Basics of SCSI: Firmware Applications and Beyond

Mark S. Kolich

Computer Science Department
Loyola Marymount University,
Los Angeles

Abstract

Since 1981, SCSI has been one of the most powerful and reliable peripheral technologies used in high-performance computing environments. Five years after its conception, A.N.S.I. (American National Standards Institute) approved the SCSI interface as an official industry connectivity standard. Today, SCSI technology continues to grow and is used across the world in the most demanding computing environments. While newer storage technologies, such as Fibre Channel, continue to outperform and outsell SCSI storage systems, the demand for fast and reliable SCSI products remains in the marketplace. [8] This paper is intended to provide a fairly in-depth exploration of SCSI technologies, and the applications behind a SCSI infrastructure.

1 Introduction

Pronounced “skuzzy”, the Small Computer Systems Interface was created to be a universal and intelligent peripheral device connectivity standard. SCSI devices include magnetic disk drives, scanners, printers, optical disk drives, and other communication devices. The SCSI standard, adopted by ANSI in 1986, defines a strict set of rules and guidelines for all SCSI-class devices. For example, the SCSI standard defines the data transfer process over a SCSI bus, arbitration policies, and even device addressing. However, the true advantage of a SCSI interface is the flexibility and ease of which new devices can be added to a SCSI bus. Without the SCSI standard, each new peripheral device would require its own infrastructure and unique device interface. For example, each computer would have to understand how to manipulate the device hardware to read and write data. Unfortunately, with thousands of peripheral devices on the market, this makes developing an Operating System capable of understanding each unique device virtually impossible. Furthermore, the introduction of each new device would require patches and updates to keep Operating Systems current on the latest standards.

Fortunately, the connectivity logic

specified by the SCSI standard resides in the device itself, not in the host computer. To transfer data, the host and peripheral device use a simple set of standard SCSI commands. Additionally, new SCSI devices can be attached to an existing computer with no hardware changes or additional parts. As long as a new device adheres to the SCSI standard, it will function correctly when added to a current SCSI bus.

Not only is SCSI a popular standard because of its interoperability with multiple devices, it also has an incredible track record of backward compatibility. In fact, each new generation of SCSI products are backward compatible with all previous generations. Not only does this keep the demand for SCSI products constant, it also allows businesses and companies who use a SCSI storage infrastructure to extend the life of their SCSI assets. In other words, the introduction of each new SCSI standard does not require the overhaul of existing SCSI components. Newer components can be easily integrated into an older environment.

1.1 Important Benefits of SCSI

Most experts would agree there are four key benefits of the Small Computer Systems Interface:

(1)**Performance** – The newest generation of SCSI, Ultra320 SCSI, supports an average data throughput of 305.175 MB/sec per data channel. The overall throughput of a storage solution helps model large, sequential data transfers similar to a remote backup system, a multimedia file-server, or any other environment which must simultaneously transfer large amounts of data. Additionally, Ultra320 compatibility allows a user to connect the fastest and most reliable peripheral devices to their system.

(2)**Connectivity** – SCSI connectivity support for internal and external peripheral devices is unmatched by any other basic storage standard. A single SCSI PCI or PCI-X adapter can connect up to fifteen devices per channel to extend the value of a SCSI investment.

(3)**Compatibility** – As aforementioned, newer generations of SCSI products must adhere to the standards of previous generation SCSI devices. Therefore, SCSI allows older peripherals to co-exist with the latest technology without hampering speed or performance.

(4)**Reliability** – Since its conception, SCSI has been one of the most reliable storage system technologies.

Unmatched data integrity, low component failure rates, and overall product quality has made SCSI an obvious choice for quality conscious Information Technology professionals.

1.2 SCSI With PCI-X Technologies

As the demand for fast and reliable storage architectures grew in the technology industry, it became clear that conventional PCI standards would not enable multiple devices to perform at their highest levels. The bottlenecks of a standard PCI 1.0 bus, became a large limitation of current Gigabit, SCSI, and InfiniBand technologies. In an effort to dramatically increase the bandwidth supported by an average PCI bus, the technology industry developed a next-generation PCI standard, known as PCI-X. In early 2003, PCI-X 2.0 was released and quickly implemented as an industry standard for higher-end servers and workstations.

The new PCI-X standard is capable of supporting signaling speeds of up to 533 mega transfers per second. Additionally, PCI-X 2.0 is capable of reaching bandwidths more than thirty-two times greater than first generation PCI technologies. Furthermore, PCI-X 2.0 is built upon “the same architecture, protocols, signals, and connectors at traditional PCI.” [9]

The release of PCI-X allowed storage systems to transfer data at unprecedented rates, which led the way for the eventual release of a faster SCSI standard. Not surprisingly, soon after PCI-X became an industry standard, Ultra320 SCSI was introduced as one of the first technologies to benefit from increased system bus speeds. Together PCI-X and Ultra320 SCSI provide the bandwidth necessary for today's applications. Currently, new PCI-X 3.0 technologies are under development, which may lead to the release of Ultra640 SCSI.

1.3 Current SCSI Technologies

To better understand the overall history and development of the SCSI standard, it's important to visualize how SCSI has changed since its conception. The original SCSI bus, known as SCSI-2, supported a throughput of only 10 MB/sec and was typically used for various slower peripheral devices. The latest generation of SCSI, Ultra320 SCSI, supports throughputs of up to 320 MB/sec and is most often used in high-end hard disks. Table 1.3.1 provides a brief snapshot into the history of the SCSI interface.

Type/Bus	Approx. Speed	Mainly Used For
SCSI-2 (8-bit Narrow)	10 MB/sec	Scanners, ZIP-Drives, CD-ROMs
UltraSCSI (8-bit Narrow)	20 MB/sec	CD-Recorders, Tape Drives, DVD Drives
Ultra Wide SCSI (16-bit Wide)	40 MB/sec	Lower end Hard Disks
Ultra2 SCSI (16-bit Wide)	80 MB/sec	Mid range Hard Disks
Ultra160 SCSI (16-bit Wide)	160 MB/sec	High end Hard Disks and Tape Drives
Ultra320 SCSI (16-bit Wide)	320 MB/sec	State-of-the-art Hard Disks, RAID backup applications

Table 1.3.1. Snapshot of SCSI history.

The latest SCSI technology, known as Ultra320, takes advantage of several new features to increase reliability and overall SCSI bus performance. Ultra320 employs a packet protocol which allows SCSI to better control data flow over the bus and increase system speed. Quick Arbitration Select, otherwise known as QAS, “increases bus utilization by streamlining release and re-use of the bus by various peripherals.” [2] Finally, Cyclic Redundancy Check, or CRC, helps “improve data integrity by detecting data integrity errors” [2] for all SCSI phases.

For the remainder of this paper, we will use “SCSI”, to refer directly to Ultra320 SCSI unless otherwise noted.

1.4 Paper Organization

The remainder of this paper will introduce several important concepts behind SCSI, SCSI applications, and SCSI firmware. Section 2 describes, in detail, the basics of a SCSI bus. Section 3 discusses the recent advances of SCSI in the technology industry. Section 4 analyzes several applications of SCSI, and discusses the results of a Ultra320 SCSI performance evaluation. Finally, Section 5 briefly discusses SCSI controller firmware and the benefits of thin drivers.

2 Bus Basics

The fundamental concepts and arbitration policies of a SCSI bus are fairly straightforward once one understands the basic logical flow of events on the bus.

It's important to note that on any SCSI bus, there are two types of devices: SCSI initiators which make an I/O request, and targets which respond to an I/O request. For example, at the highest conceptual level, an operating system will use a PCI SCSI adapter to initiate an I/O process which is directed at a specific target on the SCSI bus. In this case the SCSI target, a hard disk, responds to the request to initialize the I/O command. Now acting as the controller, or “master” of the bus, the disk issues a request to the initiator requesting an I/O command. To complete the process, the initiator responds by sending a command code, known as a Command Descriptor Block, or CDB. However, unlike other data transfer standards, SCSI devices inter-operate as “slaves” and “masters” during a single arbitration cycle. In other words, each device can act as an initiator or target when necessary.

2.1 Device Addressing

As with any data transfer mechanism, SCSI provides an easy and convenient way to address devices on the SCSI bus by assigning each device a unique ID. A SCSI device ID is used during the arbitration and selection process to properly identify the target and initiator of a given SCSI I/O operation. In fact, the number of devices on a single SCSI bus is a direct limitation of the need for a unique SCSI ID for each device. For example, when Narrow-SCSI was introduced, the standard only called for eight data lines, hence, limiting the number of devices on a Narrow-SCSI bus to only eight. The latest SCSI technology, Ultra320 SCSI, operates with sixteen data-lines and can support a maximum of sixteen devices. However, it's important to remember that a SCSI adapter is also considered a device on the bus, and is always assigned a SCSI ID. Therefore, SCSI storage systems are limited to fifteen peripherals, not including the PCI adapter itself.

Not only is the SCSI device ID important for identification, they are also critical for determining device priority. Most often, SCSI initiators (adapters) are assigned device ID 7, while other devices are assigned the remaining device IDs. Device ID 7 is usually considered the device with the

highest priority over the SCSI bus, allowing it to gain control of the bus most often. SCSI experts will typically recommend assigning higher priority SCSI device IDs to slower devices. For example, SCSI tape drives need higher priority device IDs to prevent high performance devices from over-utilizing the bus.

2.2 SCSI Bus Control

The first stage of a SCSI data transfer is known as arbitration. In its purest form, arbitration is the process of selecting a single device from a collection of devices that require concurrent use of the SCSI bus. Because all physical wires of the SCSI bus are shared with multiple devices, a systematic process must be in place to control the flow of electrical signals connecting the bus and its peripherals. The full process by which a device "obtains permission" from all other devices to transfer data or communication with the controller is known as the **arbitration phase**. Within the arbitration phase, a device patiently waits for the SCSI bus to enter a "free phase", or idle phase. When the bus becomes idle, no further data transfers are taking place which allows a device to "raise a flag" alerting all members of the bus it has a request to gain control.

Interestingly, the SCSI arbitration phase is analogous to a well-ordered discussion group. A moderator, in this case, the SCSI controller, is in charge of moderating the discussion to ensure that only one member of the panel speaks at any given time. If a member of the discussion wishes to speak, he or she must raise their hand. Similarly, the firmware of a SCSI controller acts as the bus moderator by ensuring that only one device uses the bus at any given moment, and that each device obeys the fundamental arbitration policies of the SCSI standard. Obviously, if more than one person attempts to speak in the discussion group at the same time, communication becomes incoherent for all discussion attendees. On a SCSI bus, multiple devices attempting concurrent information transfers will cause data corruption which often leads to massive system failures and headaches for SCSI engineers.

Once a device has successfully "won" control of the bus from the controller, it must signal a device it wishes to communicate with in the **selection phase**. The selection phase connects two devices on the bus, and initiates a data transfer. In relation to our discussion group example, the selection phase is equivalent to a moderator selecting the next person to speak to the group. Again, this process

is very important and all participants must agree to wait their turn to prevent confusion.

Once all data transfers have completed, the bus returns to an idle state and waits for the next arbitration request. This arbitration and selection cycle continues indefinitely as data requests enter the SCSI controller from the Operating System.

2.3 Information Transfer

As previously mentioned in Section 2.2 of this paper, the **information transfer process** initializes immediately following the selection phase. Behind the scenes of the SCSI sub-system, several key steps are needed to successfully complete and coordinate the information transfer process.

For example, the information phase itself is divided into several smaller phases which work together to transfer data: the Msg-Out phase, the Command phase, the Data-In/Out phase, the Status phase, and the Msg-In Phase. Each phase is critically important to the successful transfer of data over a SCSI bus.

(1) **Msg-Out Phase** – An initiator sends a Command Descriptor Block (CDB) to a target to initiate a data-transfer. If the target successfully responds the information transfer process moves to the second phase, the Command phase.

(2) **Command Phase** – Within the command phase, the initiator sends a Command Descriptor Block (CDB) to the target describing the address of data to be read or written. "The first byte of the CDB is the Operation Code (OP code). It is followed by the Logical Unit Number (LUN) in the upper three bits of the second byte, and by the block address (LBA) and transfer length fields (Read and Write commands) or other parameters. The last byte of each CDB is the Control byte. This byte contains two important bits, the LINK and FLAG; these bits are used for controlling the linked commands mechanism." [1]

(3) **Data-In/Data-Out Phase** – The Data-In/Data-Out phase is used when the CDB cannot be transferred using a single bus cycle. In this case, the initiator automatically partitions the command block over two bus cycles which is sent directly to the target. Upon successfully receiving the CDB, the target will begin to process the specified SCSI command.

(4) **Status Phase** – After the command has executed, the target returns the status to the initiator alerting it of execution success, or execution failure. If the target returns an execution failure or problem report, the initiator will re-issue the request. Unfortunately, a corrupt hard disk or improperly configured SCSI bus can result in an "endless loop" of failures and retries. This often locks up an entire bus as the initiator continues to repeatedly re-send the request in an infinite timeout sequence. Fortunately, newer SCSI systems configured with RAID support will recognize this data failure timeout sequence and initialize a fail over recovery process to resume normal operation.

(5) **Msg-In Phase** – The message in-phase completes the SCSI information transfer process, and requires that the target send a final status report to the initiator. If the command was executed successfully, the initiator can make additional data requests when the SCSI bus returns to a free state.

2.4 Bus Termination

In a nutshell, **bus termination** is the process of terminating electrical signals which may reflect off the ends of physical cabling and travel back to the source, colliding with other signals on their way. To better understand the need for termination on a SCSI bus, imagine you and a friend are holding taut, a piece of rope approximately six feet long. If you were to "pluck" the taut string at one end, a wave will travel from the source, reflect off the other end, and return. The wave will continue to travel back and forth across the string as it slowly decreases in amplitude and eventually disappears.

Not surprisingly, electrical signals travel across wires much like a wave travels across a taut rope. When an electrical signal is generated on a bus, it will continue to oscillate within the wire until it is terminated, or naturally loses energy and disappears due to resistance in the cabling. Obviously, reflection on a SCSI bus will cause multiple signals to collide, creating unwanted abnormalities and potentially devastating data corruptions.

To prevent the reflection of electrical signals at the end of bus cabling, SCSI engineers designed the "SCSI terminator." A terminator is a small device physically attached to the end of the bus cable, which makes the

cabling appear to the devices as if it has an infinite length. Essentially, signals will travel along the bus to all devices and then disappear into the terminator.

It is interesting to note that other bus technologies, such as PCI, expect and even require signal reflection to help enhance the strength of data on the bus. A PCI bus is often designed such that data signals traveling over the wiring systematically merge to produce constructive interference, hence, naturally improving signal strength. Therefore, terminators are almost never used on a PCI bus.

2.5 LUN's (Logical Unit Numbers)

Around 1980 when engineers in Silicon Valley began designing the SCSI interface, future expectations for microcomputers were very limited. In fact, SCSI engineers designed the interface and standard around only two devices: massive tape drives connected to large mainframe systems. The concept that SCSI could be used in smaller, localized microcomputers with multiple devices was unheard of. Therefore, engineers were not concerned about synchronizing the bus arbitration process between more than two devices.

However, many tape drives, even though considered one device, had multiple logical units. In other words, tape drives often contained more than one set of tape reels, which required the engineers to design a unique way of identifying a device, and a reel set. Hence, the Logical Unit Number was born.

Today, LUN's are used to identify logical sub-sections on a wide variety of devices. For example, visualize the following equipment setup:

(1)**Hard Disks** – Two hard disks each with three logical volumes.

(2)**Tape Drive** – One tape drive, with four sets of reels.

(3)**CD-ROM Jukebox** – One CD duplication box with five CD-Writers.

(4)**SCSI Adapter** – One SCSI PCI-X adapter connected to all devices via one bus.

First, let's assume the host system needs to read data from LUN 2 on the tape drive. The SCSI adapter will issue the request, at which time the tape drive must automatically load the logical unit representing LUN 2. Additionally, the adapter might issue a request to the CD Jukebox to write a block of data to LUN 3, which may represent disc 3 in the device. Once the device wins control of the SCSI bus, the command is issued

using the CDB which begins the execution process.

2.6 Single Connector Attachment

Single Connector Attachment, otherwise known as S.C.A., is a very popular technology used in a variety of high-end disk arrays. Often called J.B.O.D.'s (Just-A-Bunch-Of-Disks), disk arrays allow multiple SCSI hard disks to be seamlessly connected to a SCSI bus using a special disk enclosure. Figure 2.6.1 shows a sample disk array, the HP MSA30 SB JBOD. "The Single Connector Attachment (SCA) moves wiring of the SCSI bus directly to the backplane, and allows plugging of the drives into a single socket." [1] In other words, a SCA enclosure provides the sockets for which multiple hard disks can be inserted. Once inserted, the J.B.O.D. recognizes each disk as individual devices, and automatically assigns it a unique SCSI device ID using the S.C.A.M. protocol.

SCA and SCSI disk array technology is most often used in high-end storage environments, or where any application requires large amounts of reliable storage. On average, each high-end disk array will hold anywhere from 13 to 14 hard disks in a standard rack-mount enclosure. Using R.A.I.D. (Redundant Array of Independent Disks) technology with high-end large capacity hard disks, storage capacities in enterprise disk arrays can exceed 2 Terra bytes! "A storage device that uses several magnetic or optical disks working in tandem can increase bandwidth output and provide redundant backup." [1] Furthermore, the latest version of S.C.A. adds "hot-swappable" functionality to most disk arrays. Hot swappable, or hot plugging technology, allows disk devices to be physically added to the SCSI bus without powering off, or rebooting the connected system. Similar to standard Plug-And-Play (PnP) functionality, the disk will be recognized and automatically installed for use upon insertion.

Clearly, hot-swappable technology is a very important feature for mission critical computer systems. In the event of a hardware failure, IT professionals can replace a damaged device within seconds without having to shutdown or reboot a critical system.

Figure 2.6.1. HP MSA30 Disk Array.

3 Recent Advances

For the past twenty three years, SCSI and PCI engineers have worked to improve SCSI performance, reliability, and storage capacities for thousands of devices and applications. Since SCSI's conception, great strides have been made, improving every aspect of the SCSI family. The latest generation of SCSI products, however, adds features and performance improvements unlike anything SCSI has seen before, making Ultra320 SCSI one of the world's safest and reliable storage standards.

3.1 Ultra320 SCSI Engineering

The new technology behind Ultra320 SCSI introduced a variety of complex electrical engineering related problems. Thankfully, most of these issues were successfully resolved as SCSI engineers continue to seek new discoveries in engineering. Here is a brief snapshot of Ultra320 SCSI engineering successes:

(1) **Cable Lengths** – "The faster Ultra320 SCSI speed require[d] new signaling technologies in order to maintain the high reliability required by server designs. Ultra320 SCSI signals on the SCSI bus are twice the frequency of Ultra160 SCSI signals but the cable requirements have not changed. Point to point connections can be 25 meters in length and multiple load systems can be 12 meters in length." [3] SCSI engineers were forced to better control signal strengths on the bus with a faster data frequency and long cable lengths.

(2) **Data Integrity** – "Doubling the maximum signal switching frequency in Ultra320 SCSI has pushed the SCSI bus into a frequency range that has greater attenuation in SCSI bus cables and has also required the signal slew rate to increase. The doubling of signal frequency has resulted in smaller amplitude signals and more reflections (undesired high frequency noise) on the SCSI bus." [3] Unfortunately, noise on the SCSI bus can lead to massive data corruption problems and improper negotiation between devices. Using advanced termination devices on the SCSI bus, SCSI engineers dramatically reduced signal slew.

3.2 Ultra320 Max Throughput

When referring to a SCSI adapter/controller, max channel throughput is often used as a metric to define the overall data power of a specific adapter. However, to better understand the concepts behind SCSI throughput, it's important to understand the definition of a **SCSI channel**. Essentially, a channel is nothing more than a SCSI bus on an adapter. Typically, most high-end Ultra320 SCSI controllers offer at least 2 channels (2 external connectors on the back of the PCI-X controller, one for each channel). A controller's channel count is important because it helps define how many devices can be attached to the adapter, and therefore, how much data can be simultaneously "pumped" through it.

Generally speaking, most average IT professionals and "techies" assume that Ultra320 SCSI will support up to 320 MB/sec on each SCSI data channel. This theoretical maximum of 320 MB/sec is often thought to be the upper limit of the latest SCSI adapters. Unfortunately, the SCSI naming convention is grossly misleading. In fact, current "industry standards define the Ultra320 SCSI Theoretical Maximum throughput in terms of "Mega-Transfers." [5] A Mega-Transfer, or one million transfers, is used to accurately measure the throughput rate of all Ultra320 SCSI adapters. Not surprisingly, "a U320 SCSI Mega-Transfer is equivalent to one-million transfers, or in this case, it's equivalent to 3.20×10^8 bytes/second." [5] Properly converting 3.20×10^8 bytes/second to mega-bytes/second yields approximately 305.175 MB/sec. In other words, contrary to popular perception, Ultra320 SCSI's theoretical maximum throughput per channel is 305.175 MB/sec.

It's important to remember, however, that the theoretical maximum throughput is defined in terms of channels. Therefore, fully utilizing two SCSI channels on an Ultra320 SCSI adapter will generate approximately 610.35 MB/sec of max throughput.

Similar to Ultra320 SCSI, the maximum throughput of previous generation SCSI controllers is often misunderstood as well. For example, the max theoretical throughput of an Ultra160 SCSI channel is not 160 MB/sec. It is also defined in terms of Mega-Transfers. Therefore, an Ultra160 SCSI adapter is capable of generating 1.60×10^8 Mega-Transfers, resulting in approximately 152.587 MB/sec of theoretical throughput. Rarely however, are theoretical maximums achieved in the field by any SCSI controller.

"As new computer systems increase in capability, new applications evolve to take advantage of the available power and features. For example,

desktop publishing, scientific visualization, video and audio editing, digital broadcasting and other data-hungry applications continue to push the I/O bandwidth and require a more advanced interface to handle increased data transfer." [3]

3.3 Q.A.S.

Quick Arbitration and Selection, otherwise known as Q.A.S. was introduced with Ultra320 SCSI in early 2003. On a standard SCSI bus, each data communication cycle between devices is the same: a device which needs to communicate or use the bus must wait for the bus to free itself, and then start the arbitration process with the controller. Unfortunately, this standard bus negotiation method often causes large delays, and dramatically lowers bus utilization.

Fortunately, Q.A.S. now allows devices to negotiate for the bus during a current data transfer! In other words, while processing data over the bus, the controller identifies the device which will own the bus immediately following the data-transfer. "This reduces the overhead of control release on the SCSI bus from one device to another...and reduces command overhead and maximizes bus utilization." [3] Essentially, Q.A.S. allows the bus to be used primarily for data transfers, instead of negotiation. Less negotiation between devices directly means a faster data transfer rate.

3.4 Packetized SCSI

Ultra320 SCSI supports a newer protocol which allows data to be streamed over the bus using data packets. "Packetized devices decrease command overhead by transferring commands, data, and status using DT (dual transition) data phases instead of slower asynchronous phases. This improves performance by maximizing bus utilization and minimizing command overhead. Furthermore, [the] packet protocol also enables multiple commands to be transferred in a single connection." [3]

3.5 Ultra640 SCSI?

As Ultra320 SCSI continues to establish itself a premiere storage solution, the storage needs of IT professionals continues to grow at an unprecedented rate. Newer applications and databases will push the upper limits of current storage technologies, requiring engineers to create faster and more reliable data storage mechanisms. As a result, the success of Ultra320

SCSI has prompted many leaders in the technology industry to investigate newer and faster data storage standards. An obvious choice, Ultra640 SCSI, is currently under investigation in some of Silicon Valley's top technology companies. Amazingly, Ultra640 SCSI promises an exceptional max theoretical throughput exceeding 600 MB/sec per data channel! Unfortunately, current research shows that Ultra640 will never be implemented due to the instability of electrical signals over standard copper-wiring at such high frequencies. Therefore, many companies are investigating new storage standards such as Fibre Channel, which uses fiber optic cabling to connect multiple high speed disk drives. Unlike SCSI, Fibre Channel can transfer upwards of 5 GB/sec per data channel using standard fiber optic technology.

4 Applications

Besides being an obvious choice for high-end servers and workstations, SCSI is often complemented with a variety of technologies used to improve performance and data integrity on a SCSI storage solution. Section 4 of this paper is intended to provide a brief exploration into the applications of Ultra320 SCSI.

4.1 R.A.I.D.

Perhaps one of the most important applications of any storage system is its data backup and auto-failover capabilities. As aforementioned, R.A.I.D., or Redundant Array of Independent Disks, has proven to be an obvious complement to Ultra320 SCSI storage systems. In fact, most modern RAID controllers use the highly reliable Ultra320 SCSI standard as their foundation for implementing services such as data mirroring and disk striping. “RAID subsystems are commonly used as the cost-effective foundation of a business-critical storage strategy. By employing the advanced fault tolerance of RAID technology, companies can effectively implement networked business systems that require large amounts of storage space for data and applications that must be available for their businesses to continue operating.” [4]

For example RAID, combined with the power of Ultra320 SCSI, can implement data mirroring across multiple hard disks to improve system reliability. To better understand the concept of data mirroring, imagine a workstation or a high-end server with only one installed SCSI hard disk. Unfortunately, if this hard disk unexpectedly fails, the entire system becomes completely inoperable.

However, RAID data mirroring with Ultra320 SCSI allows the same data to be simultaneously accessed on multiple hard disks. In essence, this technology creates multiple copies of the primary disk which is stored on each unique back up hard disk. If a primary hard disk failed, the RAID subsystem will automatically failover to a backup disk and immediately resume operation. RAID data mirroring combined with Ultra320 SCSI technology helps IT professionals maintain virtually 100% up times for their most critical data servers.

Another popular benefit of RAID often found in the technology industry is a technique known as disk striping. "With RAID technology, data is striped across an array of physical drives. This data-distribution scheme complements the way the operating system requests data. The collection of stripe units, from the first drive of the array to the last drive of the array, is called a stripe. The granularity at which data is stored on one drive of the array before subsequent data is stored on the next drive of the array is called the stripe-unit size." [4] In other words, data striping systematically combines two or more hard disks into a single logical RAID volume, on which I/O requests are equally distributed. For example, visualize a disk array containing three SCSI hard disks connected to the same SCSI bus. A properly configured disk striping RAID controller will systematically and equally partition each I/O request amongst the three disks. In this case, if the controller issues a request to write a block of data to the RAID volume, each disk will only be responsible for handling one-third of the data. On the other hand, if a single hard disk instead of a RAID volume received the I/O request, it would be wholly responsible for 100% of the I/O process. By equally splitting data amongst each disk in the RAID volume, bus and system performance is increased tremendously.

Unfortunately, disk striping presents virtually no failover or backup capabilities. For example, if one disk in the logical RAID volume fails and becomes inoperable, the entire RAID volume is corrupt. However, newer levels of RAID technology have added a parity disk to logical RAID volumes. The parity disk is a separate physical hard disk which records the parity of each block written to the RAID volume. In the event that a disk in the volume fails, the RAID system can be automatically rebuild itself using the parity disk as a way to determine "which bits were lost on the corrupted disk."

As SCSI and RAID evolved, most RAID adapters can now be configured using a variety of "RAID levels." A RAID level determines how the controller manages data mirroring, and disk striping. It's important to note, that

many popular forms of RAID often used in mission critical enterprise data centers, are an advanced combination of a variety of RAID technologies.

The following list highlights the most popular RAID levels used in mission critical computing environments:

(1) **Level 0 – Striped Disk Array Without Fault Tolerance –** "RAID [Level] 0 uses a technique called data striping to distribute data evenly across the physical drives in such manner as to maximize I/O performance. Striping divides the logical into data blocks called stripes, which are then distributed over the physical disk drives. The layout is such that a sequential read of data on the logical drive results in parallel reads to each of the physical drives. This results in improved performance, because multiple drives are operating simultaneously...RAID [Level] 0 offers substantial speed enhancement, but it provides no data redundancy or fault tolerance." [4]

(2) **Level 1 – Mirroring and Duplexing –** "RAID [Level] 1 provides 100% data redundancy and requires only two physical disk drives. RAID 1 employs the concept of data mirroring. Mirroring creates a single logical disk drive from two physical disk drives. With RAID 1, the first half of a stripe is the original data; the second half of a stripe is a mirror (a copy) of the data...All data written to the combined logical drive is written to both physical disk drives." [4]

(3) **Level 1E – Enhanced Mirroring –** "RAID 1E combines mirroring with data striping. This RAID level stripes data and copies of the data across all of the drives in the array. The first set of stripes are the data, and the second set of stripes are mirrors of the first data stripe contained within the next logical drive...RAID 1E requires a minimum of three drives and, depending upon the level of firmware and the stripe-unit size, supports of maximum of 8 or 16 drives." [4]

(4) **Level 4 – Dedicated Parity Drive –** "A commonly used implementation of RAID, Level 4 provides block-level striping (Level 0) with a parity disk. If a data disk fails, the parity data is used to create a replacement disk. A disadvantage to Level 4 is that the parity disk can create write bottlenecks." [4]

(5) **Level 5 – Data Striping and Block Interleave –** "RAID 5 employs data striping and block interleaving in a technique designed to provide fault-

tolerant data storage that does not require duplicate disk drives…RAID 5 spreads both the data and the parity information across the disks one block at a time. This enables maximum read performance when accessing large files and improves array performance in a transaction processing environment. Redundancy is also provided via parity information, which is striped across the drives to remove the bottleneck of storing all of the parity data on one drive." [4]

(6) **Level 6 – Independent Data Disks With Double Parity –** "Provides block-level striping with parity data distributed across all disks." [4]

4.2 Other Applications

Generally speaking, the SCSI standard encompasses a wide range of technologies and applications which aim to improve the speed and reliability of large storage systems. In fact, the underlying structure of newer storage systems such as Fibre Channel were derived from the development and expansion of existing SCSI technologies. In other words, the development of SCSI has prompted the technology industry to invest in research and development of promising new ideas and concepts. As storage technology and its industry continues to mature, a wide range of new applications are sure to emerge. Rest assured however, that even though different technology continues to develop, the standard foundation upon which they are built will remain the same.

5 Operating System Interfaces

Deep within the internals of every operating system, is a set of instructions and configuration details which helps the O.S. (OS) communicate with various internal and peripheral devices. This software, known as a **driver**, provides the framework interface between a computer and its installed devices. For example, imagine a computer user purchased a new ink jet printer for their home office. Once the printer is successfully connected to the computer, the operating system recognizes the new device and prompts the user to insert a CD containing the printer drivers. In this case, the drivers contained on the CD are essentially small pieces of software which integrate into the operating system instructing it how to properly communicate with the printer. Once the drivers have been successfully installed, the user can then spool print jobs to the printer. Similar to standard

printer drivers, SCSI controller drivers operate in exactly the same way. In this case, the SCSI driver within the operating system provides an interface on which the OS can issue I/O commands to the SCSI controller. Rather than access the device directly, the operating system loads the necessary driver and calls functions in the driver software to carry out actions on the device. The driver functions themselves contain the device-specific code needed to carry out actions on the device.

Most modern enterprise operating systems such as Linux, HP-UX 11.23, and Microsoft Server 2003, implement several layers of abstraction between applications, the operating system kernel, and peripheral devices. The abstraction layers, outlined in Table 5.1, increase device and operating system performance and reliability while decreasing system overhead.

Level	**Layer**	**Overview**
4	Application	An operating system manages applications and other system resources assigned to the needs of those applications.
3	Operating System	When an application issues a service request to a device, the operating system issues a command or set of commands to the device driver.
2	Device Driver	The device driver accepts the instruction from the operating system and issues the raw command to the physical device.
1	Physical Device	The resident firmware on the physical device receives and processes the command.

Table 5.1. Snapshot of various interface layers within modern operating systems.

Once a device driver receives a request from the operating system to carry out a given task, the message is then passed to the physical hardware on the device. At this point, the intelligent firmware on the device parses, analyzes, and executes the request. As it turns out, firmware is not only one of the most critical aspects of low-level hardware and software design, it's also the most intelligent. Firmware in SCSI technologies is discussed further in Section 5.2 of this paper.

5.1 Thin Drivers

When the fundamental concept of a driver was first introduced in modern computing, a significant software engineering effort was made to place intelligence within the driver software itself. Electrical and software engineers assumed this strategy would help keep the physical components on the device simple, fast, and elegant. Not surprisingly, this method quickly proved ineffective and severely impacted critical system performance. Unfortunately, as the complexity and strict performance requirements of storage devices grew, the need for extremely fast and incredibly efficient drivers became obvious. As it turns out, integrating device intelligence into the driver software created a significant performance issue within the operating system.

Unlike firmware, which is executed on a separate resident processor mounted in the device itself, intelligent device drivers are executed through the operating system using the main system CPU. As storage data transfer speeds increased to record highs, undue strain was placed on the systems CPU. In this case, the system CPU was responsible for parsing and analyzing an I/O command. This not only created a significant bottleneck in the I/O subsystem, it also used a majority of system resources allocated for other applications running on the host system. Obviously, any intense I/O operations would indirectly slow other applications running on the system, and adversely impact other important business and user applications.

To address this bottleneck and reduce the strain on system resources, software and hardware engineers began implementing **thin drivers** for various high-performance I/O devices. A thin driver eliminates the performance bottleneck by moving the processing intelligence away from the software driver within the operating system, and into the physical device itself! In other words, the driver resident within the operating system is nothing more than a very simple interface, or API, which applications can use to access a device. The request processing intelligence originally placed in the operating system driver, is now tightly integrated into the firmware on the device. As one would expect, the firmware within a physical device is executing using its own on-board processor which increases device performance and frees the main system CPU for other important applications.

As previously mentioned, modern high-performance SCSI I/O controllers now contain an on-board processor dedicated to handling the I/O requests from the operating system. By implementing thin drivers around high-

performance SCSI firmware, engineers continued to push the limit of the SCSI interface, which eventually lead to the discovery and development of Ultra320 SCSI controllers and devices.

5.2 Firmware

Most computer scientists and hardware engineers would argue that device firmware is one of the most important aspects of a physical device, or controlled system. Firmware resides in virtually every household appliance, electronic device, automobile, and computer across the world. For example, firmware within a household heating and cooling system is a strict controlled system dedicated to managing the temperature inside of a house. The on-board firmware within the electronics of the heating and cooling system controls everything from the thermostat to the large electronic relays inside of the actual air conditioning unit. Also, most modern vehicles contain a sophisticated set of firmware and microprocessors which control everything from fuel and oxygen mixtures in the engine, to the vehicles ABS (Anti Lock Breaking System).

In the case of a SCSI controller, SCSI firmware is specifically designed to manage data flow and arbitration policies on the bus. The SCSI controller firmware also receives and processes requests from the operating system over the PCI bus!

Essentially, **firmware** itself is a specifically designed piece of software, an advanced state machine, which runs on a localized microprocessor and is designed for one generalized application. Otherwise known as **embedded system firmware,** firmware is essential to the success of a physical electronic device or large scale application. The most common and most important characteristics of an embedded firmware system include:

(1) **Complex Algorithms** – "The operations performed by the [firmware] may be very sophisticated. For example, the [firmware] that controls an automobile engine must perform complicated filtering functions to optimize the performance of the car while minimizing pollution and fuel utilization." [10]

(2) **User Interface** – "Firmware is frequently used to control complex user interfaces that may include multiple menus and many options. The moving maps in Global Positioning System (GPS) navigation are good examples of sophisticated user interfaces." [10]

(3) **Critical Real Time Applications** –

"Many embedded computing systems have to perform in real time – if the data isn't ready by a certain deadline, the system breaks. In some cases, failure to meet a deadline is unsafe and can even endanger lives. In other cases, missing a deadline doesn't create safety problems but does create unhappy customers..." [10] For example, SCSI firmware must accurately control data flow and device arbitration to ensure that no data is lost over the SCSI bus. The firmware on a SCSI controller is a perfect example of a real-time embedded system.

(4)**Multirate Compatibility** – "Not only must operations be completed by deadlines, but many embedded computing systems have several real-time activities going on at the same time. They may simultaneously control some operations that run at slow rates, and others that run at high rates." [10] SCSI firmware, for example, must control a wide variety of multirate devices. A tape-drive, for example, is typically one of the slowest devices on a SCSI bus. On the other hand, an Ultra320 SCSI 15K RPM hard disk is often the fastest. Firmware manages and controls each device, regardless of power and processing speed.

5.3 Firmware Design Challenges

As devices continue to shatter storage performance records, the demand for high-performance controllers powered by fast storage firmware continues to to grow. As a result, engineers continue to face a multitude of challenges in firmware design:

(1)**Complex Testing** – "Exercising an embedded system is generally more difficult than typing in some data. [Engineers] may have to run a real machine in order to generate the proper data. The timing of data is often important, meaning that [engineers] cannot separate the testing of an embedded system from the machine in which it is embedded." [10]

(2)**Limited Observability and Controllability** – "Embedded computing systems usually do not come with keyboards and screens. This makes it more difficult to see what is going on and to affect the system's operation. [Engineers] may be forced to watch the values of electrical signals on the microprocessor bus, for example, to know what is going on inside the system. Moreover, in real-time applications [engineers] may not be

able to easily stop the system to see what is going on inside." [10]

(3) **Restricted Development Environments** – "The development environments for embedded systems are often much more limited than those available for PCs and workstations. [Engineers] generally compile code on one type of machine, such as a PC, and download it onto the embedded system. To debug the [firmware] code, we must usually rely on programs that run on the PC or workstation and then look inside the embedded system." [10]

Clearly, firmware design engineers must overcome a large variety of complex problems when designing firmware for a given application. As previously mentioned, it is critical to the success of the system that the firmware itself is flawless and operates exactly as expected. Otherwise, firmware developers are potentially placing the lives of consumers, and the stability of the global economy at risk.

5.4 Recent Advances

Throughout the firmware development process, few concepts have remained constant as technology continues to advance. Firmware engineers have begun moving away from traditional firmware development techniques and have begun to implement **real-time operating systems** for a variety of applications. Real-time operating system design, or **RTOS**, allows devices to establish and maintain a self sustaining on-board operating system environment. This allows the device to manage processes, by coordinating interprocess communication, process abstraction, and process scheduling to better meet the needs of a device.

Most modern devices contain a microprocessor, an I/O subsystem, memory, and a power supply. The firmware installed on the ROM (Read Only Memory), of a controller initializes the RTOS which then runs within the domain of the controller. Providing a fully functional RTOS environment for a device and its firmware, can increase performance and increase the overall functionality and capabilities of a specific device or application. For example, modern Ultra320 SCSI controllers now contain an on-board self-sustaining RTOS which moderates every aspect of the controllers

operations.

Within the past few years, research and development engineers have also discovered further improvements to storage systems and device firmware. Several engineers have proposed adding intelligence and a small embedded system to disk drives themselves. This would help the disks manage the file system, and improve disk performance.

6 Conclusion

In conclusion, SCSI technology is clearly one of the most versatile and powerful storage standards ever created. When combined with intelligent firmware and semantically smart disk systems, SCSI proves to be a valuable asset to any large storage infrastructure. RAID, multimedia servers, enterprise databases, file-servers, and web-servers are only a small handful of technologies which use the powerful SCSI interface. As storage technology continues to develop and change on a daily basis, one aspect remains clear: SCSI and SCSI firmware applications are critical to the continued success of the extremely popular and profitable storage technology industry.

References

[1] J. Dedek. Ancot's Basics of SCSI: Fourth Edition, Menlo Park, CA, January 2003.

[2] Adaptec Corporation. Let's Talk about SCSI. Server Storage Whitepaper, Milpitas, CA September 2002.

[3] M. Arellano. Ultra320 SCSI: New Technology – Still SCSI. Written for the SCSI Trade Associate by Adaptec Inc., San Francisco, CA, March 2001.

[4] International Business Machines. Reliability Through RAID Technology. In IBM eServer ServeRAID Technology Whitepaper, Research Triangle Park, NC, December 2001.

[5] M. Kolich. HP A7173A PCI-X Dual Channel Ultra320 SCSI Host Bus Adapter Performance Whitepaper. Written for the Hewlett-Packard Company, Cupertino, CA, August 2004.

[6] G. Ananthateerta. PCI-X Dual Port 2GB/sec Fibre Channel Host Bus Adapter Performance Whitepaper. Written for the Hewlett-Packard Company, Cupertino, CA, May 2004.

[7] Fibre Channel Industry Association. Fibre Channel Storage Area Networks, Chicago, IL, August 2001.

[8] J. Hufferd. iSCSI: The Universal Storage Connection, New York, NY, November 2002.

[9] D. Anderson, T. Shanley. PCI System Architecture (4th Edition), New York, NY, June 1999.

[10] W. Wolf. Computers as Components: Principles of Embedded Computing System Design, San Francisco, CA, 2001.