The Design and Verification of the AlphaStation 600 5-series Workstation

by John H. Zurawski, John E. Murray, and Paul J. Lemmon

ABSTRACT

The AlphaStation 600 5-series workstation is a high-performance, uniprocessor design based on the Alpha 21164 microprocessor and on the PCI bus. Six CMOS ASICs provide high-bandwidth, low-latency interconnects between the CPU, the main memory, and the I/O subsystem. The verification effort used directed, pseudorandom testing on a VERILOG software model. A hardware-based verification technique provided a test throughput that resulted in a significant improvement over software tests. This technique currently involves the use of graphics cards to emulate generic DMA devices. A PCI hardware demon is under development to further enhance the capability of the hardware-based verification.

INTRODUCTION

The high-performance AlphaStation 600 5-series workstation is based on the fastest Alpha microprocessor to date -- the Alpha 21164.[1] The I/O subsystem uses the 64-bit version of the Peripheral Component Interconnect (PCI) and the Extended Industry Standard Architecture (EISA) bus. The AlphaStation 600 supports three operating systems: Digital UNIX (formerly DEC OSF/1), OpenVMS, and Microsoft's Windows NT. This workstation series uses the DECchip 21171 chip set designed and built by Digital. These chips provide high-bandwidth, low-latency interconnects between the CPU, the main memory, and the PCI bus.

This paper describes the architecture and features of the AlphaStation 600 5-series workstation and the DECchip 21171 chip set. The system overview is first presented, followed by a detailed discussion of the chip set. The paper then describes the cache and memory designs, detailing how the memory design evolved from the workstation's requirements. The latter part of the paper describes the functional verification of the workstation. The paper concludes with a description of the hardware-based verification effort.

SYSTEM OVERVIEW

The AlphaStation 600 5-series workstation consists of the Alpha 21164 microprocessor, a third-level cache that is external to the CPU chip, and a system chip set that interfaces between the CPU, the memory, and the PCI bus. The DECchip 21171 chip set consists of three designs: a data slice, one PCI interface and memory-control chip (called the control chip), and a

miscellaneous chip that includes the PCI interrupt logic and flash read-only memory (ROM) control. The Intel 82374 and 82375 chip sets provide the bridge to the EISA bus.[2] Figure 1 shows a block diagram of the workstation.

[Figure 1 (AlphaStation 600 5-series Workstation Block Diagram) is not available in ASCII format.]

The SysData bus transfers data between the processor, the CPU's tertiary cache, and the data slices. The 128-bit-wide SysData bus is protected by error-correcting code (ECC) and is clocked every 30 nanoseconds (ns). The data slices provide a 256-bit-wide data path to memory. Data transfers between the PCI and the processor, the external cache (typically 4 megabytes [MB]), and memory take place through the control chip and four data slices. The control chip and the data slices communicate over the 64-bit, ECC-protected I/O data bus.

The major components and features of the system board are the following:

- The Alpha 21164 microprocessor supports all speed selections from 266 to 333 megahertz (MHz).
- The plug-in, external write-back cache (2 MB to 16 MB) has a block size of 64 bytes. Access time is a multiple of the processor cycle time and is dependent on the static random-access memory (SRAM) part used. With 12-ns SRAMs, typical access times are 24 ns for the first 128 bits of data, 21 ns for remaining data.
- The system board contains a 256-bit data path to memory (284 megabytes per second [MB/s] for sustained CPU reads of memory).
- From 32 MB to 1 gigabyte (GB) of main memory can be used in industry-standard, 36-bit, single in-line memory modules (SIMMs). All memory banks support single-sided and double-sided SIMMs.
- Eight option slots are available for expansion: four PCI, three EISA, and one PCI/EISA shared slot. The system design minimized logic on the mother board in favor of more expansion slots, which allow customers to configure to their requirements. The system uses option cards for small computer systems interface (SCSI), Ethernet, graphics, and audio.
- The system supports 64-bit PCI address and data capability.
- Due to its synchronous design, the system's memory, cache, and PCI timing are multiples of processor cycle time.

- The system provides an X bus for the real-time clock, keyboard controller, control panel logic, and the configuration RAM.

Data Slice Chips

Four data slice chips implement the primary data path in the system. Collectively, the data slices constitute a 32-byte bus to main memory, a 16-byte bus to the CPU and its secondary cache, and an 8-byte bus to the control chip (and then to the PCI bus).

Figure 2 shows a block diagram of the data slice chip. The data slice contains internal buffers that provide temporary storage for direct memory access (DMA), I/O, and CPU traffic. A 64-byte victim buffer holds the displaced cache entry for a CPU fill operation. The Memory-Data-In register accepts 288 bits (including ECC) of memory data every 60 ns. This register clocks the memory data on the optimal 15-ns clock to reduce memory latency. The memory data then proceeds to the CPU on the 30-ns, 144-bit bidirectional data bus. A set of four, 32-byte I/O write buffers help maximize the performance of copy operations from memory to I/O space. A 32-byte buffer holds the I/O read data. Finally, there are a pair of DMA buffers, each consisting of three 64-byte storage areas. DMA read operations use two of these three locations: the first holds the requested memory data, and the other holds the external cache data in the case of a cache hit. DMA writes use all three locations: one location holds the DMA write data, and the other two hold the memory and cache data used during a DMA write merge.

The data slice allows for simultaneous operations. For instance, the I/O write buffers can empty to the control chip (and then to the PCI) while a concurrent read from CPU to main memory is in progress.

[Figure 2 (Data Slice Block Diagram) is not available in ASCII format.]

Control Chip

The control chip controls the data slices and main memory and provides a fully compliant host interface to the 64-bit PCI local bus. The PCI local bus is a high-performance, processor-independent bus, intended to interconnect peripheral controller components to a processor and memory subsystem. The PCI local bus offers the promise of an industry-standard interconnect, suitable for a large class of computers ranging from personal computers to large servers.

Figure 3 shows a block diagram of the control chip. The control chip contains five segments of logic:

- The address and command interface to the Alpha 21164 microprocessor
- The data path from the PCI bus to the data slices by means of the I/O data bus
- DMA address logic, including a 32-entry scatter/gather (S/G) map (This is discussed in the section Scatter/Gather Address Map.)
- Programmed I/O read/write address logic
- The memory address and control logic

[Figure 3 (Control Chip Block Diagram) is not available in ASCII format.]

CPU Interface. A three-deep queue can hold two outstanding read requests, together with the address of a victim block associated with one of these read requests. During a DMA write, the Flush Address register holds the address of the cache block that the CPU must move to memory (and invalidate in the cache). In this manner, the system maintains cache coherency during DMA write operations.

PCI Address Space Windows. PCI devices use address space windows to access main memory. During discussions with the developers of the operating system, we determined that four PCI address space windows would be desirable. EISA devices use one window. S/G mapping uses a second. The third window directly maps a contiguous PCI address region to a contiguous region of main memory. The fourth window supports 64-bit PCI addresses. Future system designs may provide more than 4 GB of main memory, thus requiring the 64-bit address window.

DMA Write Buffering. The control chip provides a single-entry, 64-byte DMA write buffer. Once the buffer is full, the data is transferred to the DMA buffers in the data slices. The design can support 97-MB/s DMA write bandwidth from a 32-bit PCI device.

DMA Read Buffering. In addition to the two 64-byte buffers inside the data slice, the control chip has two 32-byte DMA read buffers. These buffers prefetch DMA read data when the initiating PCI read command so indicates. This arrangement provides data to a 64-bit PCI device at a rate of more than 260 MB/s.

Scatter/Gather Address Map. The S/G mapping address table translates contiguous PCI addresses to any arbitrary memory

address on an 8-kilobyte (KB) granularity. For software compatibility with other Alpha system designs, the S/G map uses a translation lookaside buffer (TLB).[3] The designers enhanced the TLB: First, each of the eight TLB entries holds four consecutive page table entries (PTEs). This is useful when addressing large 32-KB contiguous regions on the PCI bus. For instance, the NCR810 PCI-to-SCSI device requires nearly 24 KB of script space.[4] Second, software can lock as many as one half of the TLB entries to prevent the hardware-controlled replacement algorithm from displacing them. This feature reduces TLB thrashing.

Programmed I/O (PIO) Writes. The designers focused on improving the performance of the functionality that allows a processor to copy from memory to I/O space. High-end graphics device drivers use this functionality to load the graphics command into the device's first-in, first-out (FIFO) buffer. The data slice has four buffers, and the control chip contains the corresponding four-entry address queue. Four buffers hold enough I/O write transactions to mask the latency of the processor's read of memory. The control chip provides two additional 32-byte data buffers. While one drives data on the PCI bus, the other accepts the next 32 bytes of data from the data slices.

Memory Controller. The memory controller logic in the control chip supports as many as eight banks of dynamic random-access memory (DRAM). The current memory backplane, however, provides for only 4 banks, allowing from 32 MB to 1 GB of memory. The memory controller supports a wide range of DRAM sizes and speeds across multiple banks in a system. Registers program the DRAM timing parameters, the DRAM configuration, and the base address and size for each memory bank. The memory timing uses a 15-ns granularity and supports SIMM speeds ranging from 80 ns down to 50 ns.

CACHE DESIGN

The Alpha 21164 microprocessor contains significant on-chip caching: an 8-KB virtual instruction cache; an 8-KB data cache; and a 96-KB, 3-way, set-associative, write-back, second-level mixed instruction and data cache. The system allows for an external cache as a plug-in option. This cache is typically 2 MB to 4 MB in size, and the block size is 64 bytes. The access time for the external cache depends on the CPU frequency and the speed variant of the cache. Typically, the first data requires 7 to 8 CPU cycles; subsequent data items require 1 or 2 fewer cycles. The actual value depends on both the minimum propagation time through the cache loop and on the CPU cycle time. The external cache data bus is 16 bytes wide, providing almost 1 GB/s of bandwidth with a 333-MHz CPU and a 5-cycle cache access.

The processor always controls the external cache, but during a

cache miss, the system and the processor work together to update the cache or displace the cache victim. For an external cache miss, the system performs four 16-byte loads at 30 ns. Any dirty cache block is sent to the victim buffer in the data slices, in parallel with the read of memory. Fast page-mode memory writes are used to write the victim into memory quickly. (This is discussed in the section Memory Addressing Scheme.)

During DMA transactions, the system interrogates the CPU for relevant cache data. There is no duplicate tag in the system. DMA reads cause main memory to be read in parallel with probes of the CPU's caches. If a cache probe hits, the cache data is used for the DMA read; otherwise main memory data is used. Each DMA write to memory results in a FLUSH command to the CPU. If the block is present in any of the caches, then the data is sent to the DMA buffers in the data slice and the cache blocks are invalidated. This cache data is discarded if the DMA write is sent to a complete block. In the case of a DMA write to a partial block, the DMA write data is merged with cache data or the memory data as appropriate. In this manner, the system maintains cache coherency, removing this burden from the software.

MEMORY BANDWIDTH

The memory bandwidth realized by the CPU depends on a number of factors. These include the cache block size, the latency of the memory system, and the data bandwidth into the CPU.

Cache Block Size

The Alpha 21164 microprocessor supports either a 32- or 64-byte cache block size. The AlphaStation 600 workstation uses the 64-byte size, which is ideal for many applications, but suffers on certain vector-type programs with contiguous memory references.[5] An example of a larger block size design is the RISC System/6000 Model 590 workstation from International Business Machines Corporation.[6] This design supports a 256-byte cache block size, allowing it to amortize a long memory latency by a large memory fetch. For certain vector programs, the Model 590 performs well; but in other applications, the large block size wastes bandwidth by fetching more data than the CPU requires.

The AlphaStation 600 provides a hardware feature to gain the benefit of a larger block size when appropriate. The Alpha 21164 microprocessor can issue a pair of read requests to memory. If these two reads reside in the same memory page, the control chip treats them as a single 128-byte memory read. In this way, the system approximates the benefit of a larger block and achieves 284 MB/s of memory read bandwidth.

Memory Latency

The 180-ns memory latency consists of five parts. First, the address is transferred from the microprocessor to the control chip in 15 ns. The control chip sends the memory row-address pulse 15 ns later, and the data is received by the data slices 105 ns later. The data slices require 15 ns to merge the wider memory data onto the narrower SysData bus, and the last 30 ns are spent updating the external cache and loading the Alpha 21164 microprocessor.

Although the 105 ns to access the memory may appear to be generous, the designers had to meet the significant challenge of implementing the required 1 GB of memory with inexpensive 36-bit SIMMs. The JEDEC standard for these SIMMs only specifies the pinning and dimensions. It does not specify the etch lengths, which can vary by many inches from vendor to vendor. Neither does it specify the electrical loading distribution, nor the DRAM type or location (1-bit parts have 2 data loads whereas 4-bit parts have a single, bidirectional load). With a 1-GB memory system, the loading variation between a lightly loaded memory and a fully loaded memory is significant. All these factors contributed to significant signal-integrity problems with severe signal reflections. The memory mother-board etch was carefully placed and balanced, and numerous termination schemes were investigated to dampen the signal reflections.

Data Bandwidth

The SysData bus transfers data between the processor, the tertiary cache, and the data slices. This 128-bit bus is clocked every 30 ns to satisfy the write timing of the external cache and to be synchronous with the PCI bus. Typical memory DRAM parts cycle at 60 ns, thus requiring a 32-byte-wide memory bus to match the bandwidth of the SysData bus. The data slice chips reduce each 32-byte-wide memory data transfer to two 16-byte transfers on the SysData bus. Consequently, the memory system is logically equivalent to a 2-way interleaved memory design.

New memory technologies with superior data bandwidths are becoming available. Synchronous DRAMs are an exciting technology, but they lack a firm standard and are subject to a significant price premium over plain 5-volt DRAM parts. Extended-data-out (EDO) DRAMs allow greater burst memory bandwidth, but the latency to the first data is not reduced. Consequently, the memory bandwidth to the CPU is not significantly improved. The major advantage of using EDO parts is their easier memory timing: The output data of EDO parts is valid for a longer period than standard DRAMs. In addition, an EDO memory can be cycled at 30 ns, which allows a 128-bit memory width instead of the 256-bit width. The designers would have used EDO parts had they been available earlier.

MEMORY ADDRESSING SCHEME

The adopted addressing scheme helps improve memory bandwidth. Whenever the CPU requests a new block of data, the write-back cache may have to displace current data (the victim block) to allow space for the incoming data. The writing of the victim block to memory should occur quickly, otherwise it will impede the CPU's request for new data.

Figure 4 shows the method used to address the external cache and memory. The CPU address <31:6> directly accesses the cache: the low-order bits <19:6> form the index for a 1-MB cache, and the remaining bits <31:20> form the cache tag. The CPU address does not directly address memory. Instead, the memory address interchanges the index portion of the address field with the tag portion. The number of address bits interchanged depends on the row and column dimensions of the DRAM used.

For the sake of discussion, assume a 4-megabit (Mb) DRAM configured with 11 row address bits and 11 column address bits. Hence, bits <30:20> interchange with bits <16:6>, and the remaining bits select the memory bank. This addressing scheme has the following effect: a CPU address that is incrementing by units of 1 MB now accesses consecutive memory locations. DRAM memory provides a fast addressing mode, called page mode, whenever accessing consecutive locations. For a 1-MB cache, objects separated by a multiple of 1 MB correspond to cache victim blocks. Consequently, a CPU read request of memory that involves a victim write to memory gains the benefit of page mode and proceeds faster than it would with a traditionally addressed memory.

Although this address scheme is ideal for CPU memory accesses, it creates the converse effect for DMA transactions. It scatters consecutive DMA blocks by 1 MB in memory. These locations fall outside the DRAM page-mode region, resulting in lower performance. The solution is to enlarge the memory blocks; for example, start the memory interchange at bit <8> instead of bit <6>. This compromise allows 256-byte DMA bursts to run at full speed. Slightly fewer victim blocks, however, gain the benefit of page mode.

The bit assignment for this address scheme depends on the row and column structure of the DRAM part and on the external cache size. Power-on software automatically configures the address-interchange hardware in the system.

[Figure 4 (Memory Address Scheme) is not available in ASCII format.]

DESIGN CONSIDERATIONS

In this section, we discuss the design choices made for system clocking, timing verification, and the application-specific integrated circuit (ASIC) design.

System Clocking

The chip set is a synchronous design: The system clock is an integer multiple of the CPU cycle time. Consequently, the PCI clock, the memory clock, and the cache loop are all synchronous to each other. The designers avoided an asynchronous design for two reasons. It suffers from longer latencies due to the synchronizers, and it is more difficult to verify its timing.

Unlike the memory controller, which uses a double-frequency clock to provide a finer 15-ns resolution for the memory timing pulses, the synchronous design of the chip set uses a single-phase clock. This simplified clocking scheme eased the timing verification work. Phase-locked-loop (PLL) devices control the clock skew on the system board and in the ASICs. The PLL in the ASICs also generates the double-frequency clock.

Timing Verification

The complete system was verified for static timing. A signal-integrity tool similar to SPICE was used to analyze all the module etch and to feed the delays into the module timing verification effort. The final ASIC timing verification used the actual ASIC etch delays. This process was so successful that the actual hardware was free of any timing-related bug or signal-integrity problem.

ASIC Design

The chip designers chose to implement the gate array using the 300K technology from LSI Logic Corporation. The control chip uses over 100K gates, and each data slice consumes 24K gates. Originally, the designers considered the slower 100K technology, but it proved unable to satisfy the timing requirements for a 64-bit-wide PCI bus.

The designers used the VERILOG hardware description language to define all the logic within the ASICs. Schematics were not used. The SYNOPSIS gate-synthesizer tool generated the gates. The designers had to partition the logic into small 3,000 to 8,000 gate segments to allow SYNOPSIS to complete within 12 to 15 hours on a DECstation 5000 workstation. Currently, the same synthesis requires 1 hour on the AlphaStation 600 5/260. The designers developed a custom program that helped balance the timing constraints across these small gate segments. This allowed the SYNOPSIS tool to focus its attention on the segments with the greatest potential for improvement.

PERFORMANCE

Table 1 gives the bandwidths of the workstation for the 32-bit and 64-bit PCI options. A structural simulation model verified this data, using a 180-ns memory latency and a 30-ns system clock. The 285-MB/s read bandwidth of the CPU memory is impressive considering that the memory system is 1 GB. Eventually, the memory size will reach 4 GB when 64-Mb memory chips become available.

The I/O write bandwidth is important for certain 3D graphics options that rely on PIO to fill the command queue. Current high-end graphics devices require approximately 80 MB/s to 100 MB/s. The 213 MB/s of I/O write bandwidth on the 64-bit PCI can support a double-headed 3D graphics configuration without saturating the PCI bus. Other 3D graphics options use large DMA reads to fill their command queue. This approach offers additional bandwidth at 263 MB/s. The system did not optimize DMA writes to the same extent as DMA reads. Most options are amply satisfied with 100 MB/s of bandwidth.

Table 1 Bandwidth Data

Transaction Type	32-bit PCI	64-bit PCI
CPU memory read:		
64 bytes	284	284
I/O write:		
Contiguous 32 bytes	119	213
Random 4 bytes	44	44
I/O read:		
4 bytes	12	12
32 bytes	56	56
DMA read:		
64 bytes	79	112
8 KB	132	263
DMA write:		
64 bytes	97	102

Table 2 gives the performance for several benchmarks. The data is for a system with a 300-MHz processor and a 4-MB cache built out of 12-ns SRAM parts. The SPECmark data is preliminary and clearly world-class. The LINPACK data is for double-precision operands. Even greater performance is possible with faster cache options (for instance, a cache using 8-ns parts) and faster speed

variants of the Alpha 21164 microprocessor.

Table 2 Benchmark Performance

Benchmark	Performance
SPECint92	331
SPECfp92	503
LINPACK 100 X 100	144
LINPACK 1000 X 1000	380

FUNCTIONAL VERIFICATION

The functional verification is an ongoing effort. Three factors contribute to the need for greater, more efficient verification. First, the design complexity of each new project increases with the quest for more performance. Next, the quality expectations are rising -- the prototype hardware must boot an operating system with no hardware problems. Finally, time to market is decreasing, providing less time for functional verification.

A number of projects at Digital have successfully used the SEGUE high-level language for functional verification.[3,7] SEGUE allows simple handling of randomness and percentage weightings. As an example, a code sequence may express that 30 percent of the DMA tests should target the scatter/gather TLB, and that the DMA length should be selected at random from a specified range. Each evocation of SEGUE generates a test sequence with different random variations. These test sequences are run across many workstations to achieve a high throughput. The project used 20 workstations for 12 months.

The test suite focused on the ASIC verification in the context of the complete system. It was not a goal to verify the Alpha 21164 microprocessor; neither was the EISA logic verified (this logic was copied from other projects). The test environment used the VERILOG simulator and included the Alpha 21164 behavioral model, a PCI transactor (a bus functional model), and a memory and cache model. The SEGUE code generated C-language test programs for CPU-to-memory and CPU-to-I/O transactions, as well as DMA scripts for the PCI transactor.

The goal of verification went beyond ensuring that the prototype hardware functioned correctly. The major objective was to ensure that the hardware is reliable many years hence, when new, as yet undeveloped, PCI options populate the system. Today, the PCI bus uses only a small number of expansion option cards. It is quite probable that a perfunctory verification of the PCI logic would result in a working system at the time of hardware power-on and for many months thereafter. It is only as more option cards become available that the likelihood of system failure grows.

Consequently, the verification team developed a detailed PCI transactor and subjected the PCI interface in the control chip to heavy stressors. The complexity of the PCI transactor far exceeds that of the PCI interface logic within the ASIC. The reason is that the ASIC design implements only the subset of the PCI architecture appropriate to its design. The PCI transactor, however, has to emulate any possible PCI device and thus must implement all possible cases. Furthermore, it must model poorly designed PCI option cards (the word "should" is common in the PCI specification).

The verification experience included the following:

- Directed tests. Specific, directed tests are needed to supplement pseudorandom testing. For example, a certain intricate sequence of events is best verified with a specific test, rather than relying on the random process to generate the sequence by chance.

- Staff hours. In prior projects, the hardware team exceeded the verification team in size. Over the years, the importance of verification has grown. On this project, twice as much time was spent on the verification effort as on the hardware coding.

- Degree of randomness. Pure randomness is not always desirable. For instance, an interesting test can be conducted when a DMA write and a CPU read target the same block in memory (although, for coherency reasons, not the same data). Random addresses are unlikely to create this interaction; instead careful address selection is necessary.

- Error tests. The pseudorandom test process added a different error condition, such as a PCI addressing error, within each test. The hardware logic, upon detecting the error, would vector by sending an interrupt to the error-handling code. The handler would check if the hardware had captured the correct error status and, if it had, would resume the execution of the test program. This strategy uncovered bugs when the hardware continued functioning after an error condition, only to fail many cycles later.

- Hardware simulation accelerator. The project team did not use a hardware simulation accelerator for a number of reasons. In the early phase of verification, bugs are so frequent that there is no value in finding more bugs. The limiting resource is the isolation and fixing of the bugs. Second, porting the code onto the hardware simulator uses resources that are better spent improving the test suite: running poor tests faster is of no value. Finally, the hardware-based verification technique offers far greater performance.

- Bug curve. The project team maintained a bug curve. The first-pass ASIC was released when the bug curve was falling but was still above zero. The tests were structured to test the important functionality first. This allowed verification to continue while the operating system developers debugged their code on the prototype. To help this strategy, any performance-enhancement logic in the ASICs could be disabled in case an error was discovered in that logic. Experience on prior projects had shown that such logic has a predilection toward bugs.

HARDWARE-BASED VERIFICATION

The hardware-based verification was developed to achieve a significant, five-orders-of-magnitude improvement in test throughput. The CPU performs pseudorandom memory and I/O-space transactions, and a number of PCI graphics options emulate generic PCI devices. The hardware-based verification has so far uncovered three bugs. To further improve this technique, a hardware PCI demon is under development. This device has the capability to mimic any PCI device.

The random nature of the test suite means that the bug curve has a long tail: The probability of finding the next bug decreases as each bug is discovered. For example, an earlier project team discovered the last bug after six months but needed only one week to find the penultimate bug. Greater test throughput helps uncover the final bug(s) sooner. Our project team achieved greater throughput by migrating the test strategy onto the actual hardware.

A self-checking, pseudorandom, test-generating program runs on the hardware, testing the memory, the cache, and the PCI. On detecting a mismatch, the software triggers a digital analyzer connected to visibility points on the hardware. Currently, a number of PCI graphics cards are emulating different DMA devices. Eventually, a custom PCI test device, the PCI demon, will replace the graphics cards and provide greater flexibility and functionality (especially with regard to error cases).

The software-based verification, running across 20 workstations, averaged approximately 100 DMA transactions per minute (with concurrent memory and PIO activity). The hardware-based verification runs 60 million comparable DMA transactions per minute per workstation. This 5-orders-of-magnitude improvement suggests that all the tests performed in the past 12 months of software-based verification can be completed during the hardware-based debugging in 5 minutes.

A secondary, but very useful, advantage of hardware-based testing is the ability to stress the chips electrically. For instance, by selecting a data pattern of 1's and 0's for the DMA,

memory, and PIO tests, verification engineers can severely test the capability of the chips to switch simultaneously.

Hardware Test Strategy

The SEGUE software proved not to be useful for the hardware-based verification effort. Instead new software was written in the C language for the following reasons:

- Verification must have full control of the hardware and thus cannot run on top of an operating system. Consequently, SEGUE and the operating system functionality are not available.

- Unlike the software environment, visibility into the logic signals is restricted in the hardware environment. The test software has to be written to make debugging simpler.

- One possible strategy is to download the SEGUE tests onto the hardware and thus treat the hardware as a simulation accelerator. However, the resultant performance improvement is small: The SEGUE code takes 2 minutes to generate a 1-hour software-simulation run. These tests run across 20 workstations with a resultant throughput of 1 test every 3 minutes. Assuming the same test executed in zero time on the hardware, the total test time would equal 1 test every 2 minutes -- a minor improvement.

The hardware-based verification software relies on the following rationale: The hardware is almost totally bug free, and any remaining bugs are most likely to be due to a rare interaction of events. Indeed, one of the bugs discovered was a special-case DMA prefetch coinciding with a memory refresh. Consequently, no test is likely to detect more than one bug. For instance, if a DMA operation suffers an error, then it is unlikely that a subsequent, identical DMA operation will suffer an error. The second DMA will experience a different set of interactions inside the chip set.

The adopted test environment has two graphics cards, each performing identical DMA operations to two different regions of memory. Because of the serial nature of the PCI bus, however, these cards will perform the DMA operations at different times. Furthermore, other traffic on the PCI bus (for instance, the CPU will be executing random PIO) will further randomize the cards' behavior. While the DMA transactions run, self-checking, random CPU traffic to memory and I/O will also run. These events provide the random mix of interacting instructions. At the completion of the test, a miscomparsion of the two DMA write regions indicates an error.

Graphics Demon

A number of PCI option cards were investigated as potential PCI demon cards. The requirements for a PCI demon card are twofold: it must be able to perform DMA of various lengths, and it must have memory for the storage of DMA and PIO data. The DEC ZLXp_E1 graphics card was selected because it offers the following advantages:

- Independent DMA. Most PCI options start a DMA operation instantly after the CPU has written to a specific register in the option. This is undesirable because it makes it impossible to emulate options that start DMA operations autonomously (e.g., a network card). To break this coupling, the test program should first make the graphics card paint a portion of the screen. While the graphics device is busy, the graphics command FIFO buffer is filled with the DMA commands. The graphics device will not start the DMA until it has finished painting. Furthermore, the delay is programmable by varying the number of pixels painted.

- Programmable DMA. The graphics card allows the DMA to be any size, whereas most PCI options are constrained to a fixed length. Moreover, it is possible to arrange for PCI disconnects on a DMA read. The graphics card modifies incoming data with the contents of the frame buffer (e.g., frame buffer = frame buffer XOR data). This feature throttles the internal bandwidth of the graphics card, which disconnects it from the PCI.

- Frame buffer. The graphics frame buffer is the target of the DMA and PIO operations. A useful software debugging feature was to observe the frame buffer while running the tests.

PCI Demon

The PCI demon is designed to mimic any possible PCI device. Software has total control of the behavior of the device, including the assertion of error conditions (e.g., parity errors on any specified data word). The architecture of the PCI demon is very simple so that the debugging of the PCI demon is straightforward. (The objective is to find bugs in the chip set and not in the PCI demon.) Consequently, the complexity in using the PCI demon is completely in the software.

The ideal architecture of a PCI demon is a large memory whose output drives the PCI data and control signals directly; the software programs the desired PCI operation by loading the appropriate pattern into this memory. In reality, the architecture of the PCI demon has to diverge from this ideal model for at least two reasons. First, the PCI demon has to be

able to emulate the fastest possible PCI device, and this forces the use of an ASIC. However, ASICs have limited memory capacity. It is desirable to store the scripts for many thousands of DMAs in this memory. The scripts are approximately 100-bits wide (64-bit PCI data and control) and require several megabytes of memory. This memory requirement forces the design to use external memory. Second, the PCI architecture has a few handshake control signals that require the use of a fast state machine.

The PCI demon has the functionality to act as a histogram unit (a PCI event counter). Internal counters measure timing information such as DMA latency and the frequency of specified PCI transactions. The PCI demon observes these states by snooping the PCI bus.

SUMMARY

The AlphaStation 600 5-series workstation offers high compute performance, together with substantial I/O subsystem performance. The project team designed a low-cost, 1-GB memory system with a 180-ns memory latency. Timing verification and placement of the plug-in, external cache resulted in a workstation with considerable flexibility in memory expansion, cache variants, and I/O option slots.

The most time-consuming portion of the project was the functional verification. To date, different test programs have run concurrently across 20 high-performance workstations, day and night, for over 12 months. The release of the prototype chip set occurred after 5 months of verification; this chip set successfully booted the operating system. The remaining 7 months of verification were focused on the lower priority functionality (e.g., error cases and slow memory configurations).

The hardware-based verification approach proved its value by uncovering three bugs. The most significant bug involved the interaction of a number of events, including an optimized, prefetching DMA read and a memory refresh. The verification process helped create a very high quality product.

ACKNOWLEDGMENTS

Many individuals contributed to the success of this project. The design was a team effort involving far more people in many diverse groups than can be acknowledged here. However, recognition is due to the core hardware and verification team: Ed Arthur, Connie Bielawski, Ernie Crocker, Tracey Gustafson, J. Grady, John Hackenberg, Rick Hagen, Randy Hinrichs, Laura Mendyke, Sudhin Mishra, Sandy McPherson, Jim Nietupski, Sub Pal, Nick Paluzzi, Rick Rudman, Jim Reilley, Manoo Siarkowski, Bob Stewart, Hugh Kurth, Tony Camuso, Jim Hamel, Rick Calcagni, Carl Mower, Peter Spacek, and Ned Utzig.

REFERENCES

1. J. Edmondson et al., "Internal Organization of the Alpha 21164, a 300-MHz 64-bit Quad-issue CMOS RISC Microprocessor," Digital Technical Journal, vol. 7, no. 1 (1995, this issue): 119-135.

2. 82420/82430 PCI ISA and EISA Bridges (Santa Clara, Calif.: Intel Corporation, 1993).

3. S. Nadkarni et al., "Development of Digital's PCI Chip Sets and Evaluation Kit for the DECchip 21064 Microprocessor," Digital Technical Journal, vol. 6, no. 2 (Spring 1994): 49-61.

4. NCR 53C810 Data Manual (Dayton, Ohio: NCR Corporation, 1992).

5. A. Agarwal, Analysis of Cache Performance for Operating Systems and Multiprogramming (Boston: Kluwer Academic Publishers, 1989).

6. S. W. White and S. Dhawan, "POWER2: Next Generation of the RISC System/6000 Family," IBM Journal of Research and Development, vol. 38, no. 5 (September 1994).

7. W. Anderson, "Logical Verification of the NVAX CPU Chip Design," Digital Technical Journal, vol. 4, no. 3 (Summer 1992): 38-46.

BIOGRAPHIES

John H. Zurawski

John Zurawski was the system architect for the AlphaStation 600 5-series workstation. Prior to this project, John was the system architect for the DECstation 5000 series of MIPS R4000 workstations. He has also led the verification effort for the DEC 3000 workstation and led the team that designed the floating-point unit for the VAX 8800 family. John holds a B.Sc. degree in physics (1976), and M.Sc. (1977) and Ph.D. (1980) degrees in computer science, all from Manchester University. John is a member of IEEE. He holds seven patents and has published six papers on computer technology. He joined Digital in 1982 after completing post-doctoral research at Manchester University.

John E. Murray

A consulting engineer in the Alpha Personal Systems Group, John Murray was the logic design architect for the AlphaStation 600

5-series. In previous work, John led the design team for the instruction fetch and decode unit on the VAX 9000 system. Prior to joining Digital in 1982, he was with ICL in the United Kingdom. He holds eleven patents.

Paul J. Lemmon

Paul Lemmon joined Digital in 1987; he is a principal engineer. Paul was the ASIC team leader and the architect of the control ASIC for the AlphaStation 600 5-series. He was previously employed at Datapoint, where he was a design engineer/project engineer. Paul received a B.S. in electrical engineering from Ohio State University in 1980. He holds two patents.

TRADEMARKS

AlphaStation, DEC, DECchip, DEC OSF/1, Digital, Digital UNIX, and OpenVMS are trademarks of Digital Equipment Corporation.

Intel is a trademark of Intel Corporation.

Microsoft is a registered trademark and Windows NT is a trademark of Microsoft Corporation.

RISC System/6000 is a registered trademark of International Business Machines Corporation.

SPECfp, SPECint, and SPECmark are registered trademarks of the Standard Performance Evaluation Council.

UNIX is a registered trademark licensed exclusively by X/Open Company, Ltd.