The NVAX and NVAX+ High-performance VAX Microprocessors

By G. Michael Uhler, Debra Bernstein, Larry L. Biro, John F. Brown III, John H. Edmondson, Jeffrey D. Pickholtz, and Rebecca L. Stamm

1 Abstract

The NVAX and NVAX+ CPU chips are high-performance VAX microprocessors that use techniques traditionally associated with RISC microprocessor designs to dramatically improve VAX performance. The two chips provide an upgrade path for existing VAX systems and a migration path from VAX systems to the new Alpha AXP systems. The design evolved throughout the project as time-to-market, performance, and complexity trade-offs were made. Special design features address the issues of debug, maintenance, and analysis.

2 Introduction

The NVAX and NVAX+ CPUs are high-performance, single-chip microprocessors that implement Digital's VAX architecture.[1] The NVAX chip provides an upgrade path for existing systems that use the previous generation of VAX microprocessors. The NVAX+ chip is used in new systems that support Digital's DECchip 21064 microprocessor, which implements the Alpha AXP architecture.[2,3] These two NVAX chips share a basic design.

The high-performance, complementary metal-oxide semiconductor (CMOS) process used to implement both chips allows the application of pipelining techniques traditionally associated with reduced instruction set computer (RISC) CPUs.[4] Using these techniques dramatically improves the performance of the NVAX and NVAX+ chips as compared to previous VAX microprocessors and results in performance that approaches and may even exceed the performance of popular industry RISC microprocessors.

The chip design evolved throughout the project as the goals influenced the schedule, performance, and complexity trade-offs that were made. The two primary design goals were time-to-market, without sacrificing quality, and improved VAX CPU performance. Our internal goal was for the NVAX CPU performance to be more than 25 times the performance of a VAX-11/780 system in a datacenter system. Achieving these goals required meeting aggressive schedules and thus concentrating on the high-leverage design points and on an unprecedented verification effort.[5]

Support for multiple system environments, compatibility with previous VAX products and systems, and a means to migrate from traditional VAX systems to the new Alpha AXP platforms were also important design goals. These goals had a profound impact on the design of the cache protocols and the external bus interfaces. NVAX and NVAX+ engineers worked closely with engineers in Digital's systems groups during the definition of these operations.

The paper begins by comparing the basic features of the NVAX and NVAX+ chips and then describes in detail the chip interfaces and design elements. This description serves as the foundation for the ensuing discussion of design evolution and trade-offs. The paper concludes with information about the special design features that address the issues of debug, maintenance, and analysis.

3 Comparison of the NVAX and NVAX+ Chips

The NVAX and NVAX+ chips are identical in many respects, differing primarily in external cache and bus support. NVAX is intended for systems that use previously designed VAX microprocessors. The following systems currently use the NVAX chip: the VAXstation 4000 Model 90; the MicroVAX 3100 Model 90; the VAX 4000 Models 100, 400, 500, and 600; and the VAX 6000 Model 600.[6,7,8,9] NVAX supports an external write-back cache that implements a directory-based broadcast coherence protocol that is compatible with earlier VAX systems.[10]

NVAX+ is designed for systems that use the DECchip 21064 microprocessor implementation of the Alpha AXP architecture and is currently used in the VAX 7000 Model 600 and the VAX 10000 Model 600 systems. NVAX+ supports an external cache and bus protocol that is compatible with that of the DECchip 21064 microprocessor. In existing systems, NVAX+ is configured to support an external write-back cache that implements a conditional write-update snoopy coherence protocol.[11]

The two CPU chips provide both the means to upgrade installed VAX systems, thus protecting previous investments, and a migration path from a VAX microprocessor to a DECchip 21064 microprocessor in the new Alpha AXP systems.

4 Chip Interfaces

The NVAX chip interfaces to an external write-back cache (B-cache) through a private port with tag and data static random-access memories (RAMs) on the module, as shown in Figure 1. The size and speed of the cache are programmable, allowing the chip to accommodate a range of possible system configurations.

The NVAX data and address lines (NDAL) constitute a 64-bit bidirectional external bus with associated control signals that operates at one-third the frequency of the CPU from clocks provided by the CPU. Addresses and data are time-multiplexed and, to provide high performance, are overlapped with arbitration for future transactions and acknowledgment of previous transactions. The NDAL bus protocol allows up to two disconnected reads and multiple write-backs to be outstanding at the same time, using identifiers to distinguish the different transactions. External interrupt requests are

received through dedicated lines and arbitrated by logic in the CPU.

The NVAX+ chip interfaces to an external write-back B-cache implemented with tag and data static RAMs on the module through a port shared with system control logic, as shown in Figure 2. Responsibility for controlling the cache port is shared between NVAX+ and the system environment; the NVAX+ chip handles the common cases of read hit and exclusive write, and the system environment provides cache policy control for other events. The size and speed of the cache can be configured to allow a range of possible system configurations.

The DECchip 21064 data and address lines (EDAL) constitute a demultiplexed, bidirectional bus with 29 bits of address, 128 bits of data, and the associated control signals. This bus operates at one-half, one-third, or one-fourth the frequency of the CPU from clocks provided by the CPU. The speed of the system clocks can be programmed to accommodate various RAM and system speeds. At power-up time, initialization information, including RAM timing, and diagnostics are loaded from a serial read-only memory (ROM) into the on-chip cache. The external interrupt handling is similar to that of the NVAX chip.

5 Electrical and Physical Design

Process technology, clocking scheme, clock frequency, and die specifications are elements of the electrical and physical design of the NVAX and NVAX+ chips. Both chips are implemented in Digital's fourth-generation complementary metal-oxide semiconductor (CMOS-4) technology. CMOS-4 is a 0.75-micrometer, 3.3-volt process with support for 5-volt input signals at the pins. The CMOS-4 process is optimized for high-performance microprocessors and provides short (0.5-micrometer) channel lengths and three layers of metal interconnect. This robust and reliable process has been used to produce NVAX chips in volume for more than a year and is the same CMOS process used in the DECchip 21064 microprocessor.

NVAX and NVAX+ use a four-phase clocking scheme, driven by an oscillator that operates at four times the internal clock frequency. The oscillator frequency is divided by an on-chip, finite-state-machine clock generator; a low-skew clock distribution network is used for both internal and external clocks.

To meet the needs of the system designer, the two chips are designed for use at various frequencies. At present, NVAX is used in systems at internal clock frequencies of 83.3 megahertz (MHz) (12-nanosecond [ns] clock cycles), 74.4 MHz (14-ns clock cycles), and 62.5 MHz (16-ns clock cycles). NVAX+ is used in systems at a frequency of 90.9 MHz (11-ns clock cycles).

Each chip contains 1.3 million transistors on a die that is 16.2-by-14.6 millimeters in size. NVAX is packaged in a 339-pin, through-hole pin grid

array. NVAX+ is packaged in a 431-pin, through-hole pin grid array.

6 Architecture Design

The NVAX/NVAX+ design is partitioned into five relatively autonomous functional units: the instruction fetch and decode unit (I-box), the integer and logical instruction execution unit (E-box), the floating-point execution unit (F-box), the address translation and primary cache interface (M-box), and the external cache and system bus interface (C-box). Queues placed at critical interface boundaries normalize the rate at which the units process instructions. A block diagram of the NVAX and NVAX+ core is shown in Figure 3.

The I-box

The I-box fetches and decodes VAX instructions, evaluates operand specifiers, and queues operands in canonical form for further processing. Included in the I-box is a 2-kilobyte (KB), direct-mapped virtual instruction cache (VIC) with 32-byte cache blocks. For reliability, the VIC includes parity protection on both tags and data.

During each cycle, the I-box attempts to fetch 8 bytes of instruction data from the VIC and place this data in an empty slot in the prefetch queue (PFQ). A VIC miss incurs a three-cycle penalty if the requested data is found in the primary cache. PFQ data is then decoded into the next VAX instruction component, which may be one of the following: operation code (opcode) and first specifier or branch displacement, subsequent specifier, or implicit specifier (an imaginary specifier included to improve the performance of some instructions). The I-box enters the opcode-related information into the instruction queue, the pointers to source and destination operands into their respective source and destination queues, and the branch-related information into the branch queue.

For operand specifiers other than short literal or register mode, the I-box decode logic invokes the pipelined complex specifier unit (CSU) to compute the effective address and initiate the appropriate memory request to the M-box. The CSU is similar in function to the load/store unit on many traditional RISC machines.

The I-box automatically redirects the program counter (PC) to the target address when it decodes one of the following instruction types: unconditional branch, jump, and subroutine call and return. The branch-taken penalty is two cycles for any conditional or unconditional branch. To keep the pipeline full across conditional branches, the I-box includes a 512-bit by 4-bit branch prediction array. The prediction is entered in the branch queue by the I-box and compared with the actual branch direction by the E-box. If the I-box predicts incorrectly, the E-box invokes a trap mechanism to drain the pipeline and restart the I-box at the alternate PC. A branch mispredict incurs a four-cycle penalty for a branch that is actually taken and a six-cycle penalty for a branch that is not taken.

The E-box

The E-box is responsible for the execution of all non-floating-point instructions, for interrupt and exception handling, and for various overhead functions. All functions are microcode-controlled, i.e., driven by a microsequencer with a 1,600-word control store and a 20-word patch capability. Since the control store does not limit the cycle time, we chose to implement a single microcode control scheme, rather than hardwire control for the simple instructions and provide microcode control for the

remaining instructions.

The E-box begins instruction execution based on information taken from the instruction queue. References to specifier operands and results are made indirectly through pointers in the source and destination queues. In this way, most E-box instruction flows do not need to know whether operands or results are in register, memory, or instruction stream.

To improve the performance of certain critical instructions, the E-box contains special-purpose hardware. A mask processing unit finds the next bit set in a mask register and is used in the following instructions: FFC, FFS, CALLS, CALLG, RET, PUSHR, and POPR. A population counter provides the number of bits set in a mask and is used in the CALLS, CALLG, PUSHR, and POPR instructions. In addition, microcode can operate the arithmetic logic unit (ALU) and shifter independently to produce two computations per cycle, which can significantly improve the parallel operation of the complex instructions.

In addition to normal instruction processing, the E-box performs all power-up functions and interrupt and exception processing, directs operands to the F-box, and accepts results from the F-box. To guarantee that instructions complete in instruction stream order, the E-box orchestrates result stores and instruction completion between the E-box and F-box.

The F-box

The F-box performs longword (32-bit) integer multiply and floating-point instruction execution. The E-box supplies operands, and the F-box transmits results and status back to the E-box.

The F-box contains a four-stage, floating-point and integer-multiply pipeline, and a nonpipelined, floating-point divider. Subject to operand availability, the F-box can start a single-precision, floating-point operation during every cycle, and a double-precision, floating-point or integer-multiply operation during every other cycle.

Stage 1 of the pipeline calculates operand exponent difference, adds the fraction fields, performs recoding of the multiplier, and computes three times the multiplicand. Stage 2 performs alignment, fraction multiplication, and zero and leading-one detection of the intermediate results. Stage 3 performs normalization, fraction addition, and a miniround operation for floating-point add, subtract, and multiply instructions. Stage 4 performs rounding, exception detection, and condition code evaluation.

Stage 3 performs a miniround operation on the result calculated to that point to determine if a full-round operation is required in Stage 4. To do this, a round operation is performed on only the low-order three (for single-precision) or six (for double-precision) fraction bits of

the result. If no carry-out occurs for this operation, the remaining fraction bits are not affected and the full stage 4 round operation is not required. If the full round is not required, stage 4 is dynamically bypassed, resulting in an effective three-stage pipeline.

The M-box

The M-box is responsible for address translation, access checking, and access to the primary instruction and data cache (P-cache). The M-box accepts requests from multiple sources and processes these requests in an order that reflects both the priority of the request and the need to maintain instruction stream ordering of memory reads and writes. Address translation and cache access are fully pipelined; the M-box can start a new request at the beginning of every cycle.

The M-box performs address translation and access checking by means of a 96-entry, fully associative translation buffer (TB) with parity protection. If a TB miss occurs, the M-box automatically invokes a hardware miss sequence that calculates the address of the page table entry (PTE) that maps the page, fetches the PTE from memory, refills the TB, and restarts the reference. TB allocation is performed using a not-last-used scheme, which is similar to a round-robin but guarantees that the most recently referenced entry will not be overwritten. The M-box reports access violations and page faults to the E-box, and E-box microcode processes these misses with hardware support from the M-box.

The M-box also translates memory destination operand addresses provided by the I-box and saves the corresponding physical address in the physical address (PA) queue. When the E-box stores a result, the M-box matches the data with the next address in the PA queue and converts this data to a normal write request. The PA queue is also used to check for conflicts in read requests to a location in which nothing has been written.

The P-cache is an 8KB, two-way set-associative cache with 32-byte blocks and parity protection on tags and data. The P-cache can be configured to cache instructions, data, or both, and usually has the latter configuration. For compatibility with the DECchip 21064 microprocessor, the NVAX+ P-cache can also be configured into a direct-mapped organization.

The NVAX C-box

The NVAX C-box maintains the interface to the external B-cache and to the NDAL bus. The C-box receives read and write requests from the M-box and monitors the NDAL for activity that would require an invalidate operation in either cache. Consecutive writes to the same quadword (64 bits) are merged into a single quadword datum by packing logic placed at the input of an eight-entry quadword write queue.

The C-box can accept one instruction read request and one data read request from the M-box. Conflict logic in the write queue allows nonconflicting read requests to be processed before queued write requests are performed. Conflicts are resolved by processing write queue entries until the

conflicting write is completed.

The C-box supports four B-cache sizes: 128KB, 256KB, 512KB, and 2 megabytes (MB). The system designer can independently select tag and data RAM speeds to meet system requirements, regardless of the frequency at which the CPU is running. The B-cache block size is 32 bytes, and both tag and data RAMs are protected with error correction code (ECC) that corrects single-bit errors and detects both double-bit errors and full 4-bit RAM failures.

The B-cache implements a directory-based broadcast coherence protocol in conjunction with a memory directory containing one bit per 32-byte block. Each memory directory bit indicates if the associated block is valid in memory or has been written and exists in a cache. Unwritten blocks may exist in multiple caches in the system. Written blocks may exist in exactly one cache.

An attempt to write to a block that is not both valid and already written in the B-cache causes the C-box to request write permission from memory by means of a special NDAL bus read command. The memory controller will not respond to any NDAL bus transactions to a block that is written in a cache. Instead, it waits for the CPU, which contains an updated copy of the block, to write the block back to memory and then completes the original transaction. All CPUs in the system monitor the NDAL bus for read and write transactions and compare the address against their B-cache tags. If a match is found, the cache block is either written back to memory, invalidated, or both, depending on the transaction type and the state of the block in the cache.

The NDAL protocol fully supports multiprocessing implementations and does not require any special chip variants to construct a multiprocessor system. The C-box invokes invalidate or write-back requests as required to keep the B-cache and P-cache coherent with NDAL activity.

The NVAX+ C-box

The NVAX+ C-box provides the interface between the internal functional units and the EDAL pin bus implemented by the DECchip 21064 microprocessor. This C-box interface includes the basic interface control for the external B-cache and for the memory and I/O system. The NVAX+ C-box receives read and write requests from the M-box. These requests are queued and arbitrated within the C-box and result in cache or system access across the EDAL. The NVAX+ C-box also maintains cache coherency by sending invalidate requests to the M-box when requested by external logic.

The NVAX+ C-box implementation provides many of the same features and performance enhancements available in the NVAX C-box. Included is support for software-programmable B-cache speeds (one-half, one-third, or one-fourth times the CPU frequency) and sizes (128KB to 8MB), write packing, write queuing, and read-write reordering. In addition, the NVAX+ C-box

supports the newer platforms and increases the degree to which NVAX+ is compatible with the DECchip 21064 microprocessor. NVAX+ C-box features include programmable system clock speeds, I/O space-mapping, and a direct-mapped option on the P-cache.

A major difference between the NVAX and NVAX+ implementations is in the B-cache coherence protocol. Rather than mandate a fixed B-cache coherence protocol, the NVAX+ implementation allows systems to tailor the protocol to their particular needs. NVAX+ cache coherency is implemented jointly by off-chip system support logic and by the CPU chip, with relevant information passed between the two over the EDAL bus. To allow duplicate cache tag stores (if they exist) to be properly updated, the NVAX+ C-box provides information to off-chip logic, indicating when the internal caches are updated. External logic notifies the NVAX+ C-box when an internal cache entry needs to be invalidated because of external bus activity.

Existing systems configure the B-cache to implement a conditional write-update snoopy protocol carried out using shared and written signals on the system bus. Writes to shared blocks are broadcast to other caches for conditional update in those caches. A CPU that receives a write update checks the NVAX+ P-cache to determine if the block is also present in that cache. If the block is present, the B-cache update is accepted and written into the B-cache, and the P-cache is invalidated. If the data is not present in the P-cache, the B-cache is invalidated. This results in a write-update protocol for data that was recently referenced by a CPU (and hence is valid in the P-cache) and reduces to a write-invalidate protocol for data that was not recently referenced.

To accommodate the programmable nature of both the system and cache clock frequencies, the NVAX+ C-box supports nine different combinations of cache and system clock frequencies. This support allows efficient use of the chip in a wide range of different performance class systems.

Pipeline Operation

The NVAX and NVAX+ chips implement a macropipeline. Multiple VAX macroinstructions are processed in parallel by relatively autonomous functional units with queued interfaces at critical boundaries. Each functional unit also has an internal pipeline (micropipeline) to allow a new operation to start at the beginning of every cycle. The pipeline operation can be logically depicted, as shown in Figure 4.

In pipeline segment S0, instruction stream data is read from the VIC. The next VAX instruction component is parsed, and queue entries are made in segment S1. For short literal and register specifiers, no other processing is required. Requests for further processing for all other specifiers are queued to the CSU pipeline, which reads operand base addresses in segment S2, calculates an effective address, and makes any required M-box request contained in segment S3. If an M-box request is made, address translation and P-cache lookup occur in segments S4 and S5.

Instruction execution starts with an E-box control store lookup in segment

S2, followed by a register file read of any required operands in segment S3, an ALU and/or shifter operation in segment S4, and a potential result store or register file write in segment S5. If an M-box request is required, e.g., for a memory store, the request is made in segment S4;

address translation or PA queue access occurs in segment S5; and a P-cache access occurs in segment S6.

Floating-point and integer-multiply instruction execution starts in the E-box, which transfers operands to the F-box. The four-stage F-box pipeline is skewed by half a cycle with respect to the E-box pipeline, beginning halfway through segment S4. The fourth segment of the F-box pipeline is conditionally bypassed if a full-round operation is not required. The result is transmitted back to the E-box, logically in segment S5 of the pipeline.

Pipeline bypasses exist for all important cases in the I-box and E-box pipelines, so that there are no stalls for results feeding directly into subsequent operands. The M-box processing of memory references initiated as a result of operand specifier processing by the I-box is usually overlapped with the execution of the previous instruction in the E-box, with few or no stalls occurring on P-cache hit.

7 Design Evolution and Trade-offs

The NVAX and NVAX+ chips are the latest in a line of CMOS VAX microprocessors designed by Digital's engineers and represent a continuing evolution of architectural concepts from one implementation to the next. The preceding chip design was the CPU for the VAX 6000 Model 400 system.[12] To meet the time-to-market and performance goals, we had to modify the NVAX/NVAX+ design throughout the project.

One of the early vehicles for making design trade-offs was the NVAX performance model, which predicts CPU and system performance and aids in quantifying the performance impact of various design options. The performance model is a detailed, trace-driven model which can be easily configured by changing any of a variety of input parameters. The model stimuli used were 15 generic timesharing and 22 benchmark instruction trace files that were captured by running actual programs on existing VAX systems.

The following sections describe the evolution of the chip design, including the number of chips, the pipelining technique used, and various cache issues.

Number of CPU Chips

The VAX 6000 Model 400 core CPU implementation is a three-chip design: a processor chip, with a small on-chip primary cache; a floating-point chip; and a secondary cache controller, with internal cache tags. The initial attempt at NVAX CPU definition was a two-chip design. One chip contained the I-box (with a 4KB VIC), the E-box, the F-box, and the M-box (with a

16KB, direct-mapped P-cache). The second chip held the C-box and the B-cache tag array. The project design goals, especially time-to-market, led to a single-chip solution, rather than a two-chip design.

To condense the design from two chips to one, we halved the sizes of the VIC and the P-cache and moved the B-cache tags to external static RAMs, leaving the B-cache controller on-chip. Later, we were able to reduce the penalty of halving the size of the P-cache by making it two-way set associative rather than direct mapped. With these changes, the performance model showed a performance loss of less than 1.4 percent across all the traces, relative to the two-chip design, with a worst-case penalty of 3.9 percent.

There are strong advantages to the single-chip solution.

1. Designing a single chip takes less time.

2. This design requires the production and maintenance of only one design database and one mask set.

3. Latency to the B-cache is shorter.

4. An off-chip tag store provides more flexibility in B-cache configurations.

Macropipelining

Run-time performance is the product of the cycle time, the average time to execute an instruction (cycles per instruction [CPI]) and the number of instructions executed. CMOS process improvements made it possible to decrease the NVAX/NVAX+ cycle time with respect to the previous generation of VAX microprocessors, thus improving the first factor in run-time performance.

The VAX 6000 Model 400 CPU design uses traditional microinstruction pipelining, i.e., micropipelining, to achieve some amount of overlap and to decrease the CPI. However, using micropipelining techniques would not reduce the NVAX/NVAX+ CPI to the level required to meet the performance goals of the NVAX/NVAX+ projects. We achieved this reduction by using RISC design and implementation techniques referred to as macropipelining. In a macropipelined architecture, the I-box acts much like a load/store engine, dynamically prefetching operands prior to instruction execution. Using the macropipeline technique in the NVAX and NVAX+ CPUs makes it possible to retire one basic complex instruction set computer (CISC) macroinstruction per cycle, as in a simple RISC design. Although macropipelining introduced considerable complexity into the NVAX/NVAX+ design, this complexity resulted in a significant performance improvement over a traditional micropipelined design.

Number of Specifiers per Cycle

The NVAX/NVAX+ I-box can parse at most one opcode and one VAX specifier per cycle. The I-box design initially considered was capable of parsing two specifiers per cycle. Although this parsing scheme represented significant complexity and circuit risk, intuitively, it seemed important to quickly retire specifiers in the I-box in order to keep the macropipeline full. However, the performance model predicted a maximum performance improvement

of less than two percent on our traces, and we decided to limit complexity and schedule risk by parsing only one specifier per cycle.

F-box Design

The NVAX F-box design is highly leveraged from the VAX 6000 Model 400 F-chip design. Rather than start from scratch, we integrated the existing design onto the NVAX and NVAX+ CPU chips and added a final-stage bypass mechanism. In addition, unlike the original F-chip implementation, the NVAX/NVAX+ control of the F-box allows a fully pipelined operation, which significantly improves floating-point performance over the F-chip design. Although a totally new design would have had shorter floating-point latencies, the combination of a fully pipelined operation and a final-stage bypass allowed us to achieve our performance goal, while meeting our time-to-market goal.

Cache Coherence

Performance studies with the previous generation of VAX microprocessors clearly indicate that system bus write bandwidth limits performance unless an external write-back cache is implemented. In addition, the VAX architecture required that we implement the cache coherence protocol in hardware.

The NVAX implementation uses a directory-based coherence protocol for compatibility with existing and planned target system platforms. The NDAL bus supports multiple outstanding read and write requests, which allows the microprocessor to utilize the capability of the system bus to process these operations in a pipelined fashion. We investigated the possibility of implementing both directory-based and snoopy coherence protocols, but time-to-market considerations and the opportunity to optimize the design for performance in existing system platforms outweighed the desirability of supporting snoopy protocols.

For the NVAX+ implementation, the coherence policy is determined by hardware external to the NVAX+ chip, in the given system. The NVAX+ cache and system interface allows the system environment to implement a variety of coherence protocols. Compatibility with the DECchip 21064 interface definition required limiting NVAX+ to one outstanding external cache miss. However, this limitation is more than offset by the significantly better main memory access times achieved in target systems.

One significant advantage of the NVAX+ scheme is that most policies associated with the external cache are determined by hardware outside the NVAX+ chip (such as the coherence policy), allowing the chip to be used in a wide variety of systems. Implementing the DECchip 21064 interface on NVAX+ greatly reduces the hardware engineering investment required to

deliver a VAX CPU and an Alpha AXP CPU in the same system environment.

For both the NVAX and the NVAX+ chips, cache coherence is maintained for the P-cache by keeping it a subset of the external cache. Externally originated invalidate requests are forwarded to the P-cache only when the block is in the external cache. This minimizes the number of P-cache cycles spent processing invalidate requests. The two-way set-associative P-cache might have been slightly more effective if it were not a subset of the larger direct-mapped external cache. However, this effect is far less significant than the effect of expending a P-cache cycle for every external invalidate event.

Virtual caches almost always have lower latency than physical caches and usually do not require a dedicated translation buffer. The VAX architecture supports the use of a VIC by allowing the cache to be incoherent with respect to the data stream, i.e., not updated with recent writes by the CPU containing the VIC, or by any other CPU. However, some mechanism must be defined to make the VIC coherent with the data stream. In the VAX architecture, the execution of the VAX return from interrupt or exception (REI) instruction performs this function.

We chose to perform a complete flush of the VIC as part of the execution of every REI instruction. Because an REI always follows a process context switch, a flush during an REI removes the process-specific virtual addresses of the previous process and prevents conflict with (potentially identical) virtual addresses for the new process. We could have also chosen to keep the VIC coherent with the data stream and implement per-process qualifiers that would have made per-process virtual addresses unique. However, coherence would have required both an invalidate address and a control path to the VIC, and some form of backmap to resolve virtual address aliases. Per-process qualifiers would have required a VAX architecture change and significant operating system software changes. To reduce project risk, we chose to flush the VIC on every REI instruction.

Cache Hierarchy

The NVAX and NVAX+ chips have three levels of cache hierarchy: the VIC, the P-cache, and the B-cache. The VIC and P-cache are fully pipelined and have minimum latency, which allows instructions to be fetched and processed in parallel at very high rates.

The default P-cache configuration causes VIC misses to be looked up in the P-cache. This lookup process is advantageous since the VIC typically experiences a smaller miss penalty because latency for P-cache hits is roughly one-third that for external cache hits. The disadvantage is that instruction fills can result in a higher P-cache data stream miss rate, because they replace data that is likely to be referenced again. We used the performance model with available traces to determine that looking up VIC misses in the P-cache generally resulted in higher performance. In

specific applications, higher performance can be achieved by not looking up instruction references in the P-cache. As a result, we implemented P-cache configuration bits that allow system designers to implement either scheme. By default, NVAX and NVAX+ systems are configured to enable both

instruction and data caching in the P-cache, but this may be changed by the console software in certain systems to support prepackaged application systems.

External Cache Size

Both NVAX and NVAX+ support multiple external cache sizes to allow system designers full flexibility in selecting external cache configurations. With existing static RAM technology, smaller external cache configurations are usually faster than larger configurations. Performance modeling indicated that many applications, especially some popular benchmarks, fit entirely in in a cache whose size is 512KB or less, resulting in slightly better performance. However, many common applications utilize more memory than will fit in such caches and benefit more from an external cache whose size is 1MB to 4MB, even with the additional latency involved. As a result, our system designs use larger but slightly slower external cache configurations.

Block Size

During the analysis of the previous generation of VAX microprocessors in existing systems, we observed that the 16-byte block size was too small to achieve optimal performance on many applications. As a result, we chose a 32-byte block size for the NVAX and NVAX+ internal caches. This size provides a good balance between fill size and the number of cycles required to do the fill, given 8-byte fill data paths.

For compatibility with installed systems, the size of the NVAX external cache block and the cache fill size is 32 bytes. On NVAX+, the external cache block size may be larger and is 64 bytes in the VAX 7000 Model 600 and VAX 10000 Model 600 systems. Because both systems implement low-latency memory and high-bandwidth buses, the increase in external cache block size results in better performance.

8 Special Features

The NVAX/NVAX+ design includes several features that supplement core chip functions by providing added value in areas of debug, system maintenance, and systems analysis. Among the features are the patchable control store (PCS) and the performance monitoring hardware.

Patchable Control Store

The base machine microcode is stored in a ROM control store in the E-box. The 1,600-microword capacity of the E-box controls macroinstruction execution and exception handling. The PCS consists of 20 entries that can be configured to replace or supplement the microcode residing in ROM. Each

PCS entry contains a content-addressable memory (CAM)/RAM pair that stores the patch microword address and patch microword, respectively. The ROM control store and the PCS are accessed in parallel. Typically, words are fetched from the ROM control store, but if a microword address matches the

CAM in one of the PCS entries, then the PCS RAM for that entry supplies the microword, and the ROM output is disabled.

Privileged software controls the loading of the PCS by means of internal processor registers. In system operation, a patch file is normally loaded into the PCS early in the boot procedure, so that any minimal system capable of starting system boot can install patches to the base microcode. This feature presents a way to modify the base NVAX/NVAX+ chip through software; the majority of engineering change orders (ECOs) can be accomplished by releasing new patch files, thus alleviating the need to change the hardware design and retool for the very large-scale integration (VLSI) fabrication.

We booted the VMS operating system within 16 days of receiving first-pass wafers from fabrication, a tribute to a very thorough design verification. However, the continuing rigorous testing on prototype systems revealed several problems with the base microcode and hardware. The PCS mechanism helped to identify, isolate, and work around many of the problems during system debug and thus allowed extensive system testing to continue on first-pass chips.

For example, we used a sequence of PCS patches during system debug to isolate an obscure failure whose symptom was a transfer to virtual address 0. By patching the main microcode exception handling routine to check for this event, we identified the instruction stream sequence that was causing the failure. We refined the patch to place additional checking into various instructions in the sequence. This refinement allowed us to isolate the exact instruction that was causing the transfer to PC 0. With this information, we were then able to reproduce the problem in simulation and correct the second-pass design. Without this diagnostic capability, we probably would have needed weeks or months of additional debug time to isolate the failure.

In addition to using the powerful diagnostic capability of the PCS, we used patches to correct or work around the few functional bugs that remained in the first-pass design. For example, a microcode patch was used to correct a condition code problem caused by a microcode bug during the execution of an integer-multiply instruction. Because the E-box is central to the execution of all instructions, we were also able to use patches to correct hardware problems in other boxes. In one instance, a patch was used to inject a synchronization primitive into the M-box in order to correct an M-box design error. As a result of the simplicity and elegance of this solution, the final second-pass correction was to move the patch into microcode ROM, rather than modify the M-box hardware design.

Performance Monitoring Environment

As computer designs increase in complexity, their dynamic behavior becomes less intuitive. Computer designers rely more and more on empirical performance data to aid in the analysis of system behavior and to provide a basis for making hardware and software design decisions. In addition, multiple levels of logic integration on VLSI chips restrict the collection of this performance data, because many of the interesting events are no longer visible to external instrumentation. The NVAX/NVAX+ chip design includes hardware multiplexers and counters that can be configured to count any of a set of predetermined, internal state changes.

Two 64-bit performance counters are maintained in memory for each CPU in an NVAX/NVAX+ system. The lower 16 bits of each counter are implemented in hardware in the CPU, and at specified points, the quadword counters in memory are updated with the contents of the hardware counters. Privileged software can be used to configure the hardware counters to count any one of a basic set of internal events, such as cache access and hit, TB access and hit, cycle and instruction retire, and cycle and stall. When the 16-bit counters reach a half-full state, the performance monitor requests an interrupt. The interrupt is serviced in a normal way, i.e., between instructions (or in the middle of interruptible instructions) and at an architecturally specified interrupt priority level. Unlike other interrupts, the performance monitor logic interrupt is serviced entirely in microcode and then dismissed; no software interrupt handler is required.

The microcode component updates the counters in memory when it services the performance monitor interrupt. During a counter update, the microcode temporarily disables the counters, reads and clears the hardware counters, updates the counters in memory, enables the counters, and resumes instruction execution. The base address of the counters in memory is taken from a system vector table and offset by the specific CPU number, creating a data structure in memory that contains a pair of 64-bit counters for each CPU.

Combining the use of hardware, software, and the PCS created a versatile performance monitoring environment{-}one that goes beyond the scope of the basic hardware capabilities. In this environment, we can correlate the counts with higher-level system events and change the representation of the collected data. For example, microcode can enable the counters every time a process context is loaded and disable the counters when a process context is saved. This feature allows us to set up workloads and gather dynamic statistics on a per-process basis. We can also use PCS patches to modify the memory counter address in order to provide an additional offset based on one of the five VAX processor operating modes: interrupt, kernel, executive, supervisor, or user. This technique provides a new performance counter data structure that collects statistics on a per-mode,

per-process, per-CPU basis. Also, microcode patches can be used to add context checks that filter and count various events. For example, we can patch the VAX context switch instruction to count context switches or patch

the interlocked instructions to count the number and types of accesses to multiprocessor synchronization locks.

The performance monitoring environment is a powerful tool that we have used to collect the data required to analyze hardware and software behavior and interactions, and to develop an understanding of system performance. We have applied this knowledge to tune the performance of operating systems and application software, and continue to apply the knowledge to improve the design and performance of future hardware and software.

9 Results and Conclusions

With a focus on time-to-market, we shortened the originally projected NVAX design schedule, from the start of implementation to the completion of the chip design, by 27 percent. We booted the operating system just 16 days after prototype wafers became available. The use of the PCS allowed us to quickly debug and work around the few functional bugs that remained in the first-pass design. Because of the quality achieved in first-pass chips, we were able to shorten the schedule from chip design completion to system product delivery. As a result, systems were delivered to customers four months earlier than the originally projected date.

At the same time, we were able to dramatically improve CPU performance relative to previous VAX microprocessors by implementing a macropipelined design, in which multiple autonomous functional units cooperate to execute VAX instructions. Our internal goal was performance in excess of 25 times the performance of the VAX-11/780 system. We significantly exceeded this goal as demonstrated by the following Standard Performance Evaluation Cooperative (SPEC) Release 1.2 performance ratings:[13]

SPECmark 40.5

SPECfp 48.8

SPECint 30.4

These ratings were measured on a VAX 6000 Model 600 system at the initial announcement and are two to three times higher than those for the previous VAX microprocessor running in the same system. Software and system tuning has subsequently improved the initial numbers on all systems.

The NVAX/NVAX+ design provides an upgrade path and system investment protection to customers with installed VAX systems, as well as a migration path from an NVAX+ microprocessor to a DECchip 21064 microprocessor in the new Alpha AXP systems.

10 Acknowledgements

We would like to acknowledge the contributions of the following people:

W. Anderson, E. Arnold, D. Asher, R. Badeau, I. Bahar, M. Benoit, B. Benschneider, D. Bhavsar, M. Blaskovich, P. Boucher, W. Bowhill, L. Briggs, S. Butler, R. Calcagni, S. Carroll, M. Case, R. Castelino, A. Cave, S. Chopra, M. Coiley, E. Cooper, R. Cvijetic, R. Davies, M. Delaney, D. Deverell, A. DiPace, C. Dobriansky, D. Donchin, R. Dutta, D. DuVarney, J. J. Ellis, J. P. Ellis, H. Fair, B. Feaster, T. Fischer, T. Fox, G. Franceschi, N. Geagan, S. Goel, M. Gowan, J. Grodstein, P. Gronowski, W. Grundmann, H. Harkness, W. Herrick, R. Hicks, C. Holub, J. Huber, A. Jain, K. Jennison, J. Jensen, M. Kantrowitz, R. Khanna, R. Kiser, D. Koslow, D. Kravitz, K. Ladd, S. Levitin, J. Licklider, J. Lundberg, R. Marcello, S. Martin, T. McDermott, K. McFadden, B. McGee, J. Meyer, M. Minardi, D. Miner, E. Nangia, L. Noack, T. O'Brien, J. Pan, H. Partovi, N. Patwa, V. Peng, N. Phillips, R. Preston, R. Razdan, J. St. Laurent, S. Samudrala, B. Shah, S. Sherman, W. Sherwood, J. Siegel, K. Siegel, C. Somanathan, C. Stolicny, P. Stropparo, R. Supnik, M. Tareila, S. Thierauf, N. Wade, S. Watkins, W. Wheeler, G. Wolrich, Y. Yen

11 References

1. R. Brunner, ed.,VAX Architecture Reference Manual, Second Edition, (Bedford, MA: Digital Press, 1991).

2. D. Dobberpuhl et al., "A 200MHz 64b Dual-Issue CMOS Microprocessor," IEEE International Solid-State Circuits Conference, Digest of Technical Papers, vol. 35 (1992): 106-107.

3. R. Sites, ed., Alpha Architecture Reference Manual, (Burlington, MA: Digital Press, 1992).

4. D. Fite, Jr. et al., "Design Strategy for the VAX 9000 System," Digital Technical Journal, vol. 2, no. 4 (Fall 1990): 13-24.

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

6. M. Callander et al., "The VAXstation 4000 Model 90," Digital Technical Journal, vol. 4, no. 3 (Summer 1992, this issue): 82-91.

7. J. Crowell and D. Maruska, "The Design of the VAX 4000 Model 100 and MicroVAX 3100 Model 90 Desktop Systems," Digital Technical Journal, vol. 4, no. 3 (Summer 1992, this issue): 73-81.

8. J. Crowell et al., "Design of the VAX 4000 Model 400, 500, and 600

Systems," Digital Technical Journal, vol. 4, no. 3 (Summer 1992, this issue): 60-72.

9. L. Chisvin, G. Bouchard, and T. Wenners, "The VAX 6000 Model 600 Processor," Digital Technical Journal, vol. 4, no. 3 (Summer 1992, this issue): 47-59.

10. A. Agarwal et al., "An Evaluation of Directory Schemes for Cache Coherence," Proceedings of the 15th Annual International Symposium on Computer Architecture, (May 1988): 280-289.

11. P. Stenstrom, "A Survey of Cache Coherence Schemes for Multiprocessors," Computer (June 1990): 12-24.

12. H. Durdan et al., "An Overview of the VAX 6000 Model 400 Chip Set," Digital Technical Journal, vol. 2, no. 2 (Spring 1990): 36-51.

13. VAX 6000 Datacenter Systems Performance Report (Maynard, MA: Digital Equipment Corporation, 1991).

12 Biographies

G. Michael Uhler Michael Uhler is a senior consultant engineer in the Semiconductor Engineering Group, where he leads the advanced development effort for a new high-performance microprocessor. As chief architect for the NVAX and REX520 microprocessors, Mike was responsible for the CPU architecture, performance evaluation, behavioral modeling, CPU microcode, and CPU and system debug. He received a B.S.E.E. (1975) and an M.S.C.S (1977) from the University of Arizona and joined Digital in 1978. Mike is a member of IEEE, ACM, Tau Beta Pi, and Phi Kappa Phi and holds eight patents.

Debra Bernstein Debra Bernstein is a consultant engineer in the Semiconductor Engineering Group. She worked on the CPU design and architecture for the NVAX microprocessor and the VAX 8700/8800 systems and is currently co-architecture leader for a future Alpha AXP processor. Deb received a B.S. in computer science (1982, cum laude) from the University of Massachusetts in Amherst. She holds one patent, has two patent applications pending, and has coauthored several technical papers.

Larry L. Biro Larry Biro joined the Electronic Storage Development (ESD) Group in 1983, after receiving an M.S.E.E. from Rensselaer Polytechnic Institute. While in ESD, he contributed to the advanced development of solid-state disk products. Larry joined the Semiconductor Engineering Group as a custom circuit designer on the NVAX E-box. Later, as a member of the NVAX+ chip implementation team, Larry designed the clock and reset logic, coordinated back-end verification efforts, and co-led the chip debugging. Currently, he is the project leader for a future, single-chip VAX implementation.

John F. Brown After receiving an M.S.E.E. from Cornell University in 1980, John Brown joined the engineering staff at Digital. At present, he is a consultant engineer working on Alpha AXP microprocessor advanced development. John's previous responsibilities include managing the design

of the instruction decode section of the NVAX microprocessor. He also made technical contributions to the VAX 6000 Model 200 and 400 chip sets, and was hardware engineer for the extended floating-point enhancement to the VAX-11/780 system. John holds two patents and has seven applications pending.

John H. Edmondson John Edmondson is a principal engineer in the Semiconductor Engineering Group. At present, he is co-architect of a future RISC microprocessor. Previous to this, he was a member of the VAX 6000 Model 600 CPU chip design team. Before joining Digital in 1987, John designed minicomputers for five years at Canaan Computer Corporation. He also worked at Massachusetts General Hospital for two years, researching applications of technology to anesthesia and intensive care medicine. John received a B.S.E.E from the Massachusetts Institute of Technology in 1979.

Jeffrey D. Pickholtz Jeffrey Pickholtz received an A.S. (1977) in specialized technology from Penn Technical Institute and a B.S.E.E.T. (1989) from Central New England College. He joined Digital in 1977 as a technician and worked on a variety of midrange computers. More recently, his responsibilities have included technical contributions to the VAX 6000 Models 400 and 600, and VAX 7000 Model 600 chip sets. Currently, Jeff is a senior engineer in the Semiconductor Engineering Group, leading the implementation of a future CMOS microprocessor chip.

Rebecca L. Stamm Rebecca Stamm is a principal hardware engineer in the Semiconductor Engineering Group. She led the design of the backup cache, the bus interface, and the pin bus for the NVAX CPU chip. Rebecca then led the chip debug team from tapeout through final release of the design to manufacturing. Since joining Digital in 1983, Rebecca has been engaged in microprocessor architecture and design. She holds two patents and has seven applications pending. Rebecca received a B.A. in history from Swarthmore College and a B.S.E.E. from the Massachusetts Institute of Technology.

13 Trademarks