The Architecture and Design of HS-series StorageWorks Array Controllers

by Stephen J. Sicola

ABSTRACT

The HS series of StorageWorks array controllers is a new family of Digital products that includes models for both open systems and systems that use Digital's proprietary buses. The HS-series controllers combine performance, availability, and reliability in total storage subsystem solutions that use industry-standard storage devices. The architecture and design of StorageWorks array controllers represents a balance between the market requirements and the available technology. The engineering trade-offs led to an innovative design that incorporates product features such as a dual-active controller configurations, write-back caching, Parity RAID technology, and SCSI-2 device handling.

INTRODUCTION

The HS series of StorageWorks array controllers, a new addition to Digital's storage subsystem family, supports an open systems environment by allowing the attachment of industry-standard Small Computer Systems Interface (SCSI-2) devices to the controller.[1] Moreover, these controller products yield high availability and high performance. This paper describes the architecture and the design of the HSJ30, HSJ40, HSD30, and HSZ40 StorageWorks array controllers. These controllers interface to host computers by means of existing Digital interconnects, i.e., the Computer Interconnect (CI) and the Digital Storage System Interconnect (DSSI), as well as a SCSI-2 host interconnect to VAX, Alpha, and most other computers in the industry. The paper documents the design and development trade-offs and describes the resulting controllers and their features.

StorageWorks array controllers represent a significant change from Digital's original Hierarchical Storage Controller (HSC) subsystem, the HSC50 controller, which was designed in the late 1970s, and also from other Digital controllers such as the HSC60, HSC70, HSC90, and KDM70 controllers. The StorageWorks controllers discussed in this paper were designed to meet the following product goals:

1. Open systems capability. The goals for open systems capability were to use industry-standard storage devices attached to the controllers and to use an industry-standard host interconnect for one controller model. Using industry-standard devices would provide investment protection for customers because they would

not have to change devices when a new controller was introduced or when they changed controller modules to use a different host interconnect. Industry-standard devices would also reduce overall subsystem cost because of the competitive nature of the storage device industry. The long-term use of both Digital and non-Digital devices was desired to provide a wide variety of device choices for customers. The use of an industry-standard host interconnect would allow StorageWorks controllers to be used with Digital and non-Digital host computers, further expanding the open systems capability. The SCSI-2 interconnect was chosen as the device interface and the host interface over other industry-standard interconnects for cost and strategic reasons.

2. High availability. The goals for high availability included both controller fault tolerance and storage (disk configuration) fault tolerance.

   Controller fault tolerance was achieved by developing a dual-redundant controller configuration in combination with new StorageWorks enclosures that provide redundant power supplies and cooling fans. The goal of the dual-redundant configuration was to have the surviving controller automatically assume control of the failed controller's devices and provide I/O service to them. As a side benefit, such a configuration would provide load balancing of controller resources across shared device ports.

   The storage fault-tolerance goal was to develop firmware support for controller-based redundant array of inexpensive disks (RAID).[2] The initial Parity RAID implementation incorporated the best attributes of RAID levels 3 and 5. The design provided the basis for later implementations of other forms of RAID technology, notably mirroring. Parity RAID supports the goal of storage fault tolerance by providing for continued I/O service from an array of several disks in the event that one disk fails. StorageWorks packaging that provides redundant power supplies and cooling should be combined with the Parity RAID technology to extend storage fault tolerance.

3. High performance. The goals for high performance were to specify controller throughput (the number of I/O operations per unit of time), latency (responsiveness), and data transfer rate (controller bandwidth) for each of the three controller platforms: CI, DSSI, and SCSI. The throughput was specified in the maximum number of read and write requests executed per second. The controllers had to speed up the response time for host I/O operations and thus deliver data with lower command latency than the HSC controllers. StorageWorks controllers had to achieve

the highest possible data transfer rate and were to do so on a common platform.

The platform-specific controller throughput goals were as follows. The initial goal for the CI-to-SCSI controller was 1,100 read requests per second; the long-term goal was 1,500 to 1,700 read requests per second. The initial goal for the DSSI-to-SCSI controller was 800 read requests per second; the long-term goal was 1,300 read requests per second. The initial goal for the SCSI-to-SCSI controller was 1,400 read requests per second; the long-term goal was 2,000 read requests per second. The controller throughput for write operations was slightly lower.

To reduce latency, the controller hardware and firmware implemented controller I/O request caching. Designers initially decided to include 16 to 32 megabytes (MB) of cache memory on a separate optional cache module. Read caching was the beginning goal for the project; however, write-back caching was added during product development as a result of RAID technology investigations.

Another approach to reduce latency was to develop controller-based disk striping, i.e., implement the RAID level 0 technology.[2] Specific goals were to achieve parallel access to all RAID level 0 array members for read and write operations and to streamline firmware to increase RAID level 0 performance.

The Parity RAID performance goal was to overcome the well-known weaknesses of RAID level 3 (i.e., poor transaction throughput) and RAID level 5 (poor small-write performance) and to approach RAID level 0 striped array performance for both small and large read and write requests.[2] A combination of hardware-assisted parity computations and write-back caching helped achieve this goal. Parity calculations in hardware reduced firmware overhead to complete RAID level 5 write operations. Write-back caching minimized the effects of the RAID level 5 small-write penalty.[3] To meet the needs of customers who require high data transfer rates with RAID, RAID level 3-style algorithms must be added for the Parity RAID design.

A common controller processing core had to be architected and designed to meet the performance needs of all the planned StorageWorks controllers (based on host interface capabilities). The platform had to execute the same base firmware, coupling new host interface firmware to the specific platforms. A common platform was believed to ease product development and to maximize reuse of firmware for the same "look and feel" in all products.

OPEN SYSTEMS CAPABILITY

For StorageWorks controllers to enter the open systems market, product designers had to consider the following aspects of open systems in the controller definition: the use of industry-standard device interconnects and industry-standard devices attached to the controller, and the use of industry-standard and Digital host interconnects.

SCSI-2 Device Interconnect

The SCSI-2 interconnect was chosen for the device interconnect because of its wide acceptance in the computer industry. During the controller definition phase, the StorageWorks packaging group was concurrently designing and building storage device enclosures called shelves that would house up to seven 3.5-inch devices or two 5.25-inch devices. These shelves, connected to the controller, would allow a wide variety of SCSI-2 devices to be incorporated and would do so at a low cost because of the widespread use of SCSI-2 as a device interconnect.

StorageWorks controllers were designed to support the following types of SCSI-2 devices:

- Disk -- rotating spindle disk drives and solid-state disks
- Tape -- individual tape drives, tape loaders, and jukeboxes that contain robotic access to multiple drives from a media library
- CD-ROM
- Optical -- individual disks and jukeboxes that contain robotic access to multiple drives from a media library

StorageWorks Controllers in System Environments

The desire to produce a controller with an open system host interconnect was coupled with a commitment to protect the investments of existing Digital customers who currently use CI and DSSI host interconnects. The strategy was to produce CI, DSSI, and SCSI variants of the StorageWorks array controller, all based on a common platform. As in the selection of the device interconnect, the SCSI-2 host interconnect variant was chosen because of its widespread use and low cost.

The controllers for the CI, DSSI, and SCSI interconnects were named the HSJ30/HSJ40, the HSD30, and the HSZ40, respectively. The designations of "30" and "40" represent a code for the number of device ports attached to the controller. The HSJ30 and HSD30

controllers have three device ports each, whereas the HSJ40 and HSZ40 have six device ports each. The number of device ports selected for each controller type was based on (1) the overall capability of the host port interconnect to support the aggregate capability of a number of device ports and (2) the desire to amortize controller cost against as many attached devices as possible.

StorageWorks controller configurations depend on the controller host interface. Marked differences exist in the configurations supported by CI-based OpenVMS VAXcluster configurations, DSSI-based OpenVMS VAXcluster configurations, and SCSI-based configurations in OpenVMS, DEC OSF/1, and other industry system environments. The basic differences are the number of hosts connected and whether or not other storage devices can be on the same host interconnect as the controller and the other hosts.

The CI configuration supports up to 32 nodes per bus. Each node may be either a storage controller (i.e., an HSJ30, an HSJ40, or an HSC device) or a host computer (i.e., a VAX or an Alpha system).

The DSSI configuration supports up to 8 nodes per bus. Each node may be either a storage controller (i.e., an HSD30 or an HSD05), a storage element (e.g., an RF73 device), or a VAX or an Alpha host computer.

The SCSI configuration supports up to 8 targets per bus. The HSZ40 controller, with its standard SCSI-2 host interface, may be connected to Digital Alpha computers (i.e., DEC 3000 and DEC 7000/10000 computers running the DEC OSF/1 operating system), Sun Microsystems computers, Hewlett-Packard computers, and IBM computers. Digital qualifies the HSZ40 controller for operation with additional vendors' systems according to market demand.

HIGH AVAILABILITY

To meet the goals of controller and storage fault tolerance, the designers of StorageWorks controllers developed a number of scenarios from which the controller can be fault tolerant with respect to failures in controller or attached storage components. The first aspect of fault tolerance considered is that of controller fault tolerance; the second is configuration fault tolerance.

Controller Fault Tolerance

Designers achieved controller fault tolerance by investigating the common faults that the controller could tolerate without requiring extreme design measures and incurring high costs. The results of this investigation drove the design of what became the dual-redundant HS-series controller configuration. This

configuration incorporates several patented hardware and firmware features (patent pending).

The following faults can exist within a StorageWorks controller and the attached StorageWorks packaging and do not make host data unavailable:

- Controller failure. In a dual-redundant configuration, if one controller fails, all attached storage devices continue to be served. This is called failover. Failover occurs because the controllers in a dual-redundant configuration share SCSI-2 device ports and therefore access to all attached storage devices. If failover is to be achieved, the surviving controller should not require access to the failed controller.

- Partial memory failure. If portions of the controller buffer and cache memories fail, the controller continues normal operation. Hardware error correction in controller memory, coupled with advanced diagnostic firmware, allows the controller to survive dynamic and static memory failures. In fact, the controller will continue to operate even if a cache module fails.

- Power supply or fan failure. StorageWorks packaging supports dual power supplies and dual fans. HS-series controllers can therefore be configured to survive a failure of either of these components.

- SCSI-2 device port failure. A failure in a single SCSI-2 device port does not cause a controller to fail. The controller continues to operate on the remaining device ports.

The controller must be able to sense the failures just listed in order to notify the host of a fault-tolerant failure and then to continue to operate normally until the fault is repaired. The designers deemed this feature vital to reducing the time during which a controller configuration must operate with a failure present.

Another requirement of fault-tolerant systems is the ability to "hot swap" or "hot plug" components, i.e., to replace components while the system is still operating and thus to not cause the system to shut down during repairs. The designers made the controller and its associated cache module hot swappable. That is, one controller in the dual configuration can be replaced without shutting down the second controller, and the second controller continues to service the requests of the attached hosts. This feature, coupled with the hot-swap capability of StorageWorks devices, creates highly available systems.

Dual-redundant Controller Configuration. Like all StorageWorks

components, HS-series controllers are packaged in StorageWorks shelves. The StorageWorks controller shelf contains a backplane that accommodates one or two controllers and their associated cache modules, as well as SCSI-2 device port connectors. The packaging is common to all system environments. HS-series controllers mounted in a single shelf may be combined in pairs to form a dual-redundant controller configuration (shown in Figure 1) in which both controllers can access the same set of devices.

[Figure 1 (StorageWorks Controllers: System Block Diagram) is not available in ASCII format.]

Figure 2 shows two HS-series controllers installed in a StorageWorks controller shelf in a dual-redundant configuration. Figure 3 shows two dual-redundant controller configurations mounted in a StorageWorks cabinet with several device shelves. The controllers connect to storage devices with cables that emerge from the controller shelf and attach to the device shelves.

[Figure 2 (StorageWorks Controller Shelf) is a photograph and is not available.]

[Figure 3 (StorageWorks Cabinet) is a photograph and is not available.]

The designers had to decide how the dual-redundant controller configuration could achieve high availability through fault tolerance. To meet the high-availability goals, the team addressed the concept of controller failover early in the design process. One fault-tolerant option considered was to run with a "hot-standby" controller that would become operational only if the main controller were to fail. A second option was to design a dual-active controller configuration in which two controllers would operate simultaneously. They would share and concurrently use device port buses (not devices), thus balancing the I/O load from host computers.

Both options allow for direct failover of devices without manual intervention. The hot-standby controller option requires either automatic configuration of the attached devices when the hot-standby controller becomes operational or nonvolatile (i.e., impervious to power loss) shared memory to hold the configuration information. The dual-active controller option requires that each controller have detailed knowledge about the other controller and the device state; it does not require that the controllers share a memory. The designers chose the second option because it provided load balancing and therefore potentially greater performance. However, they faced the challenge of designing a backplane and an interface between the controllers that would achieve the dual-active configuration but would not require a shared memory. The result of the design effort was the StorageWorks controller shelf.

StorageWorks Controller Shelf. The StorageWorks controller shelf is an architected enclosure that allows a pair of StorageWorks controllers and their respective cache memory modules to be placed into the dual-redundant configuration, as shown in Figure 4. A cache module is attached to each controller for performance purposes. The controller shelf contains a backplane that includes intercontroller communication, control lines between the controllers, and shared SCSI-2 device ports. Since the two controllers share SCSI-2 device ports, the design enables continued device availability if one controller fails.

[Figure 4 (StorageWorks Controller Backplane: Controllers in a Dual-redundant Configuration) is not available in ASCII format.]

The backplane contains a direct communication path between the two controllers by means of a serial communication universal asynchronous receiver/transmitter (UART) on each controller. The controllers use this communication link to inform one another about

- Controller initialization status. In a dual-redundant configuration, a controller that is initializing or reinitializing sends information about the process to the other controller.

- "Keep alive" communication. Controllers send keep alive messages to each other at timed intervals. The cessation of communication by one controller causes a failover to occur once the surviving controller has disabled the other controller.

- Configuration information. StorageWorks controllers in a dual-redundant configuration have the same configuration information at all times. When configuration information is entered into one controller, that controller sends the new information to the other controller. Each controller stores this information in a controller-resident nonvolatile memory. If one controller fails, the surviving controller continues to serve the failed controller's devices to host computers, thus obviating shared memory access. The controller resolves any discrepancies by using the newest information.

- Synchronized operations between controllers. Specific firmware components within a controller can communicate with the other controller to synchronize special events between the hardware on both controllers. Some examples of these special events are SCSI bus resets, cache state changes, and diagnostic tests.

The other signals on the backplane pertain to the current state of the configuration within the controller shelf and to specific control lines that determine the operation of the dual-redundant

controller configuration. The backplane state and control signals include

- Status about the presence of a controller's cache module. Each controller can sense the presence or absence of its cache to set up for cache diagnostics and cache operations.

- Status about the presence of a second controller, which indicates a dual-redundant configuration. Each controller can sense the presence or absence of the other controller in a dual-redundant configuration. This assists in controller setup of dual-controller operation as well as general controller initialization of the dual-redundant configuration.

- Status about the presence of the second controller's cache. Each controller can sense the presence or absence of the other controller's cache for dual-controller setup purposes.

- The "KILL" signal. In a dual-redundant configuration, each controller has the capability to use the KILL control signal to cause a hardware reset of the other controller. However, once one controller asserts the KILL signal, the other controller loses the capability. The KILL signal ensures that a failed or failing controller will not create the possibility of data corruption to or from attached storage devices.

  The KILL signal denotes that failover to the surviving controller should occur. A controller asserts the KILL signal when the other controller sends a message that it is failing or when normally scheduled keep alive communication from the other controller ceases. The KILL signal is also used when both controllers decide to reset one another, e.g., when the communication path has failed.

  The designers had to ensure that only one controller could succeed in the KILL operation, i.e., that no window existed where both controllers could use the KILL signal. After firmware on a controller asserts the KILL signal to its dual-redundant partner, the KILL recognition circuitry within the controller that asserted the signal is disabled. The probability of true simultaneous KILL signal assertion was estimated at 10**-20, based on hardware timing and the possibility of synchronous dual-controller operation.

- The cache LOCK signals. The cache LOCK signals control access to the cache modules. The dual-controller architecture had to prevent one controller from gaining access to a cache module that was being used by the other

controller and had to allow the surviving controller to access the failed controller's cache. The access control had to be implemented in either firmware or hardware.

A firmware solution would involve a software locking mechanism that the controllers would recognize and cooperatively use to limit cache module access to the associated controller. This method had an inherent problem: firmware alone may not prevent inadvertent cache access by a failing controller. The designers therefore had to implement a hardware lock mechanism to prevent such inadvertent access.

The hardware lock mechanism was implemented with control signals from each controller. The signals are utilized by hardware to prevent inadvertent access and by firmware to limit cache module access to the associated controller. From each controller, the designers implemented two LOCK signals that extend individually to each cache module and are visible to both controllers. The cache LOCK signals are illustrated in Figure 4.

The LOCK signals allow a controller to achieve exclusive access to a specific cache module to ensure data integrity. LOCK signals from a controller that has been "killed" by its dual-redundant partner are reset so that the partner may fail over any unwritten cache data in the write-back cache.

Failover. Controller failover is a feature of the dual-redundant configuration for StorageWorks controllers. Failover of a controller's devices and cache to the other controller occurs when

- A controller fails to send the keep alive message. This situation can occur because of a controller failure in the dual UART (DUART) or in any other non-fault-tolerant portion of the controller module. In this scenario, the surviving controller uses the KILL signal to disable the other controller, communicates to the failed controller's devices, and then serves the failed controller's devices to hosts.

  The failover of a controller's cache occurs only if write-back caching was in use before the controller failure was detected. In this case, the surviving controller uses the failed controller's cache to write any previously unwritten data to the failed controller's disks before serving these disks to hosts. When the surviving controller has written the data to disks (i.e., flushed the data), it releases the cache to await the failed controller's return to the dual-redundant configuration through reinitialization or replacement.

- A customer desires to change the load balance of one or more devices attached to one controller to the other controller. This specialized use of failover provides a load-balancing feature that the designers considered valuable in a dual-active controller configuration. Load balancing is static in the controller, i.e., devices are allocated to one controller or to the other, not shared dynamically. To change allocation, the system manager must change the preferred path of device access. This is accomplished by accessing either the maintenance port of the controller or the configuration firmware through the host interface (e.g., the diagnostics and utilities protocol for CI and DSSI systems).

- The cache module battery is low or has failed. This special case of failover is used in conjunction with Parity RAID operations for the reasons described in the Parity RAID technology portion of the following section. The main issue is to continue to provide as much data protection as possible for Parity RAID disk configurations when the battery on the write-back cache is low or bad.

- The controller is unable to communicate with the devices to which it is currently allocated for host operations. This situation can occur if a device port on a controller fails.

Storage Fault Tolerance

Storage fault tolerance is achieved by ensuring that power or environmental factors do not cause devices to be unavailable for host access and by using firmware to prevent a device failure from affecting host accessibility.

Environmental Factors. StorageWorks enclosures provide for optional redundant power supplies and cooling fans to prevent power or fan failures from making devices unavailable. The SCSI-2 cables that connect device shelves to the controller shelf carry extra signals to alert the controller to power supply or fan failures so that these conditions may be reported to host computers. The enclosures must contain light-emitting diodes (LEDs) to allow a controller to identify failed devices. In addition, a cache module can fail, and the controller will continue to operate.

RAID Technology. To prevent a device failure from affecting host access to data, the designers introduced a combined firmware and hardware implementation of RAID technology.[2] The designers had to decide which RAID level to choose and what type of hardware

(if any) was required for the implementation.

The designers considered RAID levels 1 through 5 as options for solving the problem of disk failures that affect data availability. RAID level 1 (disk mirroring, which is depicted in Figure 5a) was rejected because of its higher cost, i.e., the cost of parts to implement the mirroring.[2] Each disk to be protected implies an inherent cost of one additional housed, powered, and attached disk. RAID level 1 was also discounted because software-based solutions were available for many of the hosts for which the HS-series controllers were initially targeted.

[Figure 5 (Mapping for RAID Levels 1 through 5) is not available in ASCII format.]

RAID levels 2 through 4, illustrated in Figures 5b through 5d, were rejected because they do not provide good performance over the entire range of I/O workloads for which the controllers were targeted.[4] In general, these RAID levels provide high, single-stream data transfer rates but relatively poor transaction processing performance.

RAID level 5 in its pure form was rejected because of its poor write performance, especially for small write operations.[2] The designers ultimately chose RAID level 5 data mapping (i.e., data striping with interleaved parity, as illustrated in Figure 5e) coupled with dynamic update algorithms and write-back caching to overcome the small-write penalty. This implementation is called Parity RAID.

An HS-series Parity RAID array appears to hosts as an economical, fault-tolerant virtual disk unit. A Parity RAID virtual disk unit with a storage capacity equivalent to that of n disks requires n + 1 physical disks to implement. Data and parity are distributed (striped) across all disk members in the array, primarily to equalize the overhead associated with processing concurrent small write requests.[2]

If a disk in a Parity RAID array fails, its data can be recovered by reading the corresponding blocks on the surviving disk members and performing a parity comparison (using exclusive-OR [XOR] operations on data from other members). Figure 6 illustrates this regeneration of data.[4]

[Figure 6 (Regenerating Data in a Parity RAID Array with a Failed Member Disk) in not available in ASCII format.]

HS-series controller developers overcame a number of challenges. Foremost among them was the elimination of the RAID level 5 write hole. Parity RAID with its RAID level 5 striping is susceptible to the RAID level 5 write hole. A write hole is data corruption that occurs when all the following events take place.

- A controller failure occurs with a host's write request outstanding.

- Either the updated data or the updated parity for the host's write request has been written to disk but not both.

- A failure of a different disk occurs after the controller failure has been repaired.

To eliminate this write hole, designers had to develop a method of preserving information about ongoing RAID write operations across power failures such that it could be conveyed between partner controllers in a dual-redundant configuration.

Designers decided to use nonvolatile caching of RAID write operations in progress.[5] Three alternatives were considered:

1. An uninterruptible power supply (UPS) for the controller, cache, and all attached disk devices. This choice was deemed to be a costly and unwieldy solution because of the range of possible requirements. The indeterminate amount of data in the cache to be written and the power consumption of a wide variety of devices would necessitate a very large backup power source to ensure enough time for all cached write data to be written to attached devices.

2. A battery in the controller and device enclosures (i.e., shelves) to allow enough time for the writing of cached data in the event of a power failure. StorageWorks device enclosures can accommodate either redundant power supplies or one power supply and one backup battery for configurations that do not require redundancy. There is no provision for both redundant power supplies and a battery. This conflict between fault-tolerant StorageWorks shelf configurations with dual power supplies and the desire to add a battery for write-back caching was unacceptable to the designers because of the loss of power redundancy to gain write-back cache integrity.

3. A controller-based nonvolatile cache. The options for controller-based nonvolatile caching included (a) a battery-protected cache for write data, (b) an additional nonvolatile random-access memory (NVRAM) on the controller to journal RAID writes, and (c) a battery-protected cache for both read and write data.

   With a battery-protected write cache, data must be copied if it is to be cached for subsequent read requests. Designers deemed the potential performance penalty unacceptable.

Using controller NVRAM as a RAID write journal not only closes the RAID level 5 write hole but also provides a small write cache for data. This approach also requires data copying and creates an NVRAM access problem for the surviving controller to the failed controller NVRAM to resolve any outstanding RAID write requests.

The third controller-based nonvolatile cache option, to battery-backup the entire cache, solved the copy issue of option 3a and the failover issue of option 3b.

The designers chose option 3c, the battery-protected read/write cache module. A totally nonvolatile cache had the advantage of not requiring write-cache flushing, i.e., copying data between the write cache and the read cache after the write data has been written to devices. Segregated cache approaches (part nonvolatile, part volatile) would have required either copying or discarding data after write-cache flushing. Such approaches would have resulted in a loss of part of the value of using the caching algorithm by allowing only read caching of read data already read. Another benefit of a nonvolatile read/write cache is failover of the cache module in the event of a controller failure. This further reduces the risk associated with a RAID level 5 write hole because unwritten write operations to Parity RAID arrays can be completed by the surviving controller after failover.

To achieve a total nonvolatile cache, the designers opted for the battery solution, using two 3-by-5-by-0.125-inch lead-acid batteries that supply up to 100 hours of battery backup for a 32-MB cache module. The batteries eliminated the need for a special (and costly) nonvolatile memory write cache and allowed data hold-up after power failure. The designers chose lead-acid batteries over NiCAD batteries because of their steady power retention and output over time. This option protects against most major power outages (of five minutes to five days) and all minor power outages (of less than five minutes). Most power outages (according to studies within Digital) last less than five minutes and are handled in the same manner as major outages, that is, by flushing write data immediately after power has been restored to the controller configuration. Battery status is provided to firmware, which uses this information to make policy decisions about RAID arrays and other virtual disk units with write-back caching enabled.

For an HS-series controller to support Parity RAID, its cache module must have batteries installed. The batteries make the cache nonvolatile and enable the algorithms that close the RAID level 5 write hole and permit the use of the write-back cache as a performance assist, both vital for proper Parity RAID operation. If the controller firmware detects a low- or bad-battery condition, write-back caching is disabled. The controller that detects the condition tries to fail over Parity RAID units to the other controller in the dual-redundant

configuration to keep the units available to hosts. If the other controller cache module has a low- or bad-battery condition, the Parity RAID unit is made unavailable to hosts to protect against data loss or data corruption should a power failure occur. When the batteries are no longer low, Parity RAID units are again made available to hosts. Any Parity RAID units that had been failed over to the other controller would fail back, i.e., return, to the controller that originally controlled them. The module hardware and firmware support read caching regardless of the presence of a battery.

After solving the RAID level 5 write-hole problem, the designers decided to automate the Parity RAID recovery process wherever possible. This goal was adopted so that customers would not have to understand the technology details in order to use the technology in the event of a failure. StorageWorks controller firmware developers, therefore, chose to add automatic Parity RAID management features rather than require manual intervention after failures. Controller-based automatic array management is superior to manual techniques because the controller has the best visibility into array problems and can best manage any situation given proper guidelines for operation.

An important feature of Parity RAID is the ability to automatically bring a predesignated disk into service to restore data protection as quickly as possible when a failure occurs. Other controllers in the industry mandate configurations with a hot-standby disk, i.e., a spare disk, dedicated to each Parity RAID unit. A hot-standby disk is powered and ready for firmware use if an active member disk of its Parity RAID unit fails. This concept is potentially wasteful since the probability that multiple Parity RAID units will have simultaneous single-member disk failures is low. The designers therefore had the options of making spare disks available on a per-Parity RAID unit basis or having a pool of spares, i.e., a spare set, that any configured Parity RAID unit could access. The designers chose the pool of spares option because it was simpler to implement and less costly for the customer, and it offered the opportunity to add selection criteria for spare set usage and thus maximize either performance or capacity efficiency.

To allow more flexibility in choosing spare set members, designers made two spare selection options available: best fit and best performance. The best fit option allows for disk devices of different sizes to compose the pool of spares. When a spare disk is needed after a member of a Parity RAID unit fails, the device with the best fit, that is, whose size most closely matches that of the failed disk (typically of the same size but possibly of greater capacity), is chosen. The best performance option can reduce the need for physical reconfiguration after a spare is utilized if a spare attached to the same device port as the failed array member can be allocated. The best performance option maintains operational parallelism by spreading array members across the controller device ports after a failure and

subsequent spare utilization.

These features allow automatic sparing of failed devices in Parity RAID units and automatic reconstruction after a spare device has been added to the Parity RAID unit.[6] Furthermore, any drive of at least the size of the smallest member of a Parity RAID unit is a candidate spare, which reduces the need for like devices to be used as spares. (Typically, however, spare set members are like members.)

A Parity RAID unit with a failed member will become unavailable and lose data if a second failure occurs. The HS-series automatic sparing feature reduces the window of possible data loss to the time it takes to reconstruct one Parity RAID unit. Mean time between data loss (MTBDL) is a combination of the mean time to repair (MTTR) and the failure rate of a second device in a Parity RAID unit. The automatic sparing feature reduces the MTTR and thus increases the MTBDL. Data loss can occur only in the highly unlikely event that a failure occurs in another RAID set member before the reconstruction completes on the chosen spare. During Parity RAID reconstruction, the controller immediately makes the host read or write request to the reconstructing member redundant by updating parity and data on the spare after the host read or write operation. Parity RAID firmware quickly reconstructs the rest of the Parity RAID unit as a background task in the controller. If the member that is being reconstructed happens to fail and other spare set members remain, reconstruction on a new spare begins immediately, further reducing the probability of data loss.

Parity RAID member disk failure declaration is key to the efficient use of spares and to preventing unwarranted use of spares. If a write command to a RAID set member fails, RAID firmware assumes that the SCSI-2 disk drive has exhausted all internal methods to recover from the error. SCSI-2 disk drives automatically perform bad block replacement on write operations as long as there is space available within the disk drive revector area (the area where spare data blocks can be mapped to a failed block). The designers chose this method over more complex retry algorithms that might encounter intermittent failure scenarios. Empirical information related to previous storage devices showed that localized write failures are rare and that this strategy was sound for the majority of disk access failures.

When read failures occur, data is regenerated from the remaining array members, and a forced bad block replacement is performed. Metadata on the disk is used to perform this function atomically, that is, to perform the bad block replacement even if a power failure occurs during the replacement.[7] If the disk cannot replace the block, then the Parity RAID member disk is failed out and an attempt is made to choose a suitable spare from the spare set. If no spare is available, the Parity RAID unit operates in reduced mode, regenerating data from the failed member when

requested by the hosts.[4]

Parity RAID firmware uses the metadata to detect a loss of data due to catastrophic cache failure, inappropriate device removal, or cache replacement without prior flush of write data. The designers considered it important that the controller firmware be able to detect these data loss conditions and report them to the host computers.

The failure scenarios just described occur infrequently, and the StorageWorks Parity RAID firmware is able to recover after such failures. During a typical normal operation, the main challenge for Parity RAID firmware is to achieve a high level of performance during write operations and a high level of controller performance in general.

HIGH PERFORMANCE

As discussed earlier, the performance goals for the StorageWorks controllers were in the areas of throughput and latency. Bandwidth goals were based on the architecture and technology of the controller platform. The designers met the performance goals by producing a controller that had a low command overhead and that processed requests with a high degree of parallelism. The firmware design achieves low overhead by means of the algorithms running on the controller, coupled with RAID and caching technology. The hardware design that allows for low command overhead and high data transfer rates (bandwidth) is discussed in the section Common Hardware Platform.

Command Processing

The StorageWorks designers maximized the number of requests the controller can process per second by reducing the command processing latency within the controller firmware. The firmware utilizes controller-based caching and also streamlined command processing that allows multiple outstanding commands to be present in the controller.

To meet the varying needs of customer applications, the controller supports both Parity RAID and RAID level 0. The designers decided to include RAID level 0 as a controller feature because of its inherent parallelism, even though RAID level 0 is not fault tolerant without external redundancy.

StorageWorks controllers service all device types, but the designers felt that disk device performance was the key metric for determining how well a controller supports RAID technology. The controller firmware was designed to efficiently control individual devices (commonly referred to as "just a bunch of devices" [JBOD]) and Parity RAID, prioritizing requests to each of the SCSI-2 device ports on the controller. StorageWorks

controllers comply with SCSI-2 protocols and perform advanced SCSI-2 functions, such as tagged queuing to all attached SCSI-2 storage devices for greater performance.[1]

Discussions of the RAID level 0 technology and of how the designers used Parity RAID technology to overcome some of the performance bottlenecks follow.

Striping -- RAID Level 0

Digital has used RAID level 0 technology, that is, striping, in systems for at least five years, in its host computers using software as well as in its controllers. Striping allows a set of disks to be treated as one virtual unit. Device data blocks are interleaved in strips, i.e., contiguous sets of blocks, across all disks, which provides high-speed parallel data access. Figure 7 illustrates the mapping for a RAID level 0 array.[4] Since a striped disk unit inherently lacks fault tolerance (i.e., if one device in the set fails, data is lost), controller-based striping is typically used in conjunction with host-based mirroring or in cases where data can be easily reproduced. Stripe sets achieve high performance because of the potential for parallelism by means of the device and data organization. The key difference between RAID level 0 and RAID levels 3 and higher is that striping results in the interdependence of data written to different devices.

[Figure 7 (Mapping for a RAID Level 0 Array) is not available in ASCII format.]

Controller Caching

Caching with StorageWorks controllers was originally read caching only. When the designers decided to use a nonvolatile cache to eliminate the RAID level 5 write hole, write-back caching on the controller became a viable option.

Controller Read Caching. Read caching was intended to reduce latency in the controller by minimizing the need to access devices continuously for repeated host read requests to the same locations on attached devices. Read caching must also address the issue of how to handle write data for later use. The design could have incorporated on-board controller memory to hold write data. However, this would require copying the write data to the read cache after the write data had been written to the devices and would result in inefficient use of the read cache. Therefore, the designers decided to have the read cache serve as a write-through cache as well. Read caching would be disabled/enabled per logical unit presented to the host instead of having global read caching, where a logical unit is one or more devices configured as one virtual device. Thus, customers can specify for which virtual

devices they want caching enabled.

The read and write-through caching firmware receives requests from other parts of the controller firmware (e.g., a host port, a device port, and the Parity RAID firmware) and proceeds as follows.

For reads requests, the caching firmware provides

1. The data pointers to the cached request, i.e., the cache hit

2. The data pointers for part of the request, i.e., a partial cache hit, which means that the remaining data must be retrieved from the device or devices being requested

3. A status response of cache miss, which means that storage management must retrieve the data from the device or devices being requested

For write requests, the caching firmware offers the cache manager data from the request. The cache manager places the previous data pointers into the read cache tables after the data is written through the cache to the devices. Firmware tells the device port hardware where to find write data, and the port hardware transfers the data.

Read caching in the first version of the controller firmware allowed the controller to achieve the initial throughput goals across the three controller platforms. The current software version, version 2.0, was shipped in October 1994 and exhibits even greater throughput performance. Table 1 shows the I/O performance for the three StorageWorks controller platforms with read caching enabled.

Table 1 StorageWorks Controller I/O Performance with Read Caching

Controller	Read Requests per Second	Write Requests per Second
HSJ30/HSJ40	1,550	1,050
HSD30	1,000	800
HSZ40	2,250	1,500

Write-back Caching -- Performance Aspects. As noted earlier, when the cache module contains batteries, the memory is nonvolatile for up to 100 hours. The StorageWorks controller can use the nonvolatile cache to increase controller performance by reducing latency for write request Parity RAID performance to a level similar to that of RAID level 0 (simple disk striping). The

controller can also utilize the write-back cache to reduce the latency of JBOD and RAID level 0 disk configurations. As with read caching, write-back caching is disabled/enabled per logical unit.

The write-back caching firmware controls the usage of both a surviving controller's cache module and a failed controller's cache module. When it receives a write request, the controller places the data in the cache, marks the request as complete, and writes the data based on internal controller firmware policies (write-back caching). To provide greater performance during Parity RAID operations than simple write-back caching could provide, the write-back cache firmware is also tied to the Parity RAID firmware.

In addition to dealing with the continual problem of controller latency on write commands, designers had to overcome the RAID level 5 small-write penalty with parity updates to RAID set members. Write-back caching was chosen over RAID level 3 hardware assists as a Parity RAID strategy because RAID level 3 does not provide a wide range of benefits for all customer workloads. Write-back caching provides latency reductions for RAID and non-RAID configurations. Write-back caching also increases write-request throughput. For example, the published performance numbers for write throughput with write-back caching enabled in version 2.0 firmware appear in Table 2.

Table 2 StorageWorks Controller Write Request Throughput with Write-back Caching

Controller	Write Requests per Second
HSJ30/HSJ40	1,350
HSD30	900
HSZ40	1,850

The use of write-back caching resulted in a 20 to 30 percent increase in write throughput for all platforms as compared to write-through caching. Before discussing the effect of write-back caching on latency for individual devices and for Parity RAID arrays, the paper describes how the write-back cache firmware was designed and tied directly to Parity RAID firmware.

The features chosen for write-back caching were extensively benchmarked against data integrity issues. The addition of settable timers allows customers to flush write data destined for devices that are idle for a specific length of time. To reduce the number of read/modify/writes required to update parity on small write operations, designers tied flush algorithms to RAID. Flush algorithms for write-back caching are vital to customer data integrity and to latency reduction. The flush algorithms

actually allow Parity RAID to simulate RAID level 3 operations because of the nonvolatile write-back cache.

As mentioned earlier, Parity RAID configurations suffer a penalty on small write operations that includes a series of read and write operations and XOR operations on blocks of data to update RAID parity. The write-back cache firmware was designed with specific attributes to enhance Parity RAID write operations in general, and not just to enhance small write operations. The designers intentionally chose to overcome both the small-write penalty and the inherent lack of high bandwidth that Parity RAID delivers.

The nonvolatile write-back cache module afforded the firmware designers more choices for Parity RAID write request processing and data flush algorithms. The designers pursued techniques to speed up all write operations by performing write aggregations (i.e., combining data from multiple write requests and read cache data) in three dimensions:

1. Contiguous aggregation, in which the firmware looks for consecutive block requests and ties them together into one device request, thus eliminating separate device requests.

2. Vertical aggregation, in which the firmware can detect two write operations to the same block, thus eliminating one write operation.

3. Horizontal aggregation (for Parity RAID operations only). This type of aggregation occurs when all data blocks within a Parity RAID strip are present in the write-back cache. In such cases, the firmware can write to all RAID set members at once, in combination with the FX chip (discussed later in this section) on-the-fly hardware XOR operations during the RAID set member writes. The original request can cause horizontal aggregation to take place if all blocks within a strip are part of the first write request. The firmware can also perform horizontal aggregation after processing several write requests. In this way, the parity write operation directly follows the data write operations. Horizontal write aggregation potentially cuts physical device access in half when compared to normal RAID write operations that require data members to be read.[2,8] The result is pseudo-RAID level 3 operation, because the write-back cache is combined with the horizontal aggregation cache policy.

The performance gain for individual disks and for Parity RAID arrays from using write-back caching is dramatic, resulting in higher write throughput and low latency. The write-back cache actually smoothes out differences in performance that are typical of workloads that have different read/write ratios, whether or not Parity RAID is utilized.

Figure 8 shows the relative latency for a controller with and without write-back caching enabled. The configurations tested comprised individual devices and Parity RAID units (in a five-plus-one configuration). The performance measurements were taken from a version 2.0 HSJ40 array controller.

[Figure 8 (HSJ40 Array Latency Comparisons) is not available in ASCII format.]

Workload 1 has a read/write ratio of 70/30, i.e., 70 percent of the requests were read requests and 30 percent were write requests. Workload 2 has a read/write ratio of 84/16. Workload 3 has a ratio of 20/80. In all workloads, the latency for individual devices and for Parity RAID units is lower when write-back caching is enabled than when only read caching is enabled. In fact, when write operations dominate the I/O mix, latency for Parity RAID units is the same as for the workloads in which read operations are predominant!

RAID/Compare Hardware

StorageWorks controllers contain a hardware Parity RAID and data compare accelerator called FX, a gate array that performs on-the-fly XOR operations on data buffers. Parity RAID and data compare firmware use this gate array to accelerate Parity RAID parity calculations and host data compare requests. The FX chip is programmed to (1) observe the bus, (2) "snoop" the bus for specific addresses, (3) perform the XOR operation to compare the associated data on-the-fly with data in a private memory called XBUF memory, and (4) write the data back into the XBUF memory.

XOR operations can take place as data is moving from buffer or cache memory to device ports or vice versa. The FX can also perform direct memory access (DMA) operations to move the contents of buffer or cache memory to or from XBUF memory.

The designers determined that hardware acceleration of XOR operations for Parity RAID firmware would speed up RAID parity calculations and thus further improve Parity RAID latency and throughput. The firmware also supports FX compare operations, which eliminates the need for SCSI-2 devices that have implemented compare commands and for speeding up compare requests from hosts.

Common Hardware Platform

To produce a high-performance controller in all three performance dimensions -- latency, throughput, and data transfer rate -- the designers of StorageWorks controllers faced the challenge of creating a new controller architecture and using new technology. In addition, they had to do so at a reasonable cost.

Although each has its own specific host interface hardware, the CI, DSSI, and SCSI controller variants share a common hardware core. Commonality was desired to control the development costs and schedules for such large engineering projects. To deliver high performance and commonality, the designers investigated several controller architecture alternatives. The first architecture considered was similar to Digital's HSC50-95 controller, incorporating similar bus structures, processing elements, and memories, but newer technology. Figure 9 shows the HSC architecture.[9]

[Figure 9 (Block Diagram of the HSC Architecture) is not available in ASCII format.]

The HSC architecture is a true multiprocessor system. It contains a private memory for its policy processor, which manages the work that is coming from the host port interface and queues this work to the device interface modules. Data then flows between the host port and device modules to and from hosts. The modules have two interfaces (buses) for access to command processing and data movement. These buses are called the control memory interface and the data memory interface. The policy processor queues work to the host port and device modules through the control memory interface, and then the modules process the data over the data memory interface.

Using this architecture would have been too expensive. The controller cost had to be competitive with other products in the industry, most of which currently cost considerably less than the HSC controller. The HSC bus architecture required three different memory interfaces, which would require three different, potentially large memories. The designers had to pursue other options that met the cost goals but did not significantly reduce performance. They considered single internal bus architectures, but during simulation, these options were unable to meet either the initial or the long-term cost goals.

Figure 10 shows the controller architecture option that became the common hardware base for StorageWorks controllers. This architecture contains three buses and two memories. A third small memory is used for Parity RAID and data compare operations but does not drastically increase controller cost. The architectural design allows the policy processor to access one memory while a device or host port processor accesses the other memory.

[Figure 10 (HSx40 Controller Architecture) is not available in ASCII format.]

The architecture achieves a lower overall cost than the HSC architecture yet achieves similar performance. The new architecture, with fewer memories, does not significantly reduce the performance, while the newer technology chosen to implement the controller enhances performance. The bus bandwidth of the new

controller is much higher than that of the HSC controller. Consequently, a more cost-effective solution that uses a less-costly architecture can attain similar to better performance.

The extreme integration of hardware to the very large-scale integration (VLSI) level allowed for a much smaller enclosure than that of the HSC controller, even with a dual-redundant controller configuration (see Figure 3). A StorageWorks dual-controller configuration measures 56.5 by 20.9 by 43.2 centimeters (22 by 8 by 17 inches), which is approximately one-tenth the size of the HSC controller.

Common Controller Platform. The common controller platform consists of the controller without the associated host port. The common core of hardware consists of the policy processor hardware, the SCSI-2 device port hardware, and the cache module. The controller-specific host port interface hardware includes either the CI, the DSSI, or the SCSI interface.

Policy Processor Hardware. The StorageWorks controller policy processor is Intel's 25-MHz i960CA microprocessor, which contains an internal instruction cache and is augmented by a secondary cache external to the processor. The secondary cache relieves the potential bottleneck created by shared memory between the policy processor and host/device port processors.

The designers had to make trade-offs in two areas: the memory speed/cost and the number of buses. After simulation, the external instruction and data cache showed a significant performance improvement, given the chosen shared-memory architecture. The cache covers the first 2 MB of buffer memory, where policy processor instructions and local processor data structures reside and where most of the performance gain for the policy processor would be achieved.

The policy processor uses the IBUS exclusively to fetch instructions and to access the program storage card, the NVRAM, the DUART, and the timers.

Program Storage. StorageWorks firmware is contained on a removable program card for quick code upgrades and to eliminate the need for a boot read-only memory (ROM) on the controller. The program card is a PCMCIA, 2-MB flash electrically erasable, programmable, read-only memory (EEPROM) card that contains the firmware image. Designers chose the PCMCIA card to facilitate code updates in the field, where host-based downline loading of firmware was not supported. Although the PCMCIA card cost more than EEPROM chips attached to the module, the designers felt that the benefits of such a design outweighed the additional cost.

On each initialization, the controller reads the firmware image on the program card and copies the image to the shared memory. The firmware executes from the shared buffer memory.

Dual UART (DUART). The DUART is used for two reasons:

1. Maintenance terminal connection. The maintenance terminal is a means of entering controller system management commands (with the command line interpreter, which is the user interface for controller configuration management) and is also a status and error reporting interface. Designers made extensive use of this interface for debugging controller hardware and firmware. Use of the maintenance terminal connection is optional. The interface remains on the controller so that users can direct controller management and status reporting, if desired.

2. Failover communication between two controllers in a dual-redundant configuration. The communication path is used to share configuration and status information between the controllers.

Shared Buffer and Cache Memory. The dynamic random-access memory (DRAM) buffer (or shared memory) has at its heart the dynamic RAM and arbitration (DRAB) chip. This chip supports the buffer and cache memory accesses from the policy processor and from the host and device ports. The data transfer rate supported by the shared memory is approximately 35 megabytes per second (MB/s).

The DRAB chip contains error-correcting code (ECC) hardware to correct single-bit memory, to detect multibit errors, and to check and generate bus parity. This feature allows the controller to survive partial memory failures, which was a fault-tolerant goal for the controller.

The decision to use DRAM chips in the memory design rather than static random-access memory (SRAM) chips led to the use of ECC. DRAMs were chosen because of their cost and power savings over equivalent SRAM. However, because the designers expected large amounts of DRAM (as much as 40 MB) to be present on a controller and its associated cache module, the statistical error probabilities were high enough to warrant the use of ECC on the memory. The combination of DRAM and ECC was less costly than an equivalent amount of more reliable SRAM. The use of parity on the buses is a standard feature in all StorageWorks controllers. The bus parity feature provides further error detection capability outside the bounds of the memory because it covers the path from memory to or from external host or device interfaces.

The DRAB chip also controls access to the cache module in conjunction with slave DRAB chips on the cache module associated

with the controller. These DRAB chips provide refresh signals for the DRAM buffer or cache memory that they control; whereas, the master DRAB on the controller module provides arbitration for cache accesses that originate from the various sources on the controller module. Slave DRAB chips can also be accessed by the dual-redundant partner controller, depending on the two controller LOCK signal states.

The controller firmware uses 8 MB of shared buffer memory to execute the program image, to hold the firmware data structures, and to read and write-through cache data (if no cache module is present). The i960CA policy processor and the host and device data processing elements on the NBUS can all access buffer memory.

Cache Memory. Each cache memory module contains one slave DRAB chip and 16 or 32 MB of DRAM, and also two ports into the module (one from each controller) for use in failover. Each cache module optionally contains batteries to supply power to the DRAM chips in the event of power failure for write-back caching and Parity RAID use. The cache modules are interchangeable between controller types.

Parity RAID XOR and Compare Hardware. The Parity RAID XOR and compare hardware consists of the FX gate array and 256 kilobytes (KB) of fast SRAM. The FX allows concurrent access by SCSI-2 device port hardware and the policy processor. The FX compares the XOR of a data buffer (512 bytes of data) that is entering or exiting an attached device with the XOR buffers in the fast SRAM. The policy processor uses the FX to perform compare operations at the request of a host and perform DMA operations to move data to and from memories. This hardware is common across all the controller platforms for Parity RAID and compare firmware.

SCSI-2 Device Port Hardware. The device ports (three or six, depending on the controller model) are controlled by Symbios Logic (the former NCR Microelectronic Products Division of AT&T Global Information Solutions Company) 53C710 SCSI-2 processor chips. The SCSI-2 processor chips reside on the NBUS and access the shared-memory cache for data structure and data buffer access. These processors receive their work from data structures in buffer memory and perform commands on their specific SCSI-2 bus for read or write operations.

The Symbios Logic chip provided the most processing power, when compared to the other chips available when the controllers were designed. The designers felt that direct control of SCSI-2 interfaces by the policy processor or a separate processor was too costly in terms of processor utilization and capital expense. The Symbios Logic chips do require some policy processor utilization, but the designers considered this acceptable because

high-performance architectural features in the policy processor hardware compensated for the extra processor utilization.

The SCSI-2 device port supports the SCSI fast, single-ended, 8-bit interface.[1] The data transfer rate supported by this interface is 10 MB/s.

Host Port Hardware. The host port hardware is either a CI, a DSSI, or a SCSI interface implemented with gate arrays or Symbios Logic 53C720 SCSI-2 processors. The host port hardware, the only noncommon hardware on a StorageWorks controller, requires a separate platform to support each host interface.

The CI interface is made up of a gate array and CI interface hardware that performs DMA write or read operations from shared memory or cache memory over the NBUS. The maximum data transfer rate supported by the CI hardware is approximately 8 MB/s.

The DSSI interface utilizes a Symbios Logic 53C720 chip coupled with a gate array and DSSI drivers to receive and transmit data to or from the DSSI bus. The DSSI interface is 8 bits wide, and the maximum data transfer rate supported by the DSSI hardware is 4.5 MB/s.

The SCSI interface also uses a Symbios Logic 53C720 chip coupled with differential drivers to provide a SCSI-2, fast-wide (i.e., 16-bit) differential interface to hosts. The maximum data transfer rate supported by the SCSI-2 interface is 20 MB/s for fast-wide operations.

Table 3 shows the current (version 2.0) maximum measured (at the host) data transfer rate performance numbers for StorageWorks controllers.

Table 3 SCSI-2 Host Interface Performance

Controller	Read Data Transfer Rate (Megabytes per Second)	Write Data Transfer Rate (Megabytes per Second)
HSJ30/HSJ40*	6.7	4.4
HSD30	3.2	2.8
HSZ40**	14	8.0

* In a multihost environment
** Measured for the HSZ40-B controller

SUMMARY

The StorageWorks HS-series array controllers were designed to meet the storage subsystem needs of both Digital and non-Digital systems, thereby entering the world of open systems. The

architecture for the HSJ30, HSJ40, HSD30, and HSZ40 controllers has achieved the initial project goals and provides

1. Open systems capability. A SCSI-2 device interface allows many types of disk, tape, and optical devices to be attached to the HSJ30, HSJ40, and HSD30 controllers. The HSZ40 controller, which is currently a disk-only controller, provides a SCSI-2 host interface that allows the controller to be attached to Digital and non-Digital computers.

2. High availability. Controller fault tolerance and RAID firmware yielded a highly available StorageWorks storage subsystem.

   The dual-redundant controller configuration allows each of a pair of active controllers to operate independently with host systems, while sharing device ports, configuration information, and status. This design allows both controllers to achieve maximum performance. The dual-redundant configuration also provides fault tolerance if one controller fails, because the surviving controller serves the failed controller's devices to the host computers. The dual-controller configuration, combined with StorageWorks controller packaging, results in a highly available controller configuration with built-in fault tolerance, error recovery, and battery backup features.

   Parity RAID firmware, combined with StorageWorks device packaging, allows for highly available disk configurations that are less costly than mirrored configurations. Furthermore, Parity RAID firmware performs automatic Parity RAID management and error recovery functions in the event of a failure and utilizes spare device pools in conjunction with user-defined Parity RAID configuration management policies. The StorageWorks Parity RAID implementation exceeds the requirements of the RAID Advisory Board for RAID availability features.

3. High performance. The HSJ30/HSJ40, HSD30, and HSZ40 controllers achieved the respective initial performance goals of 1,100, 800, and 1,400 I/Os per second. The controllers met the low request latency goals by streamlining firmware where possible and by introducing write-back caching. Write-back caching firmware dramatically reduces latency on all write requests, and write-back cache hardware provides battery backup for data integrity across power failures. Furthermore, the write-back cache overcomes the RAID level 5 small-write penalty and high data transfer rate inefficiencies and thus provides high performance with Parity RAID disk configurations. StorageWorks Parity RAID firmware

implements many of the RAID Advisory Board optional performance features to produce a high-performance RAID solution.

A common controller processing core was successfully developed for the HSJ30/HSJ40, HSD30, and HSZ40 controllers. More than 85 percent of the firmware is common to all three controller platforms, which allows for ease of maintenance and for the same look and feel for customers. The architecture and the technology used resulted in a core controller design that supports a high data transfer rate for all StorageWorks controller platforms.

These achievements represent the large engineering investment that Digital has made to move into the open systems market with new technology for its storage solutions. These controller platforms are the basis for future controller architectures and platforms that utilize the knowledge and experience acquired during the development of the StorageWorks HS-series array controllers.

ACKNOWLEDGMENTS

The StorageWorks array controller project was the cooperative effort of a large number of engineers who sacrificed considerable personal time to achieve the project goals. The following people and groups contributed to the success of the product: Bob Blackledge, Diana Shen, Don Anders, Richard Woerner, Ellen Lary, Jim Pherson, Richard Brame, Jim Jackson, Ron McLean, Bob Ellis, Clark Lubbers, Susan Elkington, Wayne Umland, Bruce Sardeson, Randy Marks, Randy Roberson, Diane Edmonds, Roger Oakey, Rod Lilak, Randy Fuller, Joe Keith, Mary Ruden, Mike Richard, Tom Lawlor, Jim Pulsipher, Jim Vagais, Ray Massie, Dan Watt, Jesse Yandell, Jim Zahrobsky, Mike Walker, Tom Fava, Jerry Vanderwaall, Dave Mozey, Brian Schow, Mark Lyon, Bob Pemberton, Mike Leavitt, Brenda Lieber, Mark Lewis, Reuben Martinez, John Panneton, Jerry Lucas, Richie Lary, Dave Clark, Brad Morgan, Ken Bates, Paul Massiglia, Tom Adams, Jill Gramlich, Leslie Rivera, Dave Dyer, Joe Krantz, Kelly Tappan, Charlie Zullo, Keith Woestehoff, Rachel Zhou, Kathy Meinzer, and Laura Hagar. Thanks to the CAD team, the StorageWorks packaging and manufacturing team, the software verification team, and the problem management team. A final thanks to our OpenVMS and DEC OSF/1 operating system partners and to the corporate test groups, all of whom worked with our engineering team to ensure interoperability between the operating systems and the controllers.

REFERENCES AND NOTES

1. Information Systems -- Small Computer Systems Interface-2 (SCSI-2), ANSI X1.131-1994 (New York: American National

Standards Institute, 1994).

2. D. Patterson, G. Gibson, and R. Katz, "A Case for Redundant Arrays of Inexpensive Disks (RAID)," Report No. UCB/CSD 87/391 (Berkeley: University of California, December 1987).

3. The RAID level 5 small-write penalty results when a small write operation does not write all the blocks associated with a parity block. This situation requires disk reads to recalculate parity that must then be written back to the RAID level 5 unit to achieve data redundancy.

4. P. Massiglia, ed., The RAIDbook: A Source Book for Disk Array Technology, 4th ed. (St. Peter, Minnesota: The RAID Advisory Board, September 1994).

5. P. Biswas, K. Ramakrishnan, D. Towsley, and C. Krishna, "Performance Analysis of Distributed File Systems with Non-Volatile Caches," ACM Sigmetrics (1993).

6. Parity RAID unit reconstruction of data and parity from a failed array member means regenerating the data block-by-block from the remaining array members (see Figure 6) and writing the regenerated data onto a spare disk. Reconstruction restores data redundancy in a Parity RAID unit.

7. Metadata is information written in a reserved area of a disk. The information, which takes up approximately 0.01 percent of the total disk capacity, describes the disk's configuration and state with respect to its use in a Parity RAID unit.

8. P. Biswas and K. Ramakrishnan, "Trace Driven Analysis of Write Caching Policies for Disks," ACM Sigmetrics (1993).

9. R. Lary and R. Bean, "The Hierarchical Storage Controller, A Tightly Coupled Multiprocessor as Storage Server," Digital Technical Journal, vol. 1, no. 8 (February 1989): 8-24.

BIOGRAPHY

Stephen J. Sicola Consulting engineer Stephen Sicola is a member of the the Array Controller Group in the Storage Business Unit. He is working on the next generation of controllers and was the technical leader for the current StorageWorks controller product set. In earlier work, Steve developed software and hardware for such products as the HSC, KDM70, and advanced development controller projects. Steve joined Digital in 1979 after receiving a B.S.E.E. from Stanford University. He received an M.S.C.E. from the National Technological University in 1992.

TRADEMARKS