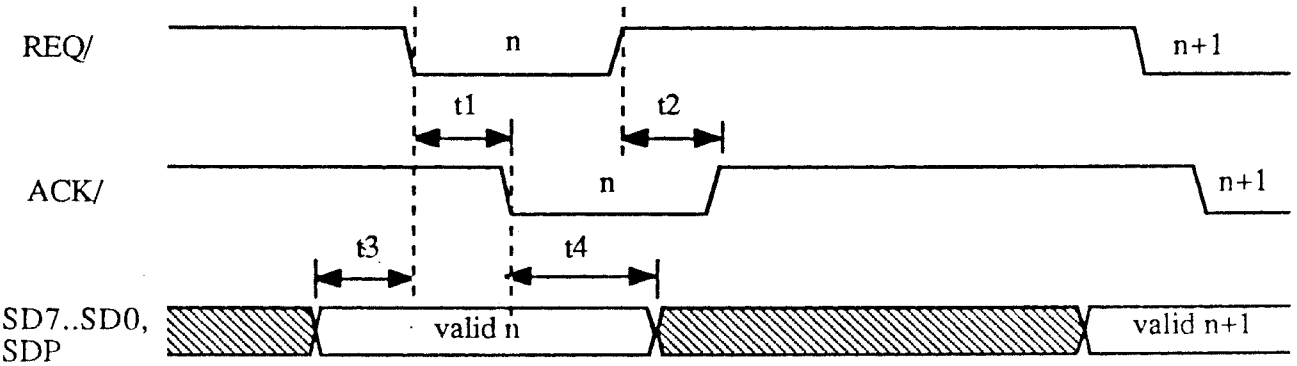
Note:
The 53C700 can transfer data at 6.25MB/sec for Synchronous Fast SCSI.

Initiator Asynchronous Send



Initiator Asynchronous Send				
Symbol	Description	Min.	Max.	Units
t1	ACK/ asserted from REQ/ asserted	10	-	ns
t2	ACK/ deasserted from REQ/ deasserted	10	-	ns
t3	Data setup to ACK/ asserted	55	-	ns
t4	Data hold from REQ/ deasserted	20	-	ns

Initiator Asynchronous Receive



Initiator Asynchronous Receive				
Symbol	Description	Min.	Max.	Units
t1	ACK/ asserted from REQ/ asserted	10	-	ns
t2	ACK/ deasserted from REQ/ deasserted	10	-	ns
t3	Data setup to REQ/ asserted	0	-	ns
t4	Data hold from ACK/ deasserted	0	-	ns

SCSI Command Control Block

- Command bytes
- Data area pointer
- Sense data area pointer
- Command, data, and sense data lengths
- Initiator generated status
- Target generated status
- Residue count
- SCSI flags -- suppress incorrect length, bus reset, device reset, etc.

Overview -- Bus Phases

- Unusual Bus Phase Changes

1. Device busy -- enter status phase after selection and identify message
2. Synchronous transfer negotiation after reset -- extended message after selection or after command
3. Device error -- enter status phase after command phase
4. Media error -- truncated data transfer
5. Power up attention -- skipping data transfer
6. Modify data pointer -- extra message phase before data transfer
7. Re-transmitting command for error recovery
8. Re-transmitting data for error recovery
9. Disconnecting before command phase
10. Disconnecting after data transfer
11. Disconnecting after status phase
12. Multiple disconnections during data transfers.

Overview -- Messages

00 -- Command complete, target to initiator

01 -- Extended messages, modify data pointer or synchronous transfer negotiation, multiple bytes

02 to 1F -- One-byte message

- 02 -- Save data pointer, target to initiator
- 03 -- Restore data pointer, target to initiator
- 04 -- Disconnect, target to initiator
- 05 -- Initiator detected error, initiator to target
- 06 -- Abort, initiator to target
- 07 -- Message reject, initiator and target
- 08 -- No Op, initiator and target
- 0C -- Bus device reset, initiator to target

20 to 2F -- Reserved for two-byte messages, 30 to 7F -- Reserved

80 to FF -- Identify message, initiator and target

Overview -- Messages

- Allow communication between devices
- Many restrictions

  "The first message sent by the initiator after the SELECTION phase shall be an IDENTIFY, ABORT, or BUS DEVICE RESET message. If target receives any other message it shall go to BUS FREE phase."

- Common usages

  Identify message after selection or reselection

  Synchronous negotiation after reset

  Save data pointer and disconnect

  Reselect and restore data pointer

  Modify data pointer for zero latency or error recovery

  Command linking or complete

One Byte Message

- Identify, send after selection or reselection

  Bit 7 -- always 1

  Bit 6 -- set to allow disconnection

  Bit 5 -- (LUNTAR, IO process, SCSI 2)

  Bit 4,3 -- Reserved

  Bit 2,0 -- Logical Unit Number (LUN)

- Save data pointers, used before disconnection

  Not needed if data transfer didn't start

- Restore pointers, used after reconnection

  Restore data and status pointers in the initiator (not really needed)

- Disconnect, used before freeing bus

  Allow others to share the bus

- Command complete

Overview -- Extended Message

- Message Format

  1st Byte -- 01

  2nd Byte -- Message length = total number of bytes minus 2

  3rd Byte -- Message code

  4th to *n*th byte -- Message argument

- Synchronous transfer negotiation

  Message length = 03, message code = 01

  Message arguments = transfer period and offset. (When offset = 0, it is asynchronous transfer)

- Modify data pointer

  Message length = 05, message code = 03

  Message argument = 4 bytes of modification in 2's complement

- ATN must be set until the last message byte

Command Format

6 Byte

Byte	Bit							
	7	6	5	4	3	2	1	0
0	Operation Code							
1	LUN				LBA (MSB)			
2	LBA							
3	LBA (LSB)							
4	Length							
5	Control Byte							

Command Format

10 Byte

Byte	Bit							
	7	6	5	4	3	2	1	0
0	Operation Code							
1	LUN							R
2	LBA (MSB)							
3	LBA							
4	LBA							
5	LBA (LSB)							
6								
7	LENGTH (MSB)							
8	LENGTH (LSB)							
9	Control Byte							

Command Format

12 Byte

Byte	Bit							
	7	6	5	4	3	2	1	0
0	Operation Code							
1	LUN							R
2	LBA (MSB)							
3	LBA							
4	LBA							
5	LBA (LSB)							
6	LENGTH (MSB)							
7	LENGTH							
8	LENGTH							
9	LENGTH (LSB)							
10								
11	Control Byte							

SCSI Command -- Group 0

- Bits 7-5 of byte 0 specify the command group
- Group 0 -- Six-byte command

Byte 0 -- Operation code

Byte 1 bits 7-5 -- Logical unit number

Byte 1 bits 4 to 0, byte 2 and 3 -- logical block number

Byte 4 -- additional parameter or data transfer length

Byte 5 -- control byte for linked command

Bit 7,6 -- Leftover from SASI

Bit 1,0 = 00, command complete message = 00

Bit 1,0 = 01, command complete message = 0A

Bit 1,0 = 11, command complete message = 0B

(21-bit logical block number addresses 2,000,000 blocks)

SCSI Command -- Group 1

- Group 1 or 2 -- Ten-byte command

Byte 0 -- Operation code

Byte 1 bits 7-5 -- Logical unit number

bits 4-0 -- command parameter

Byte 2-5 -- logical block number

(32 bit block address for 4,000,000,000 blocks)

Byte 6 -- reserved

Byte 7-8 -- transfer byte or block count*

Byte 9 -- control byte for linked command

(Same as group 0 command)

(2-byte transfer count, up to 64K bytes or blocks.)

SCSI Command -- Others

- Group 3 or 4 -- Reserved
- Group 5 -- Twelve-byte command

  Byte 0 -- Operation code

  Byte 1 bits 7-5 -- Logical unit number

  bits 4-0 -- command parameter

  Byte 2-5 -- logical block address

  (32 bit block address for 4B blocks)

  Byte 6-9 -- transfer length (4B bytes or blocks)

  Byte 10 -- reserved

  Byte 11 -- control byte for linked command

  (Same as group 0 command)

- Group 6 or 7 -- Vendor specific

  (Normally a ten-byte command)

SCSI Command -- Linked

- Multiple commands in a single IO process
- After command complete message (0A or 0B), target returns to command phase
- 0B message, tagged linked command completion, a reminder to initiator to generate interrupt.
- Save reselection overhead

  Slow initiator -- a few milliseconds

  Fast initiator -- less than 100 microseconds

  Integrated SCSI port -- may be a big saving
- Linking search command with read/write
- Very few operating systems use this function

SCSI Status

Bit	7	6	5	4	3	2	1	0	
Code									
00	R	R	0	0	0	0	0	R	complete
02	R	R	0	0	0	0	1	R	check status
04	R	R	0	0	0	1	0	R	condition met
08	R	R	0	0	1	0	0	R	busy
0A 10	R	R	0	0	0	0	0	R	linked complete
0B 14	R	R	0	0	0	0	0	R	linked cond met
18	R	R	0	1	1	0	0	R	reserve conflict
(SCSI 2, tagged queuing)									
22	R	R	1	0	0	0	1	R	cmd terminated
28	R	R	1	0	1	0	0	R	queue full

Command -- Device Type

0 -- Direct-Access, disk

1 -- Sequential-Access, tape

2 -- Printer

3 -- Processor, computer

4 -- Write-Once, WORM

5 -- CD-ROM

6 -- Scanner device

7 -- Optical memory

8 -- Medium-Changer

9 -- Communication

Device type is returned as the first byte of inquiry data.

Block Devices -- 0, 4, 5, 7

Stream Devices -- 1, 2, 3, 9

Command -- All Device

O	40	Change definition
O	39	Compare
O	18, 3A	Copy, and Copy and verity
M	12	Inquiry
O	4C, 4D	Log select, and Log sense
O	15, 1A	Mode select, and Mode sense
O	3C, 3D	Read buffer, and Write buffer
M	03	Request Sense
O	1C, 1D	Send diagnostic, and Receive diagnostic results
M	00	Test unit ready

O -- Optional M -- Mandatory

Command -- Change Definition

- A SCSI 2 command, acceptable in SCSI 1 mode

  A device is normally powered up in SCSI 1 mode.

- Modifies the operating definition

  Byte 0 -- Op code = 40, a ten byte command

  Byte 1 -- bits 7-5 = LUN

  Byte 2 -- bit 7 = 1, save the operation definition

  Byte 3 -- bits 6-0, definition parameter

  00 -- user current definition

  01 -- SCSI 1

  02 -- CCS

  03 -- SCSI 2

  04 to 3F -- reserved

  40 to 7F -- vendor unique

  Byte 8 -- parameter data length (vendor unique)

Command -- Compare, Copy

- Compare reads data from two devices

  Op code = 39, 10 byte command, byte 3-5 = parameter list length

- Copy copies data from one device to another

  Op code = 18, 6 byte command, byte 2-4 = parameter list length

- Source and target -- block or stream device

  Parameter list -- 4 byte header plus segment descriptor(s)

  Byte 0 bits 7-3 -- 00 --block device to stream device

  01-- stream device to block device

  02 -- block device to block device

  03 -- stream device to stream device

  04 -- sequential device to sequential device

- Segment Descriptors

  Steam device block size and block device address and count

Command -- Inquiry

- Request information regarding target device

  Code = 12, 6-byte command

  Byte 4 -- transfer length

- Standard Inquiry Data -- 36 bytes

  Device type and qualifier

  Device type modifier and RMB (removable media bit)

  ISO, ECMA, and ANSI versions

  Vendor identification

  Product identification

  Revision identification

  Vendor specific data (beyond 36th byte)

- Will not get power up attention status

Command -- Log Select & Sense

- Allows initiator to manage statistical data, does not define the exact data to be logged

  Op code -- 4C and 4D, 10-byte commands, (SCSI 2 only)

  Byte 7-8 -- transfer length

- PC, Page Control, Byte 1 bits 7-6

  00 -- Threshold values

  01 -- Cumulative values

  10 -- Default threshold values

  11 -- Default cumulative values

- Log Parameter

  4 byte header -- page code and length

  log pages -- each page has many log parameters

  (Pages are sent in ascending order)

Command -- Log Parameter

- Each Log Page starts with a 4 byte header

  Byte 0-1 -- parameter code

  Byte 2 -- control flags

    DU -- disable update

    DS -- disable saving

    TDS -- target disable saving (target's implicit saving)

    ETC -- enable threshold comparison

    TMC -- threshold met criteria (every update, equal, not equal, greater than)

  Byte 3 -- parameter length

Log Parameter Pages

1. 00 -- Supported log pages
2. 01 -- Buffer overrun/ underrun
3. 02 -- Write error counter
4. 03 -- Read error counter
5. 04 -- Read reverse error counter
6. 05 -- Verify error counter
7. 06 -- Non-media error
8. 07 -- Last N error events
9. 08 to 3F -- reserved
10. 30 to 3E -- vendor unique

Command -- Mode select, sense

- Allow initiator to specify and read target parms
- For both CCS and SCSI 2

  CCS -- 6-byte commands, op code = 15h and 1Ah. One byte transfer length (byte 4)

  SCSI 2 -- added 10-byte commands, op code = 55h and 5Ah. 2 byte transfer length (bytes 7 and 8).

  Command byte 1 bit 4 (PF bit) is set to 0 for SCSI 1

- Parameter List

  Save parameter flag (command byte 1 bit 0 set to 1)

  Four different types of values (command byte 2 bits 7 and 6)

  00 -- current values

  01 -- changeable values, ff changeable, 00 not changeable

  10 -- default values

  11 -- saved values

Mode Select & Sense

- 4 Byte header

  Byte 0,1 -- mode data length (only for mode sense)

  Byte 2 -- Media type (depending of device type)

  Byte 3 -- Device specific

- 8 Byte block descriptor -- optional

  Leftover from SCSI 1 -- Density code, number of blocks, block size

- Mode pages

  Byte 0 -- page code, byte 1 -- page length (page code = 3F, all pages)

  Byte 2 to n -- parameters

    Page 02 -- disconnect and reconnect page

    Page 09 -- peripheral device (ESDI,IPI, SMD etc.)

    Page 0A -- Control mode page (SCSI 2)

    Page 03 to 08, 0B to 1F-- defined by CCS (for disk drives)

    Page 20 to 3E -- Vendor unique

Mode Select & Sense

- Control Mode Page

  Queue algorithm modifier

    00 -- restricted re-ordering

    01 -- unrestricted re-ordering (don't worry about data integrity)

    02 to 07 -- reserved

    08 to 0F -- vendor unique

  DQ -- disable queue

  EECA -- enable extended contingent

  RAENP -- ready AEN permission

- Disconnect-Reconnect Page

  Bus inactivity time limit

  Disconnect time limit

  Connect time limit

  Maximum burst size

Command -- Read-Write Buffer

- Used together as a diagnostic function

  Op code = 3C and 3D, 10-byte command

- Available in SCSI 1 and CCS, Defined in SCSI 2

  Byte 1 bits 2-0, mode, specifies header and data or data along

  Byte 2 -- buffer ID

  Byte 3 to 5 -- buffer offset

  Byte 6 to 8 -- Transfer length

- Buffer Data

  4 Byte header specifies transfer length

  Data from/to the buffer

Command -- Request Sense

- Allows initiator to get detailed error sense data

  Op code = 03, 6-byte command

  Byte 4 -- transfer length

- Sense data

  Byte 0 -- sense code (70 or 71, extended. If 0, SASI, 4-byte sense)

  Byte 1 -- segment number (copy and compare commands)

  Byte 2 -- error key

    Bit 7 -- file mark

    Bit 6 -- end of media

    Bit 5 -- incorrect length indicator

    Bit 4 -- reserved

    Bits 3-0 -- Error key

  Byte 3 to 6 -- addition information, device type dependent

  Byte 7 -- additional sense data length

Request Sense -- Sense Key

00 -- No error

01 -- Recovered error

02 -- Not ready

03 -- Medium error

04 -- Hardware error

05 -- Illegal request

06 -- Unit attention

07 -- Data protect

08 -- Blank check

0A -- Copy aborted

0B -- Aborted command

0C -- Compare equal

0D -- Volume overflow

0E -- Compare not equal

Sense Code -- device specific

Sense code qualifier

Command -- Send Diagnostics

- Send Diagnostics, Receive Diagnostic Results

  Available in SCSI 1 and CCS. Defined in SCSI 2

  Byte 0, Op code = 1C and 1D

  Byte 4, Diagnostic page length

- Diagnostic Pages

  Confirms to the format of log pages or mode select pages

  -- 4 byte header followed by page data

  Device specific

Command -- Test Unit Ready

- Check out if a LUN is ready.

  Op code = 00, 6 byte command. (6 bytes of zeros)
- Use Request Sense if check status is received.
- Used to clean up any attention condition.
- Disk drives returns not ready before it spins up.
- Tape drives returns busy when rewinding tape.

Command -- Direct Access Device

- Also known as DASD, Hard Disk, or Hard File

M	04	Format drive
M	08 28	Read
M	25	Read capacity
O	37	Read defect data
O	07	Reassign block
M	17	Release
M	16	Reserve
M	01	Rezero
O	30 31 32	Search data equal, high, or low
M	0B 2B	Seek
O	1B	Start and stop unit
O	2F	Verify
M	0A 2A	Write

Direct Access -- Format Drive

- Rewrite disk drive ID and DATA fields

  Map out defective sectors

  Op code = 04, 6-byte command

  Byte 1 -- bit 4, FmtData -- defect list is provided

  bit 3, CmpLst -- use the defect list provided and P-list

  bits 2-0, Defect list format

  000 -- block format

  100 -- bytes from index

  101 -- physical sector

  Byte 2 -- vendor unique

  Byte 3-4 -- interleave factor

- Used for media error recovery

Format Drive -- Defect List

- Primary Defect List (P-List)

  List supplied by original manufacturer, stored on the disk.

- Target Certification List (C-List)

  Additional defects found during format (when certification is enabled)

- Data Defect List (D-List)

  Defect listed provided during the format command data phase

- Grown Defect List (G-List)

  D-list + C-list + reassigned blocks

  (Defect does not grow in the field. These defects are user found defects that are not found by manufacturer.)

Format Drive -- Format Options

- Format data 4-byte format header

Byte 0 -- reserved

Byte 1 -- is used only when bit 7 is set to 1

bit 7 -- FOV, format option valid

bit 6 -- DPRY, disable primary list

bit 5 -- DCRT, disable certification

bit 4 -- STPF, stop format when defect lists not available

bit 3 -- IP, Initialization pattern is provided

bit 2 -- DSP, disable mode select parameter saving

bit 1 -- Immed, return immediate status (for host adapter that can't disconnect)

bit 0 -- VS, ???

Byte 2-3 -- defect list, 8 bytes for each defect.

Format Drive -- Options

- Keep it simple stupid, let the drive handle the defects. Use P-list and G-list only.
- Use reassign block command during certification when a bad block is found.
- May have one defect per MB, 1000 defects for a 1GB drives.
- Variable number of blocks per track.

  Constant Density Record (CDR)
- One spare sector per track for mapping out defects.

Mode Select & Sense Pages

1. 08, Caching page -- pre-fetch and retention management
2. 0A, Control mode page -- queue and attention management
3. 02, Disconnect-reconnect page -- bus busy and idle time limits
4. 05, Flexible disk page -- write pre-comp, head load and unload delay
5. 03, Format drive page -- track skew, cylinder skew, interleave...
6. 0B, Medium types supported page -- up to 4 media types
7. 0C, Notch and partition page -- for drives with variable blocks per cyl
8. 09, Peripheral device page -- specifies SCSI, ESDI, SMD, or IPI2
9. 01, Read-write error recovery page -- various recovery options
10. 04, Rigid disk geometry page -- bytes per track, head & cyl counts
11. 07, Verify error recovery page -- a subset of read-write recovery
12. 3F, Return all pages
13. 00, 20-3E, Vendor specific
14. 0D-1F, Reserved

Direct Access -- Read

- Read disk blocks from specified block address

  Op code = 08 or 28, a 6-byte or 10-byte command.

  6-byte command has only a 21-bit address and 8-bit count

  10-byte command has 32-bit address and 16-bit count

- Relative address

  Byte 1 bit 0 -- RelAdr

  Used after search. Address is a 2's complement value to be added to the block address on which the search condition is met.

- SCSI 2 Options -- in the 10-byte command

  Byte 1 bit 4 -- DPO, disable page out

  Byte 1 bit 3 -- FUA, force unit access

- Read ahead function

Direct Access -- Read Capacity

- Determines the capacity of the disk drive or currently cylinder position.

  Op code = 25, 10-byte command

  Byte 9 bit 0 -- PMI, reads the last sector number of the addressed cylinder (specified by byte 2 to 5)

- Both block size and the last block address are returned.
- Used for auto-configuration.

  But, most operating system puts the configuration data in block 0.

Direct Access -- Read Defect Data

- Gets defect list back from the target

  Op code = 37, a ten-byte command

  Byte 1 bits 4 and 3 specify P-list and G-list respectively

  Byte 1 bits 2 to 0 specify defect list format -- logical block that is mapped, physical block, or byte from index

- Defect Data

  4-byte header specifies defect list length

  8 bytes for each defect

- Used for debug only.

Direct Access -- Reassign Block

- Replace defect block by a spare

  Op code = 07, a defect list is provided through data phase

- Defect list

  4-byte header, specifies the size

  Defect descriptors, each descriptor is 4-byte

- Defect management

  Transparent to initiator -- SCSI drive should not have defect visible.

  Mapped block may have long access time.

  Reformat forces block push down.

  May run out of spares. Highly unlikely.

- Media defects

  One per MB, Up to 1000 defects for 1GB drive.

  How to design defect management

Direct Access -- Reserve & Release

- Provide contention resolution in multi-initiator systems.

  Op codes = 16 and 17. Once reserved, other initiators get busy status when access reserved drive or extents.

- Reserve on extents

  An extent is a set of contiguous sectors.

- Extent lists

  Reservation type -- read, write, exclusive, shared

  Starting address and number of blocks

- Third party reservation -- reserve for an initiator

  Used in copy command so the initiator can still access it data.

- Bus device reset message clears everything

Direct Access -- Rezero & Seek

- Recalibrate disk arm

  Op code = 01

  Used exclusively for error recovery

  Many device drivers still use this command at power up

- Seek to specified block

  Op code = 0B or 2B

  Used only for testing only

Direct Access -- Start & Stop Unit

- Spin up and down the disk drive.

  Op code = 1B

  Byte 4 bit 0 specifies if start or stop is executed.

- Used to prevent current surge at power up
- Immediate completion

  Byte 1 bit 1 is set for requesting early completion status. Initiator can go on to start other drives. Used by initiators that don't disconnect.

- Media eject

  Byte 4 bit 1is set to eject a disk cartridge.

Direct Access -- Search Data

- Search one or more blocks for equal or unequal data patterns.

  Op codes = 30, 31, or 32 for search equal, high and low respectively

- Write from initiator and read from target.

  Comparison is done by target

- May link to a read or write command which has RelAdr option.

  Search 'key' and read 'data'.

- If target has a large buffer, data is fetch only once.

  Otherwise, target sends restore data pointer and re-fetches.

  Most drives today have 128K buffer.

Direct Access -- Write

- Write disk blocks to specified block address

  Op code = 0A or 2A, a 6-byte or 10-byte command.

  6-byte command has only a 21-bit address and 8-bit count

  10-byte command has 32-bit address and 16-bit count

  Relative address, Byte 1 bit 0 -- RelAdr, block address is a 2's complement value that can be added to the block number on which a search condition is met.

  SCSI 2 Options

    Byte 1 bit 4 -- DPO, disable page out

    Byte 1 bit 3 -- FUA, force unit access, bypass cache

- Write and Verify

  Op code = 2F, performs read back check after write

  Optional byte-to-byte comparison

    Byte 1 bit 1 -- BytChk, works like search equal

Command -Others

- Lock-unlock cache
- Pre-fetch
- Prevent-allow medium removal
- Read long, reads data plus ECC
- Set limits, Useful only with linked commands
- Synchronize cache -- unix like 'sync' command

Command -- Sequential Device

Also known as 1/4", 1/2", 4mm, and 8mm tapes

M	19	Erase
O	1B	Load and Unload
O	2B	Locate
M	08	Read
M	05	Read block limits
O	34	Read position
O	0F	Read reverse
O	14	Recover buffered data
M	01	Rewind
M	11	Space
O	13	Verify
M	0A	Write
M	10	Write file mark

Sequential Device -- Erase

- Causes part or all of the media to be erased

  Op code = 19, makes the tape to appear as blank

- Immediate status

  Byte 1 bit 1, when set, target returns completion status immediately.

  (Tape erase may take hours.)

- Long erase

  Byte 1 bit 0, when set, starts erase after writing buffered data to the end of media. Otherwise, write a normal erase gap.

Sequential Device -- Load Unload

- Load or unload a tape cartridge

  Op code = 1B, same as disk drive start/stop command

  Byte 4 bit 0 specifies load or unload

- Immediate completion

  Byte 1 bit 0 specifies that complete status should be returned immediately

- Re-tension

  Byte 4 bit 1 specifies that the device should rewind and forward the tape to release tension.

- End of Tape (EOT)

  Byte 4 bit 2 specifies that forward the tape to EOT before unload

Sequential Device -- Locate

- Find a partition or a block

  Op code = 2B, a ten-byte command, allows the tape to position at a specified block and/or partition using the fastest means.

  Byte 3 to 6 specify logical block number

  Byte 8 specifies partition number

- Partitions

  One or more non-overlapped mini-volumes. Each partition has its own beginning-of-partition, early warning, and end-of-partition.

  Byte 1 bit 1, Change Partition, should be set to change partition.

  Byte 1 bit 2, Device Specific, vendor unique address

- Logical block

  Allows random access -- although a little slow.

- 4mm tape drive can locate at 200 times of R/W speed.

Sequential Device -- Read

- Transfer one or more blocks

  Op code = 08. Byte 2 to 4 specify the number of blocks of bytes to be transferred. (Byte 1 bit 0, FIXED, determines bytes or blocks.)

- No block address -- sequential device
- Variable block size

  When the FIXED bit is not set, the specified size is in bytes.

  Overlength -- recorded block is greater than specified size.

  Underlenth -- recorded block is less than specified size.

  SILI bit -- Byte 1 bit 1. When set, don't report error if underlength condition exists.

- Buffered read to keep drive streaming

  Data are pre-read in the buffer.

  One must issue the next read before the buffer is full.

- Same block can be written multiple times

Sequential Device -- Read Block Limits

- Gets block length limits

  Op code = 05

- 6-byte of data are returned

  Byte 1-3 -- maximum block length limit

  Byte 4-5 -- minimum block length limit

- The largest and smallest block the initiator can read and write

Sequential Device -- Read Position

- Gets current position and number of data blocks in buffer

  Op code = 34, a ten-byte command.

  Byte 1 bit 0, BT, specifies logical or physical block number should be returned.

- 20-byte of data are returned

  Byte 0, flags -- BOP, beginning of partition, EOP, end of partition, and BPU, block position unknown.

  Byte 1 -- partition number

  Byte 4-7 -- first block number, next block to be transferred.

  Byte 8-11 -- last block number, next block to/from the tape.

  Byte 13-15 -- number of blocks in buffer

  Byte 16-19 -- number of bytes in buffer

  All other bytes are reserved

Sequential Device -- Read Reverse

- Reads tape backward

  Op code = 0F, Just like a read command except it reads backward
- All blocks and bytes are reversed!!!
- The bits within each byte are not reversed
- Support FIX bit
- Support SILI bit

(To use this command, the DMA should reverse data transfer too.)

Sequential Device -- Recover Buffered Data

- Reads data from the buffer that couldn't be written to tape.

  Op code = 14, like a read command except data are fetched from the buffer.

- Used after end-of-media error condition.
- Support FIX and SILI bits.
- End of media recovery:

1. Recover buffered data
2. Write trailer and end of file, unload the tape.
3. Load a new tape.
4. Verify and initialize the tape label.
5. Write recovered buffered data.

Sequential Device -- Rewind and Space

- Rewind to beginning-of-tape (BOT)

  Op code = 01, support immediate complete status.

- Space over blocks, Setmarks or filemarks

  Op code = 11, Byte 1 bit 2-0 specifies space code

  Space code = 000 blocks

  001 file marks

  010 sequential file marks (consecutive file marks)

  011 end-of-data

  100 setmarks

  101 sequential setmarks

- Setmark, like a book mark

  When RSmk in mode page 10, device configuration, is set, target should stop spacing and report check status.

Sequential Device -- Verify

- Verifies one or more blocks (bytes) on tape

  Op code = 13. Support immed bit. Error is reported on the next command. The last verify command should have immed bit off.

  (Not needed if initiator supports disconnect.)

- BytCmp, byte 1 bit 1, determines if data should be fetched from initiator.

  If buffer is large, data is fetched once only.

- Support variable block size.

  Byte 2-4 specifies either byte or block count.

Sequential Device -- Write

- Writes one or more blocks (bytes)

  Op code = 0A.

  No block address (sequential device)

- Support variable block sizes

  When byte 1 bit 0, FIXED, is set to zero, transfer length is specified in bytes.

- Buffered write

  It is an immediate command. Gets completion status after data transfer. To keep tape 'streaming', send another write before buffer is empty

- Read back check

  Repeated writes to correct media error.

- Continued writes to avoid stopping.

Sequential Device -- Write Filemark

- Writes file mark or setmark

  Op code = 10, transfer length specifies how many filemarks (setmarks) to write. Byte 1 bit 1, WSmk, specifies that setmarks should be written.

- Buffered write

  Immed bit, byte 1 bit 0, allows the write of filemark to be buffered. (Some tape drive will not support buffered write filemark.)

  By not using the immed bit, one assures the buffered data are written when this command is completed.

Commands --Process Device

M	08	Receive (or Read)
M	0A	Send (or Write)

- Bytes 2-4 specify transfer byte count.
- Host to host communication, SEND only.

  Each process can be either an initiator or a target.

  Target device has buffer to hold the data.
- Host to host communication, SEND and RECEIVE

  One host is an initiator and another one is a target.

  When a target gets RECEIVE, it may disconnect when there is no data.
- A 5MB or 10MB initiator-to-initiator channel?

SCSI 1 vs. SCSI 2 Comparison

- SCSI 2 == SCSI 1 + "Proliferation of Undocumented Features"
- CCS == Beginning of SCSI 2, disk only.
- SCSI 2

  Many advance features that not every operating system can use.

  Many device dependent features that an operating system shouldn't use.

  "A creation by the best minds in the business."

  "A horse designed by a committee."

  Will work with a SCSI 1 device -- if the device is properly designed.

  Many device has a jumper to disable SCSI 2 functions.

SCSI 1 vs. SCSI 2 -- Physical

- Faster Bus

  10 MB per second -- differential cable recommended.

- Wider Bus

  8-bit, 16-bit, or 32-bit data bits -- 10MB, 20MB or 40MB per second.

  50-pin or 68-pin cables.

- Smaller Bus

  Shielded high density connector.

- Safer Bus

  Parity is required.

  Initiator must provide terminator power.

SCSI 1 vs. SCSI 2 -- Features

- Removed SCSI 1 features
  - Selection without identify message
  - 4-byte sense data
  - Initiator may not supply terminator power
- New features
  - Command queuing
  - Asynchronous Event Notification (AEN)
  - Extended contingent allegiance
  - Disk caching
  - Log select and log sense
  - Diagnostic functions
  - 12-byte command for optical devices
  - Many new commands and mode pages

SCSI 2 -- Command Queuing

- SCSI 1 -- untagged queuing

  One command from each initiator at one time.

- SCSI 2 -- both tagged and untagged queuing

  An initiator can only use one type of queuing at one time.

  An target must support both at one time. Disconnection is required.

- Initiator sends a 2-byte tagged queuing message

  1st byte =

  20 -- Head of queue tag

  21 -- Ordered queue tag

  22 -- Simple queue tag

  2nd byte is a queue tag which identifies a command.

- Target reselection sends back a 2-byte simple queue tag message to revive the connection.

SCSI 2 -- Queuing

- Head of Queue

  Next command to be executed. Do not pre-empt ongoing command.

- Ordered Queue

  A fence. All commands received before are executed first. All commands received after are executed later.

- Simple Queue

  Restricted ordering or un-restricted ordering as specified by Control Mode page.

- Queue error management

  Abort all queued commands or continue after recovery.

  Specified by Control Mode page.

SCSI 2 -- Queuing, Advantages

- Minimize disk arm movement

  Elevator sort.

- Minimize disk rotational misses

  Coalesced write commands

  Sort by sector positions

  Sort by access times

- Results -- high disk throughput

  12% improvement when average queue length is 4.

SCSI 2 -- AEN

- Target informs initiator that an asynchronous event has occurred.

  Use a send command with 4-byte AEN header + request sense data.

- Initiator responds to a target Inquiry command and specifies in the inquiry data that AEN is supported.

- Use of AEN

1. Error detected after command completion.
2. Device available.
3. Power glitch, mode page changed by other initiator.
4. Media change, rewind complete, and tape loaded.
5. Device is released.

- No more polling by the initiator

SCSI 2 -- ECA

- Contingent allegiance condition

  After a check status, stay busy to other initiator until the request sense command is received -- only one set of error can be saved. (Why not save multiple sets?)

- Extended contingent allegiance

  Target sends Initiate Recovery message following check status.

  Initiator performs necessary recovery -- Recovery Buffered Data.

  Initiator send Release Recovery message.

- Not necessary for every check status
- During ECA, only untagged queuing is supported.

SCSI 2 -- Disk Cache

- Read Ahead

  Highly popular, improves sequential access performance by the amount of disk latency.

- Write cache

  Need non-volatile RAM. Has great potential

- Cache management

  Lock-unlock, pre-fetch, sync, bypass cache access.

  Is it really worthwhile to send a SCSI command to lock or unlock a cache page?

  Why not port the file management code to disk controller?

SCSI Devices -- ICs & HAs

- Adaptec AIC-6260
- NCR 53C90
- TI SN75C091
- NCR 53C700

- Seagate ST02
- ACB 1540
- DTC 3290
- NCR Raid Controller

SCSI Devices -- Disks & Tapes

- Disk Drives
  - Functions
  - Data transfer speed
  - Read ahead
  - Zero latency
  - Defect management
  - Performance
- Tape Drives
  - Characteristics
  - Operations
  - Performance

SCSI Device -- AIC-6260

- Second generation chip, no sequencer function
- Integration of SCSI and AT bus interfaces, 8-bit SCSI interface, 8 or 16 bit host interface.
- SCSI Initiator or target
- Many registers, accessible by microprocessor

  340 to 35E -- 31 registers

  SCSI control and data registers

  Synchronous data transfer control

  Interrupt and status registers

  Command registers to start SCSI handshake

  DMA control, counter and PIO registers

  Error registers
- 16 byte SCSI stack and 128 byte host FIFO

TABLE 1. REGISTER SUMMARY

DMA BYTE COUNT (R/W)		
00 (L)	01 (M)	02 (H)
7 2^7	7 2^{15}	7 2^{23}
6 2^6	6 2^{14}	6 2^{22}
5 2^5	5 2^{13}	5 2^{21}
4 2^4	4 2^{12}	4 2^{20}
3 2^3	3 2^{11}	3 2^{19}
2 2^2	2 2^{10}	2 2^{18}
1 2^1	1 2^9	1 2^{17}
0 2^0	0 2^8	0 2^{16}

03 INT MSK REG (W)
7 RESERVED
6 ARB/SEL START
5 EN AUTO ATN
4 EN ERROR INT
3 EN CMD DONE INT
2 EN SEL OUT INT
1 EN RESEL INT
0 EN SELECT INT

04 OFFSET CNTRL (W)
7 RESERVED
6 SYNC XFER RATE BIT 2
5 SYNC XFER RATE BIT 1
4 SYNC XFER RATE BIT 0
3 OFFSET BIT 3
2 OFFSET BIT 2
1 OFFSET BIT 1
0 OFFSET BIT 0

05 FIFO STATUS (R)
7 TEST SIGNAL
6 TEST SIGNAL
5 OFFSET COUNT ZERO
4 FIFO EMPTY
3 FIFO FULL
2 FIFO COUNTER BIT 2
1 FIFO COUNTER BIT 1
0 FIFO COUNTER BIT 0

05 DMA CNTRL (W)
7 RESERVED
6 RESERVED
5 RESERVED
4 RESERVED
3 RESERVED
2 ODD XFER START
1 TRANSFER DIR
0 DMA XFER EN

06 REV CNTRL (R)
7 RESERVED
6 RESERVED
5 RESERVED
4 RESERVED
3 RESERVED
2 RESERVED
1 REVISION
0 REVISION

06 INT MSK REG 1 (W)
7 RESERVED
6 EN SCSI REQ ON INT
5 EN SCSI RST INT
4 EN MEM PARITY ERR INT
3 EN PHASE MISMATCH INT
2 EN BUS FREE DETECT INT
1 EN SCSI PARITY ERR INT
0 EN PHASE CHANGE INT(INIT,
0 EN ATN ON INT (TGT)

07 STATUS REG 0 (R)
7 SCSI RST OCCURRED
6 MEMORY PARITY ERR
5 PHASE MISMATCH ERR
4 BUS FREE DETECT
3 SCSI PARITY ERR
2 SCSI REQ ON
1 SCSI PHASE CHG/ATN
0 DMA BYTE CNT ZERO

07 CONTROL REG 0 (W)
7 P MEM CYCLE REQ
6 P MEM R/W
5 TARGET MODE
4 EN PORT A INP OR OUT
3 SCSI INTERFACE MODE
2 SCSI ID 2
1 SCSI ID 1
0 SCSI ID 0

08 STATUS REG 1 (R)
7 MEM CYCLE CMPL
6 RESERVED
5 SCSI RST IN
4 ERROR
3 CMD DONE
2 SEL OUT
1 RESELECTED
0 SELECTED

08 CONTROL REG 1 (W)
7 AUTO SCSI PIO REQ
6 EN 16-BIT MEM BUS
5 RESERVED ('0' ONLY)
4 EN PORT B INP OR OUT
3 PHASE CHANGE MODE
2 CLK FREQ MODE
1 SCSI RST OUT
0 CHIP S/W RESET

09 SCSI SIGNAL REG (R)
7 SCSI C/D IN
6 SCSI I/O IN
5 SCSI MSG IN
4 SCSI ATN IN
3 SCSI SEL IN
2 SCSI BSY IN
1 SCSI REQ IN
0 SCSI ACK IN

09 SCSI SIGNAL REG (W)
7 SCSI C/D OUT
6 SCSI I/O OUT
5 SCSI MSG OUT
4 SCSI ATN OUT
3 SCSI SEL OUT
2 SCSI BSY OUT
1 SCSI ACK OUT (INIT)
1 SCSI REQ OUT (TGT)
0 RESERVED

0A SCSI ID DATA (R/W)
7 SCSI ID/DATA 7
6 SCSI ID/DATA 6
5 SCSI ID/DATA 5
4 SCSI ID/DATA 4
3 SCSI ID/DATA 3
2 SCSI ID/DATA 2
1 SCSI ID/DATA 1
0 SCSI ID/DATA 0

0B SOURCE/DEST ID (R)
7 ID 7
6 ID 6
5 ID 5
4 ID 4
3 ID 3
2 ID 2
1 ID 1
0 ID 0

0C MEMORY DATA (R/W)
7 MEM DATA 7
6 MEM DATA 6
5 MEM DATA 5
4 MEM DATA 4
3 MEM DATA 3
2 MEM DATA 2
1 MEM DATA 1
0 MEM DATA 0

0D PORT A (R/W)
7 PORT A BIT 7/$\overline{\text{LBV}}$
6 PORT A BIT 6
5 PORT A BIT 5
4 PORT A BIT 4
3 PORT A BIT 3
2 PORT A BIT 2/$\overline{\text{HBV}}$
1 PORT A BIT 1
0 PORT A BIT 0

0E PORT B (R/W)
7 PORT B BIT 7
6 PORT B BIT 6
5 PORT B BIT 5
4 PORT B BIT 4
3 PORT B BIT 3
2 PORT B BIT 2
1 PORT B BIT 1
0 PORT B BIT 0

0F SCSI LATCH DATA (R)
7 SCSI LATCHED DATA 7
6 SCSI LATCHED DATA 6
5 SCSI LATCHED DATA 5
4 SCSI LATCHED DATA 4
3 SCSI LATCHED DATA 3
2 SCSI LATCHED DATA 2
1 SCSI LATCHED DATA 1
0 SCSI LATCHED DATA 0

0F SCSI BSY RST (TGT) (W)
FOR A TARGET, A WRITE TO THIS REGISTER WILL RESET THE SCSI $\overline{\text{BSY}}$ OUT AND THE SCSI BUS WILL ENTER A BUS FREE PHASE.

Section Three Register Description

This section contains information on the AIC-6260's internal registers. Each register is described under its own heading, identified by name and address.

There are 32 registers, normally decoded from 340_h through $35E_h$. If ALTERNATE* is asserted (low), the registers are decoded from 140_h. Registers are written and read by the host processor via the host bus, in I/O address space.

The following conventions are used throughout this section:

- **set:** Indicates that the target bit was loaded with a 1.
- **clear:** Indicates that the target bit was loaded with a 0.
- **(0):** Indicates that the associated bit is set to 0 after a hard reset.
- **(1):** Indicates that the associated bit is set to 1 after a hard reset.
- **(x):** Indicates the state of the bit after a hard reset is undefined.

Table 3-1 is an address map of the AIC-6260 registers.

TABLE 3-1. AIC-6260 REGISTER ADDRESS MAP

340_h SCSISEQ	341_h SXFRCTL0	342_h SXFRCTL1	343_h SCSISIGO	343_h SCSISIGI	344_h SCSIRATE
R/W	R/W	R/W	W	R	W
7 TEMODEO	7 SCSIEN	7 BITBUCKET	7 CDO	7 CDI	7 RSVD
6 ENSELO	6 DMAEN	6 SWRAPEN	6 IOO	6 IOI	6 SXFR (2)
5 ENSELI	5 CH1/CH2	5 ENSPCHK	5 MSGO	5 MSGI	5 SXFR (1)
4 ENRESELI	4 CLRSTCNT	4 STIMESEL (1)	4 ATNO	4 ATNI	4 SXFR (0)
3 ENAUTOATNO	3 SPIOEN	3 STIMESEL (0)	3 SELO	3 SELI	3 SOFS (3)
2 ENAUTOATNI	2 RSVD	2 ENSTIMER	2 BSYO	2 BSYI	2 SOFS (2)
1 ENAUTOATNP	1 CLRCH1	1 BYTEALIGN	1 REQO	1 REQI	1 SOFS (1)
0 SCSIRSTO	0 RSVD	0 RSVD	0 ACKO	0 ACKI	0 SOFS (0)

345_h SCSIID	345_h SELID	346_h SCSIDAT	347_h SCSIBUS	348_h STCNT0	349_h STCNT1
W	R	R/W	R	R/W	R/W
7 RSVD	7 SELID 7	7 DB (7)	7 SDB (7)	7 STCNT (07)	7 STCNT (15)
6 OID (2)	6 SELID 6	6 DB (6)	6 SDB (6)	6 STCNT (06)	6 STCNT (14)
5 OID (1)	5 SELID 5	5 DB (5)	5 SDB (5)	5 STCNT (05)	5 STCNT (13)
4 OID(0)	4 SELID 4	4 DB (4)	4 SDB (4)	4 STCNT (04)	4 STCNT (12)
3 RSVD	3 SELID 3	3 DB (3)	3 SDB (3)	3 STCNT (03)	3 STCNT (11)
2 TID (2)	2 SELID 2	2 DB (2)	2 SDB (2)	2 STCNT (02)	2 STCNT (10)
1 TID (1)	1 SELID 1	1 DB (1)	1 SDB (1)	1 STCNT (01)	1 STCNT (09)
0 TID (0)	0 SELID 0	0 DB (0)	0 SDB (0)	0 STCNT (00)	0 STCNT (08)

$34A_h$ STCNT2	$34B_h$ CLRSINT0	$34B_h$ SSTAT0	$34C_h$ CLRSINT1	$34C_h$ SSTAT1	$34D_h$ SSTAT2
R/W	W	R	W	R	R
7 STCNT (23)	7 SETSDONE	7 TARGET	7 CLRSELTIMO	7 SELTO	7 RSVD
6 STCNT (22)	6 CLRSELDO	6 SELDO	6 CLRATNO	6 ATNTARG	6 RSVD
5 STCNT (21)	5 CLRSELDI	5 SELDI	5 CLRSCSIRSTI	5 SCSIRSTI	5 SOFFSET
4 STCNT (20)	4 CLRSELINGO	4 SELINGO	4 RSVD	4 PHASEMIS	4 SEMPTY
3 STCNT (19)	3 CLRSWRAP	3 SWRAP	3 CLRBUSFREE	3 BUSFREE	3 SFULL
2 STCNT (18)	2 CLRSDONE	2 SDONE	2 CLRSCSIPERR	2 SCSIPERR	2 SFCNT (2)
1 STCNT (17)	1 CLRSPIORDY	1 SPIORDY	1 CLRPHASECHG	1 PHASECHG	1 SFCNT (1)
0 STCNT (16)	0 CLRDMADONE	0 DMADONE	0 CLRREQINIT	0 REQINIT	0 SFCNT (0)

$34E_h$ SCSITEST	$34E_h$ SSTAT3	$34F_h$ CLRSERR	$34F_h$ SSTAT4	350_h SIMODE0	351_h SIMODE1
W	R	W	R	R/W	R/W
7 RSVD	7 SCSICNT (3)	7 RSVD	7 RSVD	7 RSVD	7 ENSELTIMO
6 RSVD	6 SCSICNT (2)	6 RSVD	6 RSVD	6 ENSELDO	6 ENATNTARG
5 RSVD	5 SCSICNT (1)	5 RSVD	5 RSVD	5 ENSELDI	5 ENSCSIRST
4 RSVD	4 SCSICNT (0)	4 RSVD	4 RSVD	4 ENSELINGO	4 ENPHASEMIS
3 SCTESTU	3 OFFCNT (3)	3 RSVD	3 RSVD	3 ENSWRAP	3 ENBUSFREE
2 SCTESTD	2 OFFCNT (2)	2 CLRSYNCERR	2 SYNCERR	2 ENSDONE	2 ENSCSIPERR
1 RSVD	1 OFFCNT (1)	1 CLRFWERR	1 FWERR	1 ENSPIORDY	1 ENPHASECHG
0 STCTEST	0 OFFCNT (0)	0 CLRFRERR	0 FRERR	0 ENDMADONE	0 ENREQINIT

352_h DMACNTRL0	353_h DMACNTRL1	354_h DMASTAT	355_h FIFOSTAT	356_h DATAPORTL	356_h DATAPORTH
R/W	R/W	R	R	R/W	R/W
7 ENDMA	7 PWRDWN	7 ATDONE	7 FCNT (7)	7 DATAL (07)	15 DATAH (15)
6 8BIT/-16BIT	6 RSVD	6 WORDRDY	6 FCNT (6)	6 DATAL (06)	14 DATAH (14)
5 DMA/-PIO	5 RSVD	5 INTSTAT	5 FCNT (5)	5 DATAL (05)	13 DATAH (13)
4 RSVD	4 RSVD	4 DFIFOFULL	4 FCNT (4)	4 DATAL (04)	12 DATAH (12)
3 WRITE/-READ	3 STK (3)	3 DFIFOEMP	3 FCNT (3)	3 DATAL (03)	11 DATAH (11)
2 INTEN	2 STK (2)	2 RSVD	2 FCNT (2)	2 DATAL (02)	10 DATAH (10)
1 RSTFIFO	1 STK (1)	1 RSVD	1 FCNT (1)	1 DATAL (01)	9 DATAH (09)
0 SWINT	0 STK (0)	0 RSVD	0 FCNT (0)	0 DATAL (00)	8 DATAH (08)

358_h BRSTCNTRL	35A PORTA	35B PORTB	$35C_h$ REV	$35D_h$ STACK	$35E_h$ TEST
W	R/W	R/W	R	R/W	W
7 BON (3)	7 PADAT (7)	7 PBDAT (7)	7 RSVD	7 STKDAT (7)	7 RSVD
6 BON (2)	6 PADAT (6)	6 PBDAT (6)	6 RSVD	6 STKDAT (6)	6 BOFFTMR
5 BON (1)	5 PADAT (5)	5 PBDAT (5)	5 RSVD	5 STKDAT (5)	5 BONTMR
4 BON (0)	4 PADAT (4)	4 PBDAT (4)	4 RSVD	4 STKDAT (4)	4 STCNTH
3 BOFF (3)	3 PADAT (3)	3 PBDAT (3)	3 RSVD	3 STKDAT (3)	3 STCNTM
2 BOFF (2)	2 PADAT (2)	2 PBDAT (2)	2 REV (2)	2 STKDAT (2)	2 STCNTL
1 BOFF (1)	1 PADAT (1)	1 PBDAT (1)	1 REV (1)	1 STKDAT (1)	1 SCSIBLK
0 BOFF (0)	0 PADAT (0)	0 PBDAT (0)	0 REV (0)	0 STKDAT (0)	0 DMABLK

AIC-6260 -- Command

- ENSELO -- Enable selection out

  Hardware arbitration and selection

  Set up initiator and target ID registers.

  Tell hardware to start -- TEMODEO specifies initiator or target mode

  Waiting for interrupt

- ENSELI, ENRESELI -- Enable selection in or reselection in

  Arm the chip to receive selection and reselection

  SCSI ID register must be set first

  Has a selection/reselection timer -- 256, 128, 64, or 32 milliseconds

- ENAUTOATNI --Enable auto-attention at selection (initiator only)

  Assert Attention after selection or reselection -- to start identify messages

AIC-6260 -- Command

- ENAUTOATNP -- Enable auto-attention at parity error (initiator only)

  Set ATN to inform target that parity error is detected

- SCSIRSTO -- SCSI Reset Out

  Assert SCSI reset on the bus

- SPIOEN -- SCSI PIO Enable

  Single byte handshake on the SCSI bus

- SCSIEN & DMAEN -- SCSI transfer and DMA enable

  Together to allow data or command transfer (multiple bytes)

AIC-6260 -- Programming

1. Initialize required registers and start arbitration and selection -- if not immediately successful, set up selection time out interrupt.
2. Set up SCSI PIO for one or three byte identify message.
3. Transfer command using either PIO or DMA.
4. Set up data transfer if required -- if disconnection is needed, set up SCSI PIO for save data pointer and disconnection message.
5. If disconnected, set up the chip to accept reselection. Multiple commands can be started. Must manage the reconnection from a different device.
6. Handle reconnection identify message and set up data transfer again.
7. Complete data transfer and verify everything is OK.
8. Complete status phase.
9. Complete message phase.
10. Make sure bus is free before starting another command.

AIC-6250 -- Summary

- Second generation SCSI chip

  Hardware performs arbitration, selection, and data transfer

  Firmware or BIOS must perform other SCSI handshakes -- message, command, and status phases. One event at a time.

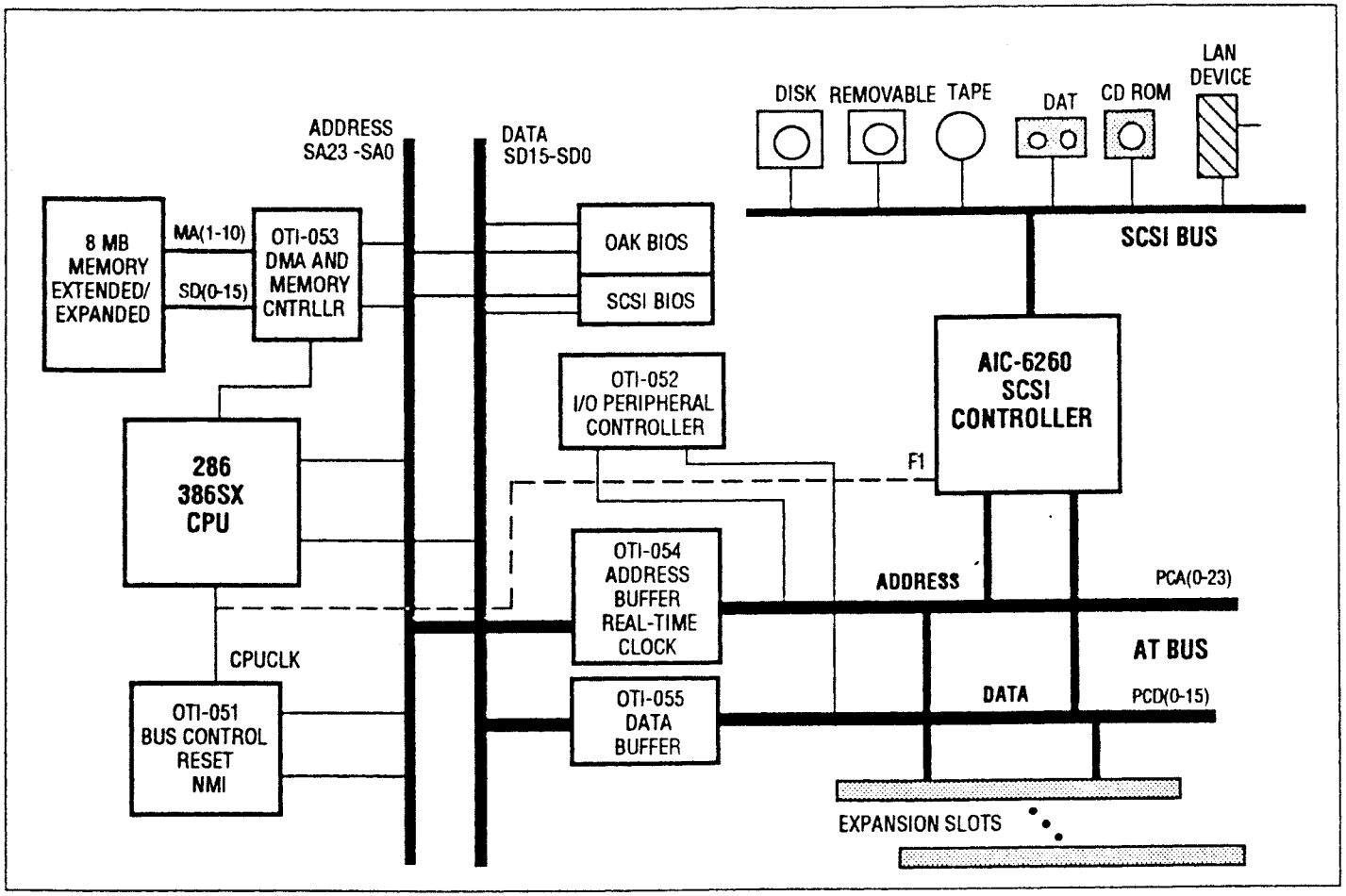
  Each handshake will take 50 to 100 microseconds depending on the microprocessor speed vs. a few microseconds by a sequencer.

  A dozen handshakes are needed to complete a SCSI command

- Works well for single task operating systems

  Requires many interrupts to complete one SCSI command

- Low cost solution for desktop computers



AIC-6260 with OakHorizon Laptop PC Chip Set

SCSI Devices -- NCR53C90

- Early third generation chip, simple sequencer to perform multiple SCSI handshakes
- SCSI initiator or target
- 8-bit SCSI and host interface (NCR53C94 for 16-bit host interface)
- Registers

  Transfer counter, DMA counters

  SCSI control and bus registers, selection time-out values

  Synchronous data transfer control

  Sequencer command register
- 16-byte FIFO

  For transferring identify message and command bytes

  For handling synchronous transfer

NCR53C90 -- Commands

NOP	Reselect sequence
Flush FIFO	Select with ATN
Reset Chip	Select with ATN and stop
Reset SCSI Bus	Select without ATN
	Enable selection/reselection
	Disable selection/reselection
	Select with ATN3 (tagged queuing)

- Available to both initiator and target modes

NCR53C90 -- Commands

Target	Initiator
Send Message	Transfer information
Send Status	Command complete sequence
Send Data	Message accepted
Disconnect sequence	Transfer pad
Command complete sequence	Set ATN
Disconnect	Reset ATN
Receive message	
Receive command	
Receive command sequence	
Receive data	
Target abort DMA	

*Use bus service interrupt for unusual conditions.

NCR 53C90A -- Sequencer

- Initiator select with ATN

Step	Descriptions
0	Arbitration complete or selection time out
0	Arbitration complete, waiting for message out
2	Message out complete
3	Command transfer, FIFO flags tells the byte count
4	Selection with ATN sequence complete

- Use bus service interrupt to handle abnormal cases.

NCR53C90 -- Programming

1. Initialize required registers, place identify message and command bytes in FIFO, and use **Select with ATN** or **Select with ATN3.**
2. If data transfer is necessary, set up SCSI and DMA controls, use **transfer information** to transfer data. If disconnection is received, bus service interrupt is generated.
3. Use **message accepted** to complete disconnection handshake
4. Enable reselection -- must manage reconnection from multiple devices.
5. If data transfer is necessary, set up SCSI and DMA controls, use **transfer information** to transfer data.
6. Use **initiator command complete** to finish a SCSI command.

NCR53C90 -- Summary

- Early third generation SCSI device

  Selection with identify message. Transferring command bytes by hardware

  Transferring data by hardware

  Sending completion status and message by one command

- Better performance than 2nd generation device

  Complete selection and command transfer in less than 100 microseconds

  Still need several interrupts to complete one SCSI command

SCSI Device -- TI SN75C091

- Late third generation chip. Has complicated sequencer to complete one SCSI command.
- SCSI initiator or target
- 8-bit SCSI and host interface

  Byte stack control - for 32 bit bus - specifies word length & byte offset
- 16-byte transmit and receive FIFOs
- Registers

  Receive and transmit FIFO registers

  SCSI control and bus registers, plus selection time out register

  Interrupt and status register

  Synchronous data transfer control

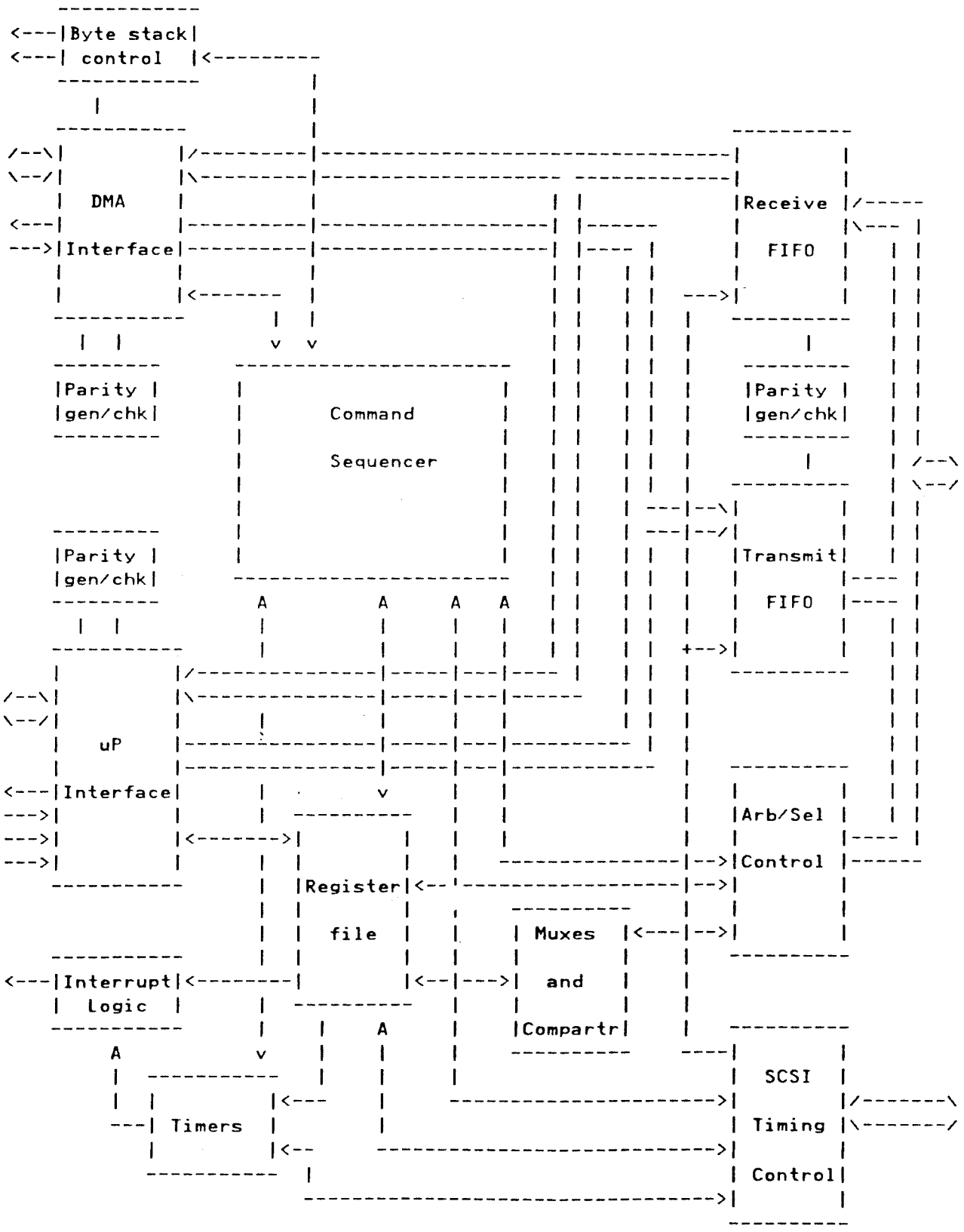
  DMA control and counters

  Command register

SECTION 3
ARCHITECTURE

3.1 Block Diagram

The chip is made up of thirteen major building blocks, as shown in Figure 3.1.



TI SN75C091 -- Command

1. Chip reset
2. Disconnect
3. Pause -- halt data transfer when error is detected
4. Assert ATN
5. Negate ACK
6. Clear receive FIFO
7. Clear transmit FIFO
8. SCSI bus reset

TI SN75C091 -- Command

Target	Initiator
Reselect	Select without ATN
Receive command	Select with ATN
Receive data	Transfer info
Receive message out	Transfer pad
Send status	Select without ATN and transfer
Send data	Select with ATN and transfer
Send message in	
Reselect and receive data	
Reselect and send data	
Wait for select without ATN and receive	
Wait for select with ATN and receive	
Conclude	
Link to next command	

TI SN75C091 -- Programming

- One command for each SCSI operation
  - Set up SCSI control registers
  - Store needed message and command byte
  - Set up DMA control and count
  - Wait for completion
- Handle abnormal conditions
  - Check function and interrupt status registers
  - Command phase registers shows where the sequencer was
- Restart after taking care abnormal conditions
  - Use command phase register
- Handle multiple devices on one SCSI bus
  - Determine which device is reconnecting

TI SN75C091 -- Sequencer

0	Arbitration/ Selection failed
1	Selection successful
2	Identify message sent
3	Start of command transfer
4	Command transfer complete
5	Start of data transfer
6	Save data pointer received
7	Disconnect message received
8	Target disconnected
9	Original target reselected
10	Correct ID message received
11	Data transfer complete
12	Status byte received
13	Command complete message received

TI SN75C091 -- Summary

- Late third generation SCSI chip

  One command for the whole SCSI operation, including disconnection and reconnection.

- Ideal for multi-tasking operating systems

  Only one interrupt is needed for 99.99% of operations.

- Ideal for work stations with build-in SCSI port
- Two interrupts for multiple SCSI devices.

  Starting new device at disconnection

  Wrong device at reconnection

- Does not handle modify data pointer message

SCSI Device -- NCR 53C700

- Fourth generation SCSI chip

  You program your own sequencer code.

- SCSI initiator or target

  8-bit SCSI and 32-bit host interfaces

- Registers

  SCSI control and data signals

  Interrupt and status register

  Synchronous data transfer control

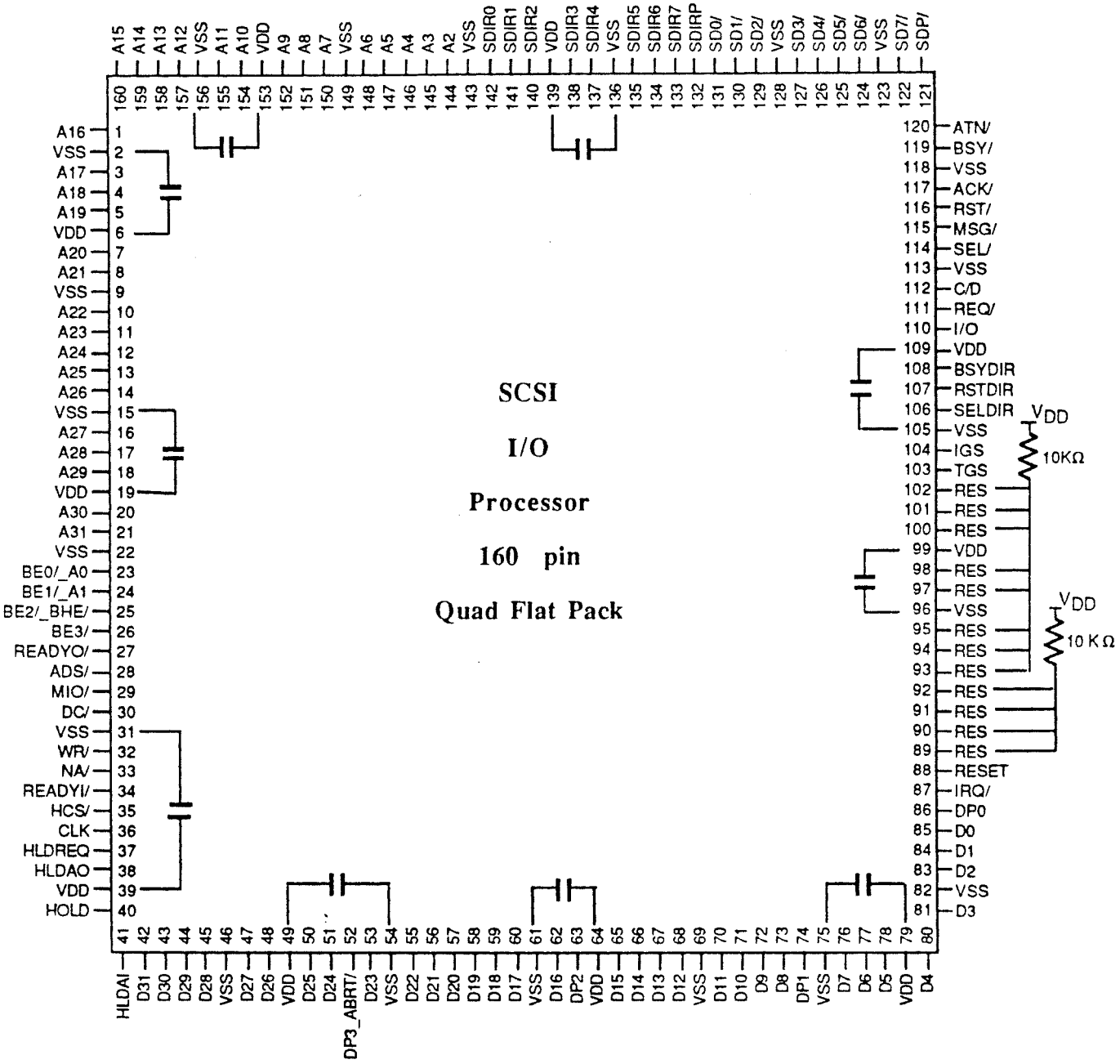
  DMA address, control and counter registers

  Scripts pointer registers

  DMA watchdog timer registers

- Operations

  Single step, pipeline, scripts modes



* NOTE: The above decoupling is recommended for optimum noise isolation. This configuration takes advantage of the separate ground planes contained in the SIOP for address and data lines, control lines, and SCSI pads. Use capacitor values of .01 uF or .1 uF.

**NOTE: Pins 102-100, 98, 97, and 95-89 are reserved for wide SCSI.

Figure 1. NCR 53C700 Pinout

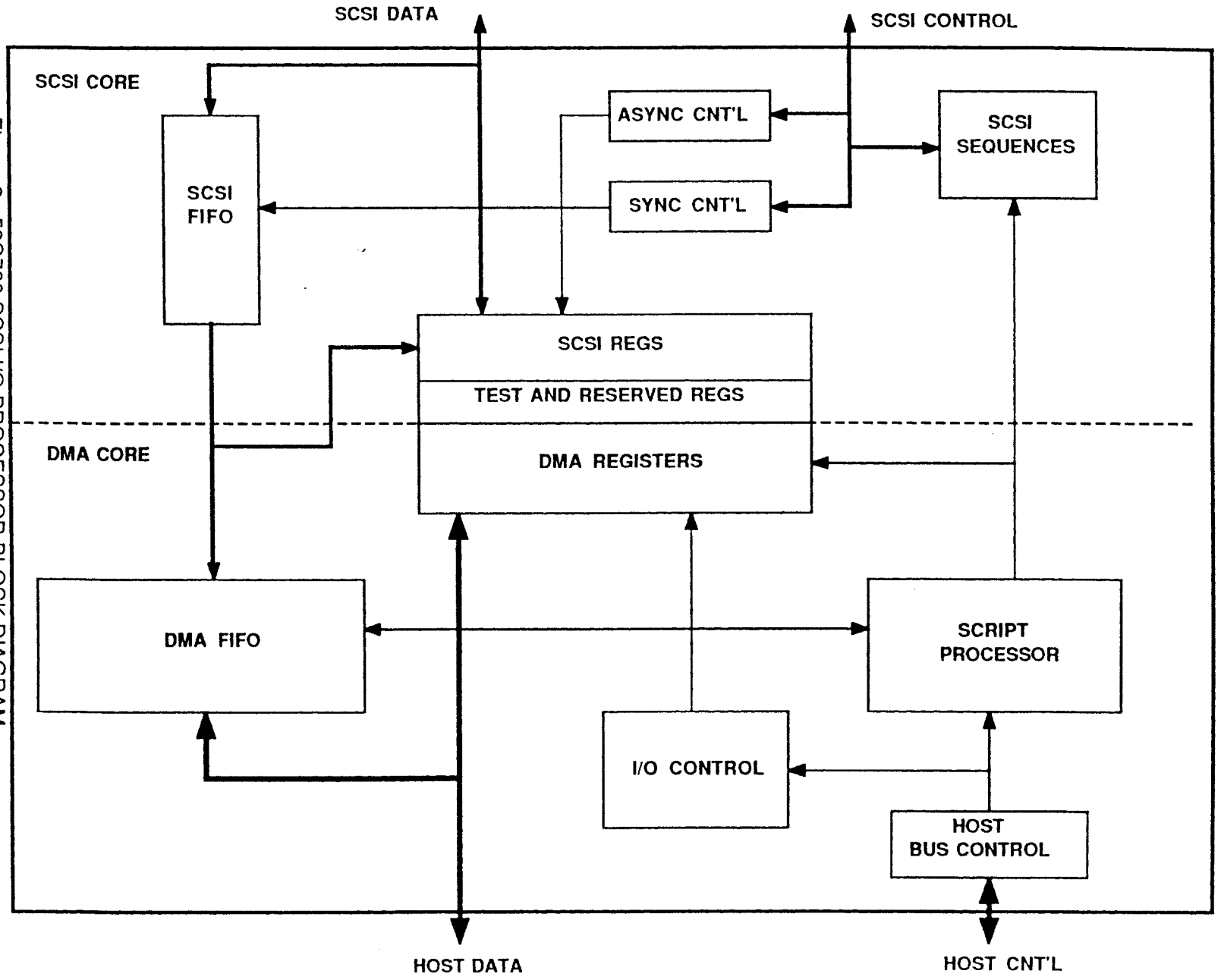


Figure 2. 53C700 SCSI I/O PROCESSOR BLOCK DIAGRAM

NCR 53C700 -- Command

- I/O Instruction -- with SCSI phase and ID, and other controls

  Reselect

  Disconnect

  Wait select

  Set (bus phases)

  Select

  Wait disconnect

  Wait reselect

  Set (expected bus phases)

  Clear ACK or ATN

- Block Move -- with count, address, & SCSI phase

  Move

  Wait Move

- Transfer Control -- with next instruction addr

  Conditional jump, call, return, and interrupt

  Only one return address is available

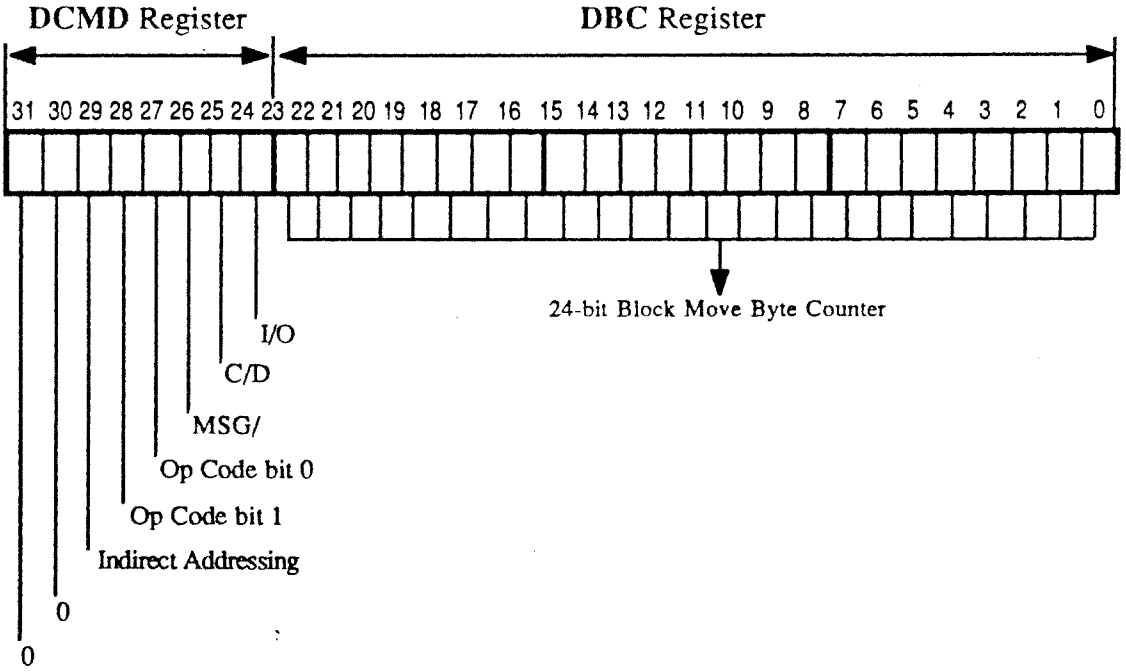
Chapter 5
Instruction Set of the SCSI I/O Processor

The SCSI I/O Processor fetches and executes its own instructions by becoming a bus master and loading two 32-bit words into its registers. This is referred to as executing SCSI SCRIPTS™. The SCSI SCRIPTS™ mode of executing instructions allows the SIOP to make decisions based on the status of the SCSI bus.

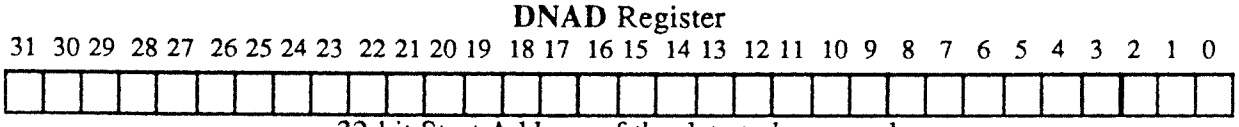
It also off-loads the microprocessor from servicing numerous interrupts. Instructions can also be executed by pipelining them in one at a time using pipeline mode.

There are three types of instructions implemented: Block Move Instructions, I/O

5.1 Block Move Instructions



First 32-bit word of the Block Move instruction

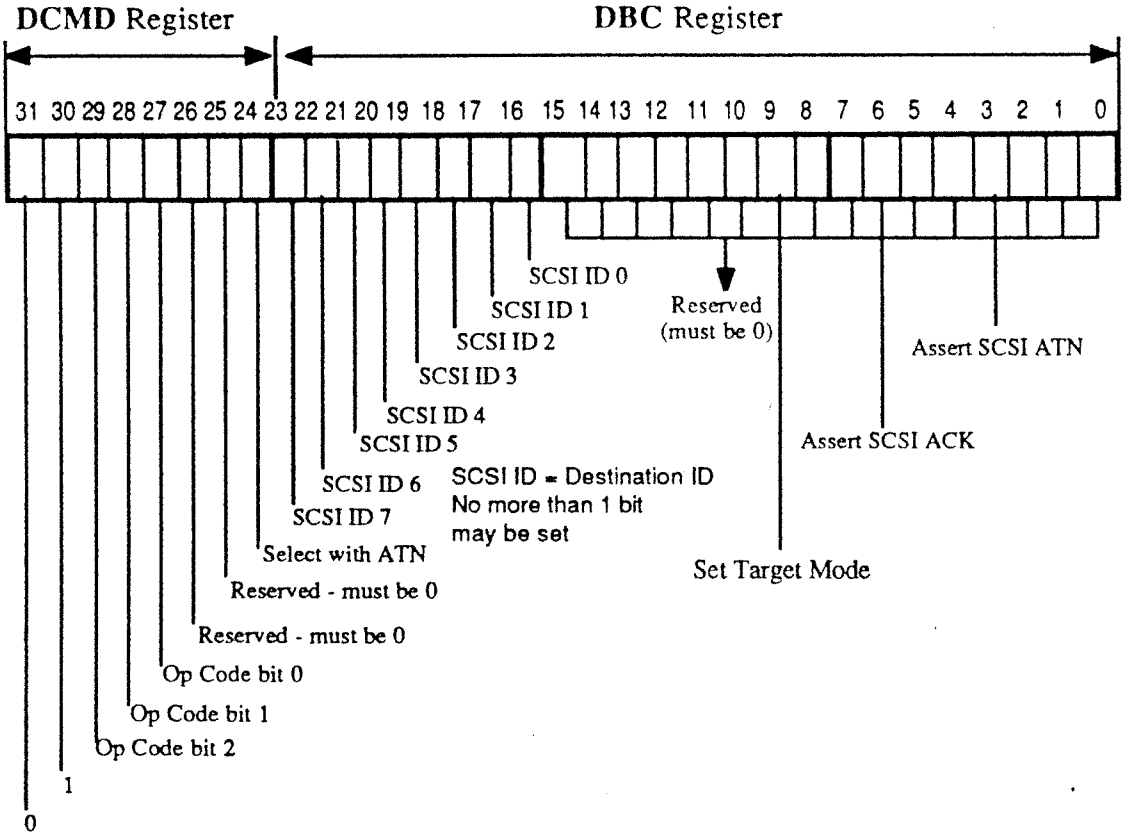


32-bit Start Address of the data to be moved

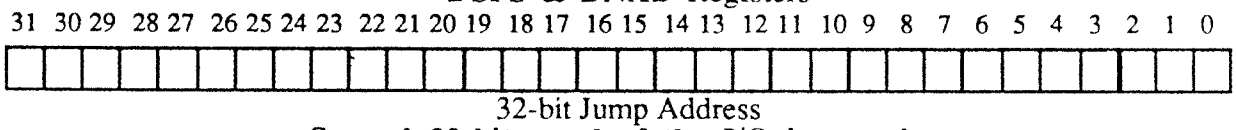
Second 32-bit word of the Block Move instruction

Figure 5. Block Move Instruction Register

5.2 I/O Instructions



First 32-bit word of the I/O instruction



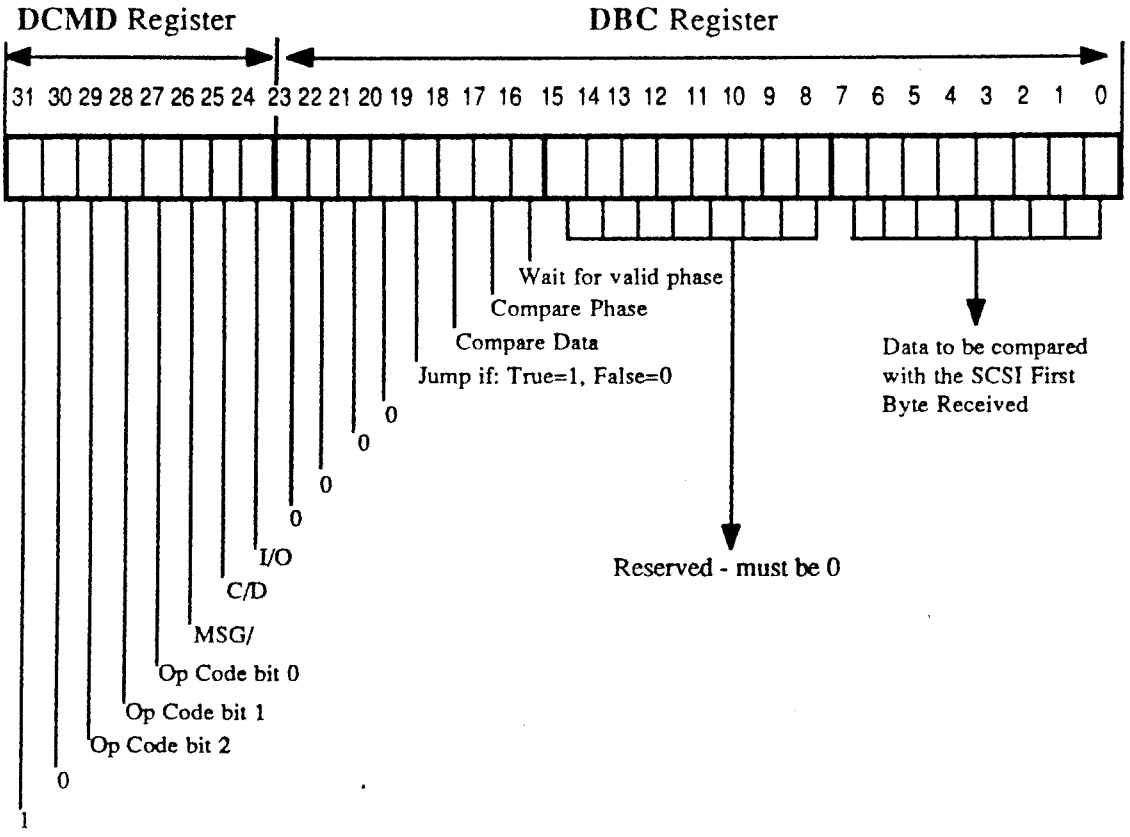
Second 32-bit word of the I/O instruction

Figure 6. I/O Instruction Register

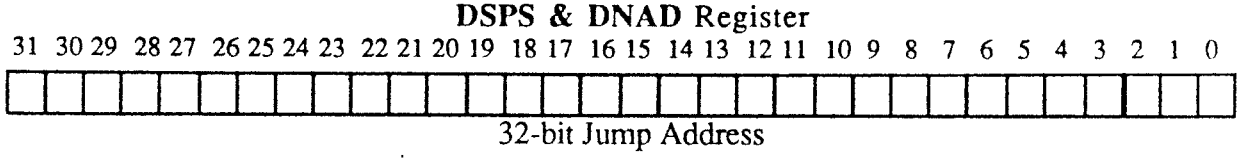
Note
In future generations of the 53C700 family, the second 32-bit word of the I/O instruction will be loaded into the DSPS only.

SII

5.3 Transfer Control Instructions



First 32-bit word of the Transfer Control instructions



Second 32-bit word of the Transfer Control instructions

Figure 7. Transfer Control Instruction Register

Note

In future generations of the 53C700 family, the second 32-bit word of the Transfer Control Instruction will be loaded into the DSPS only.

NCR 53C700 -- Programming

- 11 Steps before starting the scripts.

  Chapter 6 of the manual specifies details.

  SCSI controls, ID, DMA controls, and script execution controls.

- Building scripts

  Perform sequencer functions

    Arbitrate and select

    Transmit identify message

    Transmit command bytes

    Handle possible disconnect and reconnect

    Transmit data

    Handle command complete sequence

  Interrupt processor for abnormal conditions

- Handling multiple devices

NCR 53C700 -- Summary

- The instruction set is primitive.

  Move, IO, and Transfer

- Does the user really wish to program his/her own sequencer code?

  Sequence code is difficult to program

- Not solving the users' problems

  SCSI interface performance

  Multiple device attachments

  Error recoveries

Ideal SCSI Chip

- Hardware handles SCSI sequences and multiple devices; Firmware handles error recoveries.
- SCSI chip + a well-known microprocessor chip with downloadable RAM.
- Use cross compiler and write error recovery in C.
- Integrated DMA function -- initiator.
- Integrated buffer management function -- target.

SCSI Devices -- Disk Drives

- Attributes

  Functionality -- SCSI command set

  Performance

    Command overhead

    Access time

    Data transfer time, synchronous and asynchronous

      Track and cylinder switch time

    Rotational position delay

  Reliability, MTBF

  Advance Functions

    Read ahead

    Zero latency

    Command queuing

  Price

Disk Drives -- 1GB Drive

- Functionality -- SCSI 2

  Very few host adapters use command queuing yet.

  Very few device drivers take AEN

  Log select and sense are optional and not portable

  Has jumper to disable SCSI 2 command

- Performance

  Seek time -- 16 msec in the past, 12 msec today

  Latency -- 5400 RPM, 5.5 msec

  Transfer rate -- 3MB to 5MB, 1msec per 8K transfer

  Overhead -- 0.5 msec

  Total access time -- 16.3 msec or 61 operations per second

- Price

  $1.7/MB today. $1/MB 1992. $.75 late '92.

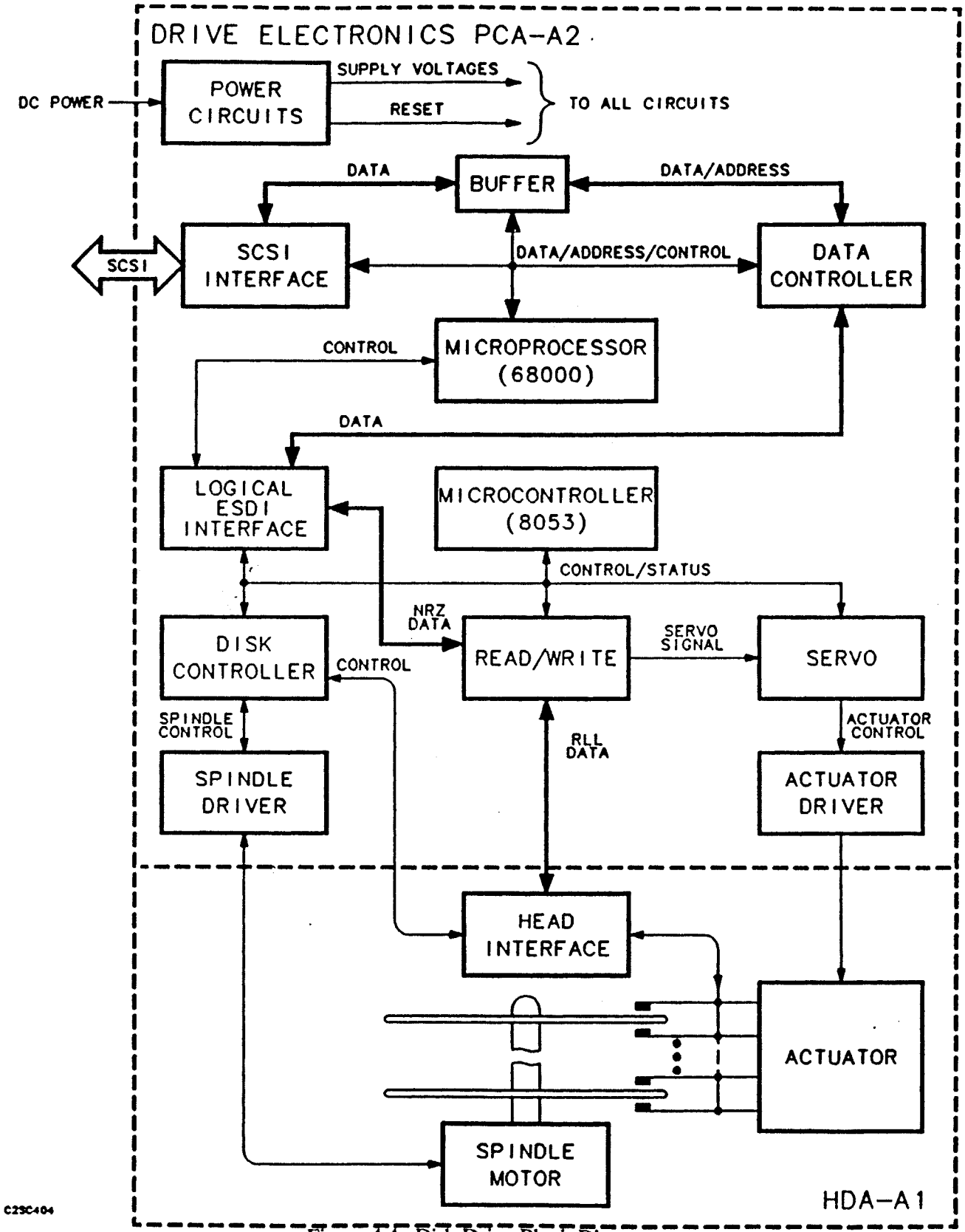


Figure 4-4. Disk Drive Block Diagram

122

Disk Drives -- Data Transfer

- Burst rate vs. average rate

  24 MHz drive -> 3MB per second burst rate.

  Every 512-byte block needs 600 byte recording area.

  average rate = 3MB * 512 / 600 = 2.5MB per second, or 83%

  Average rate is further reduced by track and cylinder switching.

- Asynchronous transfer -- 1.8MB to 2.2MB/sec
  Synchronous transfer -- 5MB to 10MB/sec

  Need synchronous transfer for high-end disk drives.

  Synchronous transfer really help when one has multiple disks on one bus.

- Buffered data transfer -- 128K or 256K

  Data to buffer at 5MB per second; data to disk at 2.5MB per second.

- Multiple disconnection/reconnection for long transfers.

Disk Drives -- Read Ahead

- While completing a SCSI disk read, keep reading disk data to buffer

  Next sequential read reads data from buffer at 5MB per second.

- Large sequential in normal applications

  SCSI drive average access time at 20 msec instead 30 msec.
  (meaning: 50 IO/second instead 33 IO/second)

- No penalty to random reads

  Read ahead should not slow the next command processing

- Very popular

  Most disk drives and SCSI controllers support it.

Disk Drives -- Zero Latency

- Save latency delay on large disk reads or writes.
- Use modify data pointer message to move processor memory address to the first sector under the disk head.
- At the last sector, use modify data pointer message again to move memory address back to the first sector.
- Revert to back to normal read when crossing track or cylinder boundary.
- Only useful for reading close to one track.

  For 2GB drive, one track has about 40K or 80 blocks.

  Can be implemented within the SCSI drives.

Disk Drives -- Defect Management

- No visible defects to user

  One spare sector per track

  Skipping defects -- sector push down

  No defects visible to user when the drive leaves the manufacturing

- Reassign block command

  Defect does not "grow", it is found by the user.

  Additional defects becomes visible because some defects are not caught in manufacturing.

  Find spares on the same track or cylinder first.

  Spare tracks in the center of a drive

- Keeping track of bad sectors

  P-list, primary manufacturing list

  G-list, list of sectors mapped out by reassign block

Disk Performance

- HP 700MB, 200,000 8K transfers, random IO

1. Asynchronous Data Transfer

	Time	Ops/sec	Ops/ch	msec/op	Ratio
1 drive	6814	29.3	29.3	34.1	1.0
2 drives	7236	27.6	55.2	36.2	1.06
3 drives	7716	25.9	77.7	38.6	1.13

2. Synchronous Data Transfer

	Time	Ops/sec	Ops/ch	msec/op	Ratio
1 drive	6643	30.1	30.1	33.2	1.0
2 drives	6820	29.3	58.6	34.1	1.03
3 drives	6962	28.7	86.1	34.8	1.05

Performance -- asynchronous IO

- HP 700MB, 200,000 8K transfers, sequential IO

1. Asynchronous Data Transfer without read ahead

	Time	Ops/sec	Ops/ch	msec/op	Ratio
1 drive	3538	57	57	17.7	1.0
2 drives	3690	54	108	18.5	1.04
3 drives	4885	41	123	23.4	1.38

2. Asynchronous Data Transfer with read ahead

	Time	Ops/sec	Ops/ch	msec/op	Ratio
1 drive	2754	73	73	13.8	1.0
2 drives	3170	63	126	15.9	1.15
3 drives	4025	50	150	20.1	1.46

Performance -- synchronous IO

1. Synchronous Data Transfer without read ahead

	Time	Ops/sec	Ops/ch	msec/op	Ratio
1 drive	3533	57	57	17.7	1.0
2 drives	3543	57	114	17.7	1.0
3 drives	3572	56	168	17.9	1.01

2. Synchronous Data Transfer with read ahead

	Time	Ops/sec	Ops/ch	msec/op	Ratio
1 drive	2176	91	91	10.7	1.0
2 drives	2365	84	168	11.8	1.08
3 drives	2751	73	219	13.8	1.26

1.2GB Disk Drive Comparison

	Wren 7	Micropolis	H.P	Fujitsu	Maxtor
Capacity	1050	1034	1004	1079	1029
Seek	15	14.5	17.5	14.5	13
Latency	8.3	8.3	7.5	8.3	8.3
Xfer rate	15-23	20	22.8	24	17-30
Overhead	1	0.5	2.0	0.5	0.5
Access time	24.3	23.3	27	23.3	21.8
16KB access	30.98	30.78	34.5	29.55	26.98
MTBF	150000	150000	150000	150000	150000

1.6GB Disk Drive Comparison

	Elite	Micropolis	H.P	Fujitsu	Maxtor
Capacity	1352	1408	1363	1620	1503
Seek	11.5	12.5	12	11	13
Latency	5.5	5.5	7.5	5.5	8.3
Xfer rate	24-35	30-40	22.8	38	20-30
Overhead	?	?	1.2	?	?
MTBF	150000	150000	150000	150000	150000

(*Drive firmware are loaded from disk. Some drives provide downloadable firmware by special write buffer command.)

Disk Drives -- 2GB Drive

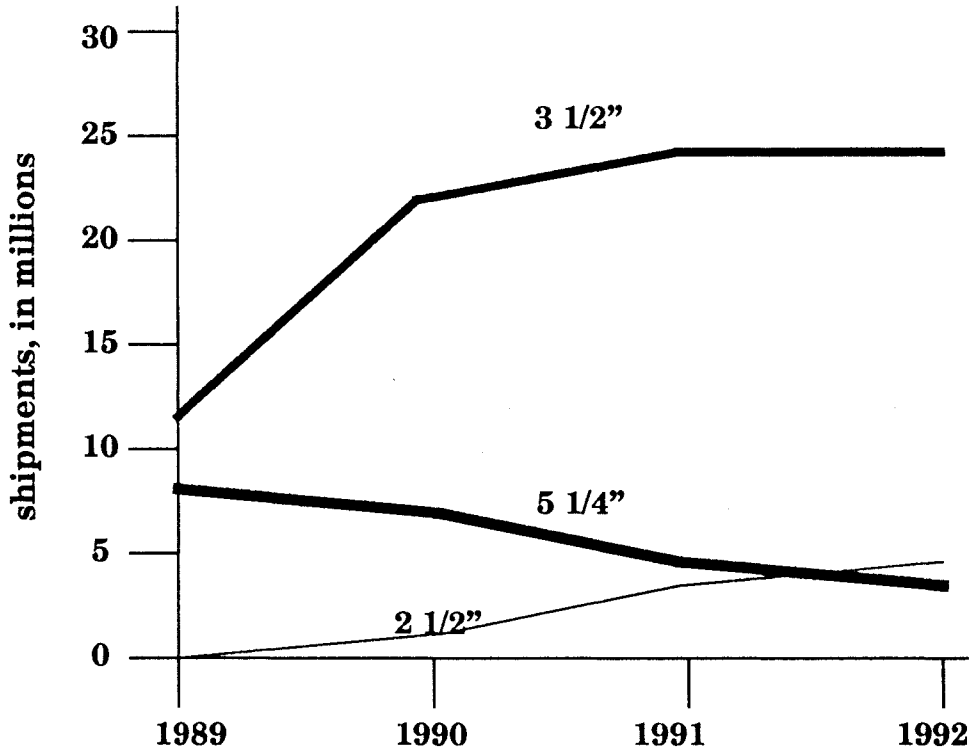
- Track capacity -- 52,000 bytes
- Data heads -- 19 or 20
- Cylinders -- 2000
- Average seek -- 11 ms, track to track 2 ms
- Latency -- 5.56 ms, 5400 rpm
- Data rate -- 4.78 MB/sec
- Average access time -- less than 20 ms.
- Power -- 40W
- Price -- >$1 per MB
- MTBF -- 200,000 hours
- Availability -- 4Q91

Disk Drives -- 3.5" 1GB Drive

- Track capacity -- 30,000 to 48,000 bytes*
- Average seek -- 11.5 ms, track to track 2 ms
- Latency -- 5.56 ms, 5400 rpm
- Data rate -- 2.5 to 4 MB/sec
- Average access time -- less than 20 ms.
- Power -- 12W
- Price -- $1.5 per MB ($1 per MB in 1992)
- MTBF -- 150,000 hours
- Availability -- 4Q91

(CDR -- constant density recording, more sectors on outer cylinders.)

SCSI Disks -- Market



Source: Dataquest

SCSI Device -- Tape Drives

	1/2" tape	1/4" tape
Density (bytes/inch)	800, 1600, or 6250	1000 or 1250
Data rate (KB/sec)	104, 208, or 769	90 or 112.5
Tape speed	130"/sec	90"/sec
Block size	Variable	Fixed -- 512 bytes
Inter-record gap	0.5" or 0.3"	0.012 -- 0.03
Capacity	*	60, 120, or 150MB
Recording tracks	9	9, 15, or 18
Start/Stop time (sec.)	1.7	0.6
Buffer size	512K	56K
Power	170 to 250 W	15W to 33 W
MTBF (20% duty cycle)	22,000	15,000
Cost	$12,000	$500

*Depending on the block size. (For 2400' reel and 1K block at 1600 bpi, the capacity is 25.6MB.)

SCSI Device -- Tape Drives

	8mm -- 8200	4mm
Density (bits/inch)	43.2K	61K
Data rate (KB/sec)	246	183
Tape speed (inch/ sec)	0.429	0.3
Effective head speed	150 ips	123 ips
Stripe size	8K	4K*
Capacity	2.3 GB	1.3 GB
Recording wrap angel	221 degrees	90 degrees
Start/Stop time (sec.)	>0.6	>0.6
Buffer size	256K	512K
Power	15 W	11 W
MTBF	30,000 (10% duty)	50,000 (25% duty)
Cost (OEM price)	$2,000	$1,300

* 22 stripes in a group with group C3 correction.

PRODUCT COMPARISON

	EXB-8200 8mm	MODEL 2600 4mm
Mechanism	8mm video modified for data storage	4mm DAT modified for data storage,computer grade
Form factor	5 1/4-inch full height	5 1/4-inch half height
Recording Density	43.2 Kbpi	61 Kbpi
Track Density	819 tpi	1869 tpi
Areal Density	35.28 Mbpi	114.07 Mbpi
Capacity	2.5 Gbytes	2.6 Gbytes MAX DC Mode
Media	Industry standard	DDS qualified
Tape size	8mm X 112 m	3.81mm X 60m
Cart dimensions	3.75 X 2.5 X .6	2.8 X 2.2 X .38
Media cost/Mbyte	$.016	$.015
Recording Format	Data storage (non standard)	DDS (standard) WangDAT compression format
Data Formatter	Custom LSI design for high performance data recording	Audio DAT LSI plus custom LSI with internal buffer manager and C3 ECC engine
Data Buffer	256 Kbyte with parity	768 Kbytes with parity
Sustained data rate	246 Kbyte/sec MAX	366 Kbytes/sec MAX
MTBF Hours	30,000 hours	50,000 hours
Error correction Code	Interleaved Reed-solomon specific design optimized for 8mm	Interleaved Reed-solomon Audio DAT for consumer Audio plus DDS group C3 ECC.
Read after Write	Read after write with perfect data	Read after write with amplitude threshold and block CRC checking
Correction capability	1024 bytes	5756 bytes
Burst Error capacity	264 bytes (23.4%)	768 bytes (14.5%)
Specified Bit error rate	10^13	10^15
Rewrite Block size	1 Kbyte max	11 Kybtes 2 track min
Minimal Writable unit	1 track or 8 kbytes	DDS 1 group (44 tracks 126 Kbytes) WangDAT compression format group size variable based on compression ratio.
Servo	Specfic design for high performance digitial applications	Processor controlled digitial servo with no adjustments.
Search speed	10X	150X
Search time	17 minutes	60 seconds (20sec average)
Search Capability	Filemark	Record/filemark/setmark
Format Structure data block	Fixed Padding	Fixed and Variable
Format Structure Filemark	2-6 Mbytes	4 Bytes
Tape handling Wrap Angle	221 Degrees	90 Degrees
Power Dissipation	18 Watts	<11 Watts
Load Time	35 seconds	20 seconds with system log update

Tape Drives -- Operations

- Using buffered read/write to keep drives streaming

  With 8mm tape, once the 256K buffer is full, the next write can come within one second. (8500 has 1MB of buffer.)

- Fast search

  4mm tape can search a block at 200 times of its read/write speed.

  8mm tape with "vendor unique" commands can search at 75 times of its read/write speed.

- Multiple partitions -- 4 mm tape drives

  Keep directory in one partition and data in another, and use fast search function.

- End of tape handling

  Recover buffered data

  Early warning error

Tape Device -- System Performance

- Why can't UNIX keep tape drive streaming?

Using dd program to copy 134 MB: (2622 IOs*)

8mm -- 220KB/sec

4mm -- 179KB/sec

4mm with data compaction -- 431KB/sec

*dd copies the whole disk partition.

Using dump program to copy 70MB (1362 IOs**)

8mm -- 170KB/sec

4mm -- 159KB/sec

4mm with data compaction -- 264KB/sec

** dump copies data files.

Restore program only restores 80KB per second.

Tape Device -- System Performance

- System Reinstruct Time (in msec) Histogram

	10	20	40	80	160	320	640	1200	2500	>>
dd	1	280	31	2189	108	12	0	0	0	0
dump	719	46	52	61	149	221	95	16	2	0

- Drive Response time (in msec) -- 8mm

	10	20	40	80	160	320	640	1200	2500	>>
dd	0	0	3	2241	0	0	0	377	0	1*
dump	0	0	2	1203	0	0	0	149	7	1*

*tape start, 38 seconds. -----> *overrun*

- Drive Response time (in msec) -- 4mm with DC

	10	20	40	80	160	320	640	1200	2500	>>
dd	0	0	1642	797	8	157	17	0	0	1
dump	0	0	918	344	25	51	19	3	1	1

Host Adapters -- ST02

- SCSI to PC XT bus
- Single chip solution -- $10 host adapter
- BIOS controls the SCSI chip
- Device driver provides attachments to disk and tape drives.
- Providing connectivity
- For single task operating system

Host Adapter -- ACB1540

- SCSI to PC/AT bus
- Market leader in PC compatible workstations

  500,000 installed. 30,000 sold per month.
- Mailbox interface to device driver

  System informs the adapter where the mailboxes are at power up.

  Mail Box Out -- multiple entries. Each entry have a "Busy" bit and a pointer to a Command Control Block (CCB).

  Mail Box In -- multiple entries. Used by the adapter to return the pointers of completed CCBs.
- For Multi-tasking Operating Systems
- CCB contains SCSI command bytes and pointers to memory locations for data and status.
- Interrupt the other party when the mailbox is changing from empty to non-empty.

ACB1540 -- Functions

- Fetching multiple CCBs to local memory
- Dispatching CCBs by first-in-first-out
- Starting multiple CCBs by allowing disconnection and reconnection
- Supporting SCSI reset and synchronous transfer negotiation
- Supporting scatter-gather of memory data
- Fetching sense data for check conditions
- Using 2nd generating SCSI chip, making up with higher performance microprocessor
- Supporting processor to processor communication?

ACB1540 -- A Total Solution

- Providing required device drivers -- UNIX, DOS, PS/2, Novell, and for tape and CD-ROM drives
- Higher performance by keeping the design simple and straight forward
- Using minimum subset of SCSI commands such that it works with every SCSI device
- Logical device geometry -- 64 sectors and 16 heads for all SCSI disk drives
- Jumper selectable DMA speed

Host Adapter -- DTC 3290

- Adaptec 1540 compatible with 512K to 4M cache
- Is cache essential?

  Certainly not in UNIX

  Only for operating systems without good cache implementation
- Benchmark for IO devices needed

  Operating system overhead

  Synthetic benchmark

  Application programs

Host Adapter -- Auspex SP

- SP -- Storage Processor, SCSI to VME bus

  20 MHz 68020 processor with 256K static RAM

  10 concurrent SCSI channels

  29000 bit slice performing DMA burst transfer

- High performance

  Keeping 20 SCSI devices busy

  With less than 300 microseconds overhead

  Re-drive a SCSI device in less than 100 microseconds

- High function

  20 elevator queues for 20 disk drives -- minimize disk arm movement

  Accepting 300 requests at one time

  Self-configuration at power up

  Complete error recovery

Auspex SP -- Functions

- Downloadable firmware
- Simple interface

  256-entry command FIFO in lieu of mailboxes

  128-byte message which details SCSI command to be executed

  Sending a message by simply writing its address to command FIFO
- Minimum SCSI commands

  Attaching any SCSI devices -- no vendor unique commands
- Simple minded recovery

  Spinning up off-line drives

  SCSI reset and restart
- Virtual disk partitions

  Concatenated, striped, and mirrored

SCSI -- Special Subjects

- Auto-configuration
- Virtual disks or partitions
- Write cache
- Data sharing
- Fault tolerant -- RAID

Special Subject -- Auto-configuration

- Jumperless design

  EEPROM -- saving configuration information, parity enable, disk synchronization, transfer negotiation, etc.

  SCSI IDs -- picking up SCSI ID from connectors

- Inquiry

  Determine device type and model number

- Read Capacity

  Determine the device size

- Mode sense

  Read special device attributes.

- Change Definition

  Switch to SCSI 2 commands

- Downloadable firmware

Special Subject -- Virtual Partitions

- Disk Partitions

  Each disk drive contains many partitions: ad1a, ad1b, ad1c,....

- Concatenated

  A virtual partition which is the concatenation of many disk partitions:

  ad2b, ad3e, ad4f,.....

- Striped

  A virtual partition which consists of equal sized partitions where *striped* data are rotating among partitions:

  ad3b, ad4b, ad5b, ad6b.....

- Mirrored

  A virtual partition which consists of two equal sized *virtual* partitions with identical data:

  vp1: ad2b, ad3e, ad4f,....

  vp2: ad5c

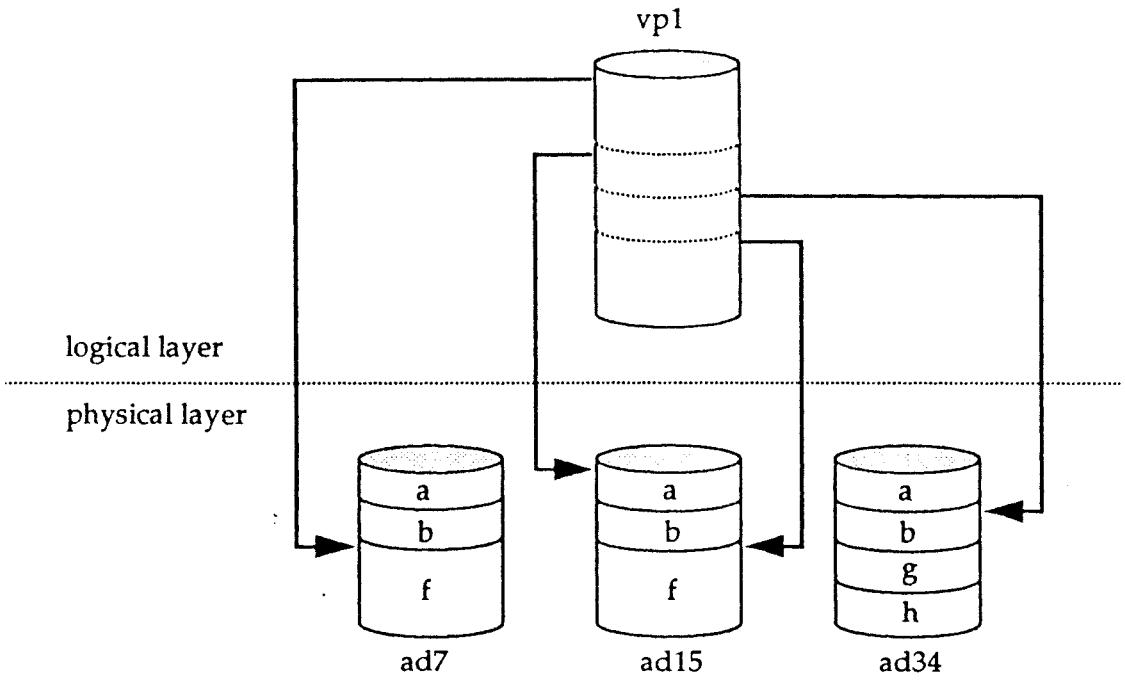


Figure 1: Logical to Physical Mapping of a Concatenated Partition.

2.2 Striped Virtual Partitions

A *striped partition* consists of two to eight physical disk partitions. The striped partition maps its logical address space across multiple physical partitions in fixed-length segments referred to as stripes. Stripe length must be a multiple of 8 KB, ranging from 1 8 KB block all the way up to the whole partition. Striping interleaves access requests to "hot" file systems over multiple drives, thus reducing access latency and improving performance. Figure 2 depicts the mapping of striped partition vp3 onto physical disk partitions. Note that all physical partitions are of the same size, as one would expect since all stripes have the same length.

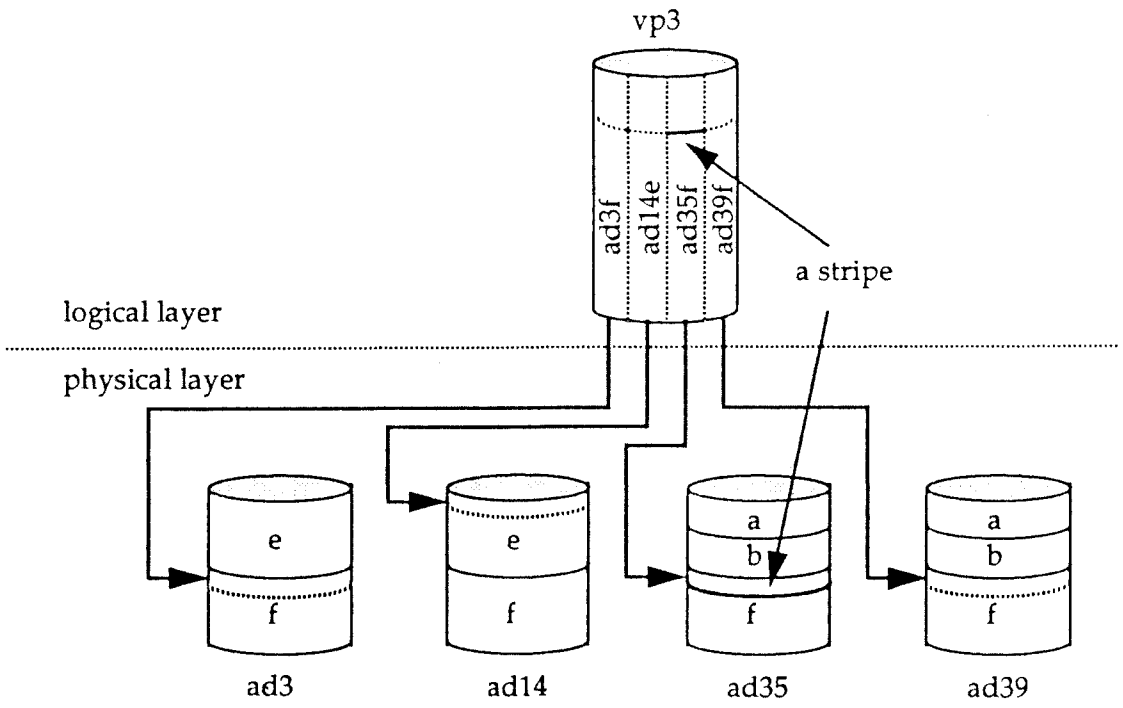


Figure 2: Logical to Physical Mapping of a Striped Partition.

Striping effectively increases the bandwidth to and from the physical disk storage by a factor of *n*, where *n* is the number of member partitions, by distributing filesystem data across all member partitions. This places the data under multiple disk actuators, enabling concurrent accesses into the same filesystem. In an NFS environment, where an 8K read or write operation requires only a single disk drive, the response latencies to client requests are significantly reduced since there are *n* disk queues, instead of one, for servicing the requests. For protocols that allow for very large data transfers in a single request, striping allows *n* data streams to flow between main memory and the disk drives, resulting in very high transfer bandwidths.

2.3 Mirrored Virtual Partitions

A *mirrored partition* usually consists of two members, both of which must be virtual partitions. The primary application of mirrored partitions is to implement high availability filesystems by maintaining duplicate copies of a filesystem on each member of the mirrored partition. In the event of a transient or permanent failure of a member partition's disk drive or its data path, filesystem requests can still be transparently serviced from the other member partition.

A secondary use of mirrored partitions is to enable on-line backups (level 0 dumps) of filesystems to be performed. The procedure for performing on-line backups requires that filesystems reside on single member mirrored partitions. This allows a second member to be attached, synchronized with the first, and then detached and dumped. This procedure is presented in more detail later.

Figure 3 below depicts mirrored partition vp5, comprised of virtual partitions vp1 and vp3. Note that both vp1 and vp3 have the same size, but vp1 is the concatenation of four partitions on three disk drives and vp3 is striped onto four partitions on four disk drives. Obviously, when mirroring is employed to provide high data availability, it is important that none of the underlying physical partitions of vp1 resides on any disk drive that also has any physical partitions belonging to vp3.

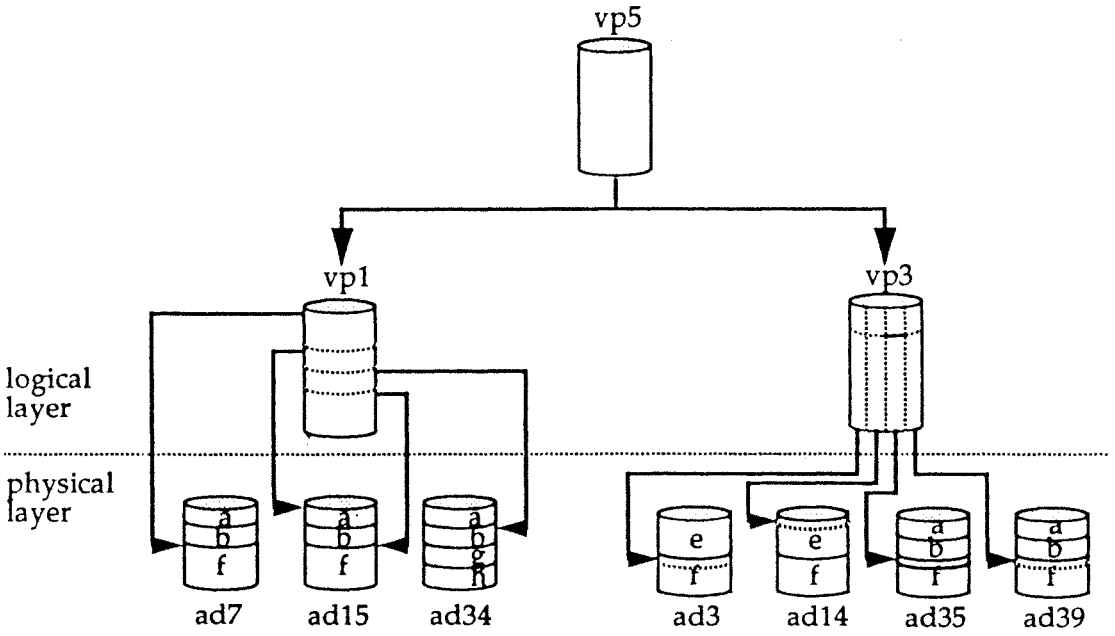


Figure 3: Logical to Physical Mapping of a Mirrored Partition.

Figure 3 illustrates the versatility with which mirrored partitions can be constructed. In practice, both of the member virtual partitions should usually be the same type.

Virtual Partitions -- Pros & Cons

- Pros

1. Creates file systems that is greater than disk physical size
2. Distributes disk accesses evenly to multiple disk arms
3. Protects against disk failures

- Cons

1. If not mirrored, when one disk fails, the whole partition fails.
2. If mirrored, one trades 50% of capacity with availability

(Disks are far more reliable than that of 10 years ago. MTBF is pushing to 200,000 to 300,000 hours.)

Virtual Partitions -- Mirrored

- Operations

  Always write to two partitions.

  Select the idle or less busy partition for reading.

  When a read fails, try the other partition.

  When a write fails, inform the operator and continue the access on the other partition.

- Recovery

  Dirty partition -- two member partitions no longer have identical data.

  Damaged partition -- one partition is inoperational.

  Restoring -- write to two partitions and read from clean one.

  Attach -- Add a second member partition.

  Detach -- Remove a second member partition.

Special Subject -- Write Cache

- Why -- 90% of reads are fetched from cache. Disk read to write ratio is 1:6. Overall average is 1:3.

  Read data are fetched from cache -- less than few msec.

  Write data must reach disk media -- takes about 30 msec.

- Battery backed Non-volatile RAM.

  Allows writes to be complete in less than few msec.

- Overwrite and coalesce

  Directory blocks will be rewritten

  Data blocks are written sequentially (if no fragmentation.)

- Recovery

  When system loses power, data are written at reboot.

  RAM module can be moved to another controller.

Write Cache -- Issues

- How much write cache is sufficient?

  If not enough, disk arm eventually becomes the bottleneck.

- Should write cache be in the drive or controller?

  In the controller to be shared among drives.

- What if write cache failed?

  No different from disk media failures.

  Write to disk ASAP.

  Removable module.

- Overhead

  Very little, if properly designed.

Write Cache -- Results

- Considered by customers as very innovative.

  Help selling products.

- Improves performance by 100 to 200% -- visible to users.

  Write tests completed in 40 seconds instead of 106.

- Increase disk throughput from 30 IOs to 70+ IOs -- depending on the % of coalesce and overwrite.
- Fast response time to *power* users.
- Needed by high MIP work stations.

Special Subject -- Data Sharing

- Multiple initiators

  Reserve and release commands ensure one owner at a time.

  AEN to notify the end of busy condition.

- Third party reservation

  Allows a target to execute the copy command

- Reservation by extents

  Reserves multiple segments of a device

- Usurp reservation

  When the initiator with the reservation dies, use bus reset message.

- Distributed cache

  Lock manger

- Is dual ported SCSI necessary?

Special Subjects -- RAID

- Redundant Array of Inexpensive Disks
- RAID 1 -- Mirrored disks
- RAID 3 -- Synchronized disks + a parity drive.

  Multiple disks operate like a single disk.
- RAID 4 -- Independent disks

  Variable stripe size.

  Allows multiple read accesses.

  Parity drive is "Hot" or very busy.
- RAID 5 -- Independent disks

  RAID 4 with parity stripe being rotated among disks.

RAID -- Pros & Cons

- RAID 1

  Trading 50% of capacity with availability.

  Writes are a bit slow, but reads are faster.

- RAID 3

  Fault tolerant disk -- performance is comparable to a single disk.

  Disk media handling -- ORing defects from all drives.

- RAID 5

  Takes four disk operations for one write -- Reading old parity and data; writing new parity and data.

  Multiple reads.

  It is programming nightmare to update a partial stripe.

  Very slow in recovery mode.

  Performance is low when write to read ratio is high.

Host Adapters -- NCR ADP-92-01

- Making five SCSI drives look like a single SCSI 2 drive.

  68020 microprocessor with 1MB dynamic RAM

  Supporting tagged queuing message

- Supporting RAID 1, RAID 3, and RAID 5

  Selectable stripe size

  Fault tolerant for single drive failure

  On-line data reconstruction

- Downloadable firmware

NCR ADP-92-01 -- Performance

- Concurrent reads to multiple drives
- RAID-1 -- trading 50% of capacity for availability

  Mirrored disks
- RAID-3 -- working like a single disk with 5 times data transfer rate

  But, only 40 IOs per second.
- RAID-5 -- heavy penalty for write operations

  Each write involve 4 disk operations

  When the system has a large cache, read to write ratio can be 1 to 6

SCSI -- Opportunities

- Operating system
  - File systems
  - Asynchronous IO
  - Storage libraries
- Interconnect
  - Physical connection
  - Protocols
- Controllers
  - Host adapters
  - Multiprocessors
  - Fault tolerant
  - Hierarchical storage
- Devices

Opportunities -- Operating System

- File systems
  - 2GB limitation on Berkeley file system
  - On-line size changes
  - Fragmentation -- reorganization
  - File count limitation
- File cache
  - Distributed cache
  - Block oriented vs. file oriented
  - Transfer size -- erase and padding
  - Pre-fetching

Opportunities -- Operating System

- Asynchronous IOs
  - Concurrent transfers from multiple channels
  - Tagged queuing in SCSI 2
- Storage libraries
  - Removable media
  - Timely backup and archival
  - Automated backup
  - Cataloging
  - 3-dimensional file system

Opportunities -- Interconnect

- Physical connection

  Fiber optical SCSI cable -- Is 100MB too slow?

  Differential vs. single ended.

  Cable length.

  P-cable for 16 devices.

- Protocols

  Cache management -- file attributes

  Supporting distributed cache

  Hierarchical storage management

  Lock manager for data sharing

  Return SCSI to standard

Opportunities -- Controllers

- Host adapters
  - How many ports are needed?
  - Tagged queuing
  - Processor to processor transfers
- Multiprocessors
  - Symmetric or Nonsymmetric
- Fault tolerant
  - Multi-ported attachment
  - How to build fast RAID controller
- Hirarchical storage
  - Jukebox -- tape and optical disks.
  - Network data management

Opportunities -- Tape Device

- Firmware overhead -- just as important as disks
- Overlap disk drive reads to tape writes

  SunOS 4.1 -- Asynchronous IO calls, return to caller when IO is started.

- Dumping to multiple drives

  At 200KB, dumping 1GB disk takes 5000 seconds.

  60GB file server takes forever.

- Restore is 3 times slower
- Saving files across network
- Optical disks for infinite storage

  3 dimensional file system -- catalog by time.

- Tape media for archiving

SCSI BUS SIGNAL QUALITY - PROBLEMS AND SOLUTIONS

Kurt Chan, Hewlett-Packard

INTRODUCTION

The SCSI-2 electrical specification has two mutually exclusive transceiver specifications:

1. Differential RS-485 transceivers which allow for up to 10 megatransfers per second at 25 meters, or
2. Single-ended TTL-threshold transceivers which allow up to 5 megatransfers per second at 6 meters.

where each transfer consists of one, two, or four bytes.

For reasons of cost, power, and size, a great majority of SCSI devices use single-ended open-collector drivers, which presents several unique problems to system integrators, especially as systems increase in size and speed. SCSI-2 working groups began addressing these problems in late 1988, and most of the recommendations from the companies involved have found their way into the current SCSI-2 draft document.

One of the most common symptoms of an unreliable single-ended interface is bus misoperation following the addition of devices or cabling to the system, characterized by:

- more failures as the number of devices increases
- more failures as the length of cable increases
- the failures are unpredictable, and not necessarily the same from system to system

This paper discusses the causes of those failures, and the practices which will correct them.

1. DESCRIPTION OF THE SINGLE-ENDED SCSI BUS

Shielded SCSI cables typically contain 25 twisted pairs, 18 of which act as transmission lines carrying low-true TTL signals and 7 of which are for additional grounds and termination power. Typically, each single-ended signal is terminated with a Thevenin circuit (Figure 1). There are exactly two bus terminator networks, one at each end of the bus, each containing 18 220-ohm pullup resistors to TERMPWR and 18 330-ohm pulldown resistors to ground. This biases quiescent signals at 0.6 times the TERMPWR voltage:

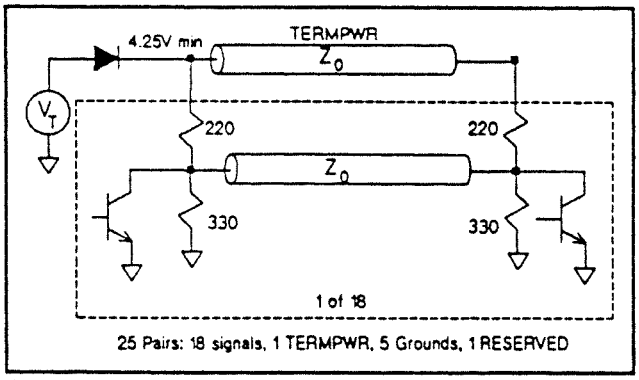


Figure 1: Single-Ended Alternative 1 Termination

$$V_{Thevenin} = V_{Term}\left(\frac{330}{220+330}\right) = 0.6V_{Term}$$

$$R_{Thevenin} = 220||330 = 132$$

The minimum SCSI-2 value for TERMPWR is 4.25V, so the lowest Thevenin equivalent termination is a voltage source of 2.55V in series with a 132 ohm resistor on each signal at each end of the bus. SCSI-1 only requires 4.0V, which results in a 2.4V Thevenin voltage. In both cases TERMPWR must be less than 5.25V. More than one device may supply TERMPWR to the bus via a diode to prevent current backflow, but only one device is required to supply this power. Typically, host adapters perform this function.

Logic levels on single-ended SCSI are compatible with TTL:

$$V_{ih}(\text{min}) = 2.0V$$

$$V_{il}(\text{max}) = 0.8V$$

$$I_{il}(\text{max}) = -400\mu A$$

$$I_{ih}(\text{max}) = 100\mu A$$

$$V_{hysteresis} = 200mV\text{min}$$

$$V_{ol}(\text{max}) = 0.5V @ 48mA$$

Many single-ended interfaces are exclusively open-collector, which makes V(oh) primarily a function of the terminator and cabling characterstics.

2. COMMON CAUSES OF FAILURE

Almost all SCSI bus failures fall into one of two categories:

1. A receiver fails to see a signal asserted/deasserted (missing pulse), or
2. A receiver sees an unexpected signal assertion/deassertion ("extra" pulse).

The first case is usually the result of a bus voltage violation: a deasserted signal fails to achieve a valid high level or an asserted signal fails to achieve a valid low level within the time delays specified by the SCSI standard. The second case usually is the result of reflections caused by impedance and stub discontinuities along the bus.

In case 1, failure to make a clean deassertion (low to high) is the most prevalent failure. While SCSI specifies a 2.0 volt minimum V(ih), it is not uncommon to see systems only pull up to 1.5V or less within the allotted time requirements. In many cases, it is only the conservative thresholds built into receivers that save the system from misoperation. In other cases, implementors have found unconventional methods of dealing with these problems, such as using analog comparators (set to 1.2-1.4V thresholds) to receive speed-critical signals.

The following sections describe the causes of missing/extra SCSI signals, and how to prevent them.

A) TERMPWR VOLTAGE DROP

The SCSI standard requires that the wire carrying TERMPWR be no less than 28AWG. A wire of 6 meters (the maximum single-ended SCSI cable length) would have a DC resistance of about 1.5 ohms. However, it is normal for SCSI devices to add some DC resistance as the TERMPWR wire gets daisy-chained from one device to the next - it is not uncommon to find TERMPWR resistances of 2 ohms or more on maximally configured systems due to device connectoring. When 15 of the 18 signals conduct, the TERMPWR wire will carry nearly 300mA to the far terminators. At 2 ohms, this means TERMPWR at the end furthest from the TERMPWR source will drop about 0.6V during these periods.

Furthermore, most SCSI devices use the conventional TERMPWR design of a TTL supply in series with a diode. Conventional silicon power diodes combined with supply tolerances can easily lower TERMPWR to 4.0V at the device connector (the SCSI-1 minimum). Subtract the 0.6V due to TERMPWR DC resistance, and far-end TERMPWR ends up near 3.4V. This would bias a quiescent signal to (0.6 * 3.4) = 2.0V and leave almost no high-level noise margin on these signals. SCSI-2 increases the minimum TERMPWR voltage to 4.25V.

Another source of TERMPWR voltage loss is in the protection diode required by SCSI to prevent current backflow in the event of multiple devices sourcing TERMPWR. A Schottky power diode such as a 1N5818-5822 will improve voltage losses over conventional silicon diodes.

As we shall see later, low TERMPWR voltages also have an effect on signal integrity by limiting pullup and pulldown transition currents.

B) TERMINATION RESISTOR TOLERANCES AND INPUT SINK CURRENT

One of the first suggestions to improve passive termination without changing any of the resistor values is to use higher precision resistors, and to avoid unnecessary DC loads on the bus (although there are better alternatives to 220/330 termination - see later sections).

As noted previously, SCSI signals are terminated with a 220/330 ohm network that has the Thevenin equivalent of a 132 ohm resistor in series with 0.6 times the TERMPWR voltage at the terminator. For SCSI-1 systems with 4.0V TERMPWR, this results in a bias point of 2.4V and a 400mV high-level noise margin. Two factors not previously discussed can lower this margin even further. The first is termination resistor tolerances. The second is input sink current. Figure 2 shows the relationship between high-level noise margin and these two parameters.

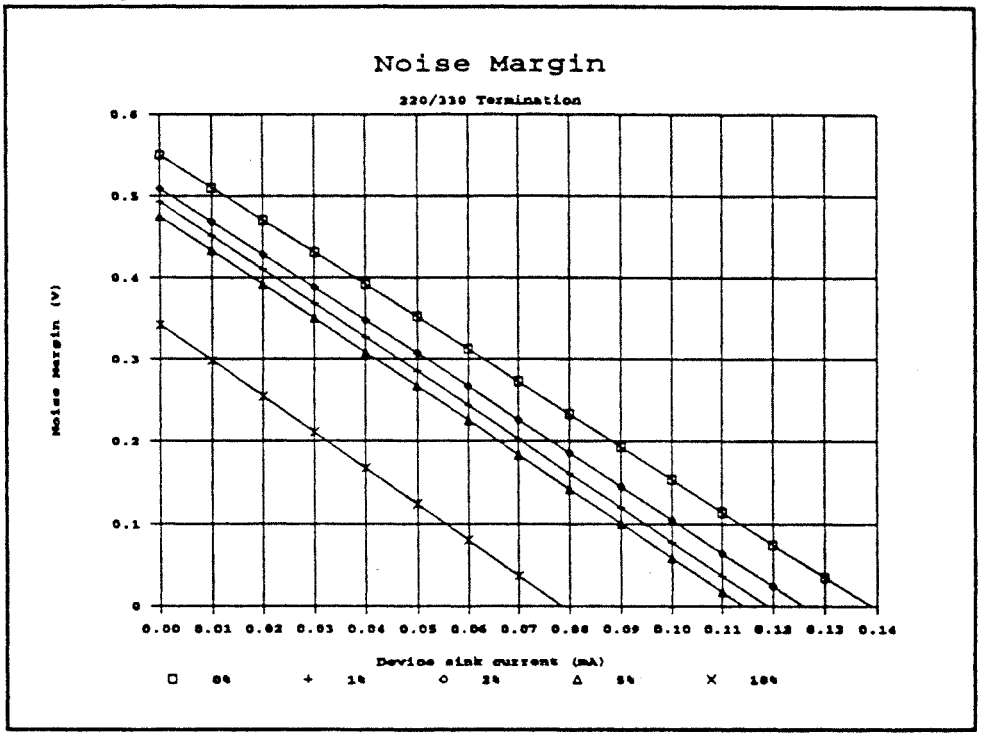


Figure 2 - Noise Margin as a Function of Resistor Tolerances, Input Sink Current

Note that using 1% resistors over 10% resistors adds about 170mV of noise margin under worst-case conditions when the 220 ohm pullup resistor is high in value and the 330 ohm pulldown resistor is low in value).

SCSI-2 has a maximum I(il) specification of 100uA per input per device. This arose primarily from a desire to continue bus operations while one or more SCSI devices are powered down. Many ESD protection schemes consist of diodes from ground to the signal, and from the signal to VDD (Figure 3). The problem is that in the absence of VDD, the signal is effectively grounded through the pullup diode and any series resistance build into the ESD protection circuit, weakening the ability of the SCSI bus to deassert signals in the low to high direction.

Figure 2 shows that about one millivolt of noise margin is lost per microamp of leakage current under worst-case conditions (all 18 inputs on all 8 devices sinking this amount of current).

Since it is not uncommon for devices which lose power to exceed the SCSI-2 recommendation of 100uA, these buses often fail with two or more unpowered devices connected. The solution is to use devices with appropriate ESD protection circuits that do not sink current on powerdown.

C) LOW INTRINSIC CABLE IMPEDANCE

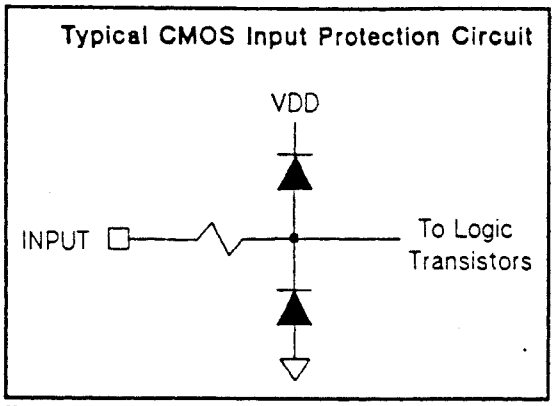


Figure 3

Shielded SCSI cables typically consist of 25 twisted-pairs bundled into a round jacket with a tape and/or a braided shield. Historically, most of these cables have measured in the range of 65-80 ohms (measured on a single-ended time domain reflectometer with the shield and all grounds tied together) due to geometry restrictions and use of conventional dielectric materials. This lower impedance is adequate for smaller SCSI systems but poses problems in larger systems, as we shall see. Recently, new cable technology and processes have demonstrated single-ended impedances as high as 90-100 ohms using 30AWG conductors. Experiments using a variety of single-ended and differential TDR measurement techniques have shown that differential impedances on shielded SCSI cables are roughly 50-60% higher than single-ended impedances.

All round shielded SCSI cables by nature will have a few higher impedance wire pairs near the core (furthest from the outer shield). Some incremental improvement can be realized by connecting the higher-impedance pairs to the speed-critical signals (REQ and ACK in particular). Note also that the LOWEST impedance wire in the cable should be used for TERMPWR to minimize transmission line effects on what is meant to be a voltage supply line. Some SCSI cable vendors have put a low-impedance conductor into the cable specifically for this purpose. Typically, a larger wire gauge along with a higher dielectric constant insulator is used on this conductor.

D) CABLE IMPEDANCE DEGRADATION

Effects of Device Capacitance and Spacing

As devices are added to a SCSI bus, capacitance is introduced to each signal from connectors, receivers, and PC board traces. SCSI-2 limits this capacitance to 25pF. The reason for this limit is that the added capacitance has the effect of lowering the impedance of the section of cable to which these devices are added, as well as adding delay.

Characteristic impedance can be calculated from the intrinsic inductance and capacitance of the cable:

$$Z_0 = \sqrt{L_0 / C_0}$$

where C0 and L0 are expressed per linear measure. A typical value for C0 is 15pF/ft for shielded SCSI cables. Adding devices, each spaced at a pitch of one foot, and each introducing 25pF to the cable signals has the effect of degrading C0 from 15pF/ft to 40pF/ft for that section of cable. The following relationship describes how Z0 is reduced as a function of device load and spacing:

$$Z_0^{loaded} = \frac{\sqrt{L_0}}{\sqrt{C_0 + C_L}} = \frac{Z_0}{\sqrt{1 + C_L / C_0}} = \frac{Z_0}{\sqrt{1 + C_{device} / pitch / C_0}}$$

where

Z_0^{loaded} = *new loaded impedance of the cable*

Z_0 = *intrinsic characteristic impedance of the cable*

C_L = *added capacitance per unit length added by devices*

C_0 = *intrinsic capacitance of the cable*

For the numbers above, Z0(loaded) falls to about 60% of Z0, resulting in some sections of typical SCSI cable falling to 40-50 ohms where devices are clustered. SCSI-2 recommends spacing devices no closer than .3 meters apart in addition to the 25pF limit. Propagation velocities are decreased in proportion to decreases in impedance.

The table below shows the relationship between trace geometry and capacitance for various multilayer PC boards measured at HP. When PC board traces constitute part of the main bus, keeping the SCSI signals on an outer layer away from ground and voltage planes, along with reducing their width, will keep trace capacitance low and impedance up.

<table>
<tr><th colspan="2">4-LAYER BOARDS</th><th colspan="4">6-LAYER BOARDS</th><th colspan="4">8-LAYER BOARDS</th></tr>
<tr><th>.012"</th><th>.008"</th><th>.012"</th><th>.008"</th><th>.012"</th><th>.008"</th><th>.012"</th><th>.008"</th><th>.012"</th><th>.008"</th></tr>
<tr><td rowspan="2">Z=55
C=13</td><td rowspan="2">Z=70
C=10.2</td><td>Z=84
C=8.66</td><td>Z=100
C=7.09</td><td>Z=55
C=13</td><td>Z=70
C=10.2</td><td>Z=88
C=8.27</td><td>Z=100
C=7.09</td><td>Z=73
C=9.84</td><td>Z=89
C=7.9</td></tr>
<tr><td rowspan="2">Z=63
C=11.4</td><td rowspan="2">Z=75
C=9.84</td><td rowspan="2">PWR</td><td rowspan="2">PWR</td><td>Z=69
C=9.84</td><td>Z=81
C=9.06</td><td>Z=46
C=15.8</td><td>Z=58
C=12.6</td></tr>
<tr><td rowspan="2">PWR</td><td rowspan="2">PWR</td><td>Z=52
C=13.8</td><td>Z=63
C=11.4</td><td>PWR</td><td>PWR</td></tr>
<tr><td>PWR</td><td>PWR</td><td>Z=50
C=14.6</td><td>Z=58
C=12.6</td><td>PWR</td><td>PWR</td><td>Z=41
C=17.7</td><td>Z=50
C=14.6</td></tr>
<tr><td rowspan="2">GND</td><td rowspan="2">GND</td><td>GND</td><td>GND</td><td>Z=50
C=14.6</td><td>Z=58
C=12.6</td><td>GND</td><td>GND</td><td>Z=41
C=17.7</td><td>Z=50
C=14.6</td></tr>
<tr><td rowspan="2">Z=63
C=11.4</td><td rowspan="2">Z=75
C=9.84</td><td rowspan="2">GND</td><td rowspan="2">GND</td><td>Z=52
C=13.8</td><td>Z=63
C=11.4</td><td>GND</td><td>GND</td></tr>
<tr><td rowspan="2">Z=55
C=13</td><td rowspan="2">Z=70
C=10.2</td><td>Z=69
C=9.84</td><td>Z=81
C=9.06</td><td>Z=46
C=15.8</td><td>Z=58
C=12.6</td></tr>
<tr><td>Z=84
C=8.66</td><td>Z=100
C=7.09</td><td>Z=55
C=13</td><td>Z=70
C=10.2</td><td>Z=88
C=8.27</td><td>Z=100
C=7.09</td><td>Z=73
C=9.84</td><td>Z=89
C=7.9</td></tr>
</table>

Also note that the low-capacitance requirement ONLY applies to the SCSI signal traces. TERMPWR is ideally transmitted on a zero resistance, high-capacitance trace.

Effects of Unshielded Cable Positioning

A problem with parallel ribbon cable on single-ended SCSI systems involves its positioning with respect to various ground planes created by grounded metal cabinetry. Attaching a 28AWG, .050"-center ribbon cable closely to a grounded metal surface will reduce its impedance. While the free air impedance of such cable is about 105 ohms, direct contact with a metal ground plane reduces it to 61 ohms.

Such impedance discontinuities along the cable will result in reflections as the signal strikes the lower impedance. In the example above, the reflection coefficient between the mismatched cable segments is:

$$\rho = \frac{61-105}{61+105} = -.265$$

As shown in the figure below, for a 1.8V step this causes a reflection back to the source of (1.8)(-.265) = -0.5V. To minimize these discontinuities it is essential that impedance be maintained along the bus by using high-impedance cables and by maintaining the spacing between unshielded ribbon cable and ground planes. The relationship between impedance and spacing is such that most of the impedance gain occurs in the first few centimeters of spacing. Maintaining .25" spacing from ground planes will keep impedance on 105-ohm ribbon cable above about 75 ohms.

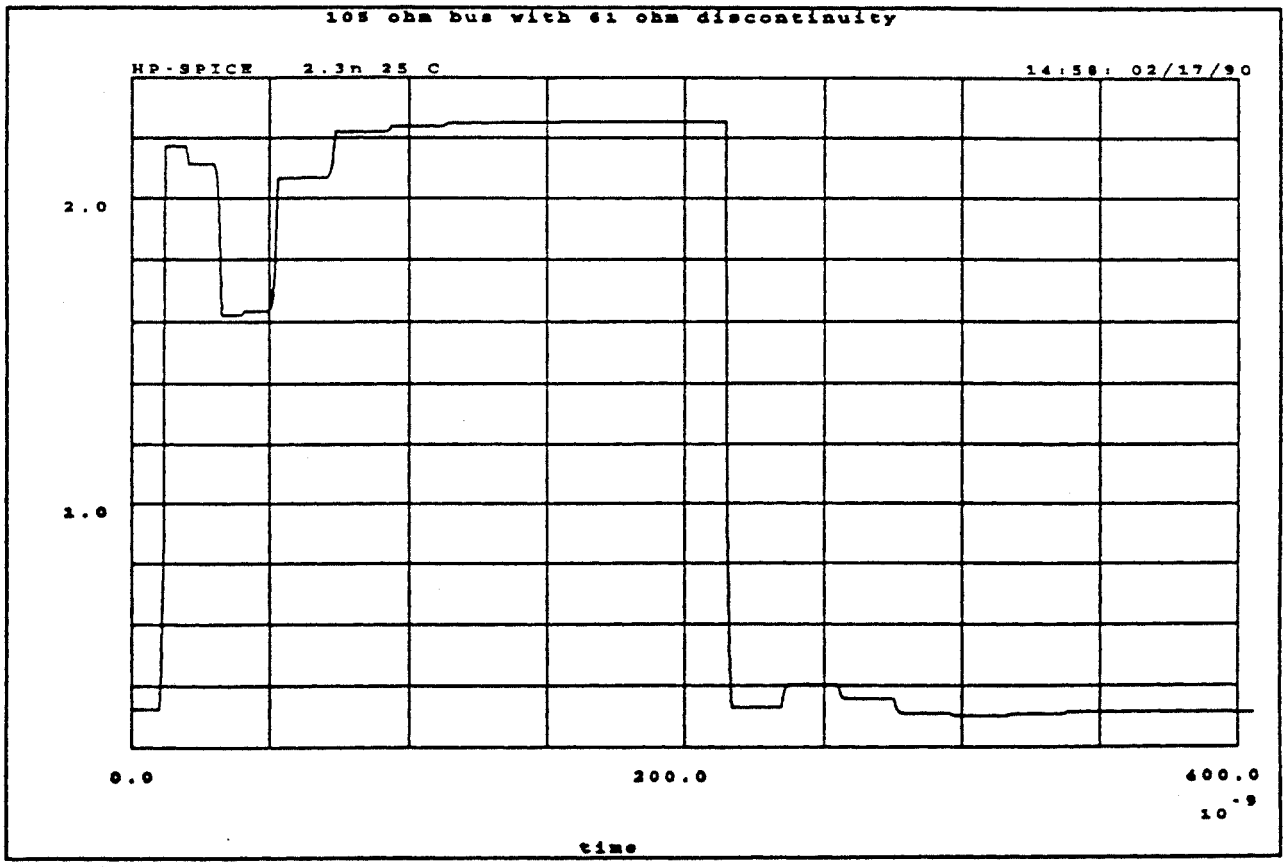


Figure 4

E) LACK OF PULLDOWN CURRENT

We've discussed various causes of cable impedance degradation, but haven't discussed why it is important to keep impedance high. Contrary to popular belief, it is not for the purposes of "matching" the termination impedance to the cable to minimize reflections.

Consider a signal being pulled down on an open-collector bus with terminations at each end (Figure 1). Some of the transistor current comes from the near terminator and some of the current comes down the cable from the far terminator. When the transistor turns off, an instantaneous voltage step at the switching transistor will be propagated down the cable from the near end to the far end. The magnitude of that step is equal to

$$V_1 = V_{ol} + Z_0 \, I_{far\_pullup}$$

where V(ol) is the voltage of the conducting transistor before turn-off, I(far_pullup) is the current flowing through the cable from the far terminator before turnoff, and Z0 is the chacteristic impedance of the cable.

I(far_pullup) can be calculated from V(ol), V(TERMPWR), R(pullup), and R(wire):

$$I_{far\ pullup} = \frac{V_{TERMPWR} - V_{ol}}{R_{pullup} + R_{wire}}$$

Assuming values of:

V(ol) = 0.2V
V(TERMPWR) = 4.25V
R(pullup) = 220+5% ohms
R(wire) = 3 ohms
Z0 = 70 ohms

we get I(far_pullup) = 17mA, and V1 = 1.5V, well below the required V(ih) value of 2.0V.

The signal will eventually achieve an adequate high level, but only after one or more propagation delays to the terminators at each end of the cable where the signal derives its current. The net result is a slow signal "risetime" as the signal bounces back and forth across the cable.

The characteristic rising "staircase" waveforms seen on weakly driven transmission lines is thus observable to some extent on SCSI systems suffering from low pulldown current and/or low cable impedance, although even on the worst SCSI systems the initial step makes it halfway to the 2.0V V(ih) specification. Figure 5 shows a signal being driven onto 60, 80, and 100-ohm cables (60 ohm waveform lowermost, 100 ohm waveform uppermost).

The primary concern on SCSI is with the FIRST step and ensuring that it exceeds the threshold of the receivers. Once a single round trip is required to ensure signal quality on full speed, fully cabled systems, the damage may already have been done with respect to SCSI bus errors and performance. Also, the longer the time spent in the undefined 0.8-2.0V region, the greater the liklihood that a single transition will be misinterpreted as multiple transitions. The problem can be even worst when the signal source is in the middle

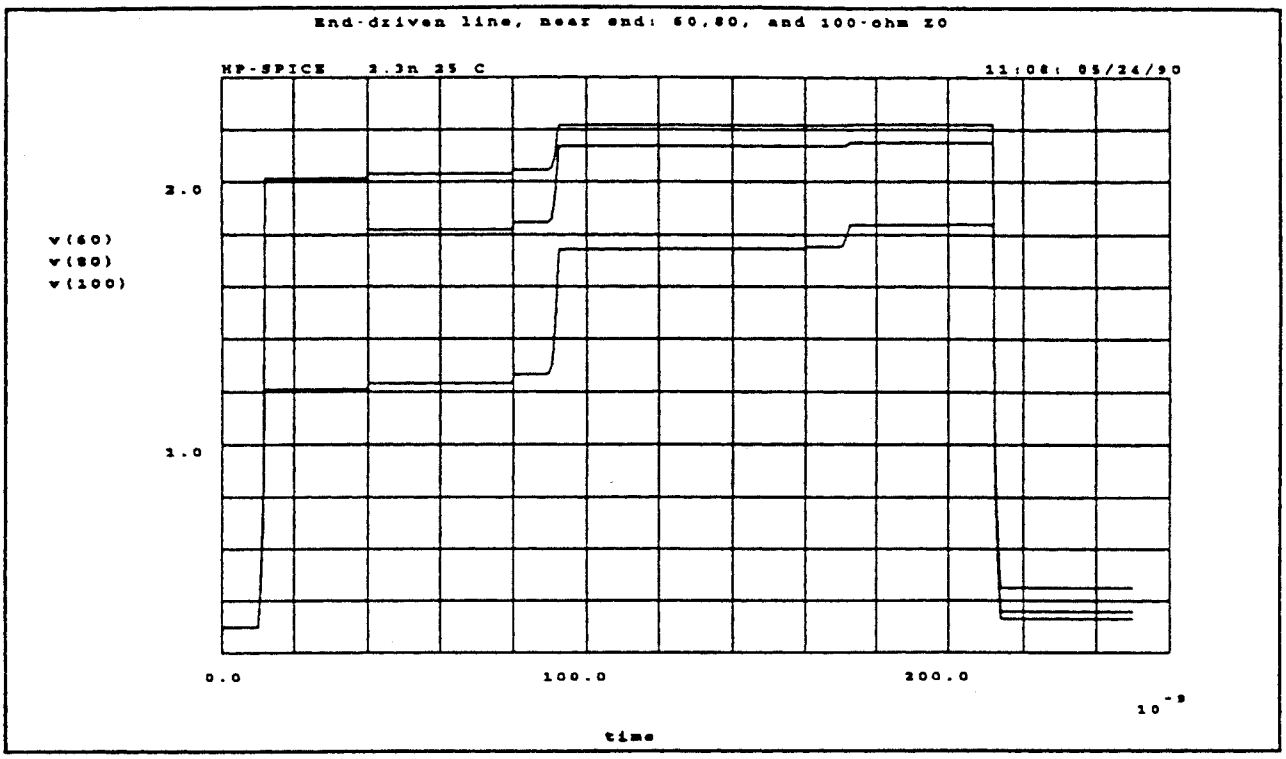


Figure 5

of the cable. The source sees two segments of transmission line in parallel, and the initial step is reduced even further (Figure 6).

Part of the reason SCSI systems work at all are because systems running at slower data rates with shorter cables can get by with lower impedance cables and lower current terminators. Also, TTL receivers typically switch near 1.4-1.5V (the middle of the V(ih) range) rather than at 2.0V.

On systems which require one cable delay to achieve full voltage, if a device at the near end of the cable is sourcing the signal the device ADJACENT to the source will be the LAST to see a full-voltage signal, while the device FURTHEST from the source will be the FIRST if the signal requires one reflection at the far-end terminator to achieve V(ih). This is because the incident voltage to the far-end terminator is below V(ih), and the signal doesn't achieve full V(ih) until it strikes the terminator.

Devices therefore don't "see" a high level on the outbound propagation, but rather see the high level in reverse order as the signal propagates back to the source.

F) UNDERSHOOT AND INADEQUATE LOW-LEVEL NOISE MARGIN

Not all problems are related to signal deassertion. The first high-to-low step at the driving end of a SCSI bus is strictly a function of Z0, internal resistance of the driver, and the quiescent voltage on the bus:

$$V_1 = V_{quiescent}\left(\frac{R_{driver}}{R_{driver}+Z_0}\right)$$

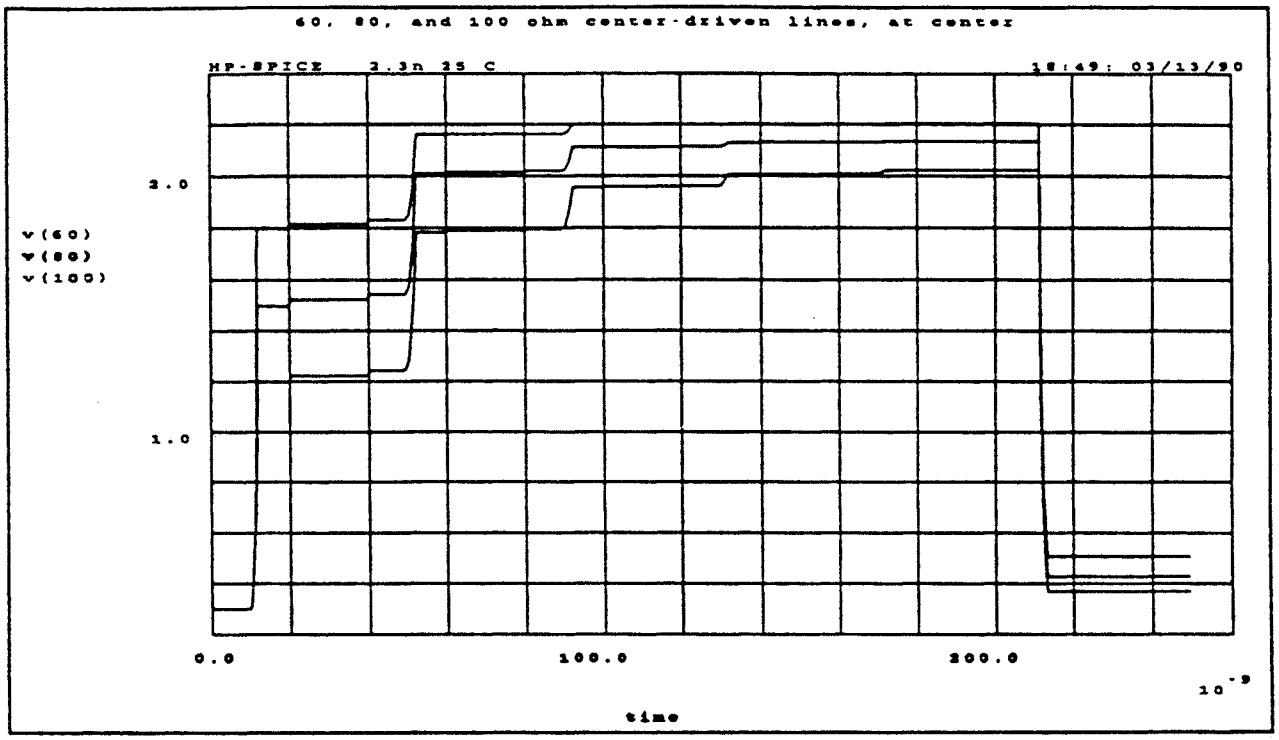


Figure 6

At time t=0 the driver and the cable form a voltage divider. The voltage level of the first assertion step is a function of how strong the driver is driving to ground relative to the Z0 of the cable. Figure 7 shows the falling transition of weakly driven signals on 60 (uppermost trace) and 100 ohm buses. Note that the 60 ohm cable does not fall below the maximum V(il) value of 0.8V on the first step, requiring a cable delay to do so. The steady state low value is dictated by the V(ol) of the driver. A common configuration error involves inadvertantly terminating the SCSI bus with more than two termination circuits. In such cases, V(ol) will rise and also degrade low-level noise margin.

Reflections due to signal assertion can result in false multiple assertions or potentially damaging undershoot. Figure 8 depicts undershoot on an unclamped SCSI signal created by a 100 ohm bus with a 1ns, 50 ohm stub in the middle. The SPICE waveform was taken from the end furthest from the source.

In this particular example, the width of the undershoot pulse is determined by stub lengths, while the magnitude is determined by cable and stub impedances. Note that the first reflection at the far terminator falls to nearly -1V, which may exceed the latch-up voltage for CMOS parts. If the latch-up current is also exceeded for that input, parasitic bipolar devices in the CMOS structure will become forward biased and may damage the device. Fortunately this is usually not a problem, since the latch-up current for the best modern CMOS devices is greater than that which is available from the SCSI bus.

For systems which use modern TTL devices this is not a problem since the intrinsic Schottky diodes on their inputs limit undershoot to -0.4V or so. Adding a Schottky diode such as a 1N5711 will have the same effect and should be used for clamping CMOS inputs which lack adequate latch-up protection.

Another reason for keeping undershoot under control are the subsequent reflections which can rise above the

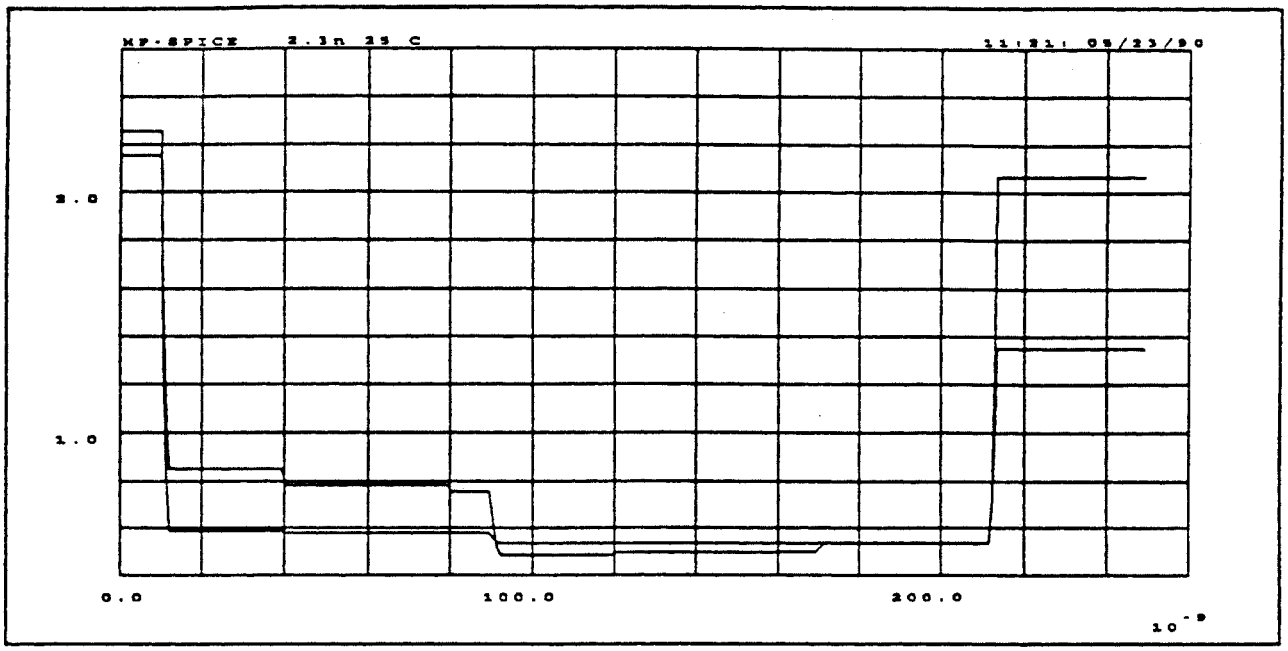


Figure 7

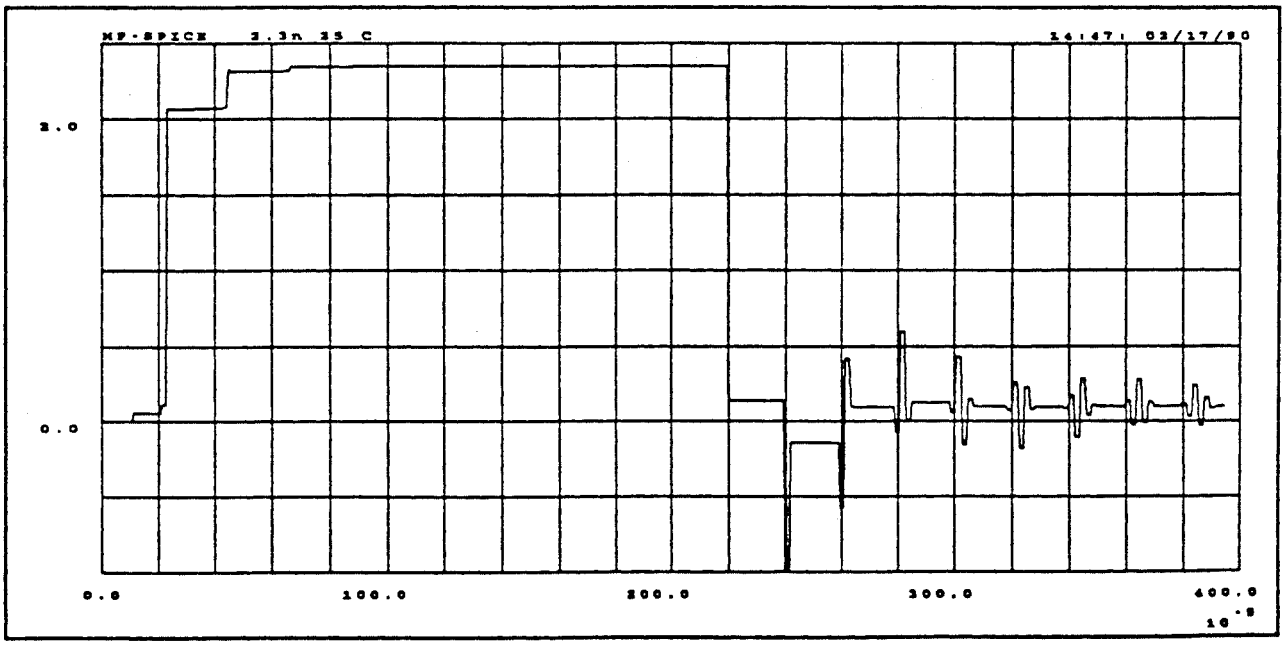


Figure 8

0.8V maximum V(il) when coupled with a high V(ol) driver, resulting in false deassertions. Such Schottky diodes have been shown to significantly reduce ringing on high speed digital signals.

G) LACK OF AC GROUND AT THE TERMINATORS

As we've noted, single-ended SCSI systems often suffer from a lack of high-level noise margin. One practice

that exascerbates this problem is a lack of TERMPWR decoupling, particularly at the terminating resistors.

TERMPWR is carried by a conductor exactly like the signal wires, and exhibits transmission line behavior as well. As with signal wires, shutting off current quickly on TERMPWR will result in a positive voltage pulse being propagated down the cable - the cable makes it impossible for the TERMPWR supply to deliver current instananeously. Likewise, open-collector drivers which conduct will not only pull down the signal, but TERMPWR with it to a lesser degree. These pulses traveling down the TERMPWR wire will appear at the terminators as noise. The solution is to add a capacitor to each end of the cable where the terminators reside. Without the capacitor TERMPWR acts simply as a high-impedance node and couples noise from signal to signal. With the capacitor an "AC ground" exists which filters this noise.

The simplest method of ensuring TERMPWR is adequately filtered at the terminators is to add an electolytic tantalum capacitor of several microfarads to each terminator. SCSI recommends 2.2uF as a minimum value. Experiments have shown that such decoupling capacitors at the terminators dramatically improve signal quality on SCSI systems.

This method may be impractical, however, since most SCSI-1 terminator blocks do not have this capacitor in them and fully encapsulate the termination network. An alternate solution is for ALL devices to add this capacitor to their TERMPWR lines, regardless of whether they are sourcing TERMPWR. This automatically places the capacitor near the bus ends, and allows the use of SCSI-1 terminators.

Note that while having two sources of TERMPWR (one at each end) will tend to minimize the large voltage losses imposed by the DC resistance of the wiring, it is not necessarily a solution for the more subtle AC problems such as this. The reason is that only one TERMPWR source will provide all the current to the bus if it differs from other TERMPWR sources by one diode drop. As long as the protective diodes on the other devices are reversed-biased, they will contribute nothing. Since the AC noise introduced by falling signals on TERMPWR are typically less than half a volt, having multiple TERMPWR sources is not the best way to address this problem - decoupling capacitors are a surer bet. Note that for those devices capable of sourcing TERMPWR, the capacitor should be placed directly on the SCSI Bus side of the diode, not behind it.

H) CROSSTALK

Single-ended Crosstalk

The most prominent type of crosstalk occurs on SCSI cables when quiescent lines are affected by multiple adjacent signals through the mutual capacitance of the signal wires. The near end crosstalk looks something like this:

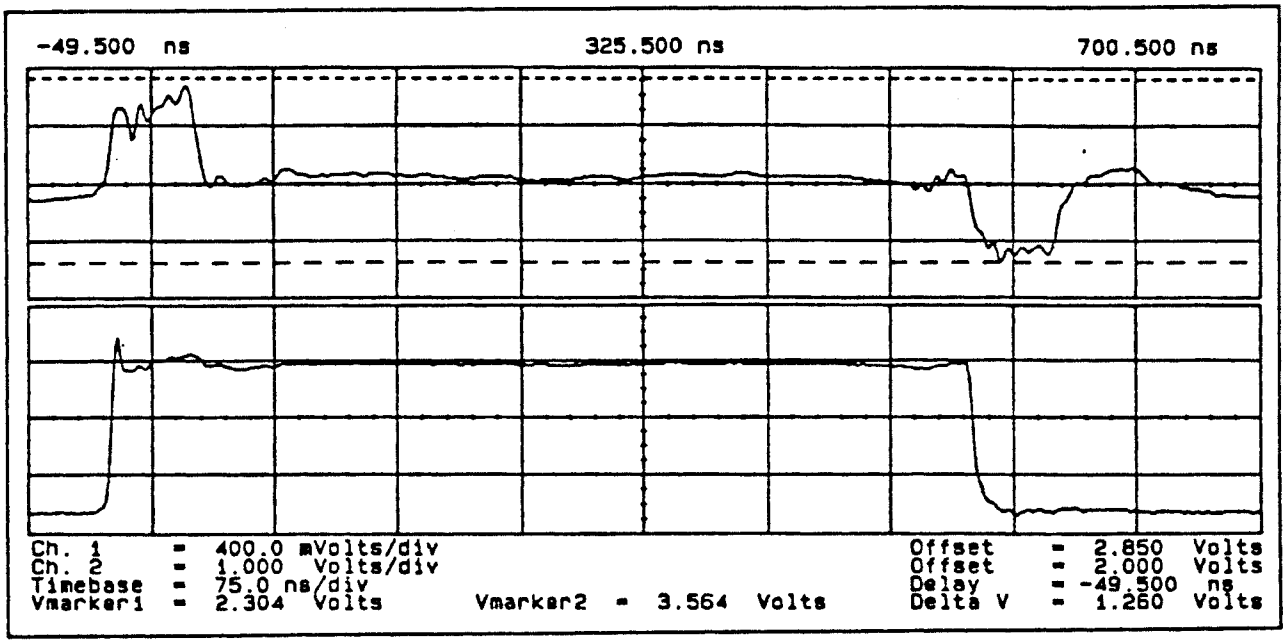


Figure 9

The lower trace shows the driving waveform present on multiple adjacent signals, and the upper trace shows their effect on a terminated but undriven line. On properly terminated signals with TERMPWR decoupling capacitors, the peaks of the crosstalk waveform shouldn't exceed about .75V. The "near end" or "backward" crosstalk pulse width is equal to two propagation delays of the cable, and the amplitude is primarily a function of the driving wave amplitude, and the ratio of the mutual pair capacitance to the intrinsic capacitance of the conductor:

$$V_{near} = \frac{V_{drive}}{4}\left(\frac{L_m}{L_0} + \frac{C_m}{C_0}\right) \quad for\ t_r < 2t_{pd}$$

"Far end" or "forward" crosstalk amplitude is a function of the cable delay and rise time, in addition to the difference of the above ratios:

$$V_{far} = \left(\frac{t_{pd}}{t_r}\right)\left(\frac{V_{drive}}{2}\right)\left(\frac{L_m}{L_0} - \frac{C_m}{C_0}\right)$$

The width of this crosstalk component is only one transition delay. Therefore the amplitude may be about the same as that of near end crosstalk, but the pulse widths are typically much narrower as shown in Figure 10. Note also that the polarity of this crosstalk component will vary according to the relative sizes of the inductance and capacitance ratios.

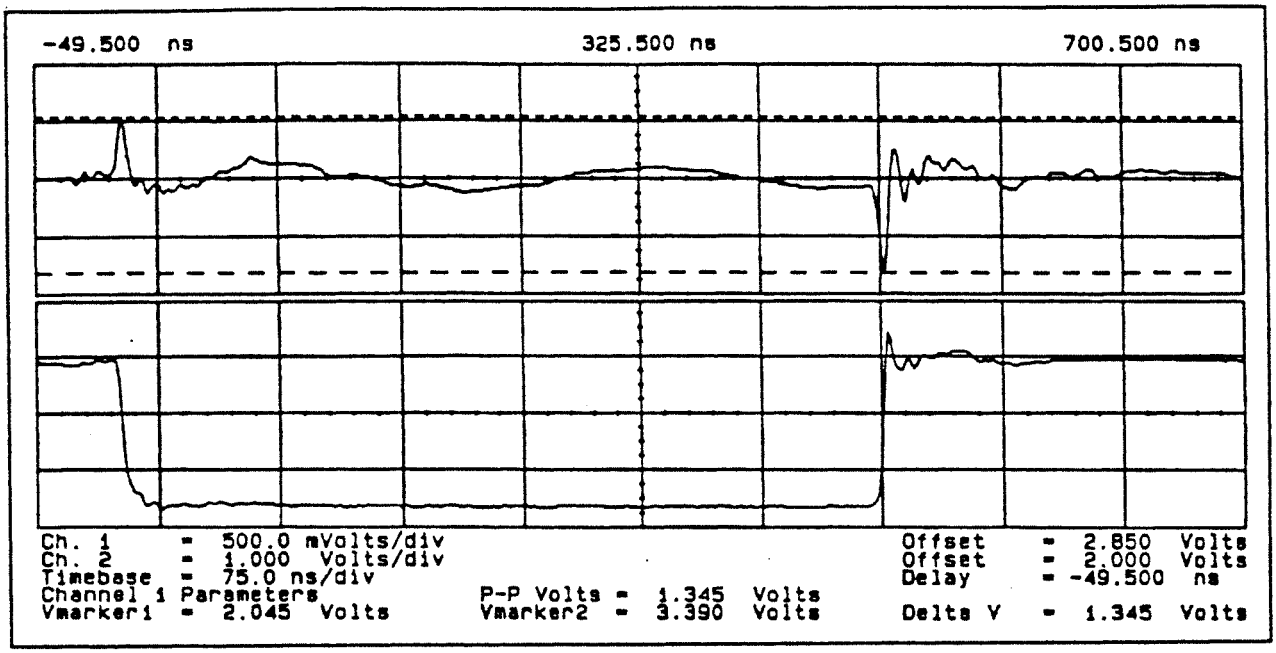


Figure 10

Proper selection of wires within the cable can help to reduce crosstalk. Ribbon cable shows fairly good crosstalk characteristics due to the GND-Signal-GND layout (next page). Shielded cables have several pairs near the shield on the outside wall, and other pairs near the center, away from the shield. Pairs near the shield are therefore typically less susceptible to crosstalk than their interior counterparts. In the diagram at right the crosstalk between the two conductors is a function of the ratio of the mutual and intrinsic capacitances, which is related to the relative spacing of the conductors. As the 'h' dimension increases, C0 will decrease and unless Cm also decreases proportionately, crosstalk will increase.

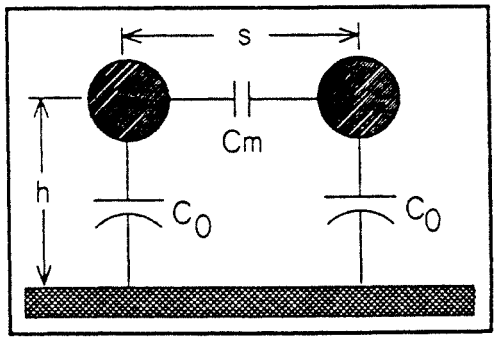


Figure 11

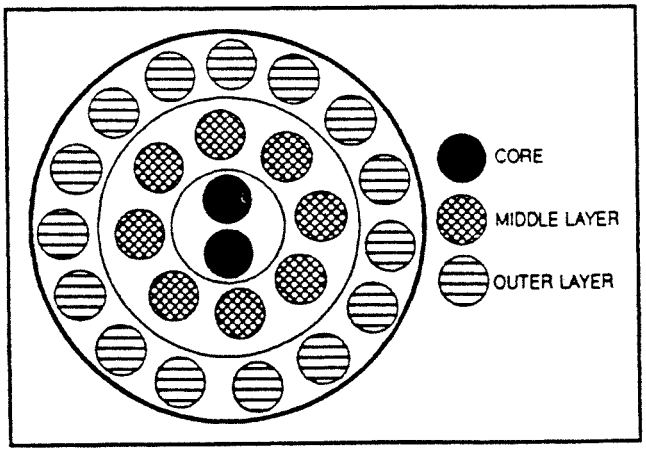


Figure 12

Increasing h is analogous to moving toward the center of a round shielded cable. Ironically, the highest impedance pairs at the core of a SCSI shielded cable are also those that are most susceptible to crosstalk. Therefore when assembling cables, it would be wise to strike a balance between high impedance and crosstalk susceptibility on the timing-sensitive REQ and ACK lines. While inner pairs may be used for these signals, care should be taken to avoid positioning the data pairs adjacent to them.

Testing at HP has shown that one acceptable configuration for round shielded cables involves separating data and handshake lines. Figure 12

shows a common 25-pair shielded cable arrangement consisting of two inner pairs, 8 middle-layer pairs, and 15 outer-layer pairs. Assuming we want the highest impedance wires to carry REQ and ACK at the core, comparing crosstalk on the REQ and ACK lines when all the data lines are toggling reveals several hundred millivolts of peak to peak crosstalk when the data pairs are in the middle layer. In contrast, when the data pairs are on the outside layer, crosstalk on REQ/ACK is almost unmeasureable. Two acceptable configurations are therefore:

Core: REQ,ACK
Middle: C/D, I/O, MSG, ATN, RST, SEL, DB(P), BSY
Outer: DB(0-7) interleaved with TERMPWR, GROUND(4), RESERVED(2)

or

Core: REQ,ACK
Middle: DB(P), TERMPWR, GROUND(4), RESERVED(2)
Outer: C/D, I/O, MSG, ATN, RST, SEL, BSY interleaved with DB(0-7)

Differential Crosstalk on Parallel Ribbon Cable

Parallel ribbon cable is one of the least expensive and most common methods of connecting SCSI devices together. Even SCSI devices which reside in shielded enclosures usually have some form of flat ribbon cable joining the bulkhead connector to the controller board. There are some key considerations in the application of these cables that affect signal quality.

The ground-signal-ground Single-Ended arrangement results in fairly low crosstalk between signal conductors:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
+G	+G	+G	+G	+G	+G	+G	+G	+G	GG	GG	GG	GG	GG	+G	GG	+G	+G	+G	+G	+G	+G	+G	+G	+G

GG = Ground or TERMPWR pair +G = Signal-Ground pair

The reason is that the capacitive coupling of each signal to the interceding ground creates a series capacitive divider. The mutual capacitance between the signals is therefore reduced by about a factor of 2 over a signal-signal layout on the cable.

A common practice for devices which support both single-ended and differential transceivers might be to use the same cable for both, which could be risky if length restrictions aren't observed. Differential receivers are naturally more immune to crosstalk than single-ended receivers when operated over twisted pairs, since any noise coupled into a twisted pair generally appears equally on both wires. Since the receivers generally respond to differences between the conductors of the twisted pair, the coupled common mode noise is rejected.

Differential SCSI on the same ribbon cable would look like:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
+-	+-	+-	+-	+-	+-	+-	+-	+-	GG	DG	GG	GG	GG	+-	GG	+-	+-	+-	+-	+-	+-	+-	+-	+-

GG = Ground or TERMPWR pair DG = Diffsense-Ground pair +- = Differential Signal pair

There are two problems with this arrangement:

1. On parallel conductors noise there is no tendency for noise to be introduced as common mode noise as in twisted pairs.

2. While the single-ended conductor arrangement naturally interleaves ground wires between signal wires, there aren't enough conductors to interleave grounds between each differential signal pair.

Both of these lead to increased crosstalk between adjacent conductors on a ribbon cable. SCSI devices using differential transceivers should therefore limit their use of parallel ribbon cable to prevent excessive crosstalk.

I) WIRED-OR GLITCH

The wired-OR glitch (Figure 13) is caused by current differences that arise between open-collector drivers that are driving the same line (such as BSY). The mechanism is the same as that discussed previously. Consider two open-collector drivers at opposite ends of a 6 meter cable driving the same line low. They are sharing the current supplied by the two terminators at each end of the bus. This current is typically 15-20 mA per driver.

The near driver turns off while the far driver remains on. A rising edge equal in amplitude to the pulldown current prior to turn off multiplied by the characteristic impedance of the bus will propagate down the cable. If I=20mA and Z0=80 ohms, a 1.6V edge will propagate down the bus; enough to cause a false high (deassertion). When the pulse reaches the far driver a cancelling reflection will occur, but devices at the near end will not see the cancellation for two full cable delays - about 65ns on a 6 meter bus. The magnitude of the wired-OR glitch is greater when cable impedances get higher. The glitch only occurs on signal deassertion.

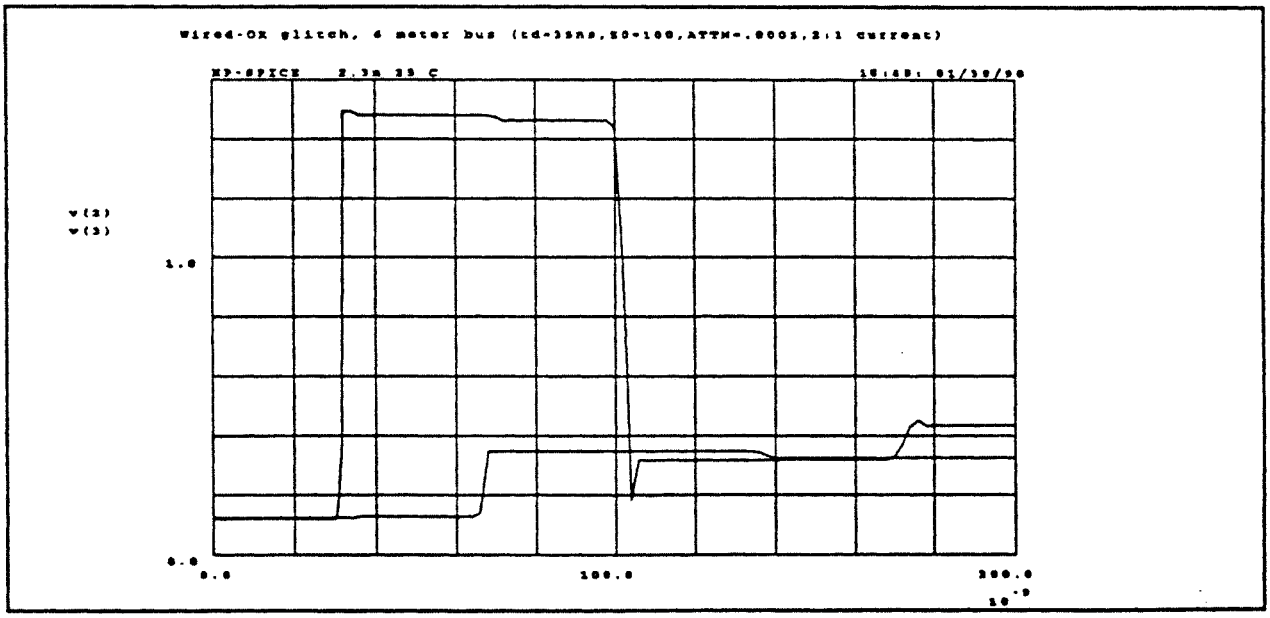


Figure 13

The glitch cannot be suppressed, and it is unwise to "filter" signals on the bus with capacitors. One solution is to use active filtering that does not load the signal itself. Another solution is to simply allow for the wired-OR glitch delay when the possibility exists of multiple open-collector drivers releasing the same line. The SCSI-2 standard describes the scenarios where a bus settle delay (400ns) must transpire before a wired-OR signal is stable. This allows two full cable delays on a 25 meter cable with some margin for bus loading.

3) ALTERNATIVE-2 "ACTIVE" TERMINATION

As we have seen, it would be ideal to lower the resistance of the termination resistors to increase currents. However, for backward compatibility with SCSI-1, open-collector drivers must not be allowed to sink more than 48mA at V(ol)=0.5V. This power rating of 24 milliwatts per driver is what prevents simply lowering pullup values from being a valid solution.

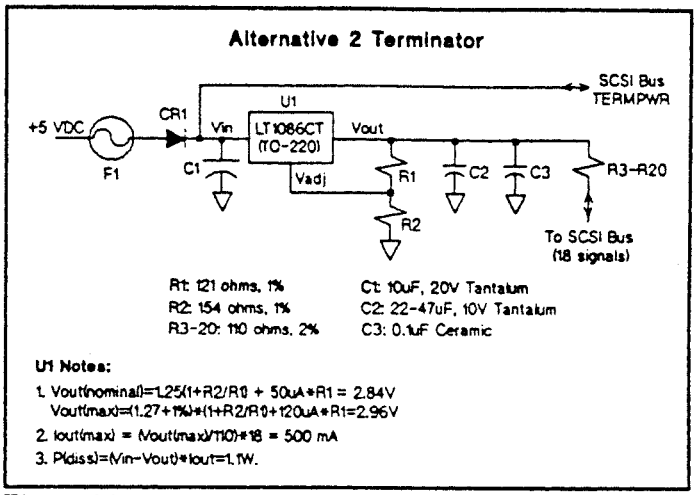


Figure 14

It is possible, however, to lower pullup values if the pullup voltage is reduced proportionally. The SCSI-2 committee recently adopted a new termination circuit designed by Paul Boulay (LMS-OSD) which solves some of the problems described above. The "Alternative 2" terminator is a voltage source termination designed not only to maintain TERMPWR voltage levels across the cable, but also to increase pulldown current.Only one 110-ohm pullup is used per signal per bus end, in contrast to the 220/330 resistor pair required on Alternative 1 terminators. In order to avoid exceeding the 48 mA maximum sink current specification from SCSI-1, the pullup voltage is reduced proportionally to about 2.85V. TERMPWR acts as an input to the low dropout voltage regulator, which produces a stable output voltage to the terminating resistors.

The benefits of the Alt-2 terminator are:

1. The termination voltage (and therefore currents) provided by the terminator are much more immune to IR voltage drops on the TERMPWR wire.

2. The termination voltage is relatively high - 2.85V/110-ohm termination is about equivalent to conventional termination with maximum TERMPWR all the way across the cable, which helps the high-level noise margin of deasserted signals and also helps to keeppulldown currents high.

3. TERMPWR is automatically decoupled.

4. No quiescent current is required, in contrast to Alt-1 termination which requires about 300 mA from a single TERMPWR source even when all signals are deasserted.

It is not uncommon for the resistance of the TERMPWR path on a 6-meter system with several devices to be as high as 2 ohms, which would result in a 0.6V drop with 300mA flowing across the cable. The Alt-2 terminator is immune to such losses, at least up until (TERMPWR-2.85) equals the dropout voltage of the regulator (about 1.1V). Even at the dropout voltage, however, the regulator output will degrade linearly (Figure 15).

Extensive error rate testing has demonstrated that systems with Alt-2 termination are much more immune to failure as TERMPWR is lowered than those with Alt-1 termination.

The average pulldown current is also kept high:

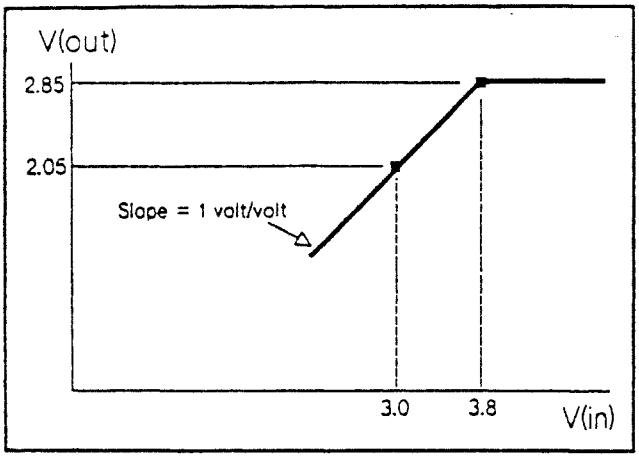


Figure 15

$$I_{pulldown} = \frac{V_{out} - V_{ol}}{R_{pullup} + R_{wire}} = \frac{2.85 - 0.2}{110 + 5} = 23mA$$

For the same Alternative-1 terminator,

$$I_{pulldown} = \frac{V_{TERMPWR} - V_{ol}}{R_{pullup} + R_{wire}} = \frac{4.25 - 0.2}{220 + 5} = 17mA$$

neglecting TERMPWR IR losses.

4) SUMMARY OF RECOMMENDATIONS

CABLING

1. Use high-impedance cables when possible (90-110 ohms single-ended, TDR).
2. Maintain at least 1/8" spacing of ribbon cable from ground planes.
3. Use the highest impedance wire pairs of a cable for the signals, but avoid positioning REQ and ACK pairs close to multiple data pairs in round shielded cable.
4. Limit the use of parallel ribbon cable when using differential transceivers.

TERMINATION

1. Use Alternative-2 single-ended termination.
2. Keep TERMPWR as high as possible at the source (regardless of which terminator is being used). SCSI-2 requires 4.25V minimum, which is too low for high-performance maximum-capacity systems using Alt-1 termination.

DEVICE CHARACTERISTICS

1. Decouple TERMPWR on all devices with a tantalum electrolytic capacitor (2.2uF min).
2. Keep signal capacitance low.
3. Use devices which whose inputs do not sink more than 100uA during powerdown.
4. Use Schottky diodes on latchup-sensitive CMOS inputs to limit undershoot.
5. Observe the 200mV minimum input hysteresis requirement in SCSI-2 (400mV is better).

SYSTEM CONSIDERATIONS

1. Keep stub lengths short to minimize the duration of resulting reflections on the main bus.
2. Allow for wire-OR glitch delays following BSY and RST deassertion.
3. Avoid separating devices by less than about 12 inches of cable.

The Myth of Transfer Rate

The Anatomy of an NFS I/O Operation, Part 1: How and Why SCSI is Better than IPI-2 for NFS

Bruce Nelson
Yu-Ping Cheng

Technical Report 6
January 1991

AUSPEX

ABSTRACT

Disk drives are often dismissed as mundane devices, but they are actually interesting, complicated, and misunderstood. As used in traditional Unix compute servers or general-purpose servers, disk storage subsystems have usually been optimized for excellent sequential-transfer performance. Counter-intuitively, however, NFS file servers exhibit marked *random-access* disk traffic. This report explores these two apparent contradictions. It investigates the anatomy of a client-to-server NFS I/O operation in complete, microsecond-level detail. It debunks the myth of disk transfer rate, and shows instead that disk-drive concurrency is the most important factor in disk storage performance for most NFS network file servers.

The paper presents a careful analysis of both SCSI and IPI disk drives. It begins with a concrete and detailed comparison of both performance-oriented and non-performance-oriented technical specifications of both drive and interface types. It offers a thorough empirical evaluation of SCSI disk drive performance, varying parameters such as synchronous or asynchronous bus transfers, random and sequential access patterns, and multiplicity of drives per SCSI channel. It discusses similar characteristics for IPI-2 drives. Measured disk traffic patterns are presented for several > 5-GB NFS servers using SCSI disk arrays. As can be expected in a 40–80 user file server, request rates are random and independent—not sequential as often benchmarked in general-purpose servers.

The report concludes with carefully benchmarked comparisons of file servers using SCSI-based disk arrays (like the Auspex NetServer family) and IPI-2 subsystems (like the Solbourne 5E/900 and Sun 4/490). The results show that NFS heavy-load throughput using SCSI disk arrays scales linearly with extra drives, whereas IPI-2 throughput scales less than proportionally with extra drives. This SCSI performance advantage, combined with its outstanding storage density, price-performance, and price-capacity, make SCSI disks the superior choice for NFS servers. IPI-2 drives, with their very high transfer rates, remain an excellent choice for compute and simulation servers executing large-file applications where sequential throughput is essential.

Revision 1.1.7, 910121.

Auspex Systems Inc.
2952 Bunker Hill Lane
Santa Clara, California 95054 USA
Phone: 408/492-0900 · Fax: 492-0909
EMail: Info@Auspex.com or uunet!Auspex!Info

A presentation based on a preliminary version of this paper appeared in the *Proceedings of the Sun User Group Conference*, San Jose, California, 3–5 December 1990.

TABLE OF CONTENTS

1 INTRODUCTION AND TERMINOLOGY

1.1 Disks and NFS

Disk drives are complex storage devices. The typical speeds of different disk functions span five decimal orders of magnitude—almost a million to one—from transfer rates of about 10 Mb/s to access times of about 10 ms. Individual disk-drive complexity is compounded when disks are used in computer systems, where *overall* storage performance is governed by hardware-software interactions. Consequently, common disk performance metrics such as transfer rate and access time can be misleading indicators of total *system* performance. This report examines disk subsystem design in the context of a specific, system-level storage application: the NFS Network File System [Sandberg86].

NFS file server traffic has an often-surprising disk access pattern: *random* I/O using small (8-KB) blocks. This is in complete contrast to the disk access patterns of compute servers or stand-alone workstations, which are usually sequential. Recognizing and embracing this random-access NFS traffic pattern is the crucial conceptual element of high-performance NFS storage design.

As mainframe database vendors have long known, random-access disk storage performance is directly related to the number of disk actuators and storage channels.

- An *actuator* is an independently operated disk arm-and-head assembly that can move to a specified disk address and transfer data there. Most disk drives have only one actuator (there are exceptions in mainframe and supercomputer products).
- A *storage channel* or *channel* is a hardware datapath that can funnel data from an actuator's heads to and from main memory. Conventional Unix-system disk controllers usually have only one channel per controller, although two is not uncommon.

Excellent random-access disk subsystem performance requires a fine balance between many actuators, multiple channels, and peak system I/O demands.

The objective of this report is to show that SCSI disk technology easily outperforms IPI-2 and SMD technology in NFS server disk subsystems. While IPI and SMD are outstanding technologies for sequential access, they are not cost effective for high-performance random-access configurations. SCSI disk technology, in contrast, offers a built-in balance of actuators and channels that intrinsically optimizes random access at low cost. Because our system-level goals include price and capacity as well as performance, this report explores these important issues in conjunction with performance.

1.2 Performance as Throughput and Access Time

In this report, throughput is the primary metric of disk subsystem performance. Disk subsystem *throughput* is the capability of the entire subsystem to perform work.

- *Sequential transfer throughput* is usually expressed as an aggregate transfer rate in MB/s. It is typically measured by summing the sequential transfer rate of each disk that can be attached to a separate, available channel. High sequential throughput is useful for the large data transfers (>> 100 KB) characteristic of data-intensive technical and scientific applications executing on compute servers.
- *Random access throughput* is expressed as an aggregate read or write I/O rate on small-block transfers in disk I/Os per second (disk IOPS). It is usually measured by summing the I/O rates of all independent disks on each separate, available channel. In this NFS-oriented

report, a disk's small-block I/O rate is the disk's ability to perform 8-KB block transfers from uniformly distributed addresses across the entire disk.

Sequential throughput favors high transfer rates and is less sensitive to initial positioning (access) times. Random throughput favors fast access times and is less sensitive to transfer rate. Unless otherwise mentioned in this report, disk throughput and disk performance refer to *random access* throughput because random throughput is the essential metric for NFS performance.

1.3 Latent SCSI Skepticism

Readers whose previous experience with SCSI disks has been limited to the Macintosh, IBM PC, or CISC-generation workstation arenas may be skeptical of "high-performance" SCSI claims. These prejudices are justified. Early SCSI drive implementations, and many contemporary low-cost, low-capacity, and/or 3.5-inch implementations, do not basic performance comparable to 8-inch SMD or IPI drive technology. However, current-generation, high-capacity (> 500 MB), 5.25-inch SCSI disks do provide performance comparable to or better than their 8-inch cousins.

Fortunately, the SCSI *specification*—as opposed to early SCSI realizations—permits a wide spectrum of performance implementations. An examination of actual high-performance SCSI drive specifications in section 3 will dispel latent SCSI prejudice.

1.4 The SCSI-1 and SCSI-2 Specifications

SCSI is an acronym for *small computer system interface*. The original SCSI-1 specification [SCSI1] defines an 8-bit-wide bus that operates at 1–2 MB/s asynchronously, and up to 5 MB/s synchronously. The SCSI-1 *common command set* comes in two parts: the mandatory commands like *inquiry*, *read*, and *write* and optional commands like *readbuffer*, *writebuffer*, and *logselect*. Because different vendors often implement non-common—or semantically incompatible common—subsets of SCSI-1 commands, many SCSI-1 drives are software plug-incompatible, especially the high-performance drives with more available commands.

The new SCSI-2 specification [SCSI2] defines 8-, 16-, and 32-bit busses, operating at maximum synchronous speeds of 10-, 20-, and 40-MB/s. Eight-bit SCSI-2 is, therefore, hardware backward-compatible with SCSI-1. The SCSI-2 command set is more precisely defined and functionally richer than SCSI-1. It is also backward compatible with SCSI-1. This has three benefits:

- SCSI-2 drives can initially be used with SCSI-1 hosts and software drivers.
- Different vendors' SCSI-2 drives will be more freely interchangeable because of accurate SCSI-2 common command set implementations.
- SCSI-2 drives, because of more built-in intelligence such as read-command reordering, will behave better in low-end computer systems that may perform no disk optimization in higher-level software (e.g., Macintoshes and PCs).

In this report, SCSI-1 and SCSI-2 can be considered equivalent unless otherwise specified.

1.5 IPI-2 and SMD Functional Equivalence

IPI and SMD are acronyms for *intelligent peripheral interface* and *storage module device*. Because IPI-2 and SMD both have *device-level* disk interfaces (explained later in section 3.4), they are functionally equivalent for the purpose of this report. Furthermore, in addition to the existing IPI-2 disk interface specification [IPI2], a general IPI-3 command interface is evolving that

incorporates controller-level functions similar to the SCSI command set. For simplicity, then, the remainder of the report mentions only IPI disk drives, and uses unsuffixed IPI to refer to IPI-2.

2 WHY NFS SERVERS EXPERIENCE RANDOM DISK TRAFFIC

2.1 A Server's View of NFS Clients

Figure 1 shows a typical large NFS network. From 40 to 100 client workstations are being served from a single NFS server ("NS 5000"). Notice that the server "sees" the aggregate, mixed NFS file traffic as it arrives spontaneously and independently from all workstations on the attached networks. Now consider two facts of NFS implementation:

- *Small blocks are mandatory.* NFS I/O requests are limited to a maximum of 8 KB per read or write operation. This 8-KB maximum is required *by the definition of NFS*—and by the operating implementations of nearly 300 licensed NFS vendors. This means that to transfer a 1-MB file between a client and server, the client must issue 128 *separate* read or write requests, and the server must respond to each and every one of these small 8-KB requests. This consumes substantial network capacity and causes much client and server software overhead—but the NFS protocol permits no shortcuts.
- *Randomized sequentiality.* Even in a best-case scenario where each workstation is *itself* sequentially reading or writing a large file, the server still receives an aggregate request stream that is unordered and random (assuming that most files are not shared, almost universally true in technical environments). A single user's sequential intentions can be lost in the randomness of the whole.

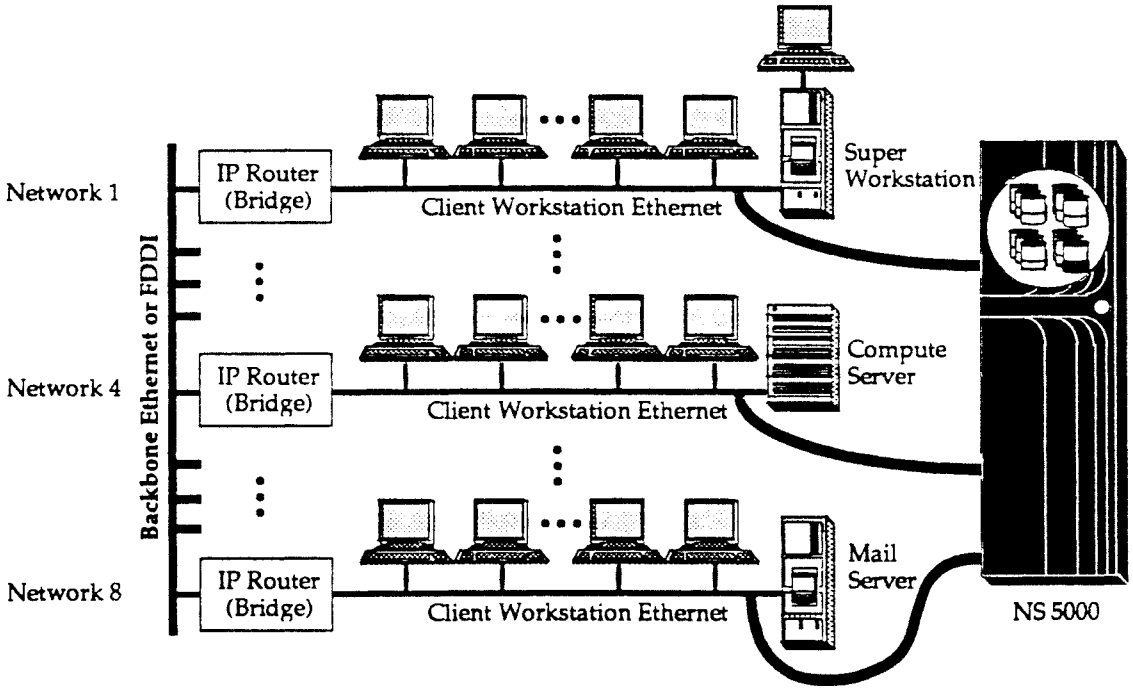


Figure 1: A large NFS network with 50–100 client workstations served by a single NFS server. The key notion is that, in medium to large Unix networks, it is quite common to attach 4–8 Ethernet segments, each loaded with RISC workstations, to a single powerful data server. The Auspex NS 5000 NetServer is one example of this new class of NFS servers. Many organizations are consolidating an overgrowth of traditional, low-NFS-throughput file servers onto large central NFS servers. The reasons are many: up to one-third lower cost per seat; a single point of administration and backup; higher NFS throughput; increased disk and network expandability (scalability); better reliability by excising NFS crossmounts; eliminated free space fragmentation; reduced /usr and project library file replication; and decreased power and floor space requirements.

In total, an NFS server sees autonomous client workstations that collectively and rapidly issue random I/O requests for tiny data blocks.

2.2 The Nature and Limits of NFS Optimizations

Mild NFS performance optimizations are possible. Client workstations can help themselves somewhat by performing read-ahead or write-behind buffering within their own file systems (typically using BIODs). Servers could perform similar optimizations, but are usually constrained by the limited read-ahead policy of the Unix File System [McKusick84, Kleiman86], the simple-and-stateless objective of most NFS servers [Sandberg86], or the synchronous write policy of NFS itself. These complex issues of buffer cache management, per-client file contexts, and fast stable storage (to mask synchrony) are beyond the scope of this paper. They have been extensively addressed in non-NFS distributed file systems such as Sprite and Andrew [Nelson88, Rosenblum90, Howard88] that have traded some reliability for much greater performance. Srinivasan and Mogul [Srinivasan89] have applied some Sprite-inspired optimizations in a "Spritely NFS" prototype with moderate results. Lyon and Sandberg [Lyon89] have shown that fast stable storage for disk-level NFS write buffering can boost performance on write-intensive workloads (writes ≥ 15%).

In practice, the most beneficial performance optimizations have been applied to non-NFS file systems. This demonstrates the limited protocol extensibility of NFS and hence the fundamental importance of a strong, underlying random-access storage subsystem for NFS itself. Keeping NFS servers simple (stateless) has explicitly forsaken some classes of software optimization for improved reliability. This is a worthwhile tradeoff, but it places strong demands on an NFS server's disk subsystem, which remains in the uncomplicated-but-unoptimized critical path of NFS server performance.

2.3 Storage Scalability and NFS

Disk storage *scalability* is a special concern in NFS servers. For example, growing an NFS server from 25 to 50 to 100 workstation users requires not only double and then quadruple the storage capacity, but also *two and then four times the random-access disk I/O rate.* It is this random-access throughput increase, more than any other factor, that limits the NFS abilities of IPI disk technology. As we will see in section 4.3, growing IPI capacity from 2 to 4 to 8 GB usually does not grow the corresponding I/O rates from 70 to 140 to 280 disk I/Os per second (disk IOPS) because of channel limitations (a fast contemporary disk drive is capable of about 35 random disk IOPS). Using SCSI disks, however, channel contention is usually not a problem and linear I/O-rate scaling is easily and economically achieved.

2.4 Storage Subsystem Design Goals for NFS

This section has established that high-performance NFS file servers using standard extent-based file systems need large-capacity, scalable-throughput, random-access-optimized disk storage subsystems. This leads to some natural goals for NFS storage subsystem design:

- Select individual disks with high random throughput (fast access times).
- Design the storage controller to initiate fully concurrent seeks and allow parallel transfers.
- Pre-sort disk seeks in "elevator" queues to minimize actuator time and motion.
- Use disk drives (like SCSI) that allow autonomous seeks and buffered read-ahead transfers.
- Use multiple storage channels that permit parallel, low-latency disk-to-memory transfers.

Finally, all of the above performance goals must be balanced against cost objectives. Price-performance and price-capacity tradeoffs must be made in the context of the storage capacity and NFS throughput requirements of the target file server.

3 SCSI AND IPI DISK DRIVE TRENDS AND COMPARISON

3.1 Form Factor and Price-Capacity

Chart 1 shows the annual decrease in disk price-capacity for four drive form factors—8.0-, 5.25-, 3.5-, and 2.5-inches. Price-capacity is a different metric from price-performance and measures normalized storage costs, usually in $/MB based on large-quantity OEM costs. The important trends in chart 2 are:

- 5.25-inch disks are the industry's current price-capacity leader.
- 8-inch disks are nearing the end of their lifecycle in 1990 as their price capacity ($7.00/MB) is higher than either 5.25-inch ($2.70/MB) or 3.5-inch ($4.00/MB) disks.
- 3.5-inch disks should overtake 5.25-inch in 1992 in raw price capacity, although equivalent performance in ≥ 1-GB sizes will lag 1–2 more years.
- 2.5-inch disks, popular in laptop computers, will remain premium-priced for several years.

Winchester Disk Price-Capacity vs. Form-Factor

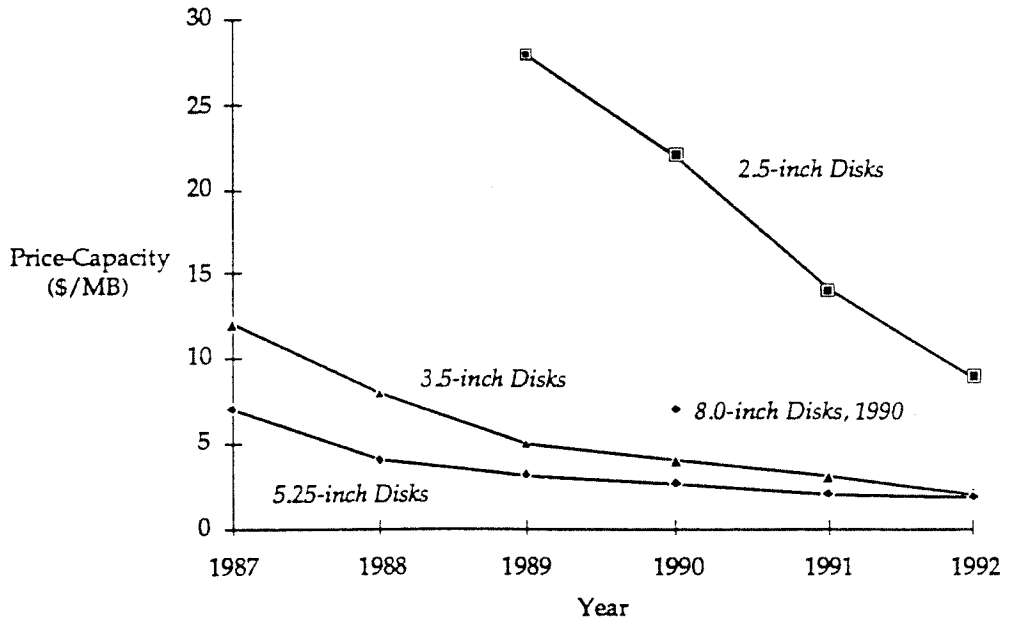


Chart 1: Disk price-capacity versus form factor. Both historical and future price trends are indicated. The 5.25-inch drive is the clear price-capacity leader, and will remain so until 3.5-inch drives are available in ≥ 1-GB sizes. Prices are wholesale, based on large-volume OEM purchases. The fundamental data is from Seagate and Dataquest.

In summary, the highest-capacity SCSI disk drives will be implemented in 5.25-inch form factors for several more years. While most IPI drives shipped today (at the end of 1990) are still in the 8-inch form factor, expect 5.25-inch IPI drives to replace 8-inch in 1991–92 computer shipments. This projection is confirmed by the recent appearance of 5.25-inch beta-version IPI data sheets by manufacturers such as Fujitsu, Hewlett Packard, and Seagate (Imprimis). The market may

even see dual 5.25-inch IPI disks packaged in an 8-inch form-factor enclosure, for backward compatibility with existing cabinets.

Interestingly, the electromechanical portion (head-disk assembly, or HDA) of a single vendor's 5.25-inch SCSI *and* IPI disks will usually be identical in coming years. Only the interface electronics will be different. This implies that the differences in SCSI and IPI manufacturing cost will be based solely on the drive interface electronics. These costs are usually (not not always, depending on volume) more expensive for SCSI because of read-ahead track buffers and more powerful embedded microprocessors. Yet SCSI drive *prices* are expected to remain substantially *lower* than IPI because of intense vendor competition and volume SCSI shipments. The massive scale of SCSI production actually reduces the cost of SCSI manufacturing. These competitive and volume factors will make SCSI the absolute leader in price-capacity, and among the leaders for price-performance.

Non-Performance Metrics	ST81236 IPI	HP 97548 SCSI-1	HP 97549 SCSI-2
Size (form factor)	8.00 inch	5.25 inch	5.25 inch
Formatted Capacity (MB)	1,049 MB @ 3 MB/s 911 MB @ 6 MB/s	663 MB	1,002 MB
Track buffer size (KB)	0 KB	64 KB	128 KB
Typical End-user List Price	$11,500	$5,900	$7,900
Price-capacity	$10.96/MB @ 3 MB/s $12.62/MB @ 6 MB/s	$8.90/MB	$7.90/MB
# of Drives in 19x72-inch rack	24	50	50
Volumetric Efficiency (GB per full-size cabinet)	25.2 GB	33.2 GB	50 GB
MTBF (rated hours)	100,000 hr	150,000 hr	150,000 hr
Power (watts)	95 W	34 W	34 W
Surfaces	15	16	16
Tracks/Surface (std + spare)	1,635 + ?	1,447 + 8	1,908 + 3
Sectors/Track (std + spare)	83 + 1	56 + 1	64 + 1
Tracks/inch	1,289	1,667	1,850
Area Bit Density (K flux/in^2)	32,565	33,953	42,180

Table 1: A SCSI-1, SCSI-2, and IPI-2 HDA comparison on non-performance metrics. Some comments on the metrics: *Formatted capacity* is measured after hardware disk formatting, but before Unix *newfs* formatting. Parallel-head drives such as the two-head 6 MB/s IPI-2 drive usually lose capacity in multihead operation. This is because more embedded control information is consumed at hardware formatting time (e.g., the union of bad spots, a doubling of spare sectors and tracks, etc.). This is why the 6 MB/s drive has lost 10% of its capacity. The advantages of SCSI's *track buffers*—not now available in IPI—are discussed in section 3.4. 8.0-inch drives need more *power* because they generate significantly more heat than 5.25-inch disks since their larger mass requires a larger motor, bigger actuator, and causes more platter-to-air friction. *Spare tracks and sectors* are assigned during hardware formatting to allow the disk controller to remap bad spots with minimal performance impact.

3.2 SCSI and IPI General HDA Comparison

Table 1 is a comparison of head-disk assemblies (HDAs) from two manufacturers: 5.25-inch SCSI from Hewlett-Packard [HP89, HP90]—as used in Auspex's file servers and as high-performance external workstation disks—and 8-inch IPI from Seagate [Seagate90]—as used by Silicon

Graphics, Solbourne, and Sun in their general-purpose servers. The HDA is the electro-mechanical part of a disk drive, consisting of platters, spindle, motor, seek actuator, arms, read-write heads, etc.

Reviewing table 1, some expected results are the slightly larger storage capacity of the standard, single-head 8-inch drive, its 3-times-higher power consumption, and its 50% higher price-capacity. Some possibly counter-intuitive results are:

- *SCSI's volumetric efficiency*. SCSI has twice the storage capacity of IPI in similar cabinets. This is a result of both smaller-form-factor drives and lower-power-dissipation drives, which can be packed tightly in rectangular arrays without cooling problems. This volumetric efficiency can result in great floor-space (footprint) savings, typically 4-to-1 because heavier IPI drives are often packaged into half-height low-boy cabinets.
- *SCSI's higher recording density*. The 1-GB SCSI drive has about 25% higher recording density than either the 663-MB SCSI or 1-GB IPI drives. Recording density and miniaturization are the primary technology thrusts of drive manufacturers, so SCSI's advantage here represents a certain amount of leap-frog activity. The important point is that competitive factors will push 5.25-inch SCSI technology the farthest for the next few years, and that consequently IPI will move from its current 8-inch to the new 5.25-inch HDA platform.
- *SCSI's higher MTBF*. Several high-performance SCSI drive vendors state 150,000-hour MTBF figures, 50% higher than IPI's MTBF here. Many vendor-specific factors determine MTBF: design-for-reliability, manufacturing precision and quality, test procedures, and form-factor-related issues like heat (which decreases with diameter) and HDA tolerances (which become easier to achieve with smaller diameter).

3.3 SCSI and IPI HDA Performance Comparison

Table 2 is a performance comparison of the two HDAs reviewed in the previous section: 5.25-inch SCSI from Hewlett-Packard and 8-inch IPI from Seagate.

A review of table 2 discloses generally minor differences: IPI has a smaller average-seek time and SCSI has a smaller short-seek time. SCSI has smaller rotational latency because it spins about 400 RPM faster than 8-inch IPI. Command overhead times (which reflect controller software and hardware overhead) have similar estimates. Single-head transfer rates are similar, since both drives are limited by the same magnetic recording physics. This leads to two final conclusions:

- *Similar small-block I/O rate (random-access throughput)*. For both drive types, the sum of average seek, rotational latency, and 8-KB transfer times is nearly identical. This leads to essentially equivalent random-access throughput of about 35 IOPS on a single-drive, single-head (i.e., 3 MB/s) basis. This means that factors other than performance—for instance, price and reliability—should determine drive choice for random-access applications like NFS.
- *Superior IPI transfer rate (sequential-transfer throughput)*. IPI drives are readily available with parallel-head transfer capability. By multiplexing the individual data streams of two or three read/write heads, aggregate transfer rates of 6 or 9 MB/s (or more) are possible. (Parallel-head SCSI drives could be easily built, but are not available today.) This high IPI sequential throughput can be important for the data-intensive, large-file applications typically run on supercomputers and minisupercomputers. Such throughput is largely wasted using NFS, as we will quantitatively discuss in section 4.1.

Performance Metrics	Seagate ST81236 IPI-2	HP 97549 SCSI-2
Command Overhead	500 µs est.	500 µs est.
Transfer Rate (raw disk head MB/s)	3.0 or 6.0 MB/s	2.85 MB/s
Transfer Rate (sector-to-sector MB/s)	2.58 MB/s @ 3 MB/s	2.22 MB/s
Seek (average)	15 ms	17.5 ms
Seek (short—track-to-track)	4.0 ms	3.5 ms
Rotational Speed (RPM)	3,600 RPM	4,002 RPM
Rotational Latency (1/2 revolution)	8.3 ms	7.5 ms
Seek+Rotate Time (average)	23.3 ms	25.0 ms
Data Transfer Time for 8-KB (ms) (sector-to-sector)	3.2 ms @ 3 MB/s est. 1.6 ms @ 6 MB/s	3.7 ms
Random I/O Rate (8-KB I/Os/second)	37.7 IOPS @ 3 MB/s 40.2 IOPS @ 6 MB/s	34.8 IOPS

Table 2: A SCSI-2 and IPI-2 HDA performance comparison. Some comments on the metrics: *Command overhead* for IPI drives is dependent on the controller, and is thus estimated here. For SCSI drives, command overhead is usually a specification published by the manufacturer. The HP 97549 drive used here is so new that HP is still estimating the final figure. *Transfer rate* is an intricate, complex number, because the raw transfer rates quoted by vendors usually exceed the sector-to-sector data rate by 15–25% and the sustained cylinder-to-cylinder data rate by up to 35%. These rate reductions are caused by: format information that must be read, but that is not data; the need to switch between tracks and cylinders, which takes time but transfers no data; and occasional bad spots, requiring time to process remappings. The *sector-to-sector transfer rate* is the real rate at which data is transferred to or from a track on the disk. It is computed by dividing the total data bytes in a track by the time of a disk revolution. The *average seek time* is computed over all possible seek distances, a uniform distribution. Thus the average seek is the same as a seek over one-third the total cylinders. Air friction and motor design in 8.0-inch drives usually limits the maximum *rotational speed* to 3600 RPM. Smaller drives can rotate faster without heat buildup. Expect 3.5-inch drives to routinely exceed 5,000 RPM later in the 1990s. The 8-KB *data transfer time* uses the realistic sector-to-sector transfer rate and assumes that all 8 KB is on the same disk track. The *random I/O rates* indicated are computed by taking the inverse of the (average seek time + average rotational latency + data transfer time for 8 KB).

3.4 SCSI and IPI Interface Definitions and Operation

Comparing the SCSI and IPI interfaces is a bit like comparing bread and flour. The inquisitive reader has undoubtedly been waiting for *something* to be different, for previous sections have concluded that typical SCSI and IPI HDA performance is essentially identical. The wait is over: it is the SCSI and IPI interfaces that duel as competitors, not the HDAs. This section examines how the two interfaces work. Section 3.5 will compare their specifications and limits.

The fundamental difference between SCSI and IPI-2 is that SCSI is a *bus* interface, and IPI-2 is a *device* interface. Refer to figure 2.

3.4.1 The SCSI Bus Interface

SCSI, as a bus interface, defines a shared packet-like bus that peer devices such as disks and tapes use to conduct and communicate their business. The SCSI-1 bus is 8-bits wide and transfers data at ≤ 5 MB/s synchronously, and about 2 MB/s asynchronously. There can be eight devices per SCSI-1 and SCSI-2 bus—addressed 0 to 7 (only two disks are shown in figure 2). The implementation of the SCSI interface in each device is called the SCSI *controller*, or just controller. Device 7 (the highest priority) is usually reserved for host system access, and this host interface is conventionally called the SCSI *host adapter*, not the "host SCSI controller." In figure 2, the Storage Processor has 10 host adaptors on a single board. Since the SCSI bus fills the traditional

channel role in SCSI storage subsystems, this report occasionally uses the less-precise term "SCSI channel" to refer to the SCSI bus.

The SCSI disk controller built into SCSI disks is frequently called "intelligent." This is because the controller responds to very high-level, almost un-disk-like commands. For example, *read (logical block number L, number of sectors S)* is a SCSI command to read S 512B sectors of data from the disk starting at 512B-block L. Note that no disk-style cylinder, track, or sector addresses are given—just a linear block number. Also, multiple blocks (sectors) can be conveniently read—even if they cross track or cylinder boundaries, or include remapped bad spots. In a SCSI disk drive, the controller imposes a simple linear addressing scheme on top of complex random-access media. The controller also transparently deals with issues of error detection and correction, bad-sector remapping, error logging, and disk buffering (see below). This intelligence is integrated *directly* into the drive by the disk vendor, who presumably best knows how to deal with these issues—in the lowest-overhead manner—for his own HDA.

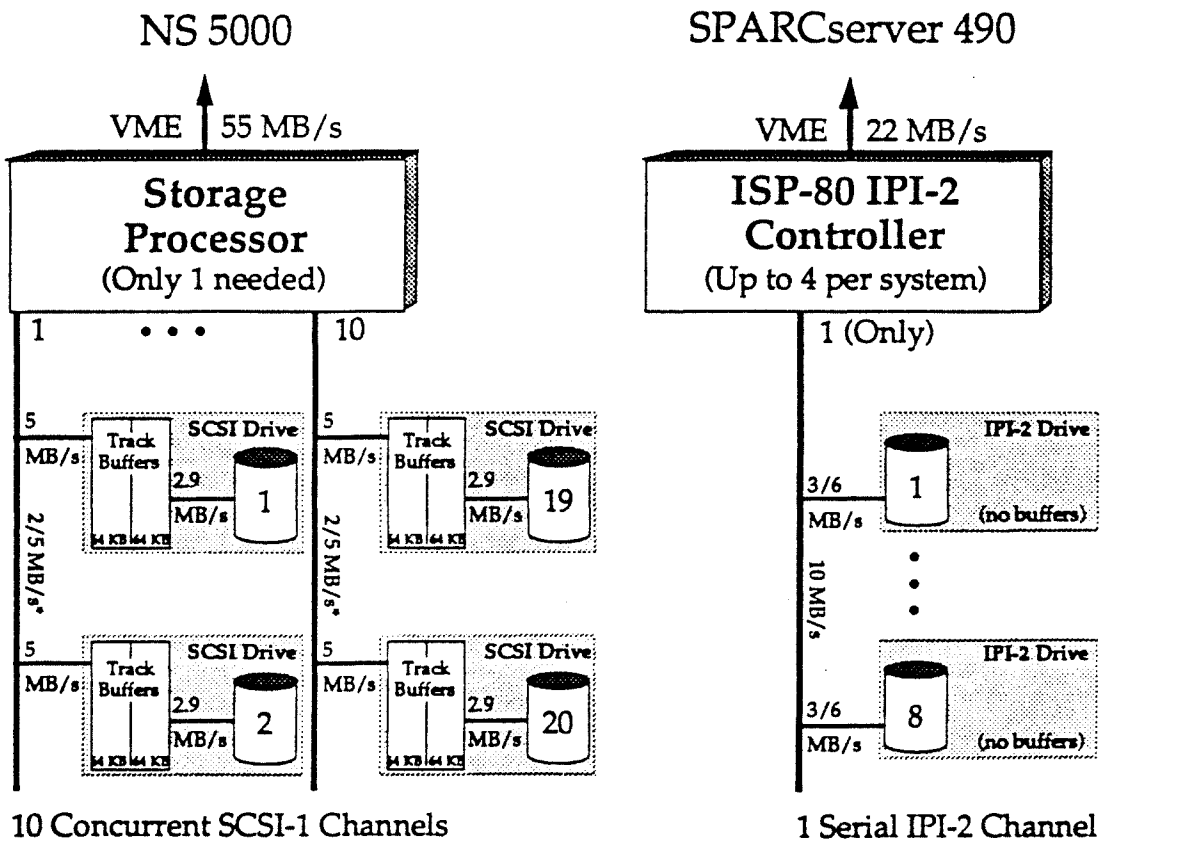


Figure 2: Typical SCSI disk array and IPI-2 disk subsystem architectures. The storage subsystems of the Auspex NetServer and the Sun 4/490 are illustrated as concrete examples of each architecture. The NetServer has a maximum of 20 drives per Storage Processor—2 per SCSI channel—and each drive can read and write concurrently as described in section 3.4.1. Do not construe the term *disk array* too elaborately: it simply refers to a disk subsystem organization that has high drive packing density (volumetric efficiency) with the drives attached to multiple parallel channels (typically ≥ 5). (For the curious, Katz and Patterson define disk arrays in detail [Patterson88].) The NetServer Storage Processor [Nelson91] is a sophisticated, intelligent, 10-SCSI-channel controller that supports 10 × 5 MB/s = 50 MB/s aggregate channel rate. (The asterisk (*) indicates 5 MB/s synchronous or 2 MB/s asynchronous transfer rates.) The 4/490 can have up to 8 disks per controller, but each controller has strictly serial read and write access to each disk, as described in section 3.4.2. There is a 4/490 limit of 4 IPI controllers per system, with only a single IPI channel per controller, although other vendors can support more controllers or more channels per controller. The Sun ISP-80 controllers do support a 10 MB/s channel transfer rate and are optimized (with controller buffering) for sequential read-ahead. Despite this sequential transfer advantage, section 4.3 discusses empirical testing that confirms that the limited drive concurrency of this style of IPI subsystem greatly restricts NFS (random-access) throughput.

SCSI is not a master-slave bus. Each device can assume control as the bus master and transact business in a request-reply fashion with any other device on the bus—usually the host, through the host adaptor. Consequently, SCSI device communication is interrupt driven, not polled, and this reduces bus contention. Because the single SCSI bus is used to pass both control and data information, the SCSI bus is the storage channel (defined in section 1.1) for all attached devices.

The SCSI bus permits substantial device concurrency, since, for instance:

- The host can connect to a tape drive and give it a command to read data. The host can then disconnect from the tape while the tape executes the command and fills its internal buffers.
- While the tape is at work (and off the bus), the host can *immediately* connect to another device—say, a disk drive—and give it a command to write data (including the data). The host can now disconnect from the disk.
- While both the tape and disk are at work (and still off the bus), the host can request another disk to read data without waiting for the previous disk or tape. Again, once the disk has received the read command, it can disconnect from the host and actually perform the read into internal buffers.
- Now, when the tape has actually read its data, it becomes bus master, connects to the host adaptor, and transmits the data requested by the host over the SCSI channel.
- While the tape is sending its data, assume that the disk finishes its read operation. The disk tries to connect to the bus, but finds it busy with tape traffic. The disk non-intrusively monitors the bus until the tape finishes, then takes its turn as bus master and sends its data to the host adaptor.
- As the disk transfers the requested data to the host, many SCSI controllers will automatically read the remainder of the current disk track into its internal buffers. This is called *read ahead*. If the host should soon ask for more data on the same track (e.g., sequentially read the next few sectors), it permits the drive to quickly respond from its buffers rather than from the (slower) disk itself. If the host asks for other data (not read ahead), read-ahead operations can be instantly aborted and the requested data sought without delay.

Notice that the intelligent nature of the SCSI interface permits and even encourages a complete decoupling of a device's operation requests, operation execution, and operation replies. Further, requests and replies to different SCSI devices can be freely intermixed in time—device access is not *serialized* (funnelled down to a single active device), and device execution is completely autonomous and overlapped. For electromechanical devices like disks and tapes, buffering in the device's controller matches the *millisecond* speeds of the recording hardware to the *microsecond* speeds of the SCSI bus—analogous to a perfect freeway on-ramp. This thousand-fold impedance matching between SCSI devices and the SCSI bus is an extraordinary SCSI advantage: it works like a charm, it's economical since it is built into the device, and it enables massive device parallelism. As we will see in section 4.4, arrays of SCSI disks can easily deliver this massive throughput at the system level—just the ticket for NFS service.

3.4.2 The IPI-2 Device Interface

IPI-2 is a 16-bit parallel synchronous data interface with five separate control signals for disk device control. Up to eight IPI devices can be attached to a single 10-MB/s IPI *channel*, addressed 0–7. IPI is a *master-slave* channel, unlike SCSI, which has a peer-to-peer protocol. As a result, an IPI channel attaches disks to a host system through a distinguished *disk controller*. This controller is the channel master and needs no device address of its own since it strictly controls slave devices (e.g., disk drives 0–7).

Compared to SCSI, the IPI interface is an unintelligent interface. IPI drives do not offer an error-free, linear-address disk model to the IPI channel. Rather, the IPI controller assumes these tasks, and uses the many data and control signals of the IPI channel to implement a higher-level model to the host. In essence, while SCSI puts the intelligence in each drive (in the drive's SCSI controller), IPI pulls out intelligence from the drive and puts it up on the disk channel controller, where intelligent functions are shared among the up to eight drives per channel. While this may appear to have some economic benefit, in practice the low volumes of IPI drives—and even lower volumes of frequently host-vendor-specific controllers—keep IPI prices well above SCSI prices.

Pushing IPI intelligence back to the controller also has serious performance disadvantages. Typical IPI controllers can command disk drives to perform concurrent seeks. But the IPI channel allows only strictly serialized data transfers. Furthermore, IPI drives do not (yet) have individual read-head (track) buffers—buffering is pushed back to the shared controller. The result is that IPI drives on the same channel end up waiting for each other—delayed waiting to initiate raw transfers to the controller.

Consider this typical sequence of events for the IPI subsystem in figure 2:

- The IPI controller commands one disk to seek to a specified track.
- The controller immediately commands another disk on the same channel to a different track.
- The controller waits for the two disks until one of them is ready to transfer data at the desired sector on the target track and interrupts the controller. (The implications of so-called *zero-latency reads* (ZLR) are not considered here. In a ZLR transfer, data movement can start from the middle sectors—not just the first—of a multisector transfer. The remaining data—i.e., the initial sectors requested—are read later when the disk head arrives there. The piecemeal transfer is spliced together by the controller before being passed, contiguous and intact, to the host. ZLR is an advantage for large, multi-track (> 50 KB) transfers, but is of marginal help for small NFS-style 8-KB block transfers.)
- The controller accepts raw data from the ready drive, processing the preamble and postamble, performing error detection (typically by computing a Reed-Solomon ECC code), buffering, etc. The controller is fully occupied servicing this single drive for the duration of the transfer.
- During the above drive's transfer, the second drive becomes ready. But because the controller is already busy with a transfer, the second drive waits—and the read-write heads spin past the desired data sectors. This is where SCSI has a remarkable advantage: a SCSI drive that finds its channel busy *never* waits, because the SCSI HDA can transfer the target data into local-to-the-drive buffers. An IPI drive, on the contrary, must rotate completely around—wasting about 16 milliseconds—before it can begin the transfer again. This is an enormous penalty. It lets us predict that *adding more drives to an IPI channel will* decrease *random-access throughput*—a counterintuitive hypothesis that is tested in section 4.3.
- After spinning around, the second drive executes its transfer, as long as the first drive is finished and no other drive has grabbed the IPI channel in the 16-ms interim. The necessarily *serialized* access to the IPI channel for data transfers severely limits IPI's random-access throughput. This is the classic and well-known *channel contention problem* in the mainframe world, where disk drives must be coupled with a balanced number of channels to maintain unhindered storage-to-host throughput.
- Finally, consider the effect of IPI read-ahead. If an IPI controller has been programmed to perform track (or greater) read-ahead—typical of sequential-transfer environments—other waiting disks will wait even longer. Furthermore, if the read-ahead data is never used, all

that extra waiting time is *really* wasted. Contrast this with SCSI, where read-head occurs inside the drive, creating neither channel contention nor unnecessary waiting.

Some remedies are available for IPI's serial-channel bottleneck. The easiest remedy is to use only one IPI disk per controller (which is how SCSI operates already). While computer vendors often benchmark their IPI disk subsystems using this method, it is far too expensive for most customers. A second remedy is to build multiple storage channels into one controller. Unfortunately, it takes substantial hardware real estate to cope with the low-level IPI-2 device interface, resulting in a limit on how many channels can be packed onto a printed-circuit board. Still, some clever vendors are building two-channel IPI controllers for the Unix market. But this does not match the dense channel packing possible with SCSI, where 5- and even 10-SCSI-channel host adaptors exist.

Finally, remember that the serial-channel bottleneck mainly applies to random I/O patterns. One of IPI's *raisons d'etre* is quick, massive, single-drive sequential transfers using parallel heads. IPI will do this well even with multiple drives per IPI channel because the transfer time—for suitably long transfers—outweighs the pretransfer positioning time.

3.5 SCSI and IPI Interface Comparison

The previous section defined the SCSI-bus and IPI-device interfaces. It continued to describe some performance-critical operational issues of host-to-disk transfers. This section returns to basics by summarizing and comparing SCSI and IPI interface specifications. See table 3.

Attribute	IPI-2	SCSI-1	SCSI-2
Device or Bus Interface	Device	Bus	Bus
Arbitration Scheme	Master-slave	Peer-Peer	Peer-Peer
Track Buffering (2 x 64KB)	No	Yes	Yes
Autonomous Device Operation	No	Yes	Yes
Multiple Transfers per Bus/Channel	No	Yes	Yes
Multiple Command Queueing	No*	No	Yes
Read-ahead Possible	No*	Yes	Yes
Zero-latency Reads Likely	No*	No	Yes
Bus/Channel Width (bits)	16	8	8, 16, 32
Max Bus/Channel Bandwidth (MB/s)	10 MB/s	5 MB/s	10, 20, 40 MB/s
Bus/Channel Parity?	Yes	Yes	Yes

Table 3: A SCSI-1, SCSI-2, and IPI-2 interface comparison. Some comments on the attributes: Section 3.4 defines *device and bus interfaces* and discusses SCSI's and IPI's *arbitration schemes*. *Zero-latency reads* are defined in section 3.4.2. All IPI attributes marked with an asterisk (*) are optionally implemented in the IPI controller and are not available in the drives themselves.

Reviewing table 3, there are five key attributes that distinguish SCSI from IPI:

- *Multiple transfers per bus/channel.* Because SCSI devices are buffered and intelligent, physical device data transfers can proceed in parallel decoupled from the channel. IPI transfers, as we have seen, are direct-coupled, serial, and will often miss their optimal starting points.
- *Multiple command queueing.* With both IPI and SCSI-1, command queueing (and sorting) is left as a controller function, where it is usually performed in high-performance systems.

With SCSI-2, queueing can also occur within the SCSI-2 device's own controller (see section 1.4). As well as aiding device optimization, this queueing has the added benefit of reducing bus contention by eliminating unnecessary bus handshaking when multiple commands are sent together using *command linking*.

- *Read-ahead possible.* With its internal buffering, read-ahead is natural and easy in SCSI devices. For disks, read-ahead is usually done on a track basis—that is, as a disk head sweeps an entire track, all the data passing under it is read into the local buffer. Lacking device-level buffering, IPI drives do not perform read-ahead themselves, although a high-performance controller will often do so. Controller-governed IPI read-ahead has serious performance implications in random-access environments like NFS, as discussed in section 3.4.2.
- *SCSI read-ahead better.* With its per-drive internal buffering, SCSI read-ahead will be more effective than IPI's controller-level buffering. This is because IPI-controller-driven read-ahead easily results in unnecessary disk delays and channel contention (see section 3.4.2).
- *Zero-latency reads likely.* SCSI-1 does not usually perform ZLR. Many SCSI-2 drives do. As we have seen, IPI disks do not perform either reads or zero-latency reads themselves, although the IPI-2 device interface provides enough information for controllers to perform both. In an IPI disk subsystem, then, ZLR can optionally be implemented in the controller.

3.6 A SCSI-1 Drive Laboratory Performance Analysis

Examining disk specifications is no substitute for an empirical performance analysis. Chart 2 presents the results of extensive SCSI testing [Cheng90a]. Hewlett-Packard high-performance 97548 663-MB SCSI-1 drives were used [HP89]. These drives are capable of 2 MB/s asynchronous SCSI bus transfers, and 5 MB/s synchronous transfers (the drive tested was an earlier manufacturing revision capable of only 4 MB/s synchronously). The host adapter was a 10-channel Auspex Storage Processor [Nelson91] attached to special instrumentation.

Here are some important conclusions from this SCSI-1 evaluation:

- *Multiple drives on one SCSI bus perform well.* For random access there is only a 5% fall-off in throughput linearity with three synchronous drives—i.e., three drives on one SCSI channel perform *almost* three times as many random I/Os (curves 5&6). For sequential access, where SCSI bus data transfer time is larger, three-drive sequential-synchronous throughput linearity decreases 20% with read-ahead (curve 1) but only 1% with no read-ahead (curve 2).
- *The cost of random access (over sequential access) is 16 ms,* or exactly the average seek time of the HP 97548 disk drive—the expected result (curves 2&6, 4&7).
- *Sorting seeks into elevator queues works well,* reducing random-access service time from 33 ms to 28 ms, or 15%, for a single synchronous drive (curves 5&6). Conversely, sorting seeks increases random-access throughput from 30 read IOPS to 36 read IOPS on one drive, and from 86 to 98 read IOPS on three drives.
- *Read-ahead is crucial for excellent sequential throughput.* Using only one drive per channel, sequential synchronous throughput increased from 467 KB/s without read-ahead to 745 KB/s (60% more) with read-ahead of sequential 8-KB blocks (curves 1&2). On a per-operation basis, read-ahead drops a read's response time by about 7 ms, which is nearly the average rotational delay. So *not* reading ahead requires rotating around again, as expected.
- *For random disk transfers, asynchronous SCSI transfers are only slightly slower than synchronous ones.* On a single disk, asynchronous random-access reads take an extra 0.9 ms (3%) of transfer time per 16-sector 8-KB block (curves 6&7). On three disks, the added time rises to 3.8 ms (11%) per drive.

- *For large sequential disk transfers, asynchronous SCSI transfers are somewhat slower than synchronous ones.* On a single disk, asynchronous sequential transfers take an extra 3.1 ms (29%) of transfer time per 16-sector 8-KB block (curves 1&3). This corresponds to a drop in sequential throughput from 745 to 598 KB/s. On three disks, the asynchronous slowdown is pronounced, taking 6.3 ms (46%) longer per drive.

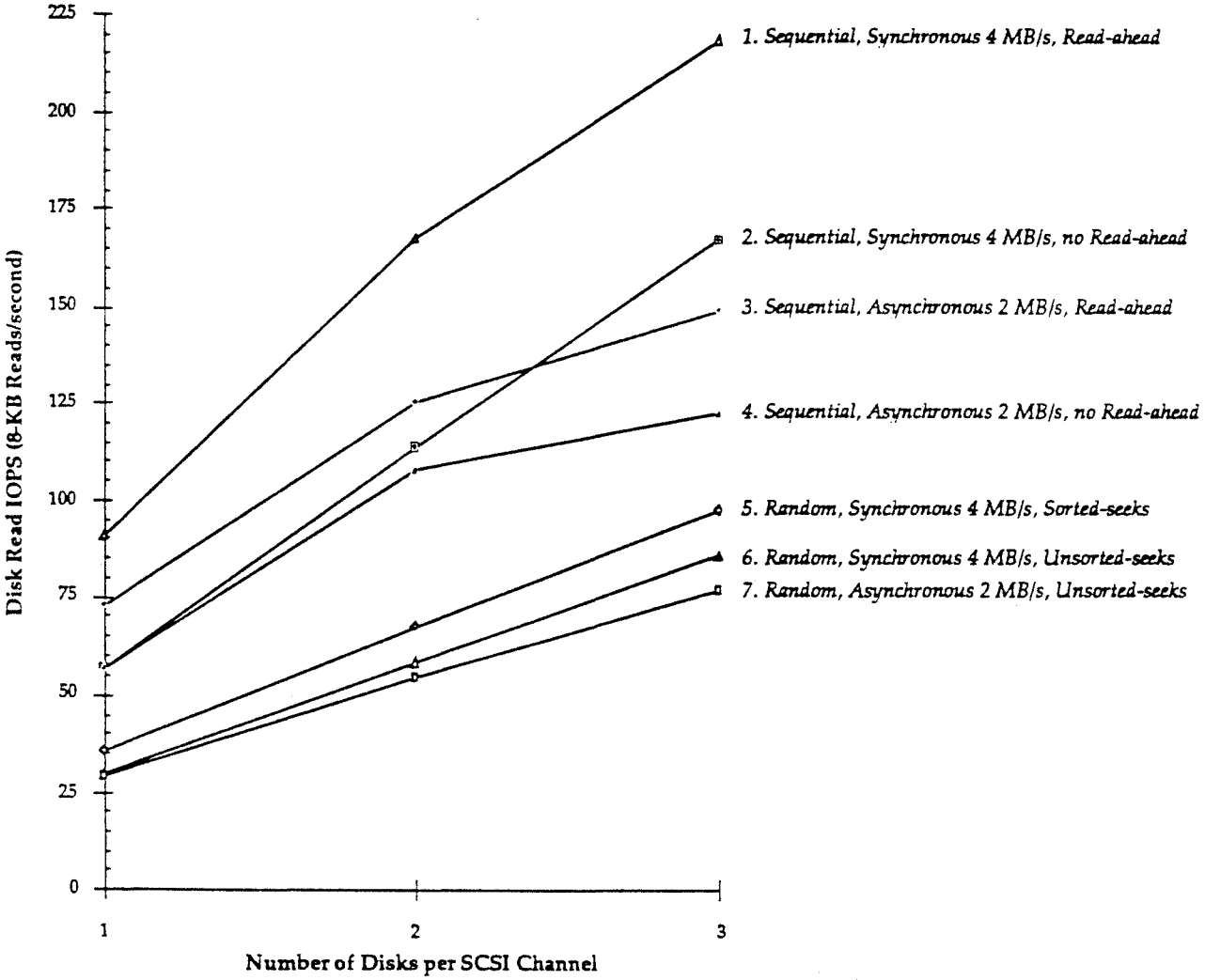


Chart 2: An extensive test-bench evaluation of SCSI-1 disk drive read performance. In this graphical analysis, using HP 97548 SCSI drive data measured by Cheng [Cheng90a], curves 1–7 investigate the influence of five parameters on read performance: (1) synchronous and asynchronous bus transfers, (2) random and sequential access patterns, (3) sorted and unsorted seek queues, (4) read-ahead enabled and disabled, and (5) multiplicity of drives per SCSI channel. In particular, observe that SCSI random-access throughput scales almost linearly as more drives are added to the same SCSI channel. As we will discover in section 4.3, however, normal IPI subsystems like the one in figure 2 exhibit the opposite behavior. More detailed conclusions are discussed in the text. Note that the service time for a disk operation in milliseconds is simply the inverse of its operation rate in IOPS (which is plotted on the Y-axis, above).

3.7 SCSI and IPI Comparison Summary

Section 3 has examined SCSI and IPI from specifications to testing, from HDA to interface, from price-capacity to price-performance, from the cursory to the detailed. Table 4 summarizes these findings for the disk subsystem organizations shown in figure 2.

Attribute Advantage ➨	IPI-2 Disk & Ctlr	SCSI Array
Transfer Rate Unstriped (MB/s)	✔	
Transfer Rate Striped (MB/s)	✔	✔
Disk Drive Concurrency (disk operations/second)		✔
Disk Channel Utilization (disk operations/second)		✔
Storage Controller or Processor (disk operations/s)		✔
Price-Capacity ($/MB)		✔
Volumetric Efficiency (GB/m^3)		✔
Reliability (disk MTBF)	✔	✔

Table 4: A summary comparison of SCSI-array and IPI disk subsystems. Check marks in this table are a consequence of conclusions reached in earlier sections and tables.

Some comments on table 4:

- Since disk striping (interleaving) can be applied to both multicontroller (e.g, multichannel) IPI subsystems and multi-SCSI-bus subsystems, both IPI and SCSI can attain >> 10 MB/s aggregate transfer rates. For those unfamiliar with disk striping, read Cheng and Pitt's report on striping, mirroring, spanning, and other aspects of logical volume and partition storage management [Cheng91].
- At MTBF ratings of 100K hours for IPI and 150K for SCSI, both drives are extremely reliable.

4 SCSI AND IPI DISK SUBSYSTEM PERFORMANCE COMPARISON

4.1 The Anatomy of an NFS Operation—Measuring Transfer Rate & NFS Throughput

Figure 3 dissects the anatomy of an NFS *Read* operation from when the *Read* request is initiated on a client workstation until the *Read*'s 8 KB of data returns to the client from a disk. The resulting timing analysis is done in the context of an Auspex NetServer [Nelson91, Hitz90a, Schwartz90], a functional multiprocessor NFS file server that uses SCSI disk arrays for its NFS storage subsystem (illustrated previously in figure 2). NetServer timings are undoubtedly representative of similar timings that could be measured on other vendor's NFS servers. The goal of this timing analysis, presented in detail below, is to clearly demonstrate that the actual data-transfer time of an 8-KB NFS *disk* read is such a small part of overall *system-level NFS Read* time that the disk's transfer rate has negligible performance impact.

This analysis explicitly ignores all system-, controller-, and drive-level caching optimizations. This is necessary to study the influence of actual disk transfer rate: If a disk's not involved in an I/O operation because a cache-hit stepped conveniently into the way—at any level in the system—then transfer rate doesn't matter at all. Caching is indeed crucial for system performance and is used in all real systems. But this study examines what happens when there is a cache miss on a read.

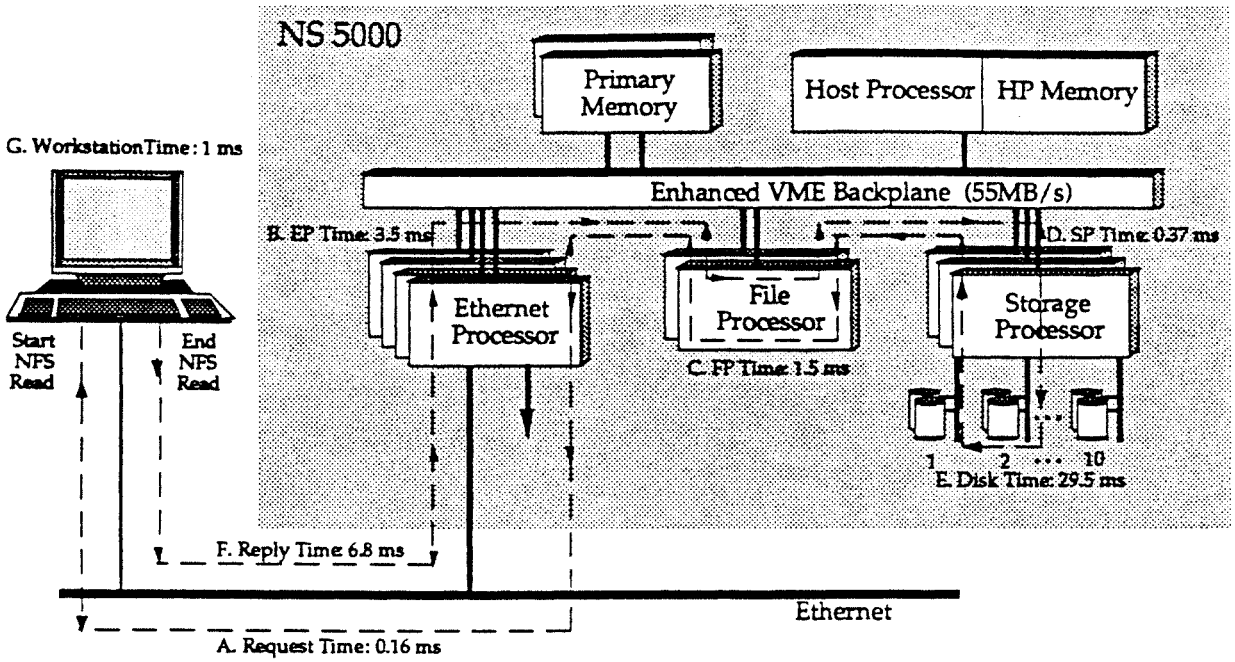


Figure 3: The anatomy of an uncached NFS *Read* operation. The indicated 29.5ms of disk time dominates the overall 41.8ms required for the complete client-to-server-to-client NFS *Read*. However, the disk data-transfer time for an 8-KB NFS block is only 3.7ms of the 29.5ms total disk time, or just 9% of the entire client-to-server-to-client NFS *Read* time. This 9% transfer-rate contribution is not significant, and becomes *less* so as disk transfer rate increases further beyond 3.0 MB/s. Client workstation time is not included in the total time. Detailed discussion and further conclusions appear in the text. The file server used in this timing analysis is an NS 5000 multiprocessor NFS server from Auspex [Nelson91], running Auspex's 1.1 software release. The timings indicated for the NS 5000 are not estimates but were measured with software monitors or logic analyzers. The disk timings are based on the HP SCSI-2 drives defined in table 2. Ethernet timings are for transmission times assuming no access contention or collisions.

Let's examine each step in the timing path of figure 3's non-cached *Read* operation, and make some key observations along the way:

A *Ethernet request time: 0.16ms, inbound to server only.* This is the Ethernet transmission time required to send a single IP packet (UDP datagram) that contains the NFS *Read* remote procedure call (RPC). This is a best-possible time that assumes no Ethernet contention or collisions. *Read* requests are about 100 bytes long and contain IP, UDP and RPC headers, NFS file handle and offset information, and user identification and (optionally) authentication information.

B *Ethernet Processor (EP) time: 3.5 ms, inbound and outbound.* This is the complete NFS protocol processing time required for the *Read* (or of many other NFS operations). In the NetServer, protocol processing includes Ethernet, IP, UDP, RPC, XDR, and NFS processing time. It also incorporates preliminary local file system (LFS [Schwartz90]) processing time (equivalent to much of what happens in a conventional NFS server's NFSDs) and block-transfer DMA time between primary memory and the Ethernet Processor.

Observe that NFS protocol processing time (EP time) dominates all other NFS file server software overhead—the EP's 3.5ms exceeds the FP's 1.5ms or the SP's 370µs. (This is why Auspex NetServers integrate protocol processing directly into the network interface—for lower response times and excellent network scalability.)

C *File Processor (FP) time: 1.5 ms, inbound and outbound.* This includes all file-system processing time, including cache management. In the NetServer, it also includes the time for kernel-level message-passing communication [Hitz90a] between the Ethernet, File, and Storage Processors, which takes about 100µs per one-way message.

There are two request-reply message pairs exchanged for an NFS disk *Read* (four one-way messages): a *Read* request from the EP to the FP, an 8-KB disk-read request from the FP to the SP, a trivial "done" reply from the SP back to the EP, and a *Read*-data reply from the FP back to the EP.

D *Storage Processor (SP) time: 370µs, inbound and outbound.* This includes all Storage Processor kernel message-processing time (230µs), elevator queuing software overhead (67µs), SCSI bus arbitration (1µs), and SP SCSI command overhead (70µs). Put in conventional SCSI terms, the NetServer's SCSI host adapter overhead is 138µs, an exceptionally low figure since many host adapters have overheads > 1 *milli*second.

E *Disk time: 29.5 ms total.* This total disk-read time for a randomly-accessed 8-KB NFS data block includes six components:

1. *70µs of SCSI device selection and command transfer* (overlapped with SP overhead in D).
2. *500µs (estimated) of SCSI drive command decode and processing;*
3. *17.5ms average seek time;*
4. *7.5ms average rotational latency;* and
5. *3.7ms data-transfer time* for 16 sectors (8 KB), all assumed to be on the same disk track (a realistic best case).
6. *260µs of SCSI cleanup and handshake* (some overlapped with the SP).

Observe that this nearly 30ms of disk time overwhelms all other times, and that transfer rate is a small fraction—13%—of disk time. We will study the influence of increasing this transfer rate below in table 5. The drive-specific timings for items 2–5 above are from table 2 for the HP 97549 1-GB SCSI-2 drive, which has a 2.85-MB/s raw disk transfer rate.

F *Ethernet reply time: 6.8ms, outbound from server only.* This is the Ethernet transmission time required to send the 6 IP packets that contain the server's reply to the client's NFS *Read* (RPC). This is again a best-possible time that assumes no Ethernet contention or collisions. *Read* replies are about 8,500 bytes long and contain IP, UDP and RPC headers, NFS file information, and the actual 8 KB (8,192 bytes) of file data. Although the *Read* RPC reply is a single UDP datagram, it is too large to be encapsulated in a single IP packet. Thus it is fragmented into 5 full size (1,512-byte) IP packets and a final, sixth partial packet.

Observe that at nearly 7ms, Ethernet transmission time is a significant factor (16%) of overall *Read* time. On an Ethernet experiencing moderate loads (> 10%), deferred transmissions and collisions can easily add 10–30ms to this 7ms of unhindered transmission time [Boggs88]. Moral: don't overload your Ethernets or both throughput and NFS response times will suffer.

G *Client workstation time: 1 ms total.* Client workstation time is highly variable, and depends not only on workstation CPU power, but also on the capabilities of the workstation's Ethernet interface, software driver, and Ethernet-to-memory I/O datapath. As a consequence, this analysis assumes an optimistic 1ms total client NFS processing time. This is not obtained in previous generation 2-MIPS CISC workstations, but might be in contemporary 20-MIPS RISC workstations.

Σ *Total time: 41.8 ms,* not including client workstation time.

Table 5 shows the performance impact of increasing disk drive transfer rate on the system measured in figure 3. Note that a 6 MB/s SCSI drive would only improve performance by 5%, and a 12 MB/s drive by 7%. But even these improvements would be worthwhile *only on* SCSI *drives*: if an IPI subsystem was used instead of a SCSI disk array, random-access throughput would suffer greatly because of IPI channel-access contention, as discussed in section 3.4.2 and shown empirically later in section 4.3.

Raw Transfer Rate	Transfer Time	Total NFS *Read* Time	Δ Performance
2.9 MB/s	3.7 ms	41.8 ms	Baseline
3.0 MB/s	3.5 ms	41.6 ms	0 %
4.0 MB/s	2.6 ms	40.7 ms	+3 %
6.0 MB/s	1.7 ms	39.8 ms	+5 %
9.0 MB/s	1.2 ms	39.3 ms	+6 %
12.0 MB/s	0.9 ms	39.0 ms	+7 %

Table 5: The influence of disk data-transfer rate on system-level NFS performance. The 5% gain from using 6 MB/s drives—if available—is negligible. Furthermore, this gain would only be useful in SCSI-style disk arrays: Using an IPI disk subsystem, channel contention would negate all transfer rate advantages and kill random-access throughput as shown in section 4.3. The transfer times shown are the data-transfer times for a 16-sector, 8-KB block (see table 2). The total NFS *Read* time consists of a fixed, 38.1ms portion (the sum of all but the data-transfer and client-workstation times in figure 3), to which the data-transfer time is added.

4.2 NFS Benchmarking, NFS IOPS, and the NFS Operation Mixture

Some simple fundamentals of NFS benchmark methodology are needed for the following sections that examine disk subsystem performance in the context of system-level NFS testing. A complete discussion of accurate NFS benchmarking is a complex subject beyond the scope of this paper. This section briefly repeats some definitions from Horton's detailed NFS performance study [Horton90].

NFS IOPS—I/O operations per second—are an NFS-specific measure of NFS load. NFS IOPS are computed by dividing the total number of NFS operations executed by a server (or client) in a given time interval by that time itself. Each type of NFS operation—*Read*, *Create*, *GetAttr*, etc. (see table 6)—is included equally in the total operation count. NFS IOPS are an appropriate measure of an NFS file server's I/O performance since server I/O throughput is dependent on system-level I/O architecture, not computational CPU MIPS.

For reference, a single DECstation 3100 or SPARCstation1 performing moderate NFS I/O (e.g., compiling remote sources) generates a load of 10–20 IOPS to an NFS server, depending on local disk and memory configuration.

NFS Operation Mix	*GetAttr*	*SetAttr*	*Lookup*	*ReadLink*	*ReadDir*	*Read*	*Write*	*Create*	*Remove*	*FSStat*	*Total*
NHFSStone Default	13%	1%	34%	8%	3%	22%	15%	2%	1%	1%	100%
Software Company	14%	1%	59%	2%	9%	8%	4%	1%	1%	1%	100%

Table 6: Two distinct NFS operation mixtures. NHFSStone's default operation mix is widely used [Lyon89]. It is moderately I/O intense and characterizes large-file, swapfull, or diskless client workstation environments. On the NHFSStone mix, *Read* and *Write* combined constitute 37%, or just over one-third of all operations. Directory operations (*Lookup*, *GetAttr*, *SetAttr*, *ReadLink*, and *ReadDir*) constitute 59%, or approach two-thirds. The Software Company's mix is particular to chart 4. It has an extremely high *Lookup* fraction and unusually low *Read* and *Write* ratios—a lightweight mix reflecting a low-use diskless environment with very deep directory trees. The remaining seven (of the seventeen total) NFS operations unrepresented in this table are executed so infrequently as to be negligible.

The benchmark workloads used in this report consist of synthetic NFS traffic generated in an average operation mixture that corresponds to the mixtures actually encountered on servers handling client workstations executing CASE, CAD, and other applications. The two sets of NFS

operation mixes used for this paper's tests appear in table 6. Auspex's standard-benchmark NFS operation mixture (unless specified otherwise) is the well-known NHFSStone "default" mix.

It is often useful to derive a single NFS throughput figure-of-merit from the two-dimensional throughput versus response-time curves shown in the accompanying charts. Auspex's careful methodology requires measuring a server's NFS throughput (in NFS IOPS) at a 70 millisecond response time. The 70-millisecond response time cutoff is a comfortable maximum: workstation users prefer 20–40ms and notice a marked slowdown at ≥ 100ms.

4.3 Why IPI is Poor at NFS—Measuring Multiple IPI Drives per Controller

4.3.1 Multiple Drives per Controller without Write Caching

In section 3.4.2 we predicted that adding drives to the same IPI disk channel would *decrease* random-access throughput. Chart 3 is empirical proof of this counterintuitive hypothesis. For reasons of convenient access, Sun's SPARCserver IPI disk subsystem was used for this test. The same results are expected from any sequentially-optimized IPI (or SMD) disk subsystem, although the relative ratios would change depending on a particular vendor's implementation.

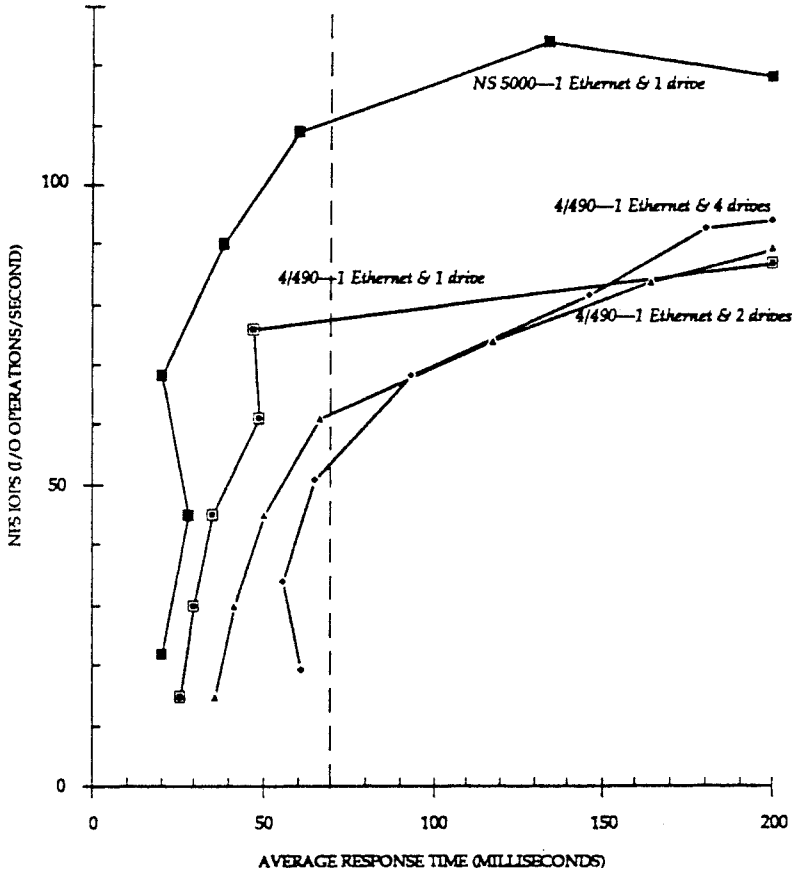


Chart 3: An NFS performance comparison of multiple IPI drives on a single IPI controller *without* write caching. Notice how the 4/490's NFS throughput actually *decreases*—and response times *increase*—as more disks are added to the same IPI channel. This is because of bottlenecked IPI channel access. Both this negative throughput scaling and the saturated-disk, 150-ms throughput crossover are explained in the text. All disk writes performed in this test were synchronous and not buffered asynchronously through fast stable storage such as a non-volatile RAM cache. This test uses the default NHFSStone NFS mix (table 6). NFS IOPS—I/O operations per second—and the test's NFS operation mixture are explained in section 4.2. The test configurations were: a Sun 4/490 running SunOS 4.0.3 with one ISP-80 IPI-2 controller and 1–4 Sun IPI-2 disks; an Auspex NS 5000 running version 1.1 software and SunOS 4.0.3 with an Auspex Storage Processor and 1 663-MB SCSI-1 disk.

This comparison is a system-level test—not an isolated disk subsystem test. The NFS load driving the server originates on workstations, so server network and CPU characteristics will play a part in the results. But from test trial to test trial, only the number of disks was varied. Three drive configurations were tested on a 4/490—1, 2, and 4 drives, all on the same IPI controller. No asynchronous write-caching (buffering) was used—NFS *Write* operations are performed synchronously to disk. A 1-SCSI-drive Auspex NS 5000 NetServer curve is included for comparison.

Here are the critical observations in chart 3:

- With only 1 drive, response times stay between 30–50ms until about 80 NFS IOPS, and then throughput levels off (see section 4.2 for a discussion of NFS IOPS). Note that even 1-drive SCSI performance on the NetServer exceeds 1-drive 490 IPI throughput on this random-access test.
- With 2 drives, response times increase to 40–70ms and throughput drops to about 65 NFS IOPS at a reasonable 70ms response time. *Random-access throughput and response time is worse for 2 drives than 1*, probably because the 2 drives are waiting for each other to perform data transfers (perhaps including excessive and unnecessary read-ahead), as discussed in section 3.4.2.
- With 4 drives, response times increase further, to 50–100ms, and throughput drops further to about 55 NFS IOPS. *Four drives are worse than 2 drives*, because there are now 3 drives waiting for the 1 active drive on the IPI channel to finish—as in section 3.4.2.
- In total overload, at the response time region of 150–200ms, 4-drive throughput exceeds 2-drive throughput exceeds 1-drive throughput. This is because each disk finally has enough queued work (in the controller) that whenever one drive finishes a data transfer, one of the waiting disks (probably) has queued data under its heads and can begin transferring it immediately. But the high response times of this overload region are completely unacceptable for most applications.

These three IPI performance curves prove our prediction that IPI disk subsystems do not exhibit good random-access performance unless there is a single-drive-per-controller pairing—an expensive proposition.

4.3.2 Multiple Drives per Controller with Write Caching

This section continues the previous study of the Sun 4/490 IPI subsystem, this time with system-level write caching (described below) included. While 4 IPI controllers were available in the test configuration, only 1 and 2 drives per controller were tested. However, as chart 4 shows, write buffering does not effectively mask the multiple-drive-per-controller contention problem on the 4/490 under heavy NFS loads. Two configurations of Auspex server are included for comparison.

In detail, this section repeats the test of the last section with three major changes:

- The IPI disk subsystem is "fronted" by a stable-storage write buffer that permits the disk subsystem to perform writes asynchronously from the system (and the user's) point of view. Asynchronous NFS writes are usually not performed by NFS servers because strict NFS semantics require synchronous writes to maintain a stateless server guarantee. However, a write cache (or write buffer) that can survive crashes—and therefore appear like a zero-latency disk—is completely acceptable [Lyon89]. Since a write cache is a random-access device, it should increase the random-access performance—i.e., boost throughput and reduce response time—of (sequentially oriented) IPI subsystems.

- Third-party Ethernet controllers that incorporate some NFS protocol processing are used instead of the 4/490's built-in Ethernet interface. This change should not influence the disk subsystem, although it may free some 490 CPU cycles for extra NFS processing.
- The NFS operation mix (section 4.2) is *significantly* different and less server-intensive in these tests, so that the total NFS loads measured are *much* higher. Because of the different mixes used in charts 3 and 4, *their throughput curves cannot be compared directly.* (Chart 4 uses benchmark data based on a different mix because write-cached data using the same mix as chart 3 is not available.)

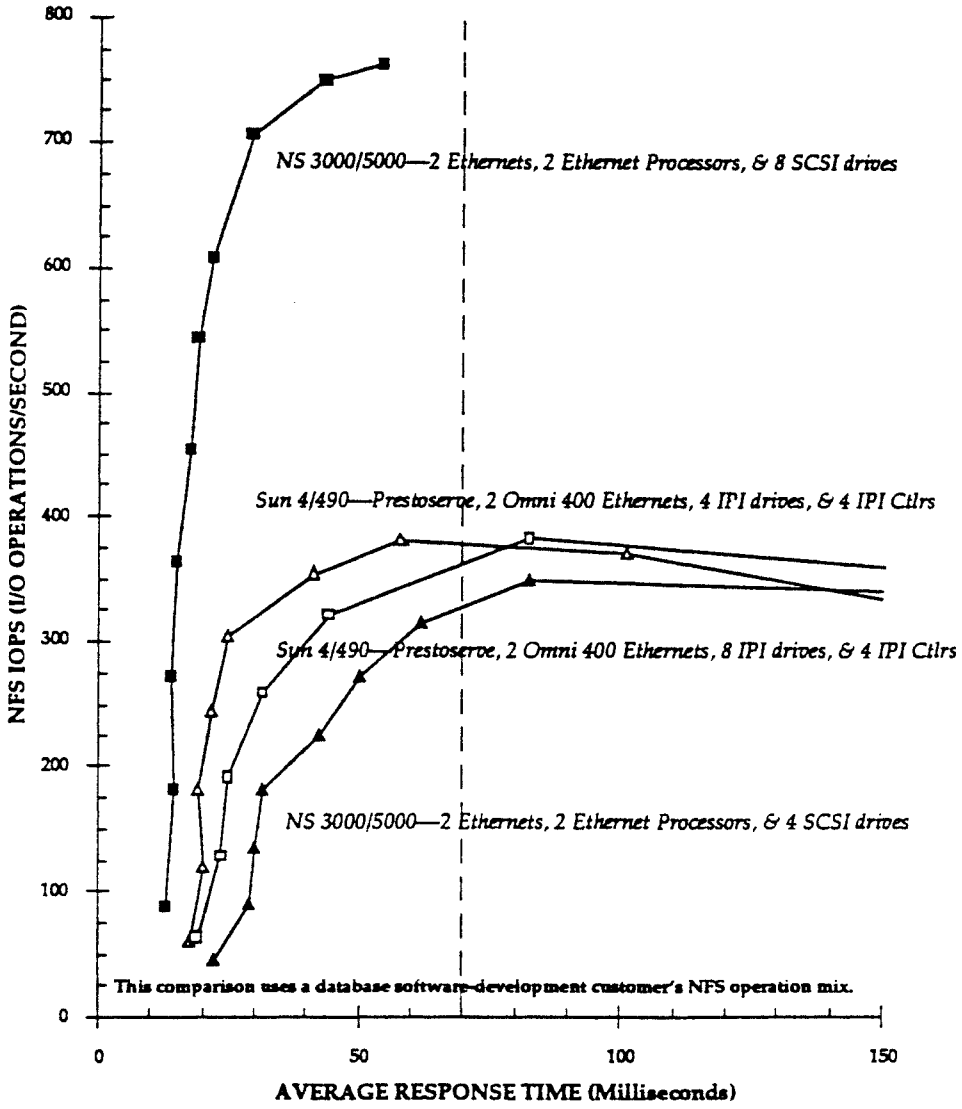


Chart 4: An NFS performance comparison of 1–2 IPI drives on 4 IPI controllers *with* write caching. The 4/490 has 4 controllers and is tested either with 1 drive per controller ("4 drives" in the chart), or 2 per controller ("8 drives"). Notice how the 4/490's NFS throughput still *decreases* as a second disk is added to each IPI channel (from 4 to 8 total). This is because even the Prestoserve write cache cannot totally overcome IPI's serialized-channel transfer contention under heavy NFS load. On the other hand, notice that SCSI disk array throughput on the NetServer doubles as disks are doubled. Further conclusions appear in the text. All 4/490 disk writes performed in this test were buffered asynchronously through a Prestoserve write cache. This test uses a software company's specific NFS operation mix (table 6). This mix is so different from that used in chart 3 that *throughput curves from the two charts are not directly comparable.* NFS IOPS—I/O operations per second—and the customer's NFS operation mixture are explained in section 4.2. The test configurations were: a Sun 4/490 running SunOS 4.1 with a Prestoserve 2.0 write cache, 2 Omni 400 Network Controllers, 4 ISP-80 IPI-2 controllers, and 4–8 Sun IPI-2 disks; an Auspex NS 5000 NetServer running version 1.2 software and SunOS 4.0.3 with an Auspex Storage Processor and 4–8 1-GB SCSI-2 disks.

Consider these chart 4 observations:

- With only 1 drive per channel (4 total) on the 4/490, NFS response times stay between 20–40ms until about 380 NFS IOPS, and then throughput levels off (see section 4.2 for a discussion of NFS IOPS). These response times are relatively lower than in chart 3—the write cache is working.
- With 2 drives per channel (8 total), *once again response times increase, and NFS throughput decreases*. Under heavy NFS load, we speculate that the Prestoserve write cache has a high miss ratio, and that the writes going directly to the disks are immediately stalled because of multiple drive-per-controller contention, as before.
- This leads to the hypothesis that write caching will be most effective for low to medium NFS loads. This is tentatively confirmed by the enhanced 490's better performance than the 4-drive NetServer. Write buffering has masked the IPI random-access shortcomings in a small, 4-drive configuration. Because of the zero-latency "disk" access, the 490 response times are shorter than the NetServer. Maximum NFS throughput of both the buffered 4/490 and unbuffered NetServer is similar—within 8%—showing that both systems are probably disk limited as they enter overload.
- The NetServer scales remarkably, doubling throughput as the number of disks double from 4 to 8. This is in stark contrast to the 490, where moving from 4 to 8 drives decreases throughput. The NetServer response time also drops to less than either 490 configuration. With SCSI disk arrays like those used in the Auspex servers, write caching is not needed to achieve high throughput on normal operation mixes. There is probably some merit in using a write cache to reduce latencies at low load levels. (These levels are encountered more in isolated benchmark tests than in real-life customer environments, however.)

For a further discussion of the systems in chart 4, including NFS performance on the standard NHFSStone operation mix, read Auspex's Benchmark Brief 5 [Auspex90].

In conclusion, IPI disk subsystems will benefit from write caching on small configurations and for low to moderate NFS loads. But our initial premise—that IPI random-access throughput is limited by serial IPI channel contention—has been empirically validated even with system-level write caching. Furthermore, the superior scalability of SCSI-array random-access performance has been demonstrated on the same workload.

With or without write caching, IPI is poor at heavy-load NFS.

4.4 The Advantages of SCSI Disk Arrays for NFS

This section finishes our empirical disk subsystem investigations by examining the scalability of SCSI disk arrays as more drives are added. As chart 5 shows, it's possible to achieve nearly linear throughput scalability on NFS workloads.

Once again, the benchmarked SCSI disk array is within an Auspex server. For readers familiar with Patterson and Katz's work on Redundant Arrays of Inexpensive Disks (RAID [Patterson88]), Auspex's disk arrays are organized in the simplest fashion: RAID 0 (independent disks) and RAID 1 (dual-disk mirroring)—no striped parity scheme is used. For NFS, which requires small-block random I/Os, RAID 0 and 1 work extremely well. Questions of file-system load balancing across a multiple-disk array are addressed below.

Chart 5 shows the scalability of the NetServer disk array over 4–16 disks and very high offered NFS loads. Disk array throughput scales nearly linearly (1.9:2.0). This is outstanding: There must be no bottlenecks in the entire system to achieve this overall client-to-server NFS throughput linearity.

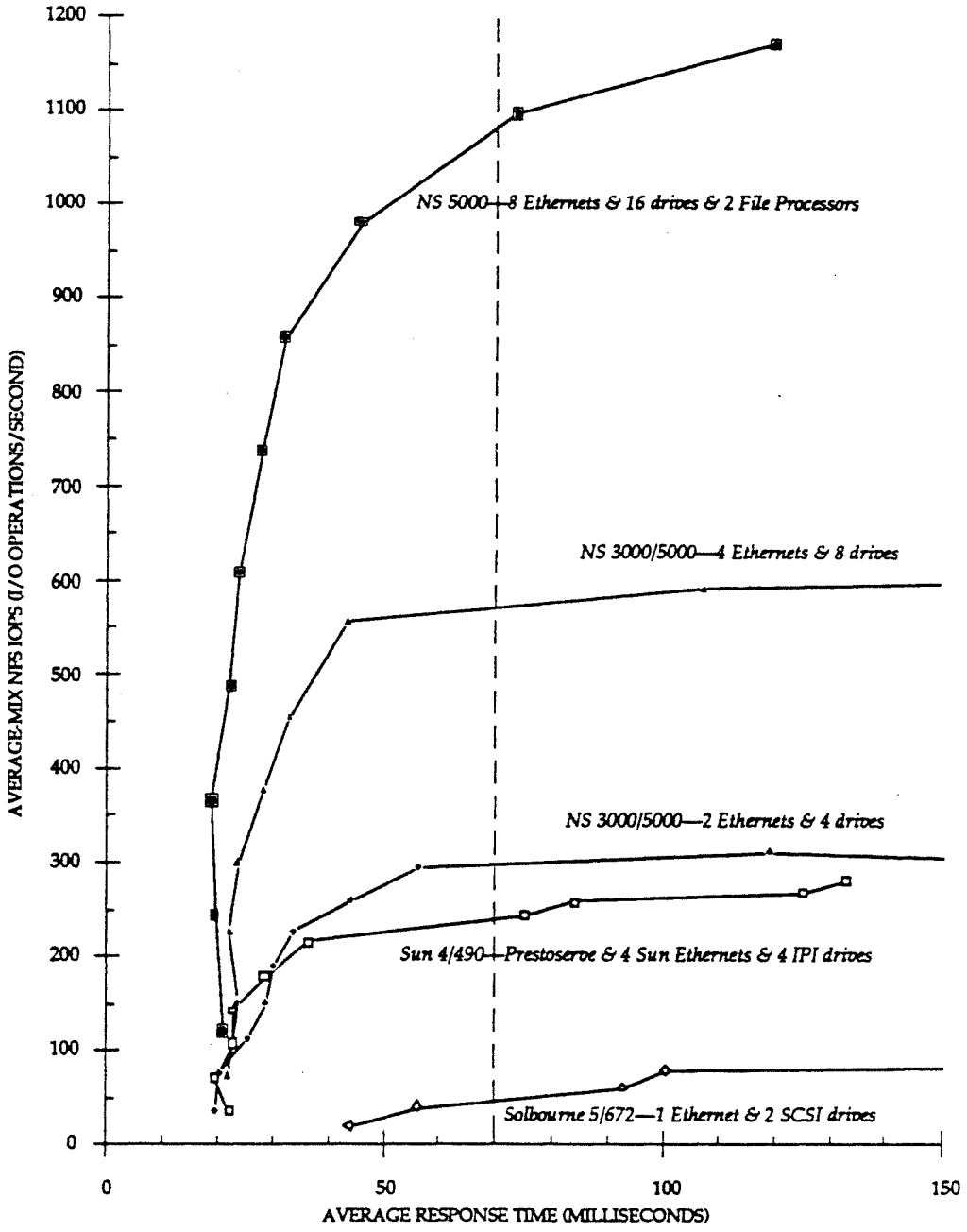


Chart 5: A scalability study of the NetServer SCSI disk arrays. In scaling a NetServer from a 1-Ethernet, 2-disk-drive system to 2 nets and 4 drives, 4 nets and 8 drives, and 8 nets and 16 drives (NS 5000 only), the NetServer has a remarkably linear NFS throughput increase of 1.9 (geometric mean) per step. This demonstrates that SCSI disk arrays can provide almost-perfectly scalable random-access throughput. This is due to two factors: the ability of the NetServer's extra Ethernet Processors to feed NFS load into the system, and—importantly for this report—the ability of the Storage Processor's disk array to respond to that load. (The 1-network, 2-disk NetServer curve is not shown; this configuration attains 165 IOPS at 70ms.) This test uses the NHFSStone default NFS operation mix (table 6). NFS IOPS—I/O operations per second—and the customer's NFS operation mixture are explained in section 4.2. The Auspex test configuration was: an NS 5000 running version 1.2 software and SunOS 4.0.3 with 2–16 1-GB SCSI-2 disks.

Organizing a file server's disk arrays as independent disks (RAID 0) begs the question of how a system administrator balances file system I/O loads across those many disks. In practice, Auspex has found that system administrators have good intuition about their user's requirements.

Table 7 shows the disk activity levels of several NetServer customers. These levels are long-term (several day) averages. They were measured by special software instrumentation inside the Storage Processor. Here is a brief interpretation of this operational data:

- The queue and service times are very uniform across all disks. This shows that the disks are evenly loaded, and that the respective system administrators have done good jobs. Two major exceptions are shown in bold: Auspex disk 11 has 0 queue time, so it is probably very infrequently used or read-only (i.e., never written; see below). 3Com disk 6 has a deep queue, so it is probably heavily written (e.g., an overloaded swap disk, since service times are so short that the writes are close together on the disk).
- Average disk service times are 15–22ms, less than the 29ms average random I/O times computed in table 2. This means that most NFS I/Os probably require less than an average seek, and that the NetServer's SCSI-drive read-head is finding *some* sequential access traffic to optimize. So it appears that even large NFS servers will see some locality of disk reference, probably because the Unix File System's cylinder group notion is working well.
- The average queue time is 10–30ms—about the above-measured time of a single 8-KB I/O. This subtle finding probably indicates that two disk I/Os frequently arrive as pairs. We speculate that they are often writes, because an NFS *Write* is usually transformed into two disk writes—one for the file's inode, and another for the 8-KB data block.
- The previous point implies that the NetServer disk array usually encounters NFS *Write* operations, that such writes are usually not themselves queued (i.e., that the disk array is not heavily loaded), and that most NFS *Reads* come from the server's file cache. Other operational statistics indicate that the average read-to-write ratio *at the disk* is 1:3, rising to as much as 1:6 for /tmp file systems and swap partitions. This shows the effectiveness of the primary memory buffer cache, which services most reads without any disk activity. These conclusions are tentative and need more study, but are our best estimates at present.

System	Disk Drive ➡	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Adobe	Queue time (avg. ms)	18	17	27	22	25	24	23	38	14	27	17	27				
	Service time (avg. ms)	20	20	21	20	18	16	19	17	18	19	18	16				
Auspex	Queue time (avg. ms)	7	13	8	15	9	31	22	21	21	17	0	46	34	9	29	18
	Service time (avg. ms)	22	21	21	20	19	17	17	19	16	17	26	22	21	15	22	19
3Com	Queue time (avg. ms)	13	12	6	45	33	92	24	33								
	Service time (avg. ms)	22	15	19	19	20	20	33	18								

Table 7: Sample disk-array queue times and disk service times from operational NS 5000 systems. In general, each system's individual disk loads are very well balanced (the two exceptions, the bold 0 and 92, are discussed in the text). This indicates that experienced system administrators have good intuition about how to assign swap partitions and file systems to disk drives for even load balancing. In addition, the low disk service times indicate that there is a mild and unexpectedly pleasant degree of locality of disk reference, even for heavy NFS loads. See the text for further discussion.

In Auspex's current server software, two built-in capabilities make it easy for system administrators to balance their disk and file-system loads:

- Disk concatenation, striping, and mirroring are fully supported as *virtual partitions* [Cheng91]. Both mirroring (for reads) and striping (for reads and writes) distribute disk traffic across multiple physical disks and eliminate hot-spot disks. The two techniques can

be used together to construct a mirrored, striped, maximal-size file system—larger than a physical disk.

- Auspex's servers come with a comprehensive performance monitor [Hitz90b] that, in the storage area, shows disk I/O rates, I/O types, and virtual partition access information. Similar information is provided for file systems, including file-operation rates, cache hit rates, and cache age distributions. This makes it easy to spot currently overloaded drives or file systems, and to predict forthcoming add-more-users overloads.

5 CONCLUSION

This paper began with two objectives: explore and explain the complex behavior of modern disk drives, and show that NFS file server disk traffic has a random—not sequential—access pattern. Conclusions from these apparently dissimilar investigations were checked in a performance evaluation of IPI and SCSI disk subsystems in high-performance NFS file servers. SCSI disk arrays—even using first-generation SCSI-1 disk drives—easily outperformed contemporary, optimized IPI drive subsystems on NFS workloads. The SCSI subsystem also demonstrated nearly linear throughput scalability, a remarkable achievement given its low cost.

Some further conclusions:

- Disk transfer rate is the *least* critical aspect of disk performance for NFS workloads in file servers using modern extent-based file systems like the UFS Unix file system. Transfer rate is important for data-intensive compute-server applications where a small file is >> 1 MB.
- High per-disk random I/O rates and fully concurrent disk activities are essential for good NFS performance. SCSI disks have built-in controllers for concurrent, autonomous activity. IPI-2 disks do not.
- Disk array organization gives superior random I/O rates and can deliver extreme transfer rates (20–40 MB/s) with striping. SCSI disk-array controllers supporting this functionality can be readily built because of the sophisticated, merchant-market SCSI interface ASICs available. These controllers have 5–10 SCSI channels per controller.
- SCSI disk arrays have exceptional scalability, capacity, and performance. In large part, this is because of their built-in controllers and track buffering. But attractive pricing of both the SCSI disks and SCSI interface components—induced by high-volume shipments—is key as well.

SCSI *is* better than IPI for NFS.

6 REFERENCES

[Auspex90] *The Auspex NS 3000/5000 vs. the Sun/Presto/Omni 4/490 in Customer NFS Benchmark Showdown #1.*
Benchmark Brief 5, Auspex Systems Inc., December 1990.

[Boggs88] David R. Boggs, Jeffrey C. Mogul, and Christopher A. Kent.
Measured Capacity of an Ethernet: Myths and Reality.
Proceedings of the SIGCOMM '88 Symposium on Communications Architectures and Protocols, ACM SIGCOMM, Stanford, CA, August 1988.
Also Digital Western Research Laboratory Research Report 88/4.

[Cheng90] Yu-Ping Cheng.
[SCSI] Disk Drive Performance.
Internal Memorandum, Auspex Systems Inc., 22 June 1990.

[Cheng91] Yu-Ping Cheng, Guy Harris, and William M. Pitts.
Using Virtual Partitions for Disk Storage Administration in Network Servers.
Application Note 2, Auspex Systems Inc., in preparation.

[Hitz90a] David Hitz, Guy Harris, James K. Lau, and Allan M. Schwartz.
Using Unix as One Component of a Lightweight Distributed Kernel for Multiprocessor File Servers.
Proceedings of the Winter 1990 USENIX Conference, Washington,DC, 23–26 January 1990.
Also Technical Report 5, Auspex Systems Inc., January 1990.

[Hitz90b] David Hitz.
A System Administrator's Performance Monitor for Tuning NFS Network Servers.
Proceedings of the Sun User Group, San Jose, CA, 3–5 December 1990.
Also Technical Report 7, Auspex Systems Inc., in preparation.

[Horton90] William A. Horton and Bruce Nelson.
An NFS Benchmark Comparison of the NS 5000 Network Server, with the Solbourne Series 5/672, and Sun 4/490 and 4/280 File Servers.
Performance Report 2, Auspex Systems Inc., June 1990.

[Howard88] John H. Howard, Michael L. Kazar, Sherri G. Menees, David A. Nichols, M. Satyanarayanan, Robert N. Sidebotham, and Michael J. West.
Scale and Performance in a Distributed File System.
ACM Transactions on Computer Systems 6(1): 51–81, February 1988.

[HP89] Hewlett-Packard Company.
HP 9754XS/D SCSI Disk Drives OEM Product Manual.
Manual Order Number: HP 19530, August 1989.

[HP90] Hewlett-Packard Company.
HP 97540T/P Series 5.25-inch Winchester Disk Drives Product Description Manual.
Draft manual, no order number available, August 1990.

[IPI2] American National Standards Institute.
American National Standard for Information Systems—Intelligent Peripheral Interface 2 (IPI-2).
Documents ANSI X3.129-1986 (physical) and X3.130-1986 (device).

[Kleiman86] Steven R. Kleiman.
Vnodes: An Architecture for Multiple File System Types in Sun Unix.
Proceedings of the Summer 1986 USENIX Conference, Atlanta, Georgia, June 1986.

[Lyon89] Bob Lyon and Russel Sandberg.
Breaking Through the NFS Performance Barrier.
SunTech Journal 2(4): 21–27, Autumn 1989.

[McKusick84] Marshall K. McKusick.
A Fast File System for Unix.
Transactions on Computer Systems 2(3): 181–97, August 1984.

[Nelson88] Michael N. Nelson, Brent B. Welch, and John K. Ousterhout.
Caching in the Sprite Network File System.
ACM Transactions on Computer Systems 6(1): 134–54, February 1988.

[Nelson91] Bruce Nelson.
An Overview of Functional Multiprocessing for Network Servers.
SunTech Journal 4(1): 84–89, January 1991 (edited version).
Also technical Report 1, 3rd edition, Auspex Systems Inc., November 1990.

[Patterson88] David A. Patterson, Garth Gibson, and Randy H. Katz.
A Case for Redundant Arrays of Inexpensive Disks (RAID).
Proceedings of the ACM SIGMOD Conference, pp. 109–16, 1–3 June 1988.

[Rosenblum90] Mendel Rosenblum and John K. Ousterhout.
The LFS Storage Manager.
Proceedings of the Summer 1990 USENIX Conference, Anaheim,CA, 11–15 June 1990.

[Sandberg85] Russel Sandberg, David Goldberg, Steve Kleiman, Dan Walsh, and Bob Lyon.
Design and Implementation of the Sun Network File System.
Proceedings of the Summer 1985 USENIX Conference, pp. 119–30, Portland, OR, June 1985.

[Schwartz90] Allan M. Schwartz, David Hitz, and William M. Pitts.
LFS—A Local File System for Multiprocessor NFS Network Servers.
Proceedings of the IEEE Systems Design & Networks Conference '90, Santa Clara, California, 8–10 May 1990.
Also Technical Report 4, Auspex Systems Inc., December 1989.

[SCSI1] American National Standards Institute.
American National Standard for Information Systems—Small Computer System Interface (SCSI).
Document ANSI X3.131-1986.
This SCSI specification should be read in conjunction with the *Common Command Set (CCS) of the Small Computer System Interface (SCSI)*, ANSI document X3T9.2/85-52 (revision 4B).

[SCSI2] American National Standards Institute.
Draft Proposed American National Standard for Information Systems—Small Computer System Interface (SCSI).
Documents ANSI X3T9.2/86-109 (revision 10C) and X3T9/89-042.

[Seagate90] Seagate Technology.
Sabre Series 8-inch Disk Drives.
Data sheet document number 1330-001 (204702), March 1990.

[Srinivasan89] V. Srinivasan and Jeffrey C. Mogul.
Spritely NFS: Experiments with Cache-Consistency Protocols.
Operating Systems Review 23(5): 45–57, December 1989.

The Auspex NS 5000 and the Sun SPARCserver 490 in One- and Two-Ethernet NFS Performance Comparisons

William A Horton
Bruce Nelson

Performance Report 2
May 1990

AUSPEX

ABSTRACT

This performance report compares the abilities of the Auspex NS 5000 and the Sun SPARCserver 490 and 4/280 Data Center Servers to provide NFS file service to client workstations on Ethernet networks. The benchmark comparing the two servers uses a synthetic workload that is calibrated to closely approximate the workload actually seen by NFS servers in CASE and CAD environments. This workload is generated by a special-purpose NFS traffic generator that is capable of saturating eight Ethernets with NFS traffic.

The results of this NS 5000 performance study demonstrate that the NS 5000 has: 3.7–10.0 times the NFS throughput of the 4/490 on two or more Ethernets (the minimum NS 5000 configuration); one-half the response time (15–20 milliseconds); and can handle 770 average-mixture NFS IOPS (I/O operations/second) at a 70-ms response time. The results also show that parallel SCSI disk arrays—as used in the NS 5000—offer extremely good NFS performance. This is in marked contrast to typical IPI-2 disk subsystems, which demonstrate poor NFS throughput.

Revision 1.5, 910314.

Auspex Systems Inc.
2952 Bunker Hill Lane
Santa Clara, California 95054 USA
Phone: 408/492-0900 · Fax: 492-0909
Email: Info@Auspex.com or uunet!Auspex!Info

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Overview of Benchmark

This performance report presents an empirical benchmark study comparing the Auspex NS 5000 Network Server to the Sun SPARCserver 490 and 4/280 Data Center Servers. These three servers are compared at their ability to provide NFS file service [Sandberg86] to remote workstations directly connected to the servers via Ethernets. Neither local I/O nor compute capabilities are considered in this strictly client-server NFS I/O study. A detailed description of benchmark methodology and results appears in the next sections.

The NS 5000 is a high-performance Unix network server that achieves exceptional NFS throughput using a unique *functional multiprocessor* architecture [Auspex89a]. The NS 5000 supports up to eight Ethernets with full protocol processing, offers standard, concurrent SCSI disk arrays, and has software I/O functions completely factored out of Unix [Hitz90, Schwartz90].

1.2 Summary of Results

The benchmark results are summarized in table 1 below. The 4/490 and NS 5000 data have been verified at multiple sites.

NFS Server Configuration	*# Ethernets*	*1*	*1*	*1*	*2*	*2*	*4*	*8*
	# Disks	*1*	*2*	*4*	*2*	*4*	*8*	*16*
NFS Server IOPS (@ 70ms)								
Auspex NS 5000		111	170	225	187	300	455	770
Sun 4/280		53					NA	NA
Sun 4/490		78	62	54	51	30	NA	NA
NS 5000 Throughput Increase Over								
Sun 4/280		2.1					NA	NA
Sun 4/490		1.4	2.7	4.2	3.7	10.0	NA	NA

Table 1: A summary of the NS 5000 and 4/490 NFS benchmark comparison. Performance of the Sun 4/280 file server is included for reference as well. The NS 5000 is from 1.4–10.0 times faster than the 4/490. Notice that the throughput of the 4/490 *decreases* as either networks or disks are added to the system. This perhaps surprising but easily explained result is a consequence of the 4/490's workstation-as-server architecture and is discussed further in the text. Because the 4/490 and 4/280 are not available in 4 and 8 Ethernet configurations, they are not compared with the NS 5000, although their throughput would apparently be < 10% of the NS 5000's. Finally, the NS 5000's 1-Ethernet, 4-drive throughput is actually higher than 225 IOPS, but is so great that it cannot be fully measured with Auspex's Traffic Generator. Each server's NFS throughput is measured in average-NFS-mixture I/O operations per second (IOPS) at a 70 millisecond response time as explained later in section 3.

2 DESCRIPTION OF RESULTS

2.1 Single Ethernet Results

The major single-Ethernet highlights of table 1 follow.

- **The NS 5000 has 1.4–4.2 times the throughput of the 4/490 on one Ethernet.** Table 1 shows NS 5000 and 4/490 throughput using one Ethernet and 1–4 disk drives; this data is extracted from chart 1. Using a liberal response time cutoff of 70 milliseconds (see *Response Time Cutoff*, later), the NS 5000 has a throughput advantage over the 4/490 of 1.4 with 1 drive, 2.7

with 2 drives, and 4.2 with 4 drives. These factors illustrate the NS 5000's dramatic architectural advantage over enhanced conventional server designs like the 4/490. The 4/490 is in a typical configuration, while the NS 5000 is underutilized because it has an idle, unused Ethernet port in this comparison.

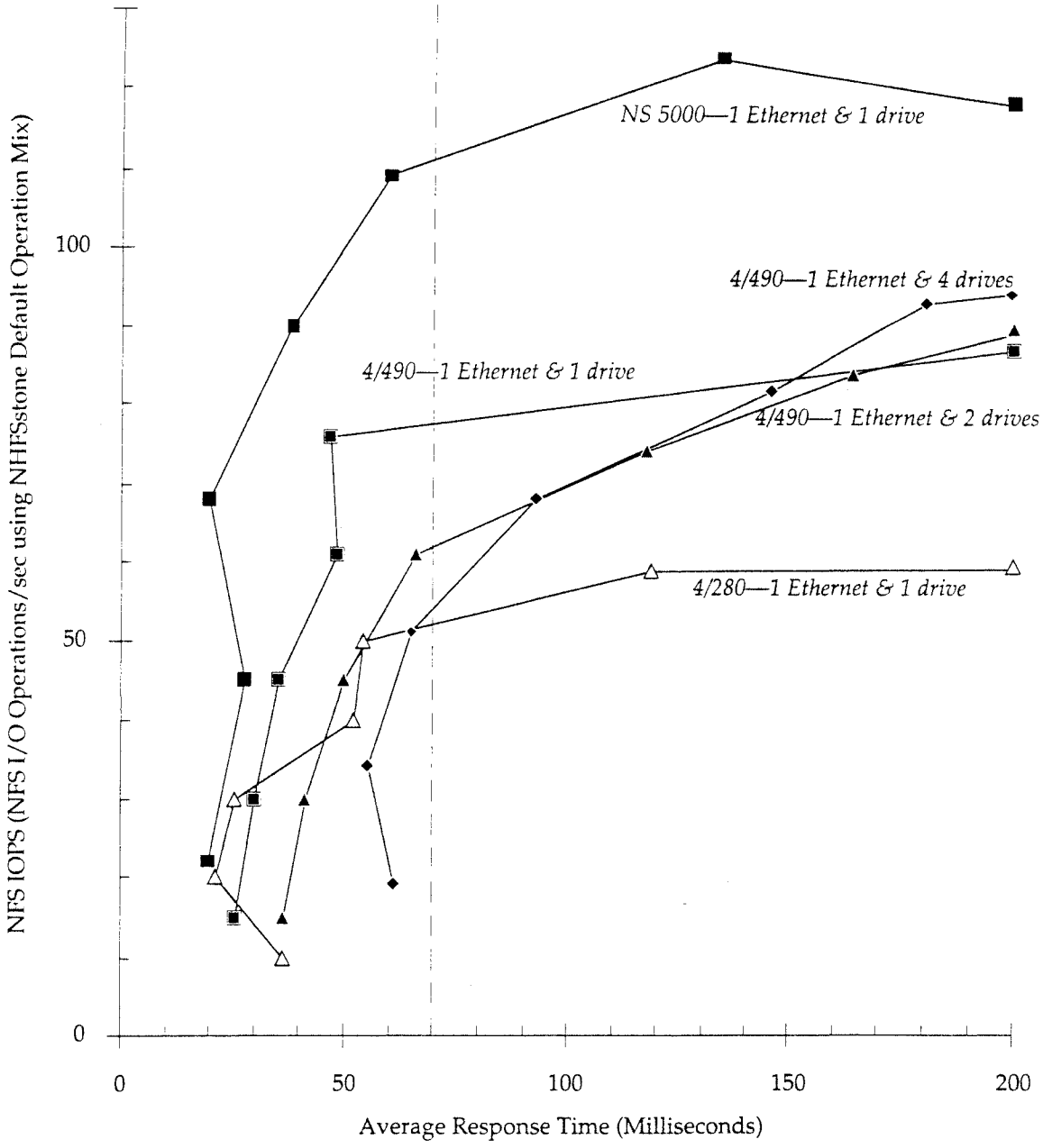


Chart 1: An NFS throughput comparison of the NS 5000 and the 4/490 on a single Ethernet. In an identical configuration of 1 Ethernet and 1 disk drive, the NS 5000 executes 111 NFS IOPS, 1.4 times as many 4/490's 78 IOPS and 2.1 times as many as the 4/280's 53 IOPS. On a single Ethernet, NS 5000 throughput increases from 111 to 170 to 225 IOPS as the number of disk drives increase from 1 to 2 to 4. Notice how the 4/490's throughput actually *decreases* with more disks because of serial IPI-2 disk operation. Both this decrease and the saturated-disk, 150-ms throughput crossover are explained in the text. NFS IOPS—I/O operations per second—are the appropriate measure of an NFS file server's I/O performance since server I/O throughput is dependent on system-level I/O architecture, not computational CPU MIPS. The IOPS ratings in this note are measured at 70 milliseconds response time as explained under *Response Time Cutoff*.

- **The NS 5000's SCSI disk arrays are demonstrably superior to the 4/490's IPI disks for NFS service.** Table 1 indicates that 4/490 NFS throughput decreases as disk drives are added. This behavior is clearly shown in chart 1. It is readily explained by the 4/490's IPI-2 disk and controller implementation as explained later in section 2.4.

2.2 Dual Ethernet Results

The major dual-Ethernet highlights of table 1 are:

- **The NS 5000 has 3.7–10.0 times the performance of the 4/490 on two Ethernets.** Table 1 shows NS 5000 and 4/490 performance using 2 Ethernets and 2–4 disk drives; this data is extracted from chart 2. The NS 5000 has a throughput advantage over the 4/490 of 3.7 with 2 drives and 10.0 with 4 drives (at the same 70-ms cutoff).
- **The NS 5000 has 4.8 times the performance of the 4/490 for comparable base systems.** The base-level model NS 5000 delivers 300 NFS IOPS and has 2 Ethernet ports and 4 disk drives (2.6 GB). The corresponding SPARCserver 490 model 490-S-32-P21 supports 62 IOPS and has 1 Ethernet port and 2 disk drives (2.0 GB). The NS 5000's 300 IOPS offers 4.8 times the throughput of the 4/490's 62 IOPS.
- **The 4/490 performs worse on two Ethernets than one.** Table 1 shows that a 2-drive 4/490 supports 62 IOPS with 1 Ethernet and 51 IOPS with 2 Ethernets. This 18% *decrease* in NFS throughput—demonstrated side-by-side in chart 2—is probably caused by long 4/490 VME interrupt latencies and/or the simple, VME Ethernet interface lacking DMA. This nonintuitive result is confirmed by the 4/280's more marked 62% NFS capacity decrease going from 1 to 2 Ethernets with a single disk drive (from 53 to 20 IOPS in charts 1 and 2). Recall that both the 4/490 and the 4/280, as retrofitted-workstation architecture servers, include their first Ethernet port on the CPU motherboard, and additional ports as VME option boards.
- **The NS 5000 has less than half the delay due to NFS response time.** The left-hand portion of chart 2 indicates that the NS 5000 latency is less than half the delay of the 4/490 on 2 Ethernets. In particular, near the origin the NS 5000's average response time is about 15–20 milliseconds and the 4/490's is about 30–40 milliseconds. The NS 5000's quicker response time translates directly into a more responsive workstation for interactive users, who report that this speed is indeed perceptible. The NS 5000 achieves this substantial response-time improvement because its functional multiprocessor architecture is optimized for NFS service.

2.3 Multiple Ethernet Results

Chart 3 compares multiple-Ethernet configurations of the NS 5000 with the best-performing configuration of 4/490 (1-network, 1-drive from charts 1 and 2). The major highlights are:

- **Linear scalability of networks and storage.** Chart 3 clearly shows that as the number of networks and disks double in an NS 5000, the IOPS capacity of the system increases from 170 to 300 to 455 to 770 at the same 70ms response time cutoff. The geometric mean of this capacity increase is 1.7 for each doubling of disk and Ethernet resources. Because performance increases directly with increased capacity in the NS 5000, it has expansion potential that cannot be matched by general purpose file servers like the 4/490. Again, this advantage is due to the NS 5000's functional multiprocessor I/O architecture.
- **Excellent overload stability.** As response times exceed 70ms, the I/O performance of all the servers gracefully plateau under intentional overload. This stable behavior is desirable.

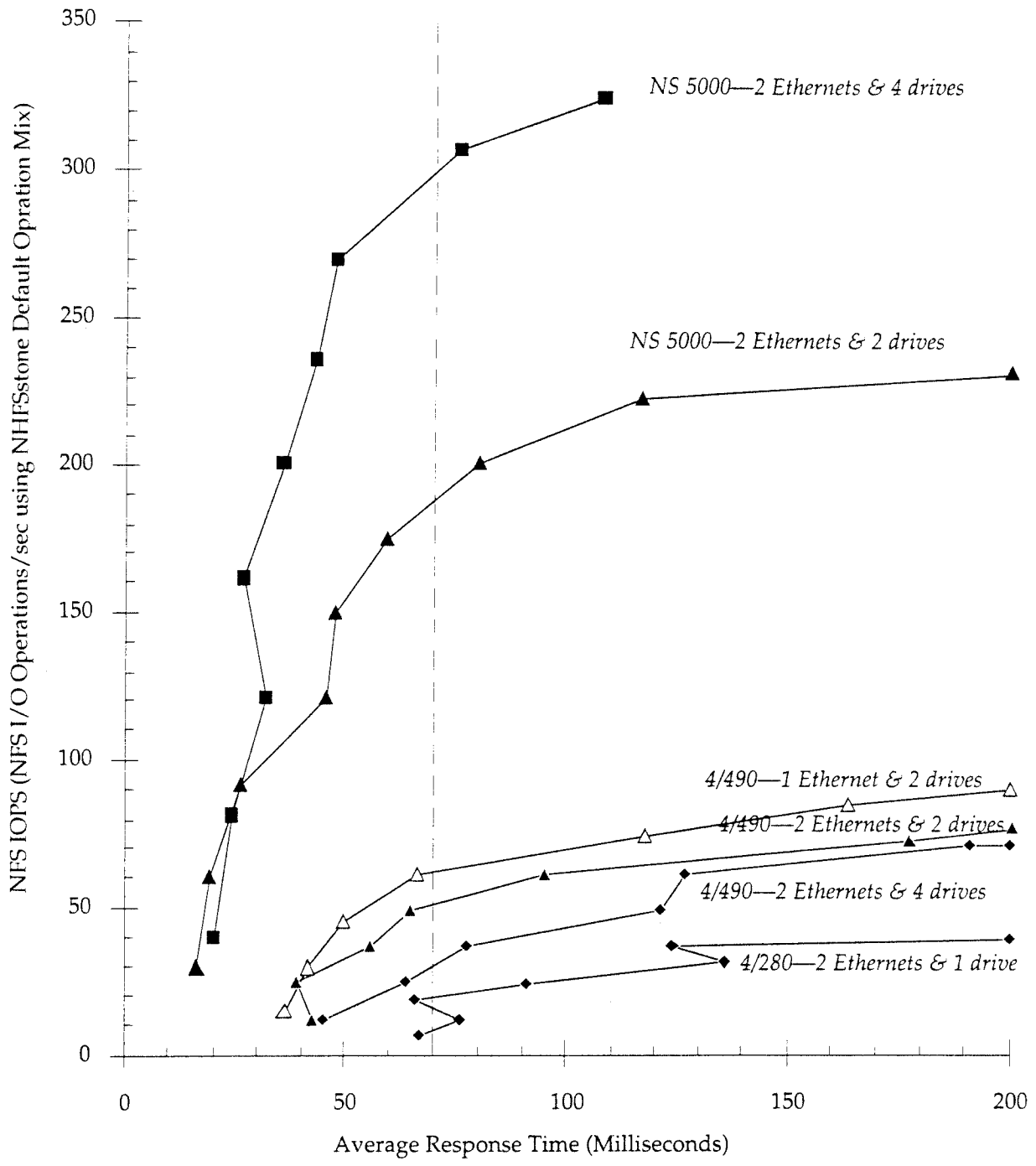


Chart 2: An NFS throughput comparison of the NS 5000 and the 4/490 on two Ethernets. In comparable base systems, the NS 5000 (with 2 Ethernets and 4 disk drives) executes 300 NFS IOPS, 4.8 times as many as the 4/490's 62 IOPS (with 1 Ethernet and 2 drives). Notice that both 4/490 and 4/280 throughput *decreases* as the second Ethernet is added; this is explained in the text (for the 4/280, compare charts 1 and 2). The NS 5000's greatly reduced NFS operation latencies (quicker response times) are discussed in the text.

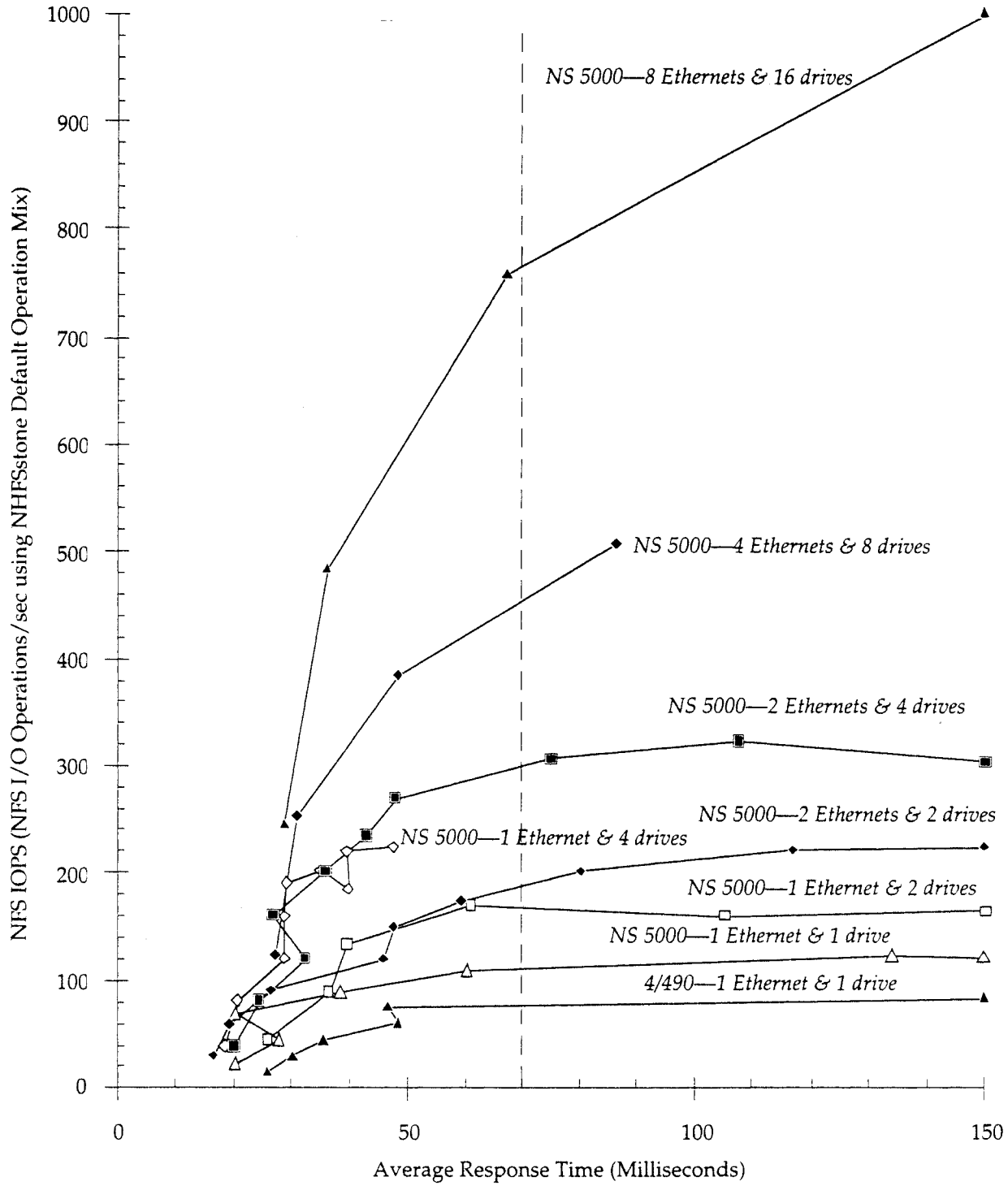


Chart 3: An NFS throughput comparison of the NS 5000 on multiple Ethernets. In doubling and redoubling the number of networks and drives, the NS 5000's linear performance scaling (by a factor of 1.7) is clearly indicated. In its 8-Ethernet, 16-drive configuration (the sole configuration using 2 File Processors), the NS 5000 achieves 770 NFS IOPS at 70ms and 1,000 IOPS at 146ms. The NS 5000's 1-Ethernet, 4-drive throughput is actually higher than 225 IOPS, but is so great that it cannot be fully measured by Auspex's Traffic Generator and appears somewhat erratic in the limit.

2.4 SCSI and IPI-2 Disk Subsystem Characteristics

SCSI disk arrays and IPI-2 storage subsystems are illustrated in figure 1. Notice the bandwidths of the different bus and device interfaces, the presence or absence of disk-level buffering, and the number of channels per controller.

Despite their slightly higher transfer rates, the IPI disk drives are unbuffered at the disks and are serial slaves of the IPI-2 controller's single channel. This means that on a given IPI controller only one disk actually operates at a time. The NS 5000's SCSI disks, in contrast, *each* have independent, fully-buffered controllers attached to a separate SCSI channel—with 10 concurrent channels per Storage Processor. These architectural tradeoffs boil down to the fact that the NS 5000 can run 20 disks transferring data in parallel, but the 4/490—if equipped with 4 $5,700 (list price) IPI controllers—can have at most 4 disks transferring in parallel. This beneficial SCSI behavior is confirmed by recent Solbourne file server benchmarks using SCSI disks (these results will be fully reported in a future performance report). Providing rapid NFS service to many simultaneous workstation users requires exceptional parallel I/O-request capability.

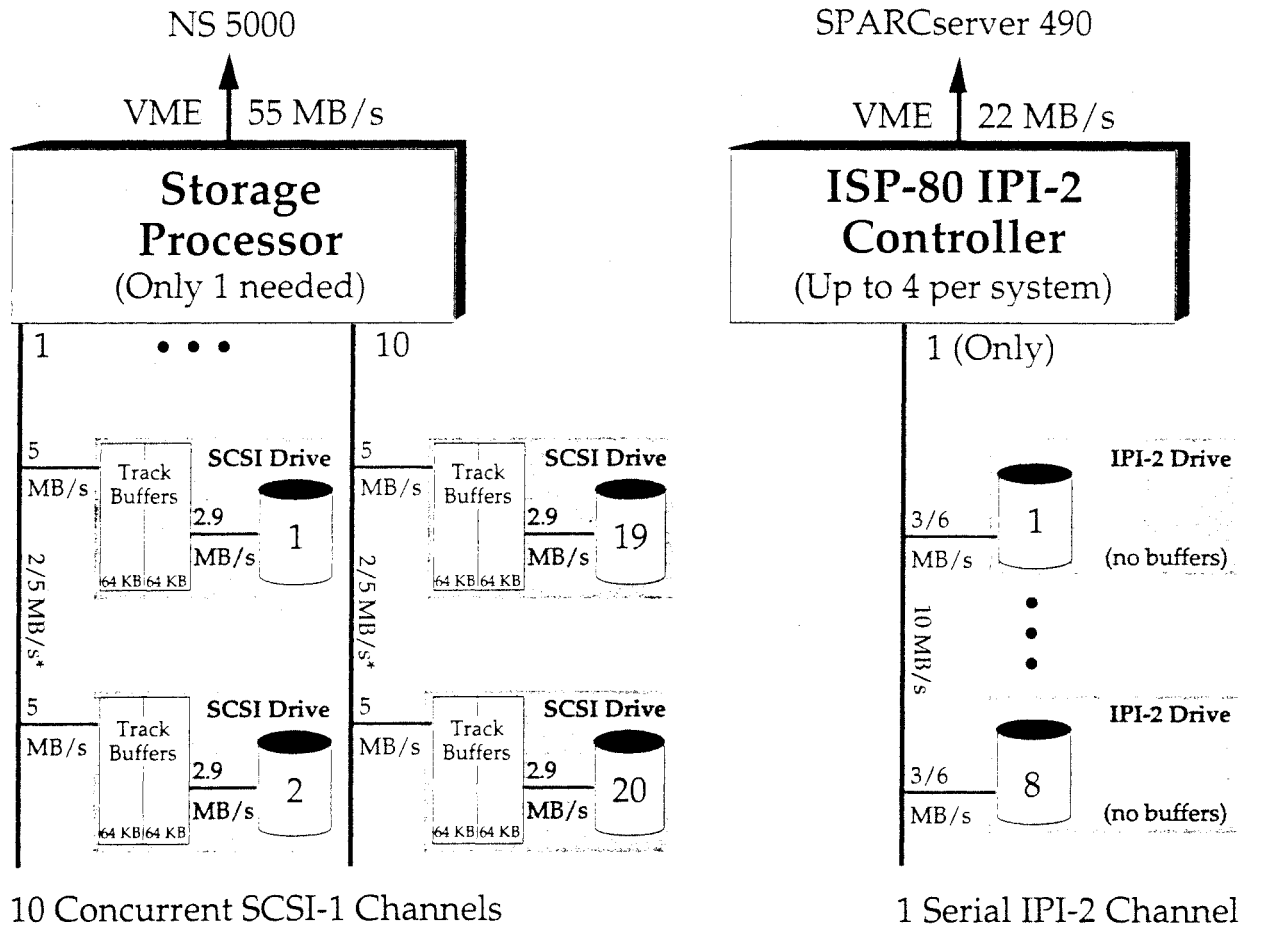


Figure 1: An architectural comparison of SCSI disk arrays and IPI-2 disk subsystem architectures. The storage subsystems of the NS 5000 and the Sun 4/490 are shown as concrete examples. The NS 5000 has a maximum of 20 drives in its disk array, and each can read and write concurrently. The 4/490 can have up to 4 IPI controllers, but each controller must read and write from each of its disks serially. For an NFS server handling the independent requests of 50–100 different users, the fully parallel operation of SCSI disk arrays has demonstrably superior throughput. The text explains these NFS performance implications further.

3 BENCHMARK METHOD AND CONFIGURATIONS

Auspex's general benchmark methodology is elaborated in another report [Auspex89b]. The benchmark used for this comparison measures NFS server response times under increasing NFS traffic load.

3.1 Benchmark Environment and Conditions

In this study, Auspex's special-purpose NFS Traffic Generator—illustrated in figure 2—generates carefully metered synthetic NFS workloads to each server configuration. Each Traffic Generator Ethernet contains three nodes (as shown in figure 2) so there is no extraneous, interfering traffic from other workstations or servers. All file systems used for testing are fresh and empty, eliminating storage fragmentation concerns. The servers under test have no active jobs other than the normal background daemons necessary for standard Unix and ONC services. While some of these test conditions do not reflect actual operating conditions, they are necessary to perform accurate *relative* performance comparisons.

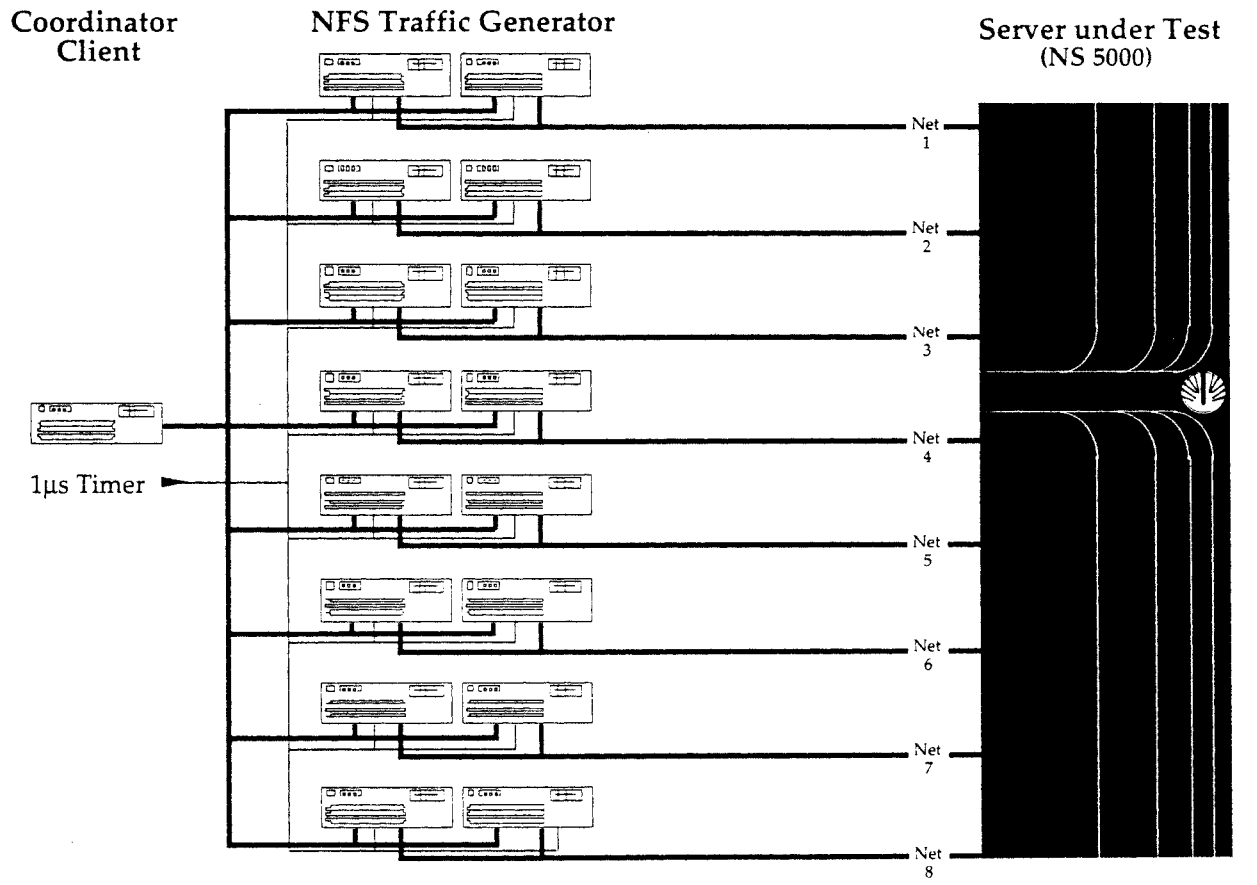


Figure 2: The Auspex NFS Traffic Generator connected to the NS 5000. The Traffic Generator can saturate eight Ethernets with NFS traffic, varying both the load and the operation mixtures dynamically. The Coordinator node runs scripts and logs statistics so that exactly repeatable tests can be run against different servers. The eight traffic networks—each with two traffic-generating clients—are isolated from all other Ethernet traffic to prevent interference. The server's response time for each NFS operation is measured at the initiating client's Ethernet interface by a 1-µs resolution timer that is distributed from the Coordinator.

3.2 Benchmark Method

Traffic Generator benchmark trials are conducted by:

1. Defining a series of fixed system *configurations* (e.g., Traffic Generator, server, Ethernets, disks) and performing steps 2–3 for each configuration in turn.
2. Increasing the NFS load in separate *trials* from 0% to 100%, usually in 10% steps, where 100% is predetermined to drive the configuration well into overload (i.e., increasing response time with no increase in IOPS).
3. Measuring the *response times* in milliseconds and *operation rates* in IOPS for each trial. Each benchmark trial has a 90-second duration to eliminate start-up anomalies. Response time and operation rates are computed as follows:
   - *Average response time.* At the conclusion of each trial, the average response time for each NFS operation is computed by dividing the accumulated individual response times for that operation by the total executions of that operation during the trial (typically thousands). These arithmetic averages are in turn weighted by the average-mixture NFS operation ratios in table 2 to obtain a single response time figure (i.e., the final response time is the sum of the weighted averages). Response time is measured with a 1-microsecond resolution timer at the client's (Traffic Generator's) Ethernet, not in the server or at the client's application level.
   - *Weighted NFS IOPS.* At the finish of a trial, the total number of executions of each separate NFS operation is divided by the trial's duration to compute the individual IOPS for each operation. These rates, which were generated at the average-mixture ratios during the trial itself, are then simply summed to give the final weighted NFS IOPS result for the trial.

Quantitative results for selected configurations are then plotted, as in charts 1–3.

3.3 Response Time Cutoff

NFS response time is the most critical performance metric for a workstation user. Most users would be satisfied (if not elated) with NFS I/O response times of 20–40 milliseconds. These same users, however, would quickly notice and complain if response times exceeded 100ms. Because 4/490 NFS throughput is low at response times < 50 ms (see charts 1 and 2), this study uses a 70ms cutoff time to measure performance.

3.4 Average NFS Operation Mixtures and Traffic

The benchmark workload in this Auspex study consists of synthetic NFS traffic generated in an average operation mixture that corresponds to the mixture encountered on servers handling client workstations executing CASE, CAD, and publishing applications. The NFS operation mixture ratios used in this study appear in table 2.

Lookup	34%	GetAttr	13%	Create	2%	StatFS	1%
Read	22%	ReadLink	8%	Remove	1%		
Write	15%	ReadDir	3%	SetAttr	1%	*Total*	*100%*

Table 2: Typical average-mixture NFS operation ratios. Note that read and write combined constitute 37%, or just over one-third of all operations. Directory operations (Lookup, GetAttr, SetAttr, ReadLink, and ReadDir) constitute 59%, or approach two-thirds. The remaining seven (of the seventeen total) NFS operations unrepresented here are executed so infrequently as to be negligible.

These ratios are surprisingly invariant under workload and usually change little (±5%) from those presented here. They can, however, change substantially with client configuration, notably that read traffic is often greatly decreased for dataless clients or clients with large (≥8MB) primary memories. The ratios in table 2 characterize a server connected to Sun 3/50-class client workstations.

RISC-based Unix workstations such as the DECstation 3100 and SPARCstation 1 generate an NFS load of about 10–20 IOPS when performing moderate NFS I/O to an NFS server. This range of 10–20 was empirically measured on an application workload that compiled remote C sources. Client traffic does depend significantly on the workstation's local disk and memory configuration, as shown in table 3.

Client Workstation Configuration	Client NFS Load Performing Moderate I/O
Diskless with 8-MB memory	22 NFS IOPS
Diskless with 16-MB memory	13 NFS IOPS
Dataless with 8-MB memory	11 NFS IOPS
Dataless with 16-MB memory	9 NFS IOPS

Table 3: The NFS load generated by different RISC-based workstation configurations performing moderate I/O (e.g., compiling remote source files). This data is based on a *single* simple application; actual traffic will vary significantly, at least by a factor of 2. Diskless workstations obviously generate significant swap traffic. Dataless workstations have /swap and /tmp (and possibly /root) directories on a local disk attached to the workstation. The local /tmp file system reduces remote traffic during compilations, which use many temporary files.

3.5 Workload Details

The Traffic Generator's workload executes by reading and writing files from standard Unix file systems, each mounted on a single large partition per disk. Each Traffic Generator client performs operations at two mount points on different disks, and thus there are two clients generating test traffic to two separate mount points on each disk. The NFS workload is uniformly distributed across all available networks and partitions. Although each trial begins with cold caches, the read cache (Unix buffer cache) will fill early during each trial, so NFS reads come primarily from the buffer cache.

The workload used in this 4/490 study is different from and more elaborate than the workload used in Performance Report 1 [Horton90], which compared the NS 5000 to just the 4/280. The workload increments were also more carefully controlled in this study than in earlier tests. The result of this heavier-duty, more carefully controlled workload is that the absolute IOPS ratings have changed slightly from Report 1 although the relative performance comparisons are unchanged. In particular, in this study:

- Each traffic generator client performs NFS operations that cycle through a private subtree of about 1,000 files and 12 MB of total read-write data. Since the traffic generator simulates multiple clients accessing a single file system on each disk, these large data sets cause substantial seek activity.
- All NFS writes are *append* operations to a series of small files (≤96KB) stored in the same directory. This implies that each NFS write operation requires two different disk writes. Since typical applications write files by truncating and then appending, *append* writes are the most realistic write workload.

- For low response times on the 4/490, the load was increased gradually to study the 4/490's multiple disk behavior. This clarified the 4/490's inconsistent response times ≤ 50ms. It also verified that, in the limit of response times ≥ 150ms, multiple 4/490 disks in saturation (constantly busy) do perform better than a single disk (the 150ms crossover in chart 1).

Auspex believes that this synthetic benchmark workload, based on empirical average-mixture NFS measurements taken with *nfsstat*, is an accurate indication of *relative* performance between the NS 5000 and other NFS servers. But of course for a particular application, client-server network configuration, or server file system organization, absolute results will vary.

3.6 Test Configuration Details

The Auspex NS 5000 benchmark configuration was:

- Hardware: 1–4 Ethernet Processors, 1–2 File Processors, 1 Storage Processor, 1 16-MB Primary Memory module, 1 Sun 3E120 Host Processor, and 1–16 663-MB (formatted) SCSI disk drives.
- Software: NS 5000 system software 1.0 with SunOS 4.0.3 on the Host Processor.

The Sun 4/490S benchmark configurations were:

- Hardware: Site1: 32-MB ECC memory, the CPU's Ethernet port, 1 Sun ISP-80 IPI-2 disk controller, and 4 Sun 706A 1-GB (formatted) IPI-2 disk drives (4th root drive never used for testing). Site 2: 64-MB ECC memory, the CPU's Ethernet port, 1 452A VME Ethernet controller, 1 Sun ISP-80 IPI-2 disk controller, and 4 CDC 9720 1-GB (formatted) IPI-2 disk drives.
- Software: Sites 1 & 2: SunOS 4.0.3 with 8 NFSDs and 4 BIODs.

The Sun 4/280S-32 benchmark configuration was:

- Hardware: 32-MB ECC memory, the CPU's Ethernet port, 1 450A VME Ethernet controller, 1 Xylogics 451 SMD disk controller, and 1 Hitachi DK815 892-MB (formatted) SMD disk drive.
- Software: SunOS 4.0.3.

4 REFERENCES

[Auspex89a] Auspex Systems Inc.
An Overview of Functional Multiprocessing for Network Servers.
Technical Report 1, Fourth Edition, Auspex Systems Inc., February 1991.

[Auspex89b] Auspex Systems Inc.
Benchmark Methodology and Preliminary Performance Specifications for the Auspex NS 5000 *Network Server.*
Technical Report 2, Auspex Systems Inc., October 1989.

[Hitz90] David Hitz, Guy Harris, James K Lau, and Allan M Schwartz.
Using Unix as One Component of a Lightweight Distributed Kernel for Multiprocessor File Servers.
Proceedings of the Winter 1990 USENIX Conference, Washington, DC, 23–26 January 1990.
Also Technical Report 5, Auspex Systems Inc., January 1990.

[Horton90] William A. Horton, Bruce Nelson, and John Yee.
An NFS Benchmark Comparison of the Auspex NS 5000 Network Server and the Sun 4/280 File Server.
Performance Report 1, Auspex Systems Inc., January 1990.

[Sandberg86] Russel Sandberg, David Goldberg, Steve Kleiman, Dan Walsh, and Bob Lyon.
Design and Implementation of the Sun Network File System.
Proceedings of the Summer 1985 USENIX Conference, pp. 119–30, Portland, OR, June 1985.

[Schwartz90] Allan M. Schwartz, David Hitz, and William M. Pitts.
LFS—A Local File System for Multiprocessor NFS Network Servers.
Proceedings of the IEEE Systems Design & Networks Conference '90, Santa Clara, California, 8–10 May 1990.
Also Technical Report 4, Auspex Systems Inc., December 1989.

Virtual Partitions: an Effective Disk Storage Administration Tool

Yu-Ping Cheng
Guy Harris
Dave Hitz
William M. Pitts

Technical Report 8
November 1990

AUSPEX

ABSTRACT

As the performance of file servers has been extended by an order of magnitude and the number of users supported by a single file server often exceeds 100 users, the amount of physical disk storage attached to the file server has increased ten fold. Managing the data resident on this enormous amount of storage presents system administrators with a significant challenge. Problems that were manageable in the range of 1 to 5 gigabytes become increasingly unmanageable as the storage capacity climbs to 50 gigabytes.

This report describes the problems associated with managing a very high capacity file server and then presents Auspex's Virtual Partition Manager as a new tool that can greatly assist the system administrator in his task. Virtual partitions enable the system administrator to:

- Create disk partitions larger than the maximum size of a single disk.
- Distribute heavily used filesystems over multiple disk drives to reduce disk access times and improve performance.
- Define mirrored partitions, to protect filesystems from disk or media failure and improve data availability.

Revision 0.01, 901022

Auspex Systems, Inc.
2952 Bunker Hill Lane
Santa Clara, California 95054 USA
Phone: 408/492-0900 · Fax: 408/492-0909

TABLE OF CONTENTS

1 INTRODUCTION

The first phase of client-server network computing, distinguished by the use of reconfigured workstations to provide the various server functions, is now transitioning into the second phase where specialized machines, designed expressly for their tasks, are beginning to assume the server functions. Some examples of these specialized servers are Imagen's XXXX and Sun's YYYY print servers, the EPOCH 1 archival file server, the NETstor backup server from Zetaco, and the NS 3000 and NS 5000 high performance NFS file servers from Auspex Systems, Inc.

These specialized servers often deliver an order of magnitude improvement in one or more of the dimensions by which price/performance is measured. In the case of the NS 5000, both the throughput and data connectivity dimensions have been extended ten fold. However, improvements of this scale tend to magnify any flaws or deficiencies that may have lain dormant in the earlier, smaller scale server implementations. This has been the case as the NS 5000 has extended its on-line disk storage capacity to over 50 gigabytes.

Problems that were manageable in the range of 1 to 5 gigabytes become increasingly unmanageable as the storage capacity climbs to 50 gigabytes. The following list identifies deficiencies that scale into problems as the amount of on-line disk storage becomes very large:

- The system administrator has to manage an increasingly large number of filesystems. Using 1 GB disk drives, a minimum of 50 partitions are required to accommodate 50 GB of disk space.
- Filesystem activity should be monitored to detect hot filesystems and hot disk drives. One or more of the filesystems on a hot disk can be moved to other drives so that client accesses are more evenly distributed across all the drives. However, for every filesystem that is moved, the */etc/exports* file on the server and the */etc/fstab* on *every* client must be updated. For large servers with 50 to 100 clients, moving a filesystem is a tedious, time consuming task.
- Backing up a single 1 GB disk drive onto an 8mm tape takes over an hour. For conventional servers with 2 to 4 gigabytes of storage, this is acceptable . All filesystems can be backed up during the midnight hours. But when capacities are far in excess of what can be dumped in the off hours, backup becomes a more formidable task.

For all of the reasons just listed, the task of administering a 50 GB file server requires more time and effort than a conventional server. Please note, however, that this "extra effort" is much less than what would be required to manage nine additional file servers, assuming that a comparison were being made between one 50 GB server or ten 5 GB servers. Even so, large capacity file servers require additional capabilities to assist the system administrator in his task of managing enormous amounts of data.

Recognizing the system administrator's requirements for better server management tools, Auspex has augmented its file servers with a virtual partition capability. This report, presenting Auspex's new Virtual Partition Manager, focuses on the definition, implementation, benefits, and use of this new capability.

2 DEFINITIONS

A *physical disk partition* is defined as a contiguous region of physical disk space. Each physical partition can be used to provide the storage for a single filesystem or it can be a *member* of a *virtual partition*. Virtual partitions provide a level of indirection, enabling the file system to address logical filesystems. By employing different methods of mapping logical filesystems onto physical device space, the Virtual Partition Manager delivers several benefits and features which will be discussed in this paper. The various mapping methods are referred to as *concatenation*, *striping*, and *mirroring*. The following sections describe each of these virtual partition types.

2.1 Concatenated Virtual Partitions

A *concatenated partition* is a concatenation of one to eight physical disk partitions from one or more disk drives. Figure 1 below illustrates the mapping of vp1 (virtual partition 1) onto the physical partitions of ad7, ad15, and ad34, allowing the construction of a virtual partition that is larger than any single disk drive.

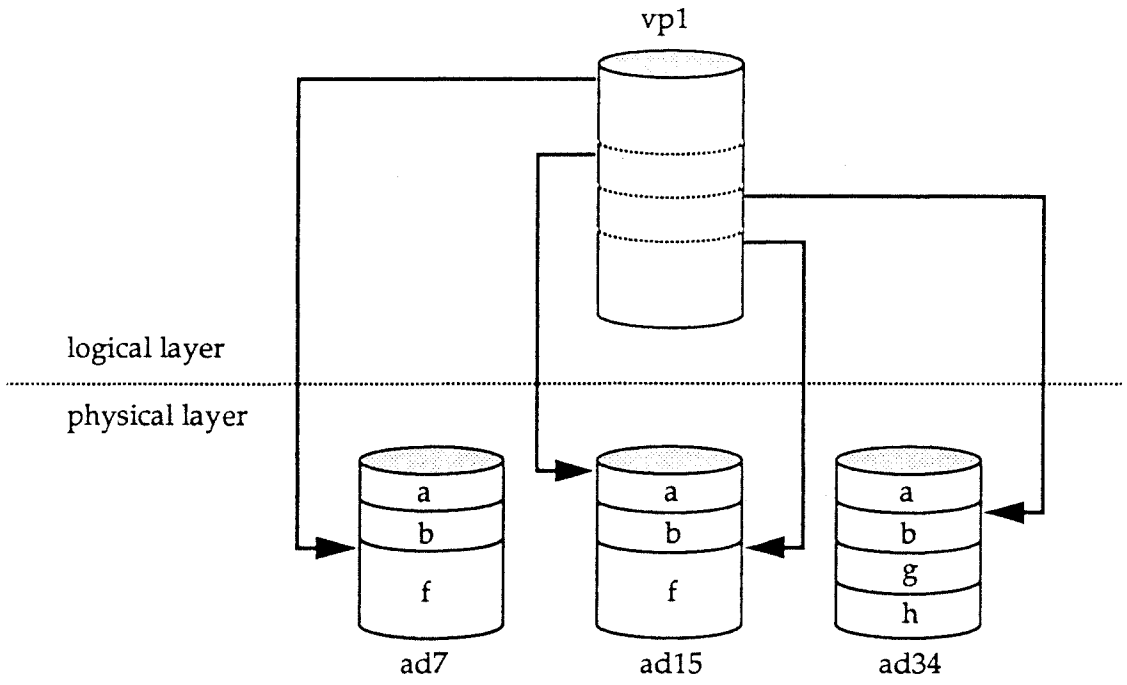


Figure 1: Logical to Physical Mapping of a Concatenated Partition.

2.2 Striped Virtual Partitions

A *striped partition* consists of two to eight physical disk partitions. The striped partition maps its logical address space across multiple physical partitions in fixed-length segments referred to as stripes. Stripe length must be a multiple of 8 KB, ranging from 1 8 KB block all the way up to the whole partition. Striping interleaves access requests to "hot" file systems over multiple drives, thus reducing access latency and improving performance. Figure 2 depicts the mapping of striped partition vp3 onto physical disk partitions. Note that all physical partitions are of the same size, as one would expect since all stripes have the same length.

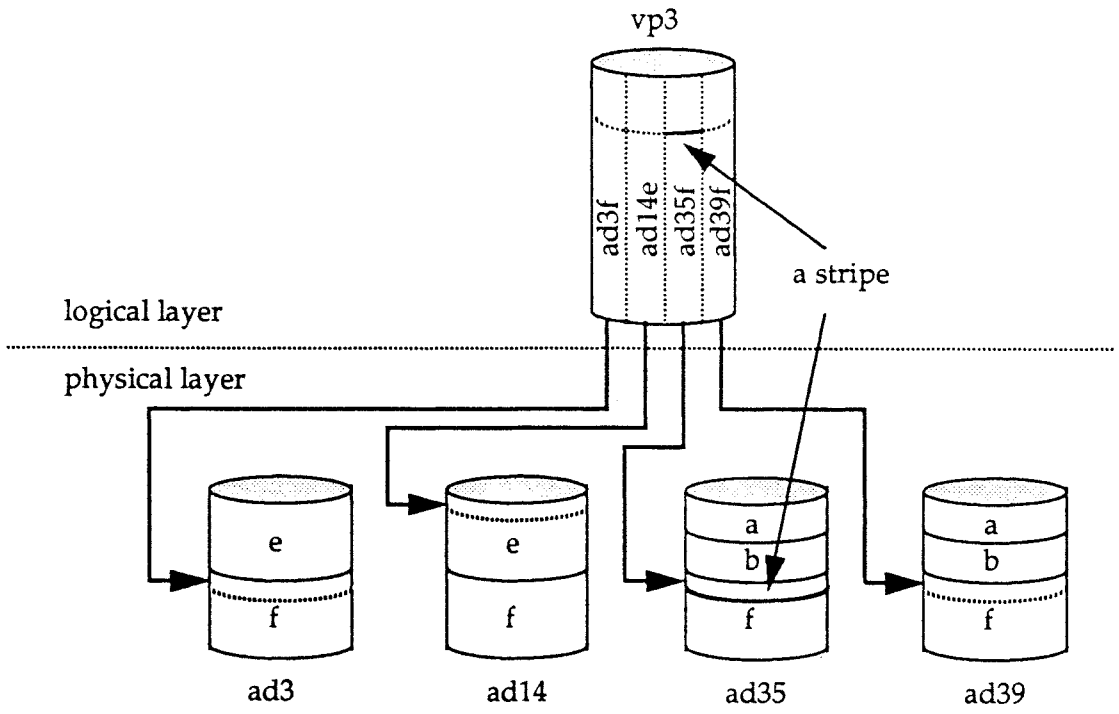


Figure 2: Logical to Physical Mapping of a Striped Partition.

Striping effectively increases the bandwidth to and from the physical disk storage by a factor of *n*, where *n* is the number of member partitions, by distributing filesystem data across all member partitions. This places the data under multiple disk actuators, enabling concurrent accesses into the same filesystem. In an NFS environment, where an 8K read or write operation requires only a single disk drive, the response latencies to client requests are significantly reduced since there are *n* disk queues, instead of one, for servicing the requests. For protocols that allow for very large data transfers in a single request, striping allows *n* data streams to flow between main memory and the disk drives, resulting in very high transfer bandwidths.

2.3 Mirrored Virtual Partitions

A *mirrored partition* usually consists of two members, both of which must be virtual partitions. The primary application of mirrored partitions is to implement high availability filesystems by maintaining duplicate copies of a filesystem on each member of the mirrored partition. In the event of a transient or permanent failure of a member partition's disk drive or its data path, filesystem requests can still be transparently serviced from the other member partition.

A secondary use of mirrored partitions is to enable on-line backups (level 0 dumps) of filesystems to be performed. The procedure for performing on-line backups requires that filesystems reside on single member mirrored partitions. This allows a second member to be attached, synchronized with the first, and then detached and dumped. This procedure is presented in more detail later.

Figure 3 below depicts mirrored partition vp5, comprised of virtual partitions vp1 and vp3. Note that both vp1 and vp3 have the same size, but vp1 is the concatenation of four partitions on three disk drives and vp3 is striped onto four partitions on four disk drives. Obviously, when mirroring is employed to provide high data availability, it is important that none of the underlying physical partitions of vp1 resides on any disk drive that also has any physical partitions belonging to vp3.

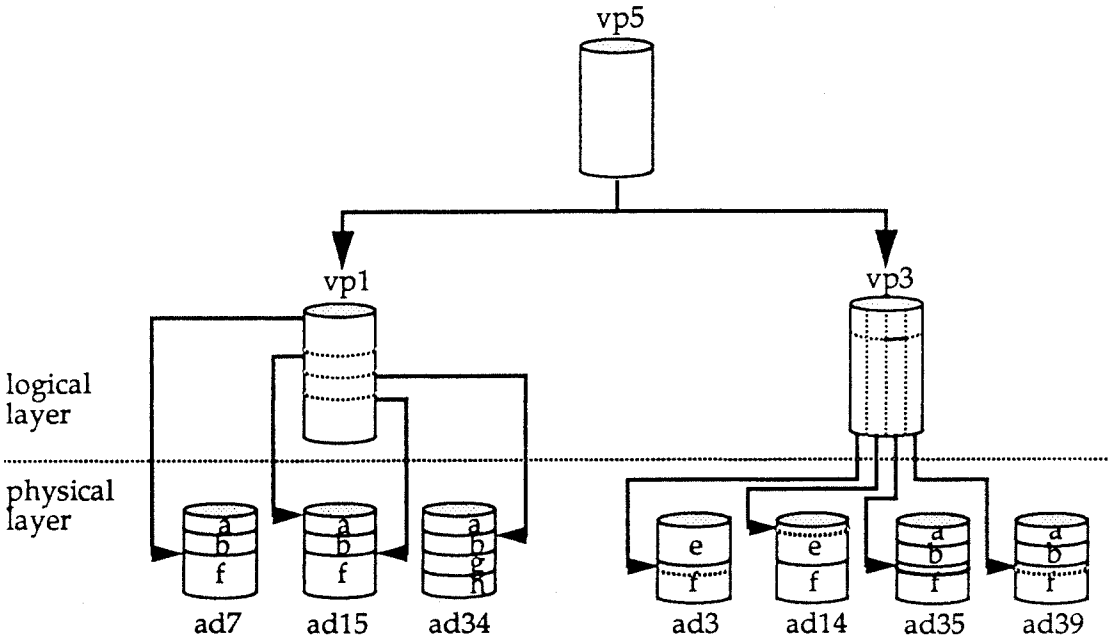


Figure 3: Logical to Physical Mapping of a Mirrored Partition.

Figure 3 illustrates the versatility with which mirrored partitions can be constructed. In practice, both of the member virtual partitions should usually be the same type.

3 OPERATION AND IMPLEMENTATION

Virtual partitions are implemented by the *virtual partition driver,* executing within the storage processor. With both the "top" and "bottom" interfaces to the vp driver being device driver interfaces, the task of the vp driver is to perform the mapping from virtual to physical partitions and, in the case of mirrored partitions, to redirect a request to the surviving mirrored partition after one partition has failed to satisfy a request due to either media or drive failure.

The vp driver is the main component of the Virtual Partition Manager (VPM). There are, however, some additional components. The complete list of VPM components is:

- **vp(4)**, the virtual partition driver that receives read/write requests to virtual partitions, performs the virtual to physical partition mapping, and generates calls to the physical device driver to actually transfer the requested data.
- */etc/vpartab*, a file created and edited by the system administrator that defines the virtual to physical mapping.
- Six Unix application layer processes that are invoked by the system administrator, the *rc.auspex* script, or by other VPM application processes. These processes monitor and control the vp driver.

Figure 4 presents the software architectural overview of an Auspex file server that depicts the various components of the Virtual Partition Manager (shaded boxes) and their relationship to the overall system.

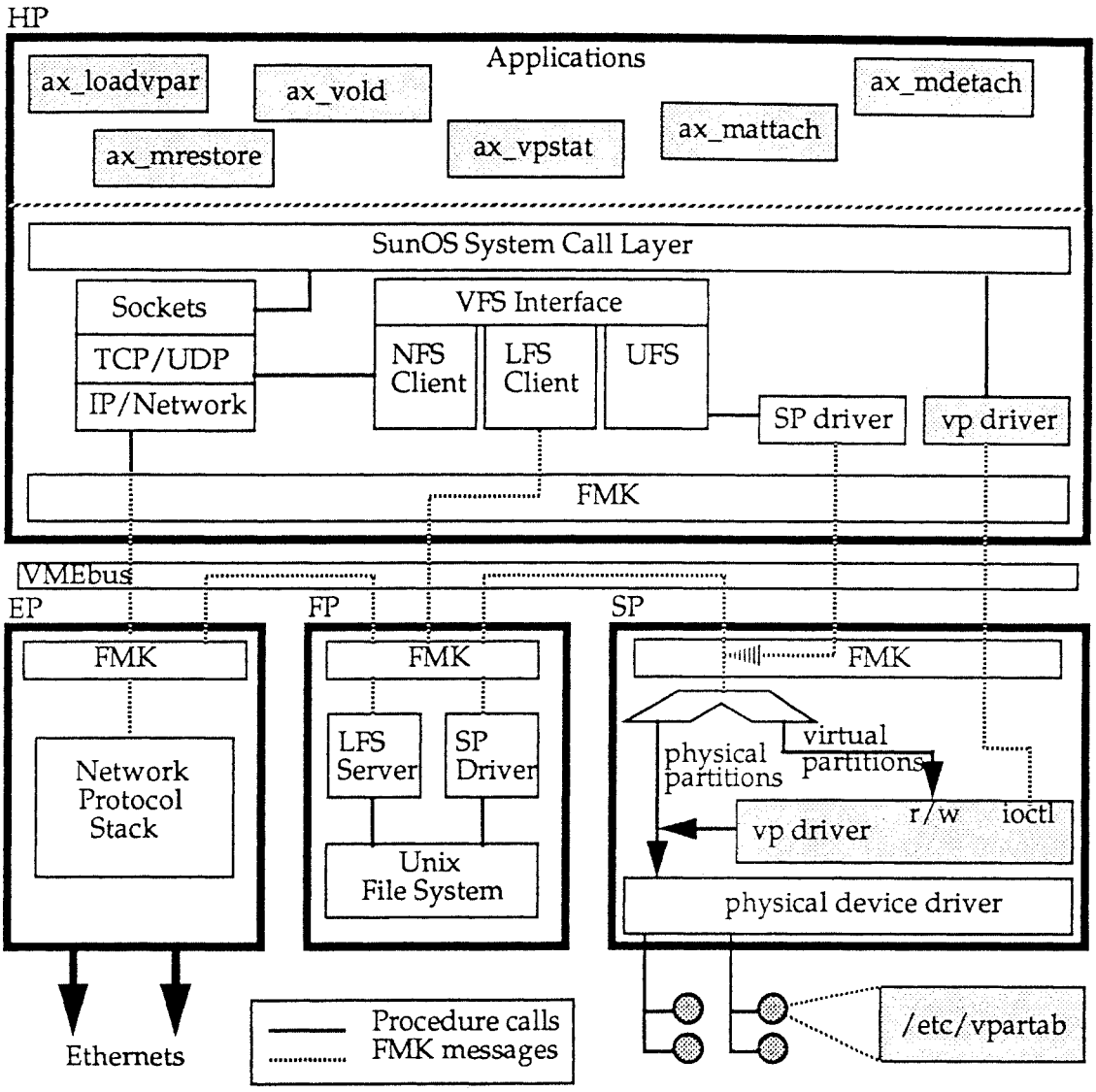


Figure 4: The components of the Virtual Partition Manager (shaded), shown in relationship to the overall system architecture of an NS 5000 NetServer. Notice that the **vp driver** is primarily implemented within the Storage Processor, but the driver's ioctl interface code resides in the Host Processor. The two vp drivers depicted above actually constitute a single driver. The */etc/vpartab* file is created by the system administrator to define the virtual partitions. **ax_loadvpar** reads this file, builds the partition mapping table, and downloads it into the Storage Processor by calling the HP resident component of the **vp driver**. Once the table has been downloaded, virtual partition read/write requests can be submitted by the SP drivers. These requests are identical to "normal" read/write requests except for the value of the *virtual partition flag* parameter in the request message. **ax_mattach**, **ax_mdetach**, and **ax_mrestore** enable attaching, detaching, and restoration of mirrored virtual partitions. **ax_vold** maintains a table reflecting the current state of all members of mirrored partitions, and **ax_vpstat** reports on the status of virtual partitions.

NOTE: Figure 4 depicts the NS 5000 NetServer architecture from a control flow perspective. The primary memory board, which is used for data caching, is not shown here.

Figure 4 is a fairly busy picture. Its intent is to illustrate how the various components of the Virtual Partition Manager are distributed throughout the system. Please refer to Auspex Technical Reports Number 1 [Auspex89], Number 4 [Schwartz89], and Number 5 [Hitz90] for a detailed understanding of the complete system.

The following sections will describe in brief detail the Virtual Partition Manager, the Unix support processes, and the */etc/vpartab* file.

3.1 Virtual Partition Driver

As shown in Figure 4 above, the virtual partition driver is layered above the physical device driver in the Storage Processor. Once the file system code has determined that it requires a block of data, it calls the bio (block I/O) routine *getblk* to fetch the desired block. *Getblk* first searches the buffer cache to determine if the block is already cached. If the block is not cached, *getblk* calls the SP driver, which builds an SP read request message and dispatches it to the Storage Processor. The message parameters that identify the data to be transferred are: *virtual partition flag, partition identifier, offset* into the partition of the first byte being requested, and the *transfer byte count.*

The Storage Processor receives the incoming SP read request and performs the following actions:

- Checks the virtual partition flag. If it's not set, the *virtual partition driver* is bypassed and control is passed directly to the *physical device driver,* which performs the requested operation and then issues a response back to the caller. This completes the processing of the SP read request.
- Otherwise, control passes into the virtual partition driver. The vp driver uses the partition identifier to index into an internal mapping table to retrieve the necessary information to map the request from virtual space into physical disk space.
- Having performed this translation, vp then calls the physical device driver to perform the requested data transfer.
- Control returns to vp after the physical device driver has completed the transfer. If there were any errors associated with the transfer, the error indication will be propagated back to the requesting client unless the virtual partition was a mirrored partition with both mirrors in the ACTIVE state. In this case, the client gets one more shot. The vp driver will attempt to read the requested data from the other mirror. In all probability, this attempt will be successful. The client will receive good data and good status, and the original error will be reported on the system log.

The response to the SP read request is assembled and dispatched. This completes the processing of the SP read request

3.2 Unix Support Processes

The six support commands, and the functions they provide, are:

- **ax_loadvpar(8)** reads */etc/vpartab* and checks the validity of each entry. If no errors are detected, the mapping table is downloaded into the Storage Processor. Any detected error will inhibit downloading the mapping table.

- **ax_vpstat(8)** reports information on all virtual partitions. This information includes:
  - o virtual partition type and components,
  - o stripe size (for striped partitions),
  - o status (for members of a mirrored partition):

    ACTIVE — This partition is identical to the other member of the mirrored partition (if there is more than one member), and no unrecovered write error has occurred on the partition. The mirrored partition is currently mounted *read/write* or *open for writing*.

    SYNCED — This status is similar to ACTIVE, but the partition is not currently mounted *read/write* or *open for writing*.

    DIRTY — This partition is not identical to the other member of the mirrored partition because there was a failure while this partition was ACTIVE or RESTORING. Writes to the mirrored partition are not being performed on this member partition.

    DAMAGED — An unrecoverable read or write error has occurred on this partition. This partition is not identical to the other member of the mirrored partition, since no writes have been performed on this partition since the error was detected.

    RESTORING — This partition is not identical to the other member of the mirrored partition, but writes to the mirrored partition are being performed on this partition. This state indicates that ax_mrestore is in the process of restoring this partition. At some point in the near future, when ax_mrestore completes, the state of this partition will change to ACTIVE.

- **ax_mrestore(8)** updates a DIRTY or DAMAGED member of a mirrored partition by copying the good member (ACTIVE or SYNCED) onto the bad member. When the copy is initiated, the state of the bad partition is changed to RESTORING. While in this state, any writes performed on the ACTIVE partition will also be performed on the RESTORING partition. After the copy completes, the RESTORING partition assumes the same state as the good partition (either ACTIVE or SYNCED). This command is automatically executed when the system boots, when ax_loadvpar executes if a new member has just been added to an existing mirrored partition, or by ax_mattach when it attaches a new member to an existing

mirrored partition. It can also be executed by the system administrator at any time.

- **ax_vold(8)** a daemon that maintains a table reflecting the current state of all members of mirrored partitions. The Storage Processor, the VPM, LFS, and ax_mrestore all direct messages to ax_vold to keep him informed of the current status.

- **ax_mattach(8)** attaches a concatenated or striped virtual partition to a mirrored virtual partition. Both partitions must have the same size. If the attach operation is successful, ax_mrestore will be invoked to bring the new partition up to ACTIVE status.
  NOTE: ax_mattach does not update the file */etc/vpartab*. It is intended to be used, for example, to temporarily attach a new member to an existing one-member mirrored partition, so that when the ax_mrestore completes it can be detached again with ax_mdetach, and a backup made of the now-static member, which is a "snapshot" of the mirrored partition.

- **ax_mdetach(8)** allows the system administrator to detach one of the member partitions of a mirrored partition. This is usually done to perform an on-line level 0 dump. The procedure for this is detailed later.

3.3 The /etc/vpartab File

The mapping performed by the Virtual Partition Manager is controlled by the vpartab table. This table is a text file (*/etc/vpartab*) that is edited by the system administrator to add or modify virtual partition definitions. An example vpartab file, specifying concatenated, striped, and mirrored partitions, is given in Figure 5.

```
# Virtual partitions 1 and 2 - Concatenated
/dev/vp1     concat              ad7f,ad15a,ad34b,ad15f  # Figure 1
/dev/vp2     concat              ad3d,ad4b,ad4g,ad5b
#
# Virtual partition 3 - Striped
/dev/vp3     striped,size=144k   ad3f,ad14e,ad35f,ad39f  # Figure 2
#
# Virtual partition 4 - Concatenated
# This is necessary because physical partitions cannot be
# mirrored, and virtual partition 6 is constructed by mirroring
# physical partition ad8c and virtual partition 2 defined above.
/dev/vp4     concat              ad8c
#
# Virtual partitions 5 and 6 - Mirrored
/dev/vp5     mirrored            vp1,vp3                 # Figure 3
/dev/vp6     mirrored            vp2,vp4
```

Figure 5: Example /etc/vpartab file.

Figure 5 above clearly illustrates that the specification of virtual partitions is simple and straightforward.

4 BENEFITS OF VIRTUAL PARTITIONS

By efficiently employing the capabilities of the Virtual Partition Manager, the system administrator can configure Auspex file servers to provide:

- Higher data availability,
- Better system performance, and
- Simplified system administration.

The following sections describe how the Virtual Partition Manager delivers these benefits.

4.1 Higher Data Availability

Mirrored partitions are the principal VPM feature that enables higher data availability via two different methods. First, higher data availability of critical data is enabled by mirroring the filesystem(s) that contains the data onto two member partitions of a virtual mirrored partition. In the event of media or disk drive failure of one member, the data will still be accessible on the other member.

Second, data availability can be enhanced by using mirrored partitions to allow full dumps of filesystems to be performed while the filesystem remains mounted and accessible to all users. Section 5 describes how this can be accomplished.

4.2 Better System Performance

System performance can be improved by monitoring filesystem activity, identifying hot spots, and then moving or redistributing filesystems to alleviate the disk contention. When the contention is the result of moderate activity on several filesystems sharing the same disk, the condition can be remedied by moving one or more filesystems onto other disk drives. When the contention is the result of heavy traffic into a single filesystem, the filesystem can be moved onto a striped virtual partition. This will have the effect of distributing the filesystem access requests over multiple disk drives. In both cases, mirroring can be employed to enable the on-line relocation or redistribution of filesystems. Section 5 will elaborate.

Moving or redistributing filesystems can be very effective in improving system performance. But only if the system administrator knows what filesystems to move and where to put them. Auspex has developed a tool to provide this information. It is called **ax_perfmon(8)** (performance monitor) and is described in Auspex Technical Report Number 7 [Hitz90].

4.3 Simplified System Administration

The Virtual Partition Manager can simplify the system administrator's task by:

- Providing virtual partitions up to 2 GB in size. This allows for the construction and use of 2 GB filesystems, which significantly reduces the number of filesystems with which the system administrator must contend. The 2 GB limit is imposed by the Unix lseek(2) system call, since the file displacement parameter is represented as a 32 bit signed number.
- Allowing full dumps (level 0) of filesystems to be performed while the filesystem is still mounted and accessible to all users. This obviates the need for restricting backup operations to the night time hours.
- Allowing filesystems to be relocated while remaining on-line. Since the mapping from logical to physical disk space is performed by the virtual device driver within the Storage Processor, filesystems that reside on virtual partitions have the attribute of *location transparency* with respect to their clients. This means that the filesystem can be relocated without needing to inform the clients (or alter their */etc/fstab*). Couple this attribute with the VPM's mirroring capability and what do you get? Hot moveable filesystems! The procedure for moving hot filesystems is described in Section 5.2.

5 USING VIRTUAL PARTITIONS

Using striped or concatenated partitions is very straightforward. A simple edit of */etc/vpartab* defines the virtual partition, and then running **ax_loadvpar** creates the partition. Once created, it is used in the same manner as an ordinary partition. Virtual partitions are referenced by either **vp*n*** or **rvp*n***, where *n* is the partition number (range: 0 to 63).

Mirrored partitions are a little more complicated, but that's because there are several ways to exploit the capabilities of partition mirroring. First of all, mirroring can be further classified into either *static* or *dynamic* mirroring. A static mirrored partition has two members and is fully defined in */etc/vpartab*. Static mirrors are used to provide highly reliable, highly available filesystems. Dynamic mirrors, on the other hand, are highly promiscuous. Coupling, reproducing, and then splitting, dynamic partitions provide the framework that enables on-line level 0 dumps and on-line filesystem relocation. These two on-line functions will be discussed in the next sections, followed by some suggestions on the assignment of physical space to virtual partitions.

5.1 On-line Dumps

The mirroring capability can be used to backup a mirrored virtual partition onto another virtual partition without interrupting user access to the filesystem. When both partitions reach the ACTIVE state (the virtual partition backup has completed), one of the member partitions can be detached. At this point, a full dump can be performed on the detached member partition.

The following text presents this procedure in more detail. The first four steps set up the initial virtual partitions. The remaining steps enable the filesystem to be backed up while it is still accessible to all users.

The on-line, full dump procedure is:

- Edit */etc/vpartab* to create:
  - **vp1** as a concatenated virtual partition consisting of **ad3c**,
  - **vp2** as a concatenated virtual partition consisting of **ad4c**,
  - **vp3** as a mirrored virtual partition whose only member is **vp1**.
- Run **ax_loadvpar** to load the revised virtual partition table.
- Run **newfs** to initialize **vp3**.
- Mount and export the filesystem on **vp3**.
- At some later date, when a backup of the filesystem on **vp3** is desired, run **ax_mattach** to attach **vp2** to **vp3**. **ax_mattach** will automatically invoke ax_mrestore to copy the contents of **vp1** to **vp2**. When **ax_mrestore** completes, both members of the mirrored partition are flagged as ACTIVE.
- Run **sync**, which flushes all memory resident data back to the disk drives.
- Run **ax_mdetach** to detach **vp2** from the mirrored partition **vp3**.
- Run **fsck** on **rvp2**.
- Backup **rvp2** to tape. The original partition (**vp1**) is unaffected by the backup and remains available to users.

NOTE: The **ax_mattach** and **ax_mdetach** commands do not update */etc/vpartab*, and no error

checking is performed by **ax_loadvpar**. Therefore, these commands should only be used to temporarily attach or detach one member of a mirrored partition. To define a permanent member of a mirrored partition, always edit */etc/vpartab* and then run **ax_loadvpar** to effect the change.

5.2 On-line Relocation of Filesystems

Using a procedure similar to the one just outlined, a filesystem resident on a mirrored virtual partition can be relocated while still providing continuous service to users. In detail, the procedure is:

- Edit */etc/vpartab* to create:
  - **vp1** as a concatenated virtual partition consisting of **ad3c**,
  - **vp2** as a mirrored virtual partition whose only member is **vp1**.
- Run **ax_loadvpar** to load the revised virtual partition table.
- Run **newfs** to initialize **vp2**.
- Mount and export the filesystem on **vp2**.
- At some later date, when it has been determined that there is too much contention in accessing data on **vp2**, do the following to move **vp2** to a disk (or set of disks) where there is less contention:
  - Edit */etc/vpartab* to create virtual partition **vp7** with exactly the same size as **vp1**. **vp7** may be either striped or concatenated, depending on whether the contention is localized within the filesystem or the disk drive.
  - Run **ax_loadvpar** to load the revised virtual partition table.
  - Run **ax_mattach** to attach **vp7** to **vp2**. **ax_mattach** will automatically invoke **ax_mrestore** to copy the contents of **vp1** to **vp7**. When **ax_mrestore** completes, both members of the mirrored partition are flagged as ACTIVE.
- Run **ax_mdetach** to detach **vp1**. Once again, **vp2** is a single member mirrored partition, but the underlying member is now **vp7** instead of **vp1**.

5.3 Physical Space Assignment

To effectively use the capabilities of relocating and dumping filesystems on-line, it is necessary to plan ahead so that empty virtual partitions of just the right size are available when they are needed for attaching to single member mirrored partitions. Here's a sample plan to illustrate how to organize your virtual partitioning system:

- Level 1: Format the disk drives so that all physical partitions have the same size.
- Level 2: Construct virtual partitions, all of the same size, from the physical partitions. The virtual partitions may be striped or concatenated in whatever combination you choose.
- Level 3: Construct the desired number of single member mirrored partitions. Use **newfs** and **restore** as required to build your filesystems.
- The unused virtual partitions are held in reserve for later attachment to mirrored partitions.

6 CONCLUSIONS

The Virtual Partition Manager significantly simplifies the task of managing a high capacity file server by reducing the number of filesystems that must be managed, by allowing full dumps to be performed on mounted filesystems, and by enabling the system administrator to dynamically relocate or redistribute filesystems to achieve better system performance. In addition to simplifying the administrator's task, the VPM also provides a partition mirroring capability that can be used to ensure very high availability of critical data. These benefits provided by the Virtual Partition Manager substantially assist the system administrator in his task of providing a reliable and efficient service to his organization.

7 REFERENCES

[Auspex89] Auspex Systems, Inc.
An Overview of Functional Multiprocessing for Network Servers.
Technical Report 1, Auspex Systems, Inc., October 1989.

[Schwartz89] Allan M. Schwartz, David Hitz, William M. Pitts.
LFS -- A Local File System for Multiprocessor NFS Network Servers
Technical Report 4, Auspex Systems, Inc., December 1989.

[Hitz90] David Hitz, Guy Harris, James K. Lau, Allan M. Schwartz.
Using Unix as One Component of a Lightweight Distributed Kernel for Multiprocessor File Servers
Technical Report 5, Auspex Systems, Inc., January 1990

[Hitz90] David Hitz.
Monitoring the Internal and External Performance of Network File Servers
Technical Report 7, Auspex Systems, Inc., December 1990

Report of the National Science Foundation Workshop on Advanced Data Storage Technology for Computer Systems

January 17-18, 1990

Carnegie Mellon University

Compiled and edited by
Alfred B. Bortz, Carnegie Mellon University
based on contributions from the workshop participants

Workshop organized by:

Prof. Mark H. Kryder, Director
Data Storage Systems Center
Carnegie Mellon University

Dr. Duane A. Adams, Assoc. Dean for Research
School of Computer Science
Carnegie Mellon University

and

Prof. Randy Katz
Department of Computer Science
University of California, Berkeley

Table of Contents

Preface

This report identifies several problems which should raise the concerns of anyone who believes the computer industry is economically and technically important to the U.S. At the same time it presents opportunities and recommends action to solve the problems. The National Science Foundation and other Federal Agencies, the Computer Industry and universities need to work together to see that the problems are addressed and that the opportunities are seized.

Mark H. Kryder

1 Data Storage: An Industry of Opportunity and Challenges

As 1990 begins, sales in the information industry worldwide have grown beyond one half trillion dollars annually. Although the industry is diverse, spanning personal computers through complex telecommunication networks, one element is ubiquitous across its entire spectrum: data storage.

The data storage industry is unmatched in its record of sustained technological advancement. Despite its size, it still offers enormous potential for business growth. If U.S. industry is to realize that potential, it must meet a host of technical challenges. Doing so will involve solving challenging research problems across the entire spectrum of science and engineering disciplines.

It will also require the development of mechanisms to encourage more academic researchers to participate with each other and with industry in coordinated, interdisciplinary research programs. Those mechanisms must be more than simple administrative devices; they must offer opportunities to transform imagination into reality and produce knowledge and ideas which are truly important to people's lives. Few areas of investigation can compare to data storage in those respects. The technical problems are intellectually appealing and challenging, and their solution leads quickly to new or improved products of a large, important industry. Despite this, academic and industry-university cooperative research in data storage falls far short of what industry experts would like to see.

Why is this so? Recognizing the importance of that question, the National Science Foundation (NSF) requested that Carnegie Mellon University organize a Workshop on Advanced Data Storage Technology for Computer Systems. This report of that workshop is designed to guide the NSF, as well as other government agencies and industry, in the identification of promising avenues of data storage research. Beyond that, it is intended to convey to the academic community the excitement and importance of that research.

In the simplest terms, the workshop participants attribute the lack of university involvement to the interdisciplinary nature of the field, lack of knowledge of the research area within universities, and the disciplinary cultures of most universities and Federal funding agencies. This report attempts to remedy that problem in two ways: (1) by laying out the important data storage research issues for academics; and (2) by suggesting new mechanisms to generate and fund academic or academic-industrial data storage research projects.

2 Opportunities in Data Storage

2.1 Market Opportunities

The data storage industry accounts for over 20% of revenue of the computer market. Its products include both hardware (*e.g.*, magnetic tapes, magnetic disks, optical files, solid state storage) and software (*e.g.*, data storage management, device management, performance, security, archive). And although it is apparently an established, mature business, the fraction of text and numeric information stored digitally is less than 1%. The remaining data are stored on microfilm (4%) and paper (95%). This low penetration coupled with the rapidly decreasing cost of storage has fueled a 30% compound growth rate in demand for digital storage over the past several decades, and driven a strong and continuing evolution of the technology.

During the last 10 years, the data storage industry has changed complexion. Prior to 1980, the industry was dominated by a few large, vertically integrated U.S. companies. Most devices were very large and complex, requiring both large engineering groups and large capital investments. However, the simultaneous introduction of small computers with standard interfaces and the advancement of storage technology which has allowed modest capacities to be contained within very small form factors (*i.e.*, 5.25, 3.5, and 2.25 inch devices, with cigarette-pack sized drives now in development) has lowered the entry barrier into this field. Thus the number of active companies has multiplied, reaching over 70 by the end of 1989. This has resulted in intense competition and low profit margins.

Classically, the technology (the components) from high-end devices has "trickled down" to low-end, less expensive devices. This has resulted in a significant difference in performance and cost (per bit) between the ends of the market place. Although to some degree this still continues, two other very important advances have occurred. First, engineers have realized that packaging technology (both electrical and mechanical) can offer significant opportunities to increase storage density.[1] And second, the tolerance demands on electro-mechanical systems are more easily achieved in small form factors. This combination of events has fueled the rapid growth in higher performance small devices, and the closure of the performance wedge between the low and high ends of the market.

An additional parallel trend has occurred during this timeframe: the emergence of the very large consumer video and audio market. The drive for high capacity and high bandwidth has turned the technology pyramid upside down: the low-end consumer products are the driving force behind technological advances. The capacities of consumer magnetic tape now exceed most computer digital

[1] It should be noted that the rapid advancement in packaging technology has been driven by the consumer electronics industry, and not from the data storage or computer industry.

recordings. And the move to digital audio disks has been the driving force behind improvements in reliability, cost, and performance of semiconductor lasers. Consumer products are expected to continue to drive technological advances at least as strongly in the near future, if not more so, with the introduction of high-definition television (HDTV) and fiberoptic communication lines into homes.

Finally, because the low-end market is exceptionally price-sensitive, a high volume commodity market in components has also emerged. All of this has given system integrators a ready source of low-cost building blocks from which to assemble high-capacity, highly reliable storage subsystems. This combination of events has both resulted in a rapid growth in the industry (including a shake-out because of the large price sensitivity) and an opportunity for a broader application of the technology.

2.2 Application Opportunities

Many new potential applications could drive this demand for products and technological progress even further. Some of these applications are predictable from extrapolations of present communication and computing technology: high bandwidth data transfer in parallel computing and supercomputing; portable computing in which today's workstations become tomorrow's laptop computers; digital libraries, including personal libraries in which papers now found on an indivdual's desk or in his or her file cabinet are stored electronically; integration of high-performance networking (1 gigabit/sec and higher) and storage technology to produce nationwide repositories; and large-scale storage of digitized images (10^7 bytes per page compared to 10^3 for text), digital audio, digital video (including HDTV), and perhaps even photographic images.

Clearly, this represents an enormous opportunity for industry in the U.S. and throughout the world. Those who capitalize on this opportunity will need a vision of both evolutionary and revolutionary new products to fill both broad needs and niches, for both the consumer market and the commercial market. Just as computer architectures have entered an era of major change, data storage system architectures must also change. Successful companies will be those which design data storage systems to support radically new products. Doing that will require computer architects, computer systems engineers, and data storage device researchers to work very closely together. The design will involve a multidisciplinary team including computer scientists and electrical engineers with a wide range of specialties, mechanical engineers, chemists and chemical engineers, materials scientists and engineers, physicists, and applied mathematicians.

2.3 Technological Opportunities

The figures of merit for data storage technology have maintained steady exponential improvement over the past 30 years, an amazingly long growth period. For example, the areal storage density of magnetic disks has had an uninterrupted compound growth rate of 22%. The area of the magnetic bit cell has been reduced by a factor of 10,000 during this period. This has been fueled by major advances in all elements of the technology, from components (media and recording heads) through signal processing (data compression, error correction). Similar advancements in tape technology have allowed over one gigabyte of data to be stored in an 8 mm cartridge today, compared to 20 megabytes on a ten and one-half inch reel only 25 years ago.

This rapid rate of technological advance shows no sign of abating. The fundamental limit for information density in magnetic recording technology, for example, is at least four orders of magnitude beyond what has been achieved in commercial devices, and laboratory demonstration devices indicate that there are no overwhelming obstacles to continued improvement at high rates for the remainder of the twentieth century. In magneto-optic recording, the current major goals are improved access time and volumetric data storage density, and current research indicates that engineering paths to these goals are realizable. On the horizon, alternative data storage technologies offer new capabilities for future systems, including holographic and three-dimensional storage schemes.

3 Challenges

A major purpose of the NSF Workshop on Advanced Data Storage Technology for Computer Systems was to identify areas in which academic researchers are needed to make significant contributions to both the knowledge bases of their fields and this critical industry. However, achieving success in the worldwide data storage market will require more than technological prowess. It will also require changes in the culture of the research community. Academic researchers in diverse fields must be more attuned to communicating across disciplinary boundaries with colleagues in other departments and with their industrial and government counterparts. They must also be sensitive to the needs of users and to issues of international competitiveness.

3.1 Increasing Academic-Industrial Interaction

Over the last few years, research results in areas such as magnetic disks (*e.g.*, thin-film magnetic storage media, magneto-resistive heads), magnetic tapes (*e.g.*, integrated thin film heads, rotary heads), and optical recording has provided the potential to actually increase the rate of technological advancement in data storage above the impressive growth of the previous few decades. However, strong competition from abroad threatens the U.S. position in this critical industry.

If the U.S. is to maintain and extend its current lead in digital data storage technology, it must move rapidly to capitalize on research breakthroughs, translating them into new and improved products, taking advantage of the vast data storage opportunity described above. But data storage industry experts agree that the lack of a major multi-university effort in this area, and the failure of U.S. industry to couple well with the few strong university programs that do exist, are serious obstacles to rapid innovation in data storage technology. Without product innovation, U.S. companies must compete in the international commodity market on the basis of reduced manufacturing costs alone -- something they have historically not done well.

Needed is a broad-based industrially-coupled multi-university effort capable of addressing the very complex systems issues in data storage technologies. The most significant breakthroughs and new knowledge can be expected when several components of the system and their interaction are studied and improved together. Industrial researchers understand this very well, but individual investigators of universities frequently do not. They need close coupling to industry to gain sufficient knowledge of the system to contribute effectively.

Some people question this point of view, noting that the data storage industry has made impressive advances and produced innovative products despite a level of involvement of universities that is quite small as compared to other "high-tech" industries, such as semiconductors. Their argument ignores the fact that the first two decades of advances and innovations were largely incremental and empirical. In contrast, in recent years, progress in data storage has required technological breakthroughs and fundamental new knowledge.

For example, in today's advanced data storage systems, the media are thin films of both magnetic and wear-resistant materials developed through systematic investigation of deposition techniques, magnetic and mechanical properties, and recording performance with a variety of head designs; the heads are made by extremely complex microfabrication techniques from thin-film and bulk materials; and advanced coding and signal-processing techniques have advanced the information density achievable from these heads and media. The data storage research problems of the present and on the horizon are, as described in section 4, wide-ranging and challenging. The development of new university-industry cooperative programs in data storage over the past seven years is significant, and their contributions to the technology have been important. But substantial further growth of academic research and much more industry-university interaction in this area is clearly needed.

3.1.1 Overcoming Academic Compartmentalization

A major obstacle to continuing that rapid growth in industrial-academic interaction is the presence of barriers to close interdepartmental work in many university settings. Those barriers make it difficult to establish a long-term, focussed, coherent set of projects involving professors and graduate students in different fields. The directorates of the NSF mirror this compartmentalization of academic departments. In contrast to this is the technically very diverse nature of the storage field and the requirement for simultaneous advancements in all aspects of the technology in order to advance the state of the art. This is illustrated in Table I, which lists research areas in data storage by discipline. Clearly, investigators in this field must have regular contact with persons familiar with systems issues. But such contact is discouraged by compartmentalization; there is a serious mismatch between the disciplinary cultures of the academic world (including many of the agencies that fund academic research) and this interdisciplinary industry.

Table I:
Data Storage Research Issues, Organized by Discipline

Polymer chemistry: frictional polymers, stable films

Mechanical design: small durable bearings

Servomechanisms: disk, tape servos

Mathematics/information theory: modulation/EDAC codes

Communication/Signal Processing: detection, intertrack interference, noise modelling

Analog circuit design: signal pre-amp and signal processing

Digital circuit design: microcode/controllers

Materials science: surface states, microstructure, B_s, H_c, $\Delta R/R$, $\Delta R/T$, amorphous/crystalline transitions

Physical chemistry: corrosion and rheology of coatings

Physics: Many body strong interaction field problems, B_r

Computer Software: operating systems, file systems, networks

Computer Architctures: performance modelling/measurement, CPU/memory/controller organization and interconnect

Computer Hardware: circuit/firmware design and implementation

This cultural mismatch has led to low visibility of data storage among academics. Many academic researchers with relevant expertise do not realize the potential applicability of their work to this vitally important field, especially if it could be coupled with the work of industrial researchers or other academics on their own or other campuses.

The NSF data storage workshop was, in part, intended to address this problem. The participants hope that distribution of this report to academic investigators in engineering, computer systems, control theory, materials, and tribology, among other fields, will increase those researchers' awareness of problems that have both fundamental interest and practical importance to data storage technology. One intended outcome of that distribution is for the recipients to seek and develop contacts with other researchers in academe, industry, and government laboratories and to initiate joint research programs of importance to the data storage industry. Another is that the NSF and other agencies find effective means to facilitate the evaluation and funding of data storage related proposals from academia.

3.2 Understanding User Applications

The design of a data storage system should, ideally, take into account patterns of data transfer, which depend on the nature of the users and their applications. For example, the design of current disk drive systems could be optimized if the designer knew how frequently the disk is accessed for reading, how frequently for writing, the distribution of the numbers of bytes transferred on each read or write, the likelihood of successive operations taking place on nearby areas of the disk, and so on. The usage patterns vary widely according to applications which can range from transaction processing, such as airline reservation systems, to Unix operations, to supercomputing. In addition, disk usage depends on the operating system, the file system, the disk controller, and operational constraints such as reliability, size of files, backup, *etc.*

Although such information would be of great benefit to the developers of disk controller software, disk drive control electronics, and disk drive hardware, it is rarely available. Acquiring such data would require monitoring computer systems as they work on sensitive, classified, or proprietary information. Users are reluctant to cooperate with researchers because they fear that sensitive information may be compromised, that the data gathering might interfere with operations, or that the publication of statistical analyses of their use patterns might reveal information about their operation. Furthermore, at present, the researchers cannot cite any clear advantage for the user that would result from the study.

When they approach users, researchers can, at best, cite possible intermediate or long-term advantages from such work. They can explain that existing software could be tuned to operate far more efficiently with current data storage products, that disk controller design could be improved substantially, and that tradeoffs, such as deciding whether to concentrate on improving head design or increasing

rotation rate, could be made at the disk component level. They can point out that design and development of future storage architectures such as disk arrays, intelligent controllers, and storage systems geared to large network operation, would benefit enormously from the existence of a knowledge base of disk-usage patterns. In addition, with the possibility of a proliferation of new data storage products for particular niche markets, use-pattern data will be even more valuable; the development of alternative data storage technologies may well be influenced by their suitability for a certain class of users or applications.

But to persons concerned about the integrity of their data or operations, such arguments are not likely to be convincing. Thus the workshop participants propose a strategy to gather the data needed for long-term progress by gradually overcoming the concerns of more and more users. Beginning with a cooperative effort of academic and other researchers, data storage system vendors, computer vendors, and selected applications developers, the necessary data can be gathered and analyzed statistically for a few cases. Based on that analysis, a well-tuned combination of application, operating system, and data storage system could be designed and developed. Presumably the combination would demonstrate substantial advantages. As a result, more users would be willing to cooperate with researchers, and eventually a design methodolgy could result that would enable even the most sensitive users to develop their own well-tuned computer and data storage system.

3.3 Education and Training

Industrial researchers in information storage have cited, for nearly a decade, the dearth of university graduates at all levels with knowledge and experience in areas relevant to data storage systems, such as magnetic materials and applied magnetics. In other areas, such as tribology, information theory, and signal processing, students are being trained without ever learning that their area has important applications in this major industry.

The problem exists not just at the student level but at the faculty level. Increasing academic research and the number of trained researchers in data storage will require the development of both new faculty and new programs that attract established faculty whose research is relevant to the area. The development of a handful of academic centers of excellence during the 1980s has been important, but is far from sufficient to solve the personnel problem. Data storage industry experts report that the annual output of bachelors, masters, and doctoral graduates prepared to move into entry level engineering positions, key mid-level positions, and university faculty positions continues to fall considerably short of their needs.

3.4 Achieving International Competitiveness

Data storage is a world-wide market, and the companies that succeed will do so in the face of international competition. As entrepreneurial companies enter the data storage market, they will find their futures depend on their access to new technology and trained employees and on the success of the industry as a whole.

As noted above, the access to technology and personnel requires closer ties between universities and industry as well as increased government and industrial support for university programs which produce relevant new knowledge and trained graduates. But a larger university effort with closer ties to individual companies in the industry is not the entire answer, because companies in data storage, as in any other industry, are often interdependent as well as competitive; they succeed or fail together.

Thus a whole-industry approach is needed. Universities can provide an environment in which all segments of the industry can work together with each other and with government and academic researchers, defining and probing areas in which new knowledge can bring about technological breakthroughs.

4 Research Issues and Opportunities

The major task of the workshop participants was to identify opportunities for academic research in data storage from system architecture, through the disk controller (or its equivalent for other forms of storage), down to the engineering and materials of the storage device itself. Their conclusions, set forth in this section, are intended to provide guidance to the National Science Foundation and to academic researchers. The views expressed here are those of the workshop participants and not necessarily of the NSF.

The areas of opportunity are as multidisciplinary as the industry itself. (Again, see Table I.) Academic researchers reading this report to discover where they might contribute to this field should read both sections 4.1 and 4.2 and their subsections.

Section 4.1 details system architecture issues, and it is likely to be of interest to those doing research in computer science, computer engineering, and electrical engineering. This area of data storage systems technology is largely unaddressed in academe and offers a wealth of potential research opportunities. Concerns in this area include the effective management of many levels of storage hierarchy, which leads to issues of internal transfer rate and latency, policies for migration of files, *etc.* Work will be needed in the development of novel architectures and in advancements in the hardware, software, and firmware for control and signal processing within storage devices.

Issues in data storage device technology (section 4.2) will also be of interest to researchers in a broad range of technical fields. Electrical engineering, physics, and materials science and engineering problems abound in the design and fabrication of advanced recording heads and media for vastly increased information storage density (both areal and volumetric). Improved actuators, control systems, and signal processing are needed for reading and writing information at that density (and with acceptable data rates and access times). Producing them will require researchers to address problems in both mechanical and electrical engineering. But the most demanding hardware issue, and the most critical according to industry experts, is the head-disk interface in magnetic recording, where tribological issues of friction, stiction, and wear and the aerodynamics of 50-nanometer flying heights will challenge chemists and chemical engineers, mechanical engineers, materials scientists, and physicists.

4.1 Data Storage System Architecture

High performance storage systems can only be achieved through a coordinated hardware and software attack on the problem. From this perspective, the data storage system starts at the application interface presented to users, and includes the operating system's file and networking services, storage controllers and their associated firmware, and even embedded device controllers. Thus, we define data storage system architecture as the entire collection of hardware and software systems that exist between the point at which an application makes a request to read or write bytes and the actual reading or writing of those bytes by the physical storage devices themselves.

Of course, future data storage systems must provide very high levels of performance, be it high I/O rate or high data density, but equally important attributes include high capacity in a small footprint and high data availability. Beyond that, future storage architectures must allow the system to scale gracefully to higher levels of performance and capacity over time. In general, we divide the application space into three broad areas:

1. high data rate I/O, such as numerically intensive computations and image processing applications;

2. high access rate I/O, such as engineering/software development environments, information retrieval, and many commercial data processing environments;

3. database processing.

Database processing is a hybrid of high data-rate and high access-rate input/output. While on-line transaction processing can be characterized by high I/O rate demand, a good deal of generic database processing involves sequential access, such as sorting and report generation. In addition, a database system has special needs with respect to its interaction with the storage system. A database system has much knowledge about the sequence in which it will access data, and thus it must be able to set policies for how to manage data which has been cached from the storage system.

The components of the data storage system architecture include:

1. an operating system, providing the file abstraction and implementing the protocols that allow access to network and storage controllers;

2. the interconnect, including such protocols, through which control and data passes between the components that request I/O ("clients") and those that provide I/O ("servers"); and

3. controllers, the hardware/software system that interfaces to physical storage devices and which is responsible for their control.

We will examine the research opportunities within each of these categories in the sections to follow.

From the viewpoint of data storage, the most relevant components of the operating system are the applications interface on the one hand, and the file system on the other. The former provides a procedural interface for opening and closing files, and reading and writing records or byte streams. I/O may be synchronous, in which the application must wait for its I/O request to complete, or it may be asynchronous, in which the request for I/O service is decoupled from its completion. The file system manages the mapping of names into physical locations of data, and implements the protocols for requesting service from I/O controllers, which may be anywhere in a network. In addition, the operating system is responsible for providing transparent management of the storage hierarchy, that is, the migration of files between primary (*e.g.*, in-memory cache), secondary (*e.g.*, magnetic disk), and tertiary/archival storage (*e.g.*, magnetic tape or optical disk).

Interconnect is a generic term for the "glue" that interfaces the operating system, and the host or client processor on which it runs, with the rest of the storage system. These consist of high speed hardware interfaces (such as physical wires or control registers) and associated protocols. The protocols may be interpreted by either software or hardware. The physical components may be high speed networks, processor-to-processor channels, or ports to a memory system controlled via direct memory access techniques. The computing environment and the intended application will dictate the way in which the storage system is coupled to the host/client.

The final component of the data storage system architecture is the controller. Traditionally, I/O controllers have managed the host/controller interface on one side and the controller/physical device interface on the other. They are responsible for device management and the matching of bandwidths between relatively slow physical devices and higher speed elements of the system, such as the interconnect described above. Controllers abstract away many of the physical details of the device from higher levels of control. We view controllers as one area where there are rich opportunities for migration and enhancement of functionality, and a redefinition of the interfaces that it manages.

4.1.1 Operating Systems

An important research opportunity exists in developing application-oriented file systems. Most existing file systems were designed 10 to 20 years ago, when the relationship between CPU speeds and storage system speeds and the cost trade-offs between semiconductor memory and secondary storage were quite different than they are today. Formerly, file systems exhibited many more reads than writes, but with large portions of memory being set aside for file caches, we can expect the performance of future file systems to become dominated not by reads, but by writes.

For example, researchers in the laboratory of John K. Ousterhout at the University of California, Berkeley, are building a "log-structured file system" that collects many small files within a cache memory and writes these in bulk to secondary storage in an append-only fashion, thereby taking advantage of large transfer sizes and minimizing positioning latencies. This work promises a five- to ten-fold improvement in performance for conventional UNIX-like workloads. Its general strategy is to conserve I/O rate and data rate by including more intelligence in the file system (why read 10,000 bytes if you only need to access 100). Other approaches could be geared specifically to (1) high data rate/very large files, (2) high I/O rates for small files or transactions, and (3) support of database management systems. The last of these areas offers a great challenge because a database system requires close control over the management of cached data, which is hidden by existing file systems. A promising area for research is to explore parameterized interfaces that allow applications such as these to have more control over items that are cached.

Caching is a technique that has been used with considerable success throughout the data storage system, but many issues need further study. Caching allows frequently referenced data and control structures to remain resident in a relatively smaller but higher-speed level of the memory hierarchy than the one in which it would otherwise be found. But there is a problem: we are witnessing a movement towards distributed functionality, resulting in the caching of the same data at many levels and in many places within the system. This leads to the following research question: what kinds of caching and methods of cache coherency (maintaining a consistent view of cached information in the presence of distributed updates) work best in the new environments we have envisioned?

The growing interest in multi-media information systems makes it likely that large distributed repositories of diverse types of non-traditional data will be commonplace in the future. Although we only have a limited understanding of the implications of such data for storage subsystems, this situation is likely to change in the near future. A number of projects are under way to build repositories and interfaces exploiting diverse types of data. The demands of these projects will provide the impetus for new work on storage subsystems.

A critical aspect of future file systems will be their need to manage effectively many levels of the storage hierarchy. Today's systems do an inadequate job of managing archive: there are no good tools to migrate files from long latency archival media to faster access secondary storage. The problem becomes even more complicated as new technology leads to new levels in the storage hierarchy:

- file cache;
- solid state disks (10x cost, 100x performance);
- performance magnetic disks (many actuators relative to capacity);
- capacity magnetic disks (higher transfer rate/high MB per actuator);
- variations on digital recording tapes and helican scan techniques;
- removable optical disks in a jukebox;
- *etc.*

The research issues include migration policies. There are well known techniques for managing a processor's main memory cache, including prefetching and clustering of related data, but how do these methods apply to files whose rereference rates may be measured in days or weeks?

If the operating system (OS) or applications designers are going to play a bigger role in the design of storage hierarchies, they need to be able to evaluate alternatives easily. They need early feedback on key design decisions before investing substantial implementation effort. The tools available to them today are relatively crude. Analytical and queueing tools are not particularly effective in the presence of caching, migration, and sophisticated hierarchy management algorithms.

Simulation offers greater realism, but requires considerably greater investment of time in the development and validation of models. A tool that allows vendor-supplied, parameterized, simulation models of devices to be easily integrated into a simulation model of a memory hierarchy would be of considerable value here. Such a model could be driven with traces of interest to the hierarchy designer to obtain feedback on the design of the hierarchy. Although simulation can never be a substitute for real implementation, it can serve as an important component of the design process. Although the feasibility of this approach has been demonstrated by Mahadev Satyanarayanan and coworkers in the implementation of the "Andrew" distributed file system at Carnegie Mellon University, considerably more work remains.

Extensive tracing facilities to collect data from which alternative migration strategies can be evaluated are needed throughout the storage system. These are not currently available in most existing environments. While there have been many papers on these topics in the past, the actual engineering of such systems needs considerable additional work. One resounding conclusion that emerged from this

workshop is that no one has a clear idea of the users' reference patterns -- not academics, not industrial researchers, not even the users themselves. The industry must take a fresh look at embedding performance monitoring and analysis tools into every level of the storage hierachy. These tools will support the collection of traces and statistics to drive the design of new storage system architectures, to allow tuning of existing architectures, and to support the adaptive configuration of the storage system using artificial intelligence techniques.

Another research issue permeating the entire storage system is the question of high availability and fast recovery. Many of today's workstation-oriented file systems are far from robust. A key operating system technique for achieving high availability is data replication. But replicating data carries with it costs in capacity and bandwidth, as updated data must be reflected in more than one place in the system. In fact, redundancy techniques of various kinds are implemented throughout the storage system (*e.g.*, mirrored or parity check disks, and ECC codes inside disks). An important issue is the question of optimization: where in the storage system should one place the redundancy mechanisms, to maximize data availability while suffering as little degradation in performance as possible?

Portable machines of substantial functionality and power are now emerging. Effective and widespread use of these machines hinges upon the ability of users to integrate them into their normal computing environments. An important element is access to shared data. Distributed file systems that can effectively support disconnected operation of clients will become increasingly important in the future. Disconnected operation of portable machines will, in turn, lead to a demand for lightweight disks of large capacity and low power consumption, which has, in fact, already begun with laptop computers.

File server architectures are to be found in many local area network environments. In terms of future storage systems, how will the concept of the file server change? To a large extent, it is a question of the conceptual coupling of the disks to the network and the protocols through which data is accessed from a client. The conventional view is that a file server, which contains a good deal of intelligence, couples the disks to the network and communicates via file-level protocols. An alternative is a networked disk that communicates via much lower level block-oriented protocols. Both approaches have advantages in certain applications environments. The networked disk alternative leaves the implementation of the file abstraction in the host or client.

Vendors are now bringing to market specialized file-server machines. It does not need a high resolution display. Such a machine does not need to support logins from remote users. It does need to dedicate additional resources (memory, bandwidth) to the storage system it manages. In future computing environments, what will be the hardware base of the file server? It will likely to bear little resemblance to a workstation. Much work needs to be done to understand the optimal architecture for providing the file service function.

4.1.2 Interconnect

The technology of system interconnect is undergoing rapid change, leading to many alternative system partitionings that need to be re-examined. We divide interconnect technology into three categories: network, channel, and memory ports/busses. We expect network distances to be measured in tens of kilometers, supporting gigabit transmission speeds, but being somewhat "unreliable," therefore requiring protocol support. A channel interconnect distance is shorter, spanning tens of meters. Within ten years, channel transfer rates will be in the range of a gigabyte per second and channels will be sufficiently reliable that only the simplest request/acknowledge sorts of protocols will need to be supported. Memory ports will run at channel speeds as well, but will be accessed through direct memory access techniques.

A critical need will be the development of low-overhead network protocols for high data-rate and high I/O-rate transfers, both for local area and wide area networks. The kinds of latencies we would expect to see are in the microsecond range for departmental networks, milliseconds for enterprise networks, and seconds for national networks. While we hope that the same protocol could be effective for both applications targets, it may well be the case that this cannot be achieved. An effort should be made to support research leading to the next generation of high speed interconnects beyond FDDI (Fiber Distributed Data Interface) and HPPI (High Performance Parallel Interface), which are in the process of emerging as today's high performance standards. What will the interconnect media and access hardware be? Will it be possible to leverage off of existing protocols, and perhaps migrate these to custom hardware to reduce the associated overheads?

4.1.3 Controllers

The final component of the data storage system architecture is the controller, the hardware/software system that manages the interfaces to physical devices. Many existing controller architectures have been predicated on the assumption that computer systems are CPU-limited, and it has been rare that the controller represents the bottleneck in the I/O system. This is rapidly changing, as processors and devices get faster, as functionality migrates to the controller, and as peripheral interfaces become more intelligent and thus incorporate higher latencies.

Controllers are rapidly evolving into multi-access multiprocessor computing systems in their own right, running real-time executives with integrated network interfaces. These systems must provide low control latency (always desirable) while handling either enormous numbers of interrupts per second or enormous internal data bandwidths.

There are a number of advantages in moving functionality into the controller. For example, dual non-volatile track buffers could support ping-pong transfers to/from devices while simultaneously

supporting a fast write capability (the write completes as soon as the bytes are transferred into the buffer). The effectiveness of buffering and caching throughout the device-controller-host/client must be examined and analyzed, and the role of non-volatile storage elements must be assessed.

Another important research topic is controller architectures that support high availability. Redundant arrays of disks are only one component of the availability equation. Duplicated controller hardware, error correcting codes, duplicated access paths, end-to-end data integrity through such techniques as parity protected datapaths, and expert systems for analyzing disk errors and predicting disk failures are additional techniques that need further development and analysis. Many of these techniques are known to industry, but are not available in a form that can be taught to students nor readily implemented in prototypes. Little work has been done on understanding and characterizing disk failure modes, and some of the techniques, such as disk arrays, may introduce new failure modes for which we do not yet have any experience (*e.g.*, cabling errors or pulling the wrong disk during repair operations). Computer system architects have little appreciation for the reliability issues nor a good idea of the kinds of policies that lead to effective reliability throughout the system.

One possible area for fruitful research involves reexamining the controller-to-device interface. Existing industry wisdom is to provide high-level intelligent interfaces, which abstract away much of the detailed control information about what the device is doing. The corrected error rate, which is all that is usually passed back to the controller, is significantly less than the raw error rate.

An interesting alternative is to reveal significantly more of the knowledge of the physical position of heads with respect to the rotating surfaces and the details of the data recovery process (retries after seek errors, ECC errors, phase errors in phase lock loops). This would allow better fusion of information in the controller, either to predict device failures or to choose more effective recovery procedures. For example, rather than automatically performing multiple retries with respect to a read failure, the device could reflect the error back to the controller immediately, allowing it to decide on the appropriate correction strategies, including reconstituting the data from redundantly stored information on other devices. Such a device would implement the primitives, while the controller would set the policies. Direct control of the device could lead to significant improvements in device efficiencies, throughputs, and reliability. This strategy, often effective in many areas of computer systems, has yet to be applied in storage systems.

4.2 Data Storage Device Technology

Data storage systems research is likely to result in both evolutionary and revolutionary changes to existing storage hardware technologies. Magnetic disks, magnetic tapes, and optical disk systems can be expected to be the mainstays of the digital storage industry during the 1990s, although they may begin to

take on a wider variety of very different forms. Alternative technologies may also play a significant role, although their future is less clear. We describe here the research issues and possibilities that the workshop participants identified for these different types of data storage hardware. As noted above, this involves a wealth of interesting and important research opportunities in many fields of engineering and science.

4.2.1 Magnetic Disk Storage

During the past decade, the physical volume of a 30-megabyte hard-disk drive has shrunk from a cubic foot to less than 25 cubic inches -- one seventieth as large. At the same time the price per megabyte has plunged from $140 to under $10. Extrapolation of these trends to the twenty-first century suggests that drives the size of cigarette packages will exist and will sell for 10 to 20 cents per megabyte. Quite conceivably, this trend in size will open new marketing opportunities, such as new personal portable appliances. Likewise larger devices based on the same technology will open new opportunities for both small and large computer systems.

Increased volumetric storage density is essential to continue reducing cost, weight, space, and power requirements of the storage system. Performance requirements will demand corresponding increases in data transfer rates and decreases in access time. All of these improvements will impose severe demands on the magnetic and mechanical properties of media, transducers, and the head-disk interface, as well as on the mechanical design and the electronic signal processing of the system.

An appropriate goal of university research in this area is to develop an understanding of the physical factors which can bring about the necessary advances in materials, processes, mechanical reliability, and signal processing algorithms. This multidisciplinary knowledge base will be essential to developing the ability to manufacture future systems. Some of the areas of research that will lead to more fundamental understanding are summarized here:

Magnetics

To increase recording densities both narrower tracks and shorter wavelengths are required. This has implications for media, heads, the head-disk interface, and the mechanical and electronic aspects of the system. For media, shorter wavelengths imply increasingly thinner magnetic films with higher coercivity and with smaller grain size to minimize noise. Head materials will need higher saturation magnetization and permeability, and heads will have to fly much closer to the disk. To maintain a reliable head-disk interface while decreasing flying heights, the media will need to be extremely smooth. To compensate for the poorer signal and noise properties of narrower tracks, new complex sensor structures (*e.g.* multielement heads or superlattice materials) will be needed, and perhaps different or new phenomena (*e.g.* magnetoresistive effects or enhanced properties of superlattices), to provide larger signals. For the same reasons new signal processing techniques will be useful.

Basic understanding of materials, processes, interfaces, and micromagnetic interactions -- and invention resulting from that understanding -- will be required to make these advances in media. Likewise, to maintain sufficient signal-to-noise ratio as the bit cell size decreases, the micromagnetics of smaller data bits must be understood well enough to manipulate them. Similarly, since the signal energy available to the playback head decreases with bit cell volume, the same types of understanding are required for transducers and the materials from which they are made.

Magnetic anisotropy and exchange interactions play a large role in control of the coercivity and noise in media. A better understanding of these parameters across intergranular interfaces, thin-film layer interfaces, and multiple ultra-thin film layers is needed in order to develop new ways to control these parameters. For media this means that a better understanding of the role of thin-film crystalline microstructure of the magnetic film, the underlayer, and the substrate are required. This understanding will largely come about through studies of the effect of composition of materials and processing conditions, using the most advanced analytical tools.

Some of the tools for understanding domains, microstructure, and their interaction are: Transmission Electron Microscopy, Lorentz microscopy, Scanning Electron Microscopy with Polarization Analysis, Magnetic Force Microscopy, Scanning Tunneling Microscopy, and modeling. Micromagnetic modeling has played a significant role in interpreting domain behavior and it is expected to be at least as significant in the future. Advances in currently used measurement and modeling tools and invention of new ones should be encouraged as this will no doubt speed up advances in our understanding just as it has done in the past.

Permeability and saturation magnetization are the primary figures of merit for the recording head. In thin-film heads, permalloy is the workhorse material today. Increasing the magnetization by a factor of two while maintaining high permeability will require invention of a new material. Both new compositions such as iron nitride and new structures such as superlattices need thorough investigation. Performance criteria, such as high sensitivity and low noise in playback heads, are largely determined by magnetic anisotropy, magnetostriction, and exchange interactions. For the currently used inductive transducers, better methods of observing the domains, domain walls, and their motion, and better methods of relating these these observations to the fabrication processes would be useful. New process control methods to make low magnetostriction materials and new processes which minimize stress are needed to maintain sensitivity as smaller trackwidths are incorporated. Multiple layer thin films might be one approach. New materials, devices, and structures which utilize lower noise phenomena such as magnetoresistive, giant magnetoresistive, or Hall effects may be alternatives to inductive transducers. Magnetoresistive devices have shown considerable promise recently but still suffer from instability in domain structure. Considerable work in the area of new devices is appropriate.

Modeling of the transducer domain structure, the side fringe fields, and the recorded bit and track edge patterns will be increasingly useful as the trackwidth and bit lengths are decreased. This work should shed light on the write process, noise due to both media and device domain structure, and noise and crosstalk from the region between tracks and from adjacent tracks. Understanding the degree of track crosstalk and its causes should be very useful for improving tracking servo accuracy.

Mechanics

The future disk drive requirements of smaller size, more efficient volumetric packing, shorter access time, and higher data rates imply that drives will have increased rotation rates, faster actuators, and smaller disk diameters to reduce latency and seek times. Small trackwidth and short wavelength dictate a much smaller head disk interface separation. Narrowing the track width also leads to the need to have more precise methods of staying on track.

Solid mechanics areas that require advancement include:

1. Understanding the limits on rotational rates of the spindle and bearings and developing new high speed bearing systems;

2. improving spindles to reduce both repeatable and non-repeatable runout;

3. improvements in the mechanical rigidity and dampening of vibrations of the spindle, actuator arms and bearings, the head mounting springs, and the (smaller) system mounting support structure;

4. new micro-slider designs and mounts which will allow lower flying heights with increased stability.

Achieving better understanding of how all of the mechanical components are interacting will require new numerical simulation of the components and the head-disk assembly. Issues in fluid flow and thermal mechanics include developing better models of airflow patterns and their effects on the head-disk interface, power consumption, flow of contaminants, and thermal gradients inside the drive box. Thermal distortions of the components can affect both tracking and alignment. As the head-disk interface distance approaches contact and occasionally reaches it, and as the disk is made increasingly smoother and the air bearing becomes thinner than the mean free path of molecules in air, new numerical simulation tools will be required to predict flying height and performance during actuation.

Tribology

The tribological issues for future disk drive development center around having closer head disk spacings, monolayer lubricants, thinner wear and magnetic layers, smoother surfaces, and the continued need to operate at higher rotational rates while still being able to land the head slider when the drive

stops. Tribology, the head-disk interface, and the cause of a head crash is probably the least predictable area of disk drive design. Yet this is where confidence in the reliability of the drive must come from. Contaminants or changes in the materials on the disk or slider can cause excessive wear, friction, stiction, or a crash. Hence, one needs a better understanding of the materials involved and the processes which are occurring. The needed areas of material research include developing models and better experimental tools to achieve greater understanding of the roles of the lubricant, wear layer, slider, mechanical texture, roughness, and the mechanical processes during the contact process.

Among the questions that need to be answered in the continuous quest for better materials are these: What are the mechanisms and chemistry of lubricant adhesion, redistribution, and chemical breakdown? How stable must the lubricant be to maintain lubricity with age or wear? What is the role of mechanical texture in stiction control? What new, thinner protective overcoat wear layer materials might work better than carbon? What are the microscopic failure mechanisms of the wear layer? What interfacial temperatures are reached during slider disk contact and are these temperatures high enough to destroy the lubricant?

To answer some of these questions new tools need to be developed. Some of these include instrumentation for measuring very low flying height, for detecting and determining the slider dynamics when asperities are encountered, for determining microhardness and fracture toughness, microscopic probes such as atomic force microscopy for determining microstructural changes. Modeling tools which would be useful could be directed at the effects of stress in the slider and the disk surface, thin film adhesion failure mechanisms, micro- or nano-mechanics of wear and friction mechanisms, and interface temperatures in quasi-contact recording.

4.2.2 Magnetic Tape Storage Technology

There are many types of tape recorder systems on the market using a variety of technologies. The applications for them range from inexpensive consumer analog audio, digital analog, and video to expensive high performance instrumentation, broadcast video and audio, and computer backup and main storage systems. In general, they can be divided into technology groups by transport system and signal encoding. Computer backup systems have traditionally used linear transports with parallel head arrays to provide multiple simultaneous signal channels. In theory these linear transports with parallel channels can give faster file seek times and greater data rates than helical scan systems.

The helical scan approach is more efficient in utilization of tape surface area due to narrow trackwidth and track format. The linear transport has commanded most of the computer storage applications, but recent trends show the helical format playing a larger role. The large volume consumer video and the high performance instrumentation helical recorders are pushing this technology into the

computer field. Independent of which technology prevails, computer storage applications such as transaction storage for the business world and image storage for the scientific, business, and consumer world demand less weight and greater volumetric efficient storage with improved transfer rates and better access times to both the beginning and end of a file.

The highest performance state-of-the-art systems have gap lengths of about 8 microinches (2000 Angstroms), linear densities approaching 10^5 flux changes per inch, and trackwidths of 17 micrometers. Currently the highest available information density is 117 Megabits/in^2, and the highest commercial data transfer rate per channel is 120 Megabits/sec. (Scientific computational systems which deal with images require files that are greater than a gigabit in size and should be accessed and read into the computer in about one second.) These performance levels are not commonplace and represent use of brute force and the best available technology. Research in the following areas is needed to make these performance levels available to computer data storage and to advance beyond them.

Heads

Future tape-based data recording systems will require advancement in head technology. Narrower track widths and head gaps will give higher storage densities, while parallel recording channels will yield higher data rates. Achieving these will dictate the development of either magnetic materials with higher saturation magnetizations and improved high frequency permeability or alternative transducer technology with lower-noise geometries and higher sensitivities.

To achieve these advances, researchers will need a fundamental understanding of new media materials with higher saturation magnetizations and coercivities. Maintaining low head coercivity will require lower or better-controlled magnetostriction. Materials with larger magnetoresistive effects and new multilayer thin-films would improve detection (reading) sensitivity. At the same time, to achieve reliability, the properties of amorphous and polycrystalline thin films must have low temperature dependences, while long-term stability requires better corrosion resistance and temporal stability of their nanostructures.

Theoretical work also can contribute to better head technology. Improved modeling tools and theoretical understanding of micromagnetics, domain structures, magnetoresistance phenomena, and biasing techniques should lead to lower noise and higher sensitivity devices.

Media

Rapidly increasing recording densities call for narrower track widths and shorter bit lengths. This implies the need for higher output signals and lower noise levels from the media. In addition, media

should have a long, reliable life under conditions that produce far higher data rates than those of today's best systems.

At present, most media are composed of a dispersion of fine particulate material. However, metallic thin film media have broken into the market because their higher magnetic moment per unit volume produces higher signal levels. Research can contribute to improvements in both particulate and thin-film tape media.

In particulate media, the research issues include developing improved particles and dispersions for better magnetic and mechanical properties. Particles will have to be smaller, more uniform, have higher coercivity with low magnetostriction, and have improved stability under temperature change and oxidation. Improved dispersions are needed for lower noise and overwrite properties. Achieving this will require better techniques for dispersion, better dispersion chemistry, and new tools and techniques for characterizing the state of the dispersion. Better binders and lubrication chemistry are required to prevent head wear while maintaining the close head-to-medium spacing that is needed to read and write shorter wave lengths.

Metallic thin-film media require smoother substrates since the media are produced by vacuum deposition and the substrate surface is replicated in the thin film. The substrate must also be capable of withstanding high temperatures because for low noise, the media must have small isolated, oriented crystalline grains, and producing that microstructure usually requires deposition at elevated temperatures.

Metal films also need better lubricating systems and lubrication chemistry than particulate tape media since there is no binder to hold the lubricant and since the metal film is both fragile and abrasive to the head. Both particulate and thin film media need higher coercivity and remanent magnetization in order to produce the higher signal levels needed for higher linear bit density. Increased track density will require substrates with improved dimensional stability. To minimize the demand on error-correction systems and to improve reliability, metal tapes will require better corrosion and wear resistance. For the same reason, it will be necessary to decrease the number of dropouts, which implies the need for improved techniques for manufacturing both metal and particulate tapes.

Mechanics

With tapes as with the other data storage technologies, narrower tracks, shorter wavelengths, faster access times and higher data rates have major ramifications on the mechanical tolerances of recording systems. Mechanical accuracy must be sufficient to allow transfer of tape packages from recorder to recorder. Faster random access of very large storage systems implies automated media package handling systems such as jukebox configurations.

For the media these mechanical requirements translate into

1. smoother substrates with better dimensional stability against variation in temperature, humidity, and stress;

2. improved slitting techniques to give more accurate and uniform edges upon which to guide the tape;

3. new backcoatings to allow more uniform traction between the tape and capstan;

4. new cartridge designs, such as a self-threading cartridge, to minimize exposed tape and handling damage.

For the transport system these mechanical requirements imply

1. new tape guidance and tension control;

2. improved or new tracking servo systems;

3. tapes preformatted for tracking;

4. multiple track head arrays to provide direct tracking servo signals.

Head-Medium Interface and Tribology

The future tape technology directions will likely include multiple track head arrays, shorter wavelengths (higher magnetic transition density), higher realative head-tape velocities, and thin-film inductive and magnetoresistive heads. Since the signal level of magnetic recording systems falls off exponentially with head-to-tape spacing, these advances require that the head be brought closer to or in better contact to the recording surface of the medium.

Improved basic understanding or new developments in a number of areas will be needed to help make this possible: understanding and reducing wear or tribological processes between head and tape surface; reducing material transport between tape and head; understanding and eliminating the unwanted development of polymers that accumulate on the head during operation causing friction between head and tape; measuring and understanding interfacial magnetically dead layers; measuring and minimizing head-tape separation; improving contact for the increased width of multi-track head arrays.

4.2.3 Opportunities in Optical Data Storage

The principal advantages of optical data storage devices are high storage density coupled with medium removability. The storage capacity of a drive is determined by the areal bit density of the medium and the number of disks that are stacked on the spindle. The areal bit density is determined by the wavelength of the light and the properties of the focussing optics.

To maximize the areal density, shorter wavelength sources are required along with high numerical aperture lenses with low aberration. Removability is permitted by the relatively large head-medium spacing that is possible with optical disks. The spacing is determined by the focal length of the optics used in the head, and is usually several millimeters. In contrast with conventional magnetic recording, this large spacing eliminates contact between the head and the medium and the associated tribological problems.

Present weaknesses of the technology have mostly to do with performance. Access time, data rate, and write time are inferior to conventional magnetic disks. To reduce the access time, reductions of the size and weight of the optical head and improved actuators are needed. Reductions in the height of the head may also permit stacking multiple disks on a single spindle to improve the volumetric storage density. The data rate can be improved by using multiple channels and by increasing the rotation rate of the medium. Finally, in present systems, erasing old data and writing new data are separate processes; thus two passes over a given track are required to record new data. Further research into direct overwrite techniques is needed to remove this performance bottleneck. Additional specific research topics are listed below.

1. Lasers

Solid state lasers with shorter wavelengths (400-500 nm vs. present wavelengths in the 800 nm range) are required to further increase areal bit density. Other requirements include 10^5 hour lifetimes, high power (50 mW), direct modulation capability with 2 nS rise times, low noise, low threshold, and a cost of under $5. Surface emitting lasers and laser arrays are of particular interest. The use of higher-order transverse laser modes may lead to techniques capable of surpassing the diffraction limit; thus the ability to control the transverse mode is of interest. Other research needs include: integration of lasers with detectors and/or drive electronics, electronic steering, combined laser and collimating optics structures, and external cavity lasers.

2. Low Cost Optics

Low cost precision lenses are needed in optical recording and playback heads. Research should emphasize modelling and designing for manufacturability. Components of interest include microlenses, micro-polarizers, micro-fresnel and holographic lenses, polarization selective optics, achromatic (low dispersion) optics, binary diffractive elements, optical isolators, and integrated optics. Also of interest are active components using electro-optical and/or acousto-optical interactions to achieve beam scanning and focussing without moving parts.

3. Optical Heads

Novel structures and techniques are needed to realize low profile (<6 mm), lightweight optical heads. Promising research areas include split optical head geometries, holographic structures, and integrated optics. Near-field optical scanning approaches are also of interest. Details on specific components are listed above in items 1 and 2.

4. Actuators

Small, lightweight actuators are needed to reduce the access time of optical disks. Very thin structures (<6 mm) allowing multiple disks on a single spindle and high performance servos (>12 kHz) are of particular interest. Also of interest are rotary actuators and non-mechanical beam scanners using electro-optical and/or acousto-optical techniques.

5. Media

As previously mentioned, high-performance media capable of direct overwrite at optical wavelengths of 400-500 nm are needed. Better understanding of the nano-magnetics of the media is required to realize higher carrier/noise ratios (>70 dB at 30 kHz). Advances are needed in tracking technology, including preformatted substrates that are isotropic, flat to close tolerance, and inexpensive. Research is also needed to develop alternative erasable media and structures. Possibilities include multilevel (as opposed to binary) media and three-dimensional geometries employing multiple recording layers.

6. Systems

Systems approaches are needed to conceal the access time and overwrite latency problems discussed above. Possibilities include novel caching schemes and optimal organization of data on the disk. Research is also needed in coding and signal processing and in adaptive servo techniques. Possible research topics include the use of two-dimensional modulation (length x width) and multilevel codes.

7. Other Topics

As mentioned in Topic 1 above, it may be possible to overcome the diffraction limit by using more complicated beam profiles from higher-order transverse laser modes. To investigate the possibility of achieving such super-resolution, fundamental research into the physics of diffraction from subwavelength structures is needed.

Advances in characterization technology will be needed along with advances in each of the above areas. Examples include techniques for studying the dynamics and nano-magnetics of sub-micron domains and improvements in laser-doppler vibrometer techniques for studying the dynamics of high-performance servos.

Finally, research relating to media, heads, systems, and applications for optical tape are also of interest.

4.2.4 Alternative Storage Technologies

A common way of describing technology development is the so-called "S-curve" description. That is, progress in a technology is generally slow at first, then becomes rapid after the technology is introduced into the marketplace, and finally reaches a plateau as it matures. It is in this mature phase that a technology is most likely to be displaced. Magnetic recording technology has shown a remarkable ability to improve continually over decades. And IBM's demonstration of a disk system with an areal density of 1,000 Mbit/in^2 indicates that the improvement will continue for some time.

Nevertheless, there are certain "user parameters" in magnetic storage that constitute disadvantages and invite alternative technologies. Magnetic disk and tape systems are fundamentally mechanical systems. This impacts their reliability and access times. Various solid state technologies have been developed in response to this need. The obvious approach is semiconductor technology. Ordinarily semiconductor memory is accomplished by the biasing of transistors. This scheme relies on the presence of a bias voltage, *i.e.*, this memory is lost when the power is removed. Backup batteries are a possibility, but are often not completely acceptable solutions. Another approach is to use stored charge to bias the transistor. The flash EEPROM is the latest effort in this direction. It suffers from the fact that in order to keep the memory cell simple, thereby achieving a reasonably high storage density, all the memory cells must be erased together. EEPROM technology also suffers from long write times and limited cyclability.

The incorporation of a ferroelectric material in a semiconductor memory cell also provides nonvolatility. Recent advances in the fabrication of thin-film ferroelectrics make them compatible with conventional device voltages and increases the attractiveness of this approach. However, unsolved questions on the long-term operating characteristics of these materials, specifically cyclability and fatigue, still remain.

All semiconductor alternatives today also are more than an order of magnitude more expensive per bit than magnetic disk storage. The semiconductor cost curve is decreasing rapidly, but it is not projected to approach disk costs until well after the year 2000. Cost is not the only factor that places semiconductor RAM at a disadvantage to magnetic disks for mass storage. Low transfer rates and volumetric densities are also a problem.

There are several alternatives. The promise of magnetic bubble devices reached a peak about five years ago. Their costs and total capacities were in the semiconductor range but their nonvolatility did not offset the slow performances. Today magnetic bubble technology is only found in niche markets. An attempt to enhance this technology by storing information in the magnetic domain walls as opposed to

the domains themselves is being pursued by several Japanese companies. Although all the necessary functions of such a Vertical Bloch Line (VBL) memory have been demonstrated, the actual densities are yet nowhere near the promised value.

Other promising magnetic solid state storage alternatives are non-volatile random access memory (NVRAM) devices such as magnetoresistive random access memory (M-RAM). The memory hierarchy consists of random access (RAM), block access (*e.g.* disk drives), and serial-access (*e.g.* tape) levels. Introducing NVRAM could improve the entire hierarchy since it has the capability to overcome limitations of both present RAM and block-access devices.

Optical disk technology today relies on the local transformation of a medium as the result of heating from a laser beam. The spot size is limited by diffraction, which, for a gallium arsenide laser, is of the order of a micron. Several proposals have been made to circumvent this limit to obtain extremely high bit densities. One scheme is to exploit the frequency spectrum associated with the absorption. That is, due to inhomogeneties in the material different atoms will have different absorption frequencies. By tuning the laser one can selectively excite those subsets of atoms with the same absorption frequencies within a given spot. As long as these atoms remain excited, they are unavailable for subsequent absorption, *i.e.*, they produce a "hole" in the absorption spectrum. Such "hole-burning" schemes require materials with very restrictive properties. The associated optical requirements are also very demanding. At this stage, this approach is still very exploratory.

Developments in optical components over the past several years have, however, revived an approach that was suggested shortly after the invention of the laser -- holographic storage. In this approach data is written into a medium a page at a time in the form of a hologram. The data is prepared in 2-dimensional format, for example, by a spatial light modulator. The attractive feature of such a scheme is the very high data rates that can be achieved. A typical page might consist of 64,000 bits. The average read latency is 10 microseconds which corresponds to an average sustained transfer rate of 800 MBytes/sec. For this approach to succeed, significant work will be needed on how to interface such a technology with future computer systems.

Perhaps the ultimate in information storage density is the idea of memories based on scanning tunneling microscopy (STM). In one concept, small organic molecules would be manipulated, attached at precise positions on regular carbon or silicon surfaces, and then detected by STM. The presence or absence of such a molecule would be a 1 or a 0.

In summary, it appears that magnetic disk (or tape) storage is in no danger of being displaced in the next decade. However opportunities exist for novel storage technologies to take a position in selective markets. The M-RAM and holographic storage may enter the hierarchy in niche roles; thus research on magnetoresistive and photorefractive materials should be intensified to support these technologies.

4.2.5 Communications, Signal Processing, and Coding

Common to all data storage technologies is the need to encode, transfer, and decode information in the form of electronic signals that must be processed. Achieving higher storage density and data rates will require improvements in communications, signal processing, and coding.

For shorter access times and higher individual channel data rates, future magnetic or optical tape or disk data storage systems may need to have arrays of drives. Shorter wavelengths of recorded information and narrower tracks with larger relative misregistration will result in lower signal-to-noise ratios, highly nonlinear channels, and channel crosstalk.

University research in detection, coding, and signal processing should produce knowledge and approaches that contribute to overcoming these problems. For example, better characterization of the channel could lead to developing new channel and noise models that describe the nonlinear signals and intertrack interference. Likewise, new signal processing techniques and new detection schemes for very low signal-to-noise ratios and nonlinear signals could combat intertrack or intersymbol interference. Needed is the development of compression and decompression techniques, of multilevel modulation coding and error correction, and of new adaptive techniques for equalization and channel parameter identification. Also needed is an assessment of the system implications of novel detection, coding, and signal processing techniques with arrays of multiple heads. Finally, the impact of VLSI technologies and architectures tailored for advanced magnetic and magneto-optic recording and signal processing requirements should be studied.

4.2.6 Tracking

Currently in magnetic recording disk drives, the linear spacing between magnetic transitions (along the track) is more than an order of magnitude smaller than the spacing between tracks. For example, in a typical current magnetic recording product, a magnetic cell is roughly a half-micron long and twelve microns wide. In recent years, progress toward increased information density has been mainly due to increasing the linear density. Although progress in that direction can still be made, technical obstacles to further increases in density along the track are becoming as substantial as those limiting track density. Thus it is now time to pursue research directed toward a substantial increase in the density of tracks, which means developing techniques to locate and follow narrower and more closely spaced tracks.

Current hard-disk magnetic recording products usually have a stack of disks on a spindle with the storage medium on both surfaces of each disk. A comb-like assembly carrying one head for each data surface moves in and out, deriving tracking information from a particular head that reads from the permanently written surface of one disk. This dedicated servo track approach can lead to incremental increases in track density as the size of the drive is decreased but will be limited by thermal and

mechanical effects. But these incremental increases will ultimately not be enough; major breakthroughs will be needed. Substantial increases in track density will probably require an approach in which each head is independently controllable and derives its tracking position signal from the data surface itself, as is currently done in magneto-optical recording (where the track pitch is on the order of 1 micron).

We expect university research in tracking to produce knowledge and approaches that will contribute to a major breakthrough in track density. One aspect of this work will be identifying and modeling the disturbances that must be tracked; another will be exploring the limits of current track following methods. Among the most challenging research problems wil be to model mechanical resonances in the head-disk assembly after rapid motion.

Another aspect of tracking research is the design of two-stage, coarse-fine actuators for positioning systems. In magnetic recording systems, this will involve development of very small and light second-stage servos with high bandwidth. In magneto-optical recording systems, positioning system research will include studying alternate beam-steering methods and the use of fiberoptic arrays.

Adaptive nonlinear control strategies will also be an area for fruitful investigation. The object will be to minimize seek time and settling time while achieving the very small tolerances in positioning needed during write and read operations. To do this, the researchers will strive to identify parameters that can be used to select the best tracking control strategy for a given application.

4.2.7 Packaging/Manufacturing

Manufacturability

As the penetration of magnetic storage into lower and lower segments of the marketplace continues, and with the standardization of key technologies and interfaces, the cost of the device itself rather than the cost per megabyte, is becoming the overriding issue. This trend puts increasing pressure on the manufacturing elements of the design. This problem can be, and is being, attacked on two fronts: the design of efficient and controllable manufacturing processes, and more importantly, design-for-manufacturability of the device. Although the former issue is important, the latter has far more leverage for cost reduction. The key issues are the following.

Design

The manufacturability of a device begins with design. The design determines, for example, whether the device has 100 or 500 components, whether screws or snap fasteners are used, whether the device can be assembled from one side or has complex interconnects, and whether tolerances are machined-in, cast, or determined by several final adjustments.

Process Latitude: An important manufacturability issue is the amount of latitude designed into the manufacturing process steps. Does the process require constant monitoring? Are adjustments to the tooling or chemicals needed? Are yield busts common?

In order to design complex electro-mechanical devices while achieving efficient manufacturing processes at low cost, research is required in numerous areas. Clearly advances in computer-aided design are required. Likewise advanced work in mechanical modelling is needed – in all areas, *i.e.*, tolerance, mechanical performance, vibration, thermal and air flow. Research in advanced materials is also called for, especially in processing and performance.

Academic institutions can contribute research to all of these, but their most important contribution to the manufacturing process is education. For development engineers, the curriculum should include manufacturing issues, including the design process. The intent of that curriculum should be an integrated development/manufacturing education, since it is necessary to understand and appreciate the manufacturing line and manufacturing processes before one can begin to design for manufacturability. The key issues to be addressed are:

- mechanization or automation of small assemblies;
- auto-feed and driving of fasteners;
- material movement for fragile assemblies;
- small-part logistics system.

Packaging

Although not normally considered part of the technological innovations of a product, the convenient packaging of a high function device has high leverage in its final application. One has to only examine the current state of the art of camcorders to realize the leverage that high density, flexibly mounted electronic components, or very small high-precision mechanical elements have on the ease-of-use of this device.

In recording technology, physical packaging also plays a dominant role. The number of rigid magnetic disks which can be stacked on a 3.5-inch (or smaller) spindle, the physical thickness and strength of thin polymer films used as substrates for magnetic tape, the precision of very small ball bearings, all play a role in how much information can be stored in a memory device.

Areas of research into this important topic include:

- Materials: High strength plastics, metals, and ceramics which allow thinner and smaller elements to be fabricated. Examples include glass rigid disk substrates, improved mylar tape substrates, and ceramic high strength actuator elements.

- Mechanical Engineering: Methodologies to decrease non-repeatable runout of spindles; very small high-torque motors; light, high-efficiency rotary actuators to minimize power; and non-contact bearings.

- Electrical Engineering: High density direct chip attachment to flexible cable connectors; low heat generation/high density chip designs; novel thermal conduction techniques for components buried within high-density electrical packages; and high-density, low-cost, high-reliability interconnect systems with ease of connect/disconnect.

4.3 Cross-disciplinary Connections

Sections 4.1, 4.2, and their subsections may suggest that data storage systems research is easily compartmentalized into systems architecture and device technology. But that division is artificial; it arose only as a consequence of our desire to enable researchers in different disciplines to see how they can become involved in addressing the issues of this highly interdisciplinary area.

This section is intended to encourage researchers to cross that artificial boundary. Here, we point out that the new knowledge necessary to achieve major technological breakthroughs in data storage is more likely to develop from projects requiring the expertise of more than one discipline, and we emphasize that the least explored cross-disciplinary interfaces are those which cross the boundary between systems architecture and device technology.

For example, computer architects and disk controller designers, working together, could evolve a much improved storage system. Today's disk devices contain microprocessors and small semiconductor memories of their own. What system level functions, such as key searches, can be migrated into the device? How might one level of a system be provided the "intelligence" to take advantage to the behavior of a device at a lower level, thereby improving error handling and performance? Potential improvements, suggested by these and other questions, can be developed in joint projects viewing the storage system as a whole, but might be lost if the interaction between the architects and the device designers were limited by previously developed standard interfaces.

Similarly, improvements in linear bit density and track density in disk and tape data recording devices will be accompanied by changes in technology, such as multiple heads and new signal processing, detection, equalization, and coding techniques. The new device configurations and the signals they produce may offer opportunities for significant improvement at the system level if system architects and device designers work in close cooperation.

Very few technologies are as dependent upon multidisciplinary advances as data storage systems. Thus there is a need for establishing strong cross-disciplinary connections and new avenues of communication among researchers who to date have interacted only through well-defined standard interfaces.

5 Recommendations

Despite the commercial importance of data storage systems, they have generally been neglected in federally supported research programs. The NSF's award of an Engineering Research Center (ERC) grant for the Data Storage Systems Center at Carnegie Mellon is an important step in the right direction. But that ERC, even when supplemented by the few, largely industrially-supported, university centers noted above, does not constitute an adequate national research program for this vital industry. To implement the technical program that is described in this workshop report, there must be changes within federally funded programs. The changes must begin within the funding agencies themselves, which must recognize the uniquely interdisciplinary nature of data storage research (see Table 1) as well as its importance. The following list outlines several specific suggestions for programmatic changes:

1. *Establish appropriate level of funding for data storage research.*
   The workshop identified a number of areas where research is needed in data storage technology, but the participants did not have time to propose a specific program and establish a funding level to support it. This is a task which should be done by the National Science Foundation as the next step in providing support for data storage research.

2. *Create a focal point for data storage research within NSF.*
   At present, there is no single point within NSF where researchers interested in data storage technology can go for support. With the present structure of the agency, interested researchers must go to Materials, Microstructures Engineering, Computer Science, Mechanical Engineering, or other programs to obtain funding. Typically, the program directors involved have little knowledge of data storage technology and, at most, will use a small fraction of their budgets to support work in this area. With this organization, there is no method of coordinating support to be sure that critical areas are covered and that duplicative efforts in certain areas are avoided. The proposed focal point must be cognizant of the research needed to advance data storage technology and also be in a position to leverage existing NSF programs in support of data storage technology.

3. *The NSF should provide data storage research support to system-level projects, multidisciplinary projects, and individual investigators.*
   Because of the nature of data storage systems, multidisciplinary research projects should be encouraged. Funds for data storage research projects should not be provided to multidisciplinary teams to the exclusion of single investigators, but the workshop participants believe that many of the greatest opportunities for innovations lie in areas which require knowledge or expertise from two or more disciplines to address. Hence the NSF should look for proposals involving professors and students from different disciplines, university-industry cooperative research programs, and research programs involving single investigators working with university centers doing systems level research in this area.

4. *Make data storage research an integral part of other Federal programs, such as the Federal High Performance Computing Program.*
   The proposed Federal High Performance Computing (HCP) Program represents perhaps the best opportunity to make data storage technology an integral part of a program focused on high performance computing. Storage systems will be crucial to the success of that program, yet there is little mention of research in storage systems. This problem should be corrected in the HPC Program.

5. *NSF should set up a joint industry-university advisory committee to help guide NSF in implementing these recommendations on data storage research.*
   Such a committee would consider the following questions: What is the current level of collaborative work between universities and industry? How can it be increased? What persons or organizations are best suited to initiate discussion and collaboration and how can they best do so?

Seagate Technology, Inc.
Bruce Bicknell, Manager, Subsystems
Gordon F. Hughes, Director Engineering
Norman B. Talsoe, V.P. Tech. Programs

Storage Technology Corp.
Ralph Butts, Program Manager
Michael Leonhardt, Advisory Development Engineer

3M
Durkee Richards, Division Scientist

Transarc Corp.
Alfred Z. Spector, President

University of Arizona
Masud Mansuripur, Assoc. Professor, Optical Sciences Center

University of CA - Berkeley
David Bogy, Professor of Mechanical Engineering
Randy H. Katz, Computer Science Division

University of CA - San Diego
John C. Mallinson, Director, Center for Magnetic Recording Research

University of Minnesota
Jack H. Judy, Professor of Electrical Engineering
John M. Sivertsen, Assoc. Professor of Chemical Engineering & Materials Science

Washington University
Marcel W. Muller, Professor of Electrical Engineering

Zitel Corp.
Jack King, President & CEO