The VAX 6000 Model 600 Processor

By Lawrence Chisvin, Gregg A. Bouchard, and Thomas M. Wenners

1 Abstract

The Model 600 is the newest member of the VAX 6000 series of XMI2-based, multiprocessing computers. The Model 600 processor integrates easily into existing platforms. Each processor module provides 40.5 SPECmarks of performance made possible by the NVAX CPU chip. The major VLSI interface chip, called NEXMI, was created using Digital's internal CMOS-3 design and layout process. The ability to design and fabricate the interface chip internally was critical to delivering a working CPU prototype module on schedule. The aggressive module timing goals were met by employing previous module experience in combination with extensive SPICE simulation.

2 Introduction

The Model 600 system is the latest addition to Digital's VAX 6000 family of midrange symmetric multiprocessing computers.[1] The Model 600 was designed to be integrated cost-effectively into an existing VAX 6000 platform, and to provide a significant performance improvement over the previous generation of systems. Table 1 compares the performance of the Model 600 with the previous generation Model 500 on several important benchmarks. The powerful NVAX single-chip microprocessor enables this level of performance.[2]

3 Design Goals

The primary goal of the project was to deliver a module that included the appropriate support functions, performance, and VAX 6000 system compatibility. Equally important, the module had to be delivered on schedule to prevent an adverse time-to-market impact on the program. This included the delivery of a working prototype module before the first NVAX microprocessor chips were available, since a VAX 6000-based platform would be used to debug the initial NVAX CPU chips. Furthermore, the prototype module had to allow the VMS operating system to be booted and tested.

Our goals were achieved. When the NVAX CPU chip, the prototype module, and the NEXMI support applications specific integrated circuit (ASIC) were integrated for the first time, the hardware worked almost immediately. The software team was well prepared, and the full VMS system boot took place 11 days after the hardware was put together. Moreover, the first-pass modules were used for all the system debugging, and were of sufficient quality to be used for system field test.

This paper relates the background and design process for the VAX 6000 Model 600 processor module. The module and its system context are described, as

well as many of the design decisions that were made during the project. The first section describes the module in general terms, along with some

of the trade-offs and choices associated with its development. The second section focuses on the NEXMI support chip, which is the primary interface device positioned between the NVAX CPU data and address lines (NDAL), the XMI2 system bus, and the support peripheral ROMBUS. It discusses the very large-scale integration (VLSI) design process used to create and verify the functions of this interface. The final section details some of the physical aspects of the module design, including the important work performed to ensure good module signal integrity and thermal management.

4 Description of the Processor Module

Figure 1 shows the VAX 6000 Model 600 CPU module. Figure 2 is a block diagram of the module, showing the major subsections. The CPU module contains two VLSI components: the NVAX microprocessor and the NEXMI ASIC interface chip. The module also holds the backup cache static random-access memory (SRAM) devices, the XMI2 corner bus interface, and the supporting logic necessary to implement a VAX 6000 processor node.

NOTE

Figure 1 (VAX 6000 Model 600 Processor Module) is a photograph and is unavailable.

The NVAX CPU directly controls its external backup cache SRAMs. When the data is resident in the cache, the NVAX CPU modifies it there, and implements a write-back scheme.[2] When the data misses in the backup cache, or when an I/O access is necessary, the NVAX CPU places the command on the NDAL, along with any associated control and data information. The NEXMI accepts and processes the command.

The NEXMI VLSI chip provides an interface to the functions necessary to integrate a VAX 6000 processor module into an XMI2-based system. In particular, the NEXMI chip:

- Translates the NDAL bus commands to XMI2 bus commands
- Returns read data from the XMI2 bus to the NVAX CPU (via the NDAL)
- Forwards invalidate traffic to the NVAX CPU for lookup and potential write back
- Controls NDAL bus arbitration

The NEXMI contains a programmable interval timer, reset logic, halt arbitration logic, and secure console logic on chip. It accommodates the rest of the support functions through the ROMBUS.

The ROMBUS is controlled by and interfaces to the NEXMI; it supplies a path for the low-speed devices that are necessary to create a working computer. The devices on the ROMBUS are off-the-shelf parts that contribute specific functions, including ROM for the boot diagnostic and console program, a stack RAM for storage of diagnostic/console dynamic information, an electrically erasable programmable read-only memory (EEPROM) for saved

state, a universal asynchronous receiver/transmitter (UART) for console communication, an input and output port, and a time-of-year (TOY) clock.

The bare module itself is a sophisticated printed wiring board (PWB) with the following characteristics:

- 11.024-inch by 9.18-inch module size
- 10-layer module with 0.093-inch thickness
- 4 signal layers, 4 power/ground layers and 2 component/dispersion layers
- 0.010-inch vias
- 5-mil etch/7.5 mil space minimum for signals
- 10-mil etch/40 mil space minimum for clocks

5 NEXMI Support Chip

The NEXMI chip is the routing and control interface between the three major buses that reside on the VAX 6000 Model 600 processor module. A functional diagram of the NEXMI is provided in Figure 3. The three major buses are the NDAL, XMI2, and ROMBUS. The NEXMI chip contains the following functions, each function is associated with one or more of the three major buses:

- NDAL bus arbitration
- NDAL receive and transmit logic
- NDAL/XMI2 queues
- XMI2 data/invalidate responder queue
- XMI2 bus control logic
- System support control (SSC) logic

The NDAL receive logic latches and decodes commands and data from the NVAX CPU, and routes them to one of two queues. If the command is a write back, consisting of an address and 32 bytes of write data, it is placed in the XMI2 write-back queue. All other commands (reads and 8-byte writes) are placed in the non-write-back queue. The non-write-back queue services both the XMI2 logic and the SSC logic. Depending on the function, the SSC logic might then send the command on to the ROMBUS. Each queue is loaded in the NDAL time domain, and a request signal is sent to the appropriate XMI2 or SSC logic for processing.

On read commands, data is returned from the XMI2 responder queue or SSC control section, and forwarded to the NDAL through the NDAL transmit section. The NDAL is then requested, and when bus access is granted the data is driven onto the NDAL to be accepted by the NVAX CPU.

The XMI2 logic also sends potential invalidate addresses to the NVAX, where the information is compared with the existing tag address in the indexed backup cache block. An invalidate address is nothing more than the address associated with a command initiated on the XMI2 by another processor. If the cache block matches, and if the NVAX must relinquish control of the data, the block is either invalidated or written back. The choice depends upon the type of transaction and the state of the cache block.

In the Model 600, the NDAL arbitration is handled by the NEXMI chip. The NVAX CPU does not implement the NDAL arbitration on chip because the microprocessor must accommodate many different types of system platforms. A method of arbitration that is fair and efficient on one type of system (for example, a single processor workstation with several potential NDAL master nodes implemented in off-the-shelf programmable devices) might be less than satisfactory for another (such as the NEXMI).

The NEXMI and the NVAX CPU are the only two nodes on the NDAL in this implementation, so a simple priority scheme is used. The NEXMI always has highest priority, since it is either returning data or forwarding XMI2 bus transactions for potential invalidation and write back. In both cases, some other entity on the bus is actively waiting for the information to be returned.

Choice of NEXMI Technology

The NDAL is the NVAX CPU external interface bus. It is a 64-bit, bidirectional, multiplexed address and data bus that runs synchronously with the CPU. Although it is significantly slower than the internal CPU speed (an NVAX with a 12-nanosecond [ns] clock cycle has an NDAL with a 36-ns cycle), it is still aggressive in many respects, and presented a challenge to design using standard parts. The NDAL arbitration for the next cycle happens in parallel with the data transfer for the current cycle, and the full request and grant loop must be performed in one NDAL cycle. In addition, the NDAL data path is bidirectional. An NDAL master must be able to transmit its data to the receiver within a single cycle, then allow time for the bus to become tristate before the next master can drive the bus.

The original product plan was to use several gate arrays to implement the module control logic. One gate array would have contained the XMI2 interface logic, and the other would have controlled the system support functions and interface. It became clear early in the project that the external I/O cells of the gate array could not meet the timing mandated by the NDAL specification. Furthermore, we were unable to obtain timely and accurate SPICE models of the gate array output stage, which compounded our general difficulty in determining the design trade-offs for the module.[4]

The use of an external commodity gate array implied another design-related drawback. The normal gate array design process included the submission of our design for chip layout after we had completely verified it for both logical correctness and estimated timing constraints. The timing was estimated since the actual timing could not be known until the ASIC was routed. The gate array routing would be performed by the vendor, and actual delay numbers would be used to verify the design again. This sometimes meant changing logic interconnections to fix timing-related violations, especially if the design team had been aggressive in either the chip cycle time or gate usage. The design would then have to be verified again, and sent back to the vendor where the process was repeated until everything worked at speed.

Consequently, we decided to design the NEXMI ASIC chip using Digital's CMOS-3 standard cell process at the Hudson, MA site. Once the decision had been made to use the internal process, we realized other significant benefits. We believed that we could collapse the design into one package, since we could make use of Digital's chip expertise to create full-custom sections where necessary. We would know early in the design cycle if our assumptions were wrong, since the design flow uses a subchip approach. The approximate size of each subchip is known as soon as the first pass of the structural design is finished.

Another major advantage of using Digital's internal CMOS design process

was that it afforded us direct contact with the VLSI design, process, and manufacturing groups. As our design progressed, we had constant communication with the people who wrote and supported the computer-aided design (CAD) tools, the people who had designed the circuits and standard cell library elements, and the people who would eventually fabricate the

chips. At any stage of the project, we could determine how to obtain a performance advantage without risk to either chip yield or product reliability. When a tool problem surfaced, or when a tool did not do exactly what we needed, the issue would be immediately addressed and resolved.

By having easy access to the individuals who had detailed knowledge of the circuits, we could attain the maximum possible speed and density from the design. We benefited from the experience of the internal full-custom design community, and profited from access to its complete and accurate SPICE libraries. Consequently, we were certain of how we could obtain a design advantage, yet still allow the chip to perform reliably under worst-case conditions.

The Digital semicustom design process (described in more detail in the section NEXMI Design Process) provides constant feedback on circuit layout. Therefore, our functional and logical timing simulations were always up-to-date with accurate gate and wire delays. By the time we were ready to freeze the design (after we had verified it for logic and timing correctness), only one regression run was needed on a minor change from the last iteration. This approach prevented last-minute surprises, and resulted in a smooth transition from design description, through layout, and into fabrication.

NEXMI Design Issues

During the design of the NEXMI chip, several interesting problems were addressed.

Interblock Control Signal Synchronization. The clock that drives the NVAX and NEXMI chips runs at 36 ns, and is not synchronized to the 64-ns clock that drives the XMI2 bus interface. Without any special care, the data that is generated in one clock domain can cause the target latching device outputs to enter a state called "metastable." This state is characterized by oscillations, or an output voltage level that is neither high nor low for an extended period of time. This can cause unreliable system operation.

Traditionally, a synchronizer is provided for the control signals between two different clock domains to allow communication without causing metastability. A synchronizer is a cascaded set of latches, allowing the first latch to go metastable, but characterized so that it settles down before the second latching device is sampled. The drawback to a synchronizer is that it increases the latency of communication due to the cascaded latching devices.

There are no inexpensive and easy remedies for latency of communication when the system goes from idle to active. However, we decided to reduce

the synchronizer latency on a busy system. Our method optimizes the case where information is in either the NDAL queue or the XMI2 responder queue, waiting for transmission when the current command has been successfully transmitted. For these situations, we created a set of round-robin synchronizers, with a ring of request and done signals in each direction.

While one request/done pair is being serviced, the pipeline overhead for the next pair is hidden by the overlapping synchronizers.

NEXMI Queues. For the three major queues in the NEXMI chip, we determined reasonable queue sizes based on our chip space constraints and performance simulations. System testing on the real hardware during our debugging and system integration phase confirmed that our decisions were correct. None of the queues cause performance degradation on an actual running system.

We decided to make the non-write-back and the XMI responder queues into integrated queues rather than have separate queues for each function. Queuing theory shows that a shared resource is better utilized when only one queue is served by the next available resource, rather than having a separate queue for each resource.[5]

Visibility Port. A problem that always exists with dense, complex integrated circuits is how to diagnose problems that happen in an actual running system that do not show up during simulation. Although no such problems appeared in the NEXMI chip, the designers wanted to provide visibility to as many internal states as possible. To this end, we created a parallel port to provide visibility to some important internal signals.

Given the size of the design, we could provide only the minimum number of signals for visibility. We tried to predict the internal signals that might help diagnosis and adjustment, such as the internal state machines, interblock control signals, and queue head and tail pointers. We then grouped them into similar units so that related signals were visible within one control group. Internally, the signals were segregated within their boxes, and routed so that they did not adversely affect the top-level chip routing.

6 NEXMI Design Process

The NEXMI chip contains 250,000 transistors in a die that measures 0.595 inches by 0.586 inches. It is housed in a custom-designed, 339-pin, ceramic pin grid array (PGA). This section describes the NEXMI chip design methodology and the unique CAD tools that were used to design Digital's largest CMOS-3 semicustom chip. (The chip is physically larger and has more transistors than any previous standard cell produced using the Digital semicustom process.) This section also covers many of the trade-offs made during the chip design, and explains the reasoning behind our decisions.

The design of the NEXMI chip can be characterized by three major phases:

- Behavioral modeling
- Structural design

- Physical chip implementation

The three design phases of Digital's internal CMOS design process significantly overlap each other. Each design stage is explained and analyzed in this section.

Behavioral Modeling Phase

The first major design effort focused on describing the chip functions at a high level of abstraction. This is normally referred to as behavioral or functional modeling, and the design team used Digital's internal hardware description language, DECSIM-BDS, for this task.[6] Functional design sections were allocated to different design engineers, and interfunctional block boundary descriptions were specified.

A behavioral modeling strategy was attractive for many reasons. A well-defined hierarchical partition of the design was quickly realized, and smaller subsections (or subchips) within the chip were identified. Behavioral models of each subchip were initially developed to prove functional correctness. As the design progressed, these subchips were replaced by functionally equivalent gate-level structural models. Using this mixed-mode functional simulation strategy, each designer could progress at his/her own pace, from behavioral definition to structural implementation, independent of the status of other subchips.

The behavioral modeling effort identified many architectural problems early in the design cycle. Details such as queue sizes and structure, flow control mechanisms, and interblock communication protocols were all emphasized, and each decision yielded valuable information about the feasibility of a single-chip implementation.

Behavioral modeling allowed a functional description of the NEXMI chip to be integrated into a system-level model quickly. This enabled the verification team to write and debug tests early in the design cycle. As each structural subchip was finished, it replaced its previous behavioral counterpart in the verification model, and was tested for functional correctness. This step-wise, integrated approach permitted the tests, models, and logic to be verified incrementally.

One major advantage of our behavioral modeling strategy was that it let the design progress without targeting a specific technology. The designers focused their attention on logical implementation rather than on the technology-specific details, such as the timing and loading constraints imposed by a choice of technology. The Choice of NEXMI Technology section described the process of selecting the CMOS-3 implementation path. The initial behavioral phase of the project allowed some of the design to be finished before the final choice of CMOS technology was made.

Structural Design and Chip Implementation Phases

The next major project design phase involved mapping the behavioral subchip models into their equivalent gate-level structural representations. The design methodology chosen followed directly from the decision to

implement NEXMI using Digital's CMOS-3, 1-micrometer, semicustom process. The semicustom process includes a fully specified library of primitive elements, called standard cells, similar to the cells included in a gate array library.

The advantage of the semicustom approach is that it gives the designer full control over the placement and routing of the individual primitives or groups of primitives (subchips) within the chip. This allows the engineer to easily take advantage of special placement for speed-critical paths. Because we were using the internal tool suite and fabrication process, we were also able to take advantage of a wealth of full-custom knowledge provided by our support groups. One of the original reasons for using the internal VLSI process was the tight timing on the NDAL. Therefore, the NEXMI pad ring was custom designed for speed, control, and TTL-level compatibility. Other major handcrafted sections were the dense, multiported queue structures.

To coordinate our large, multiple-person chip design, we used the organized chip design (ORCHID) file management system. The ORCHID system manages the files created by each design tool for every hierarchical subchip. It contains translation tools that convert the schematic data into file formats accepted by the simulation, layout, and verification tools. The system allowed individual designers to work independently on subchips at various stages of development (schematic entry, simulation, floor plans, layout, verification) yet still maintained a coherent hierarchical design database that could be shared by all members of the design team. The idea of a shared database facilitated the reuse of common logic (e.g., counters, parity trees, testability devices), and designers often borrowed from each other to avoid primitive design duplication.

Semicustom Design Flow

Figure 4 shows the semicustom design process and the individual tools that were used during the structural and physical implementation phases of the NEXMI chip design.

ALOE, Digital's in-house graphical editor, was our schematic entry vehicle, and was used in conjunction with the primitive symbols and models from the CMOS-3 standard cell library (SCL3). The tools within the ORCHID system were used to translate schematics into generic wirelists with estimated delays. The schematics were then input to other tools for logic and timing verification. Specific design tools included the internal logic simulator, DECSIM, a timing analysis tool, AUTODLY, and the SPICE circuit simulator.

Automated logic synthesis was used to convert some behavioral models into structural entities. OCCAM, an internal CAD tool, was used to synthesize a gate-level representation of the scattered address decode, and was able to minimize the logic to meet the aggressive bus timing.[7] Controllers embedded within the XMI and SSC subchips were synthesized using SMD2SIM, a tool developed at Digital's Boxboro, MA site for another project. This synthesizer allows large programmed logic array (PLA) structures to be realized from a simple LISP-like behavioral description. During the course

of the design, controller function changes became easier to maintain using a textual description.

After each gate-level subchip was created, the ATLAS tool suite was used to place and route the standard cells within the subchip. The TWOLF editor (TWEDT) was used to place special cells, such as clock buffers and timing-critical gates, within the subchips. The rest of the subchip was then placed automatically and globally routed using the TWOLF program, which relies on a simulated annealing placement algorithm.[8] Lastly, the standard cell assembler known as SCASM was used to assign standard cell rows and to complete the detailed routing.[9] SCASM uses both channel routing and "over-the-cell" routing, which reduced the overall size of the subchip by permitting metal routes over the underlying cells. Subchip post-layout timing information was then fed back into the logic and timing verification tools for a more detailed analysis. The ability to analyze post-layout delay information and to iterate through the layout process early in the design phase was crucial to meeting our schedule.

The top-level floor plan was progressing in parallel with the process of subchip placement and routing. FAME, another tool from the ATLAS tool suite, was used for the chip floor plan and top-level routing.[10,11] Information about top-level routing was fed back to the subchip place and route tools to change the basic shapes and subchip boundary pin placement information. This was used to keep intersubchip routing to a minimum, which in turn reduced overall loading and timing delays. Many successive iterations were necessary to achieve an optimal top-level floor plan that would meet our timing goals and fit onto a single die. Fortunately, much of this work could be done by the designers themselves.

After the final place and route iteration had been done on the the entire chip, an exhaustive set of checks was performed to ensure design integrity and circuit reliability for first-pass silicon. A VLSI design rule checker was run on the individual subchips, then on the entire chip, to verify the layout against the CMOS-3 process design rules. As a precautionary measure, the original chip schematics were extracted to SPICE wirelists using an internal wirelist tool, and another program, called HILEX/CUP, did the same thing using the geometric information in the final layout database. A wirelist comparison program, called IVCMP, compared the two wirelists to ensure that the design we had simulated was the same one we would build.

Finally, a program called XREF used the capacitance from each internal chip node and the topology of the routed interconnection database to predict the cross talk each signal could expect. Changes were made to the signal routing and the sizes of some of the driving transistors, based upon potential problems identified by the XREF program.

Figure 5 is a representation of Digital's CMOS-3 design process. It shows the chip floor plan, along with an example of the schematics used to drive the layout process, and a timing diagram created by the simulation program.

7 Verification of the VAX 6000 Model 600

Early in the design cycle, our verification team was able to integrate a behavioral chip-level model into a CPU module model and perform system-level testing of the VAX 6000 Model 600. These tests gave the designers timely feedback on the effects of their design decisions on the system as a whole.

An NDAL/XMI2 bus monitor, called the BEMAR, was written to verify the bus activity between the NDAL and XMI2 ports of the chip, and to log valuable cycle information. The NEAT, an NDAL cycle emulator, was also used to create NVAX transactions on the NDAL. It was written to reduce simulation test run times and to ease the test verification process.

Most of the tests written were focused tests, targeting a specific function within the NEXMI chip. However, a random bus exerciser was also written to test unexpected combinations, and was instrumental in catching two design flaws that the focused test cases had missed. In all, over 100 focused tests were written and verified against the simulation model, giving us a high degree of confidence that first-pass silicon would be functional. The quality of the prototype systems is due in large part to this complete verification.

A subset of the functional tests was also used by manufacturing to generate chip test vectors. Test vectors were automatically converted from DECSIM-formatted trace files to Takeda pattern sets through a special program called TEMPEST. Prior to the return of first-pass silicon, the generated pattern sets were converted back into DECSIM format, to verify that the pattern sets would run successfully on the Takeda chip tester. This test pattern generation and simulation process reduced the time needed to debug the test patterns when the NEXMI chips became available.

8 Module Design issues

During the design of the VAX 6000 Model 600 processor, many module-related issues were considered. The next section describes some of the more interesting issues that we encountered during the physical module design process. In many cases, we describe the method that was adopted by the design team to prevent problems through careful planning and analysis.

Backup Cache Size and Speed

We chose to defer the selection of the backup cache size and speed until late in the design cycle. This allowed us to make the decision based upon the pricing and availability of the SRAM devices, and the performance to be gained from combinations of these devices. We also wanted to wait until we knew the final speed of the NVAX CPU. The two most likely cache sizes

were 512 kilobytes (KB) and 2 megabytes (MB). We felt that simulation could provide insight into the performance trade-offs, and simulation studies were performed with both sizes to establish approximate performance numbers. We knew, however, that running real modules on a variety of

benchmarks under actual workloads was the most accurate way to determine the performance side of the price/performance trade-off.

The footprints for the two SRAM cache chips that represented the cache size trade-offs were different, which made it difficult to create one prototype module to accommodate both sizes. Creating two separate modules to test the different combinations of cache sizes would have burdened the rest of the project, so special surface-mount pads were designed to handle the different SRAM geometries on the same module. Once the cycle time of the NVAX CPU was determined to be 12 ns, we were able to make the final choice on the cache size and SRAM speeds. The final decision was to provide a 2MB cache with 20-ns 256KB by 4 SRAMs for the data and 15-ns 64KB by 4 SRAMs for the tag. This provided the best balance between cost and performance in the Model 600 system.

ROMBUS Decisions

General-purpose computing systems need a core of system functions to supplement the computing power of the processor. On the VAX 6000 series of computers, this includes:

- ROM to hold the boot, diagnostic, and console code
- EEPROM to hold items such as boot paths and error information
- Console terminal UART
- Time-of-year (TOY) clock
- Battery backed up RAM
- Programmable interval timer
- Time-of-day register
- Input ports to sense external switches and other state
- Output ports to drive module light-emitting diodes (LEDs)
- Reset logic for the module
- Halt detection and arbitration
- Secure console logic

Previous VAX 6000 systems used a system support chip for most of these functions. After we decided to combine as much support logic as possible

into the single ASIC device (NEXMI), we had to determine what functions to include inside the chip, and what functions to locate external to the chip. We saw a potential schedule risk if some functions, such as the UART and TOY clock, were placed inside the NEXMI. These functions were relegated to outside the chip since industry-standard, off-the-shelf components were available to perform exactly the functions we needed.

Once we decided to implement the UART and TOY clock outside the NEXMI, and added the normally external ROM, EEPROM, and input/output ports, we found that the NEXMI pin count was higher than we could afford. Since all the support devices were slow, and each one was byte-wide by its nature, we solved the pin problem by creating a single, slow-speed, bidirectional bus, which we called the ROMBUS. Figure 6 is a block diagram of the ROMBUS and its components. The ROMBUS reduced the NEXMI pin count by 40 pins for the same functions.

Using the ROMBUS for the UART and TOY clock reduced the risk entailed in designing them inside the semicustom NEXMI ASIC, but the ROMBUS itself was eventually burdened with 12 separate devices. This presented a large capacitive and direct current load to all the components on the bus. The UART and TOY chip in particular were not well suited to this large load. We attempted to find CMOS-equivalent devices for all the TTL components to eliminate the direct current problem, but were unsuccessful. We finally decided to split the bus into two sections, and place all the CMOS low-drive-capability devices on a separate segment. A transceiver connected the two segments.

Given that most of the devices on the ROMBUS were non-Digital components, we had the normal problem of obtaining accurate SPICE models to determine the ROMBUS timing. Extensive lab testing was performed on the devices to characterize their output delays under different capacitive loading conditions. These loading tests helped generate SPICE models for the ROMBUS simulations.

Sinal Integrity Considerations

Module signal integrity analysis was one of the most important aspects of the project. A system that includes components running as fast as the NVAX and NEXMI can easily see performance degradation or unreliability if the module signals are not carefully placed, routed, and terminated where necessary. The past experience of the signal integrity team played a major role in the eventual success of the process. Several approaches were taken for this aspect of the design.

Package Selection. SPICE simulations were performed on each level of the design. This included the chip, package, module, and backplane. Even though each particular simulation analysis was focused on one aspect of the design, we understood that each individual area affected the entire system. For example, the selection of a package spanned several important levels of design hierarchy. The connections between the die pads and the module signal pins, as well as the connection of the package to the board, needed to be considered; as did the module layout that accompanied each package size and type. One aspect of the signal integrity might show that a particular package type was superior to another, while another aspect

might favor a different approach to module component interconnection. Electrical SPICE simulation determined that the performance and reliability of a ceramic through-hole PGA on a 100-mil grid was the best trade-off.

NDAL and Backup Cache Simulation. The most important module-level signal integrity analysis was the extensive characterization of the NDAL and backup cache. As the interconnection bus between the NVAX and the NEXMI, the NDAL controls not only the transmission speed between the components, but also the speed at which the NVAX can run (since the NDAL is synchronous to and scales with the NVAX speed). The speed of the backup cache was a performance concern for obvious reasons.

We determined that estimates of the interconnect and routing would be insufficient for our aggressive timing goals. Several trial layouts were performed to provide accurate input for the SPICE simulations. The I/O drivers for both the NVAX and the NEXMI chips were known to be complete and accurate, since they were both designed internally. The timing requirements for reliable operation were also well understood for the same reason. The module-level SPICE simulations provided guidelines about the length and routing rules associated with each high-speed signal trace, such as:

- Daisy-chain routing of all signals
- Clock routing scheme (e.g., matched length and termination)
- Maximum length requirements for each type of network
- Treeing, or ordering, requirements for networks
- Impedance of different networks

These requirements were used as design guidelines. A cross-talk prediction program within the layout tool verified that the coupling between signals was within an acceptable range. After the prototype modules were delivered, measurements were taken of all the critical signals on the module, showing excellent correlation with the SPICE results.

Thermal Management

During the early phases of the module design process, the power dissipation of the NVAX chip was estimated to be as high as 20 watts, though the final figure for the 12-ns component that we shipped with the product was 14 watts. Cooling such a part presented a challenge to the module designers. Since the Model 600 was an upgrade option in the VAX 6000 family, there was no possibility of system modifications to improve the thermal design.

Figure 1, which is a picture of the Model 600 module, shows that the heat sink for the NVAX is larger than the package. The final dimensions of the heat sink are 3.285 inches by 3.285 inches by 0.325 inch, while the PGA package is only 2.2 inches by 2.2 inches. One of our limitations was the maximum component height of 0.420 inch on side 1 for any module in a

VAX 6000 backplane. This restriction forced the heat sink to grow wider rather than higher. The NVAX chip was also placed closer to the edge of the board, where the airflow provides maximum cooling. The final size of the NVAX heat sink was a compromise between system requirements, board area, and thermal performance.

9 Summary

The decision to use Digital's proprietary tool suite and fabrication process was proved correct by the quality of the VAX 6000 Model 600 and its delivery, on schedule, for NVAX CPU debugging and product shipment. Access to accurate information about the components allowed decisions to be made early, and entailed less risk. This advantage, coupled with a seasoned module development team and extensive functional, timing, and circuit simulation resulted in a successful project.

10 Acknowledements

The authors would like to acknowledge the following people for their contributions to the VAX 6000 Model 600 Processor: Chuck Benz, Mike Gowan, Chris Houghton, Dave Ives, Keith Johnston, Mike Kagen, Diane Kirkman, Doug Koslow, Bill LaPrade, Don MacKinnon, Lisa Noack, Del Ramey, Sharad Shah, Steve Thierauf, Mike Warren, and Beth Zeranski.

11 References and Note

1. B. Allison, "An Overview of the VAX 6200 Family of Systems," Digital Technical Journal, vol. 1, no. 7 (August 1988): 10-18.

2. G. Uhler et al., "The NVAX and NVAX+ High-performance VAX Microprocessors," Digital Technical Journal, vol. 4, no. 3 (Summer 1992, this issue): 11-23.

3. SPEC Newsletter, vol. 3, no. 4 (December 1991).

4. SPICE is a general-purpose circuit simulator program developed by Lawrence Nagel and Ellis Cohen of the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley.

5. S. Ross, Introduction to Probability Models (Orlando, Florida: Academic Press, Inc., 1985).

6. M. Kearney, "DECSIM: A Multi-Level Simulation System for Digital Design," Proceedings of IEEE International Conference on Computer Design: VLSI in Computers (1984): 206-209.

7. R. Brayton, ASV, and A. Wang, "MIS: A Multiple-Level Logic Optimization System," IEEE Transactions on Computer-Aided Design, vol. CAD-6, no. 6 (November 1987).

8. C. Sechen and A. Sangiovanni-Vincentelli, "Timberwolf3.2: A New Standard Cell Placement and Global Routing Package," Proceedings of the 23rd Design Automation Conference (1986): 432.

9. J. Reed, A. Sangiovanni-Vincentelli, and M. Santamauro, "A New Symbolic Channel Router: YACR2," IEEE Transactions on Computer-Aided Design, vol. 4 (July 1985): 208.

10.N. Chen, C. Hsu, and E. Kuh, "The Berkeley Building-Block (BBL) Layout System for VLSI Design," Digest of Technical Papers, IEEE International Conference on Computer-Aided Design (1983): 40.

11.M. Marek-Sadowska, "Two-Dimensional Router for Double Layer Layout," Proceedings of the 22nd Design Automation Conference (1985): 117.

12 Biographies

Lawrence Chisvin A principal hardware engineer in the Semiconductor Engineering Group, Larry Chisvin is involved in the design of modules and systems based on the DECchip 21064 microprocessor. Larry also provides technical support for customers, including Alpha AXP architecture presentations and example designs and application notes. Previously, he worked on processor and memory modules for the VAX 6000 Model 600. He holds a B.S.E.E. (summa cum laude) from Northeastern University and an M.S.E.E. from Worcester Polytechnic Institute. He is a member of the IEEE Computer Society and the ACM.

Gregg A. Bouchard Gregg Bouchard is a senior hardware engineer with the Semiconductor Engineering Group. His current responsibilities include the module design of a DECchip 21064 daughter card that contains a CPU-to-bus interface. Previously, Gregg worked on chip design of the NVAX-to-XMI bus interface for the VAX 6000 Model 600, and field programmable gate array chips for the VAXstation 4000 Model 90. Gregg joined Digital in 1986 after receiving his B.S.E.E. from the Rochester Institute of Technology. He also holds an M.S.E.E. from Northeastern University and has a patent pending related to hardware queue structure.

Thomas M. Wenners Thomas Wenners is a senior hardware engineer in the Semiconductor Engineering Group. He is responsible for the design of a next-generation VAX workstation CPU module and for CPU module designs of DECchip 21064 microprocessor{-}based products. Tom's previous work includes the module design of the VAX 6000 Model 600, module design and signal integrity support on ESB products, and analysis and evaluation of advanced chip and module packaging. Tom joined Digital in 1985. He received a B.S.E.E. (1985, cum laude) and an M.S.E.E. (1990) from Northeastern University.

13 Trademarks

SPICE is a trademark of the University of California at Berkeley.