Digital Technical Journal
Volume 6, Number 2
DECchip paper

Development of Digital's PCI Chip Sets and Evaluation Kit for the DECchip 21064 Microprocessor

by

Samyojita A. Nadkarni, Walker Anderson, Lauren M. Carlson, David Kravitz, Mitchell O. Norcross, and Thomas M. Wenners

ABSTRACT

The DECchip 21071 and the DECchip 21072 chip sets were designed to provide simple, competitive devices for building cost-focused or high-performance PCI-based systems using the DECchip 21064 family of Alpha AXP microprocessors. The chip sets include data slices, a bridge between the DECchip 21064 microprocessor and the PCI local bus, and a secondary cache and memory controller. The EB64+ evaluation kit, a companion product, contains an example PC mother board that was built using the DECchip 21064 microprocessor, the DECchip 21072 chip set, and other off-the-shelf PC components. The EB64+ kit provides hooks for system designers to evaluate cost/performance trade-offs. Either chip set, used with the EB64+ evaluation kit, enables system designers to develop Alpha AXP PCs with minimal design and engineering effort.

INTRODUCTION

The DECchip 21071 and the DECchip 21072 chip sets are two configurations of a core logic chip set for the DECchip 21064 family of Alpha AXP microprocessors.[1] The core logic chip set provides a 32-bit PCI local bus interface, cache/memory control functions, and all related data path functionality to the system designer. It requires minimal external logic. The EB64+ kit is an evaluation and development platform for computing systems based on the DECchip 21064 microprocessor and the core logic chip set. The EB64+ kit also served as a debug platform for the chip sets. The DECchip 21071 and the DECchip 21072 chip sets and the EB64+ evaluation kit were developed to proliferate the Alpha AXP architecture in the industry by providing system designers with a means to build a wide range of uniprocessor systems using the DECchip 21064 processor family with minimal design and

engineering effort.[2]

The core logic chip set and the EB64+ evaluation kit were developed by two teams that worked closely together. This paper describes the goals of both projects, the major features of the products, and the design decisions of the development teams.

THE CORE LOGIC CHIP SET

This section discusses the design and development of the two configurations of the core logic chip set. After presenting the project goals and the overview, the section describes partitioning alternatives and the PCI local bus interface. It then details the memory controller and the cache controller and concludes with discussions of design considerations and functional verification.

Project Goals

The primary goal of the project was to develop a core logic chip set that would demonstrate the high performance of the DECchip 21064 microprocessor in desktop and desk-side systems with entry prices less than $4,000. The chip set had to be system independent and had to provide the system designer with the flexibility to build either a cost-focused system or a high-performance system.

Another key goal was ease of system design. The chip set had to include all complex control functions and require minimal discrete logic on the module so that a system could be built using a personal computer (PC) mother board and off-the-shelf components.

Time-to-market was a major factor during the development of the chip set. The DECchip 21064 microprocessor had been announced nearly five months before we started to develop the core logic chip set. Digital wanted to proliferate the Alpha AXP architecture in the PC market segment; however, the majority of system vendors required some core logic functions in conjunction with the microprocessor to aid them in designing systems quickly and with low engineering effort. Providing these interested system vendors with core logic chip set samples as soon as possible was very important to enable the DECchip 21064 microprocessor to succeed in the industry.

To determine the feature set that would meet the project goals, we polled a number of potential chip set customers in the PC market segment to understand their needs and the relative importance of each feature. We kept this feedback in mind during the course of the design and made appropriate design decisions based on this data. The following subsections describe the final chip set partitioning, the trade-offs we had to make in the

design, and the design process.

Chip Set Overview

The chip set consists of three unique designs:

- DECchip 21071-BA data slice
- DECchip 21071-CA cache/memory controller
- DECchip 21071-DA PCI bridge

It can be used in either a four-chip or a six-chip configuration.

The DECchip 21071 chip set consists of four chips: two data slices, one cache/memory controller, and one PCI bridge. This configuration was developed for a cost-focused system; it provides a 128-bit path to secondary cache and a 64-bit path to memory. Cache and memory data have 32-bit parity protection.

The DECchip 21072 chip set consists of six chips: four data slices, one cache/memory controller, and one PCI bridge. Intended for use in a performance-focused system, this configuration provides a 128-bit path to secondary cache and a 128-bit path to memory. The system designer can choose between 32-bit parity or 32-bit error correcting code (ECC) protection on cache and memory data.

Figure 1 is a block diagram of an example system using the core logic chip set. For a list of components used in a typical system built with this chip set, see the EB64+ Kit Overview section.

[Figure 1 (Core Logic Chip Set Configurations in a System Block Diagram) is not available in ASCII format.]

The processor controls the secondary cache by default. It transfers ownership of the secondary cache to the cache controller when it encounters a read or a write that misses in the secondary cache. The cache controller is responsible for allocating the cache on CPU memory reads and writes, and for extracting victims from the cache. The cache controller is also responsible for probing and invalidating the secondary cache on direct memory access (DMA) transactions initiated by devices on the PCI local bus.[3]

The ownership of the address bus, sysAdr, is shared by the processor and the PCI bridge. The processor is the default owner of sysAdr. When the PCI bridge needs to initiate a DMA transaction, the cache controller performs the arbitration.

Data is transferred between the processor, the secondary cache, the data slices, and the cache/memory controller over the sysData bus, which is 128 bits wide. In the 4-chip configuration, each of

the two data slices connects to 64 bits of the sysData bus. In the 6-chip configuration, each of the four data slices connects to only 32 bits of the sysData bus, leaving 32 data bits available for use as ECC check bits for memory and cache data. The cache/memory controller connects to the lower 16 bits of the sysData bus to allow access to its control and status registers (CSRs).

Data transfers between the PCI and the processor, the secondary cache, and memory take place through the PCI bridge and the data slices. The PCI bridge and the data slices communicate through the epiBus. The epiBus contains 32 bits of data (epiData), 4 byte enables, and the data path control signals. We defined the epiBus control signals so that the PCI bridge chip operation is independent of the number of data slices in the system. Furthermore, the epiBus control signal definitions allow the epiData bus width to be expanded to 64 bits without changing the design of the data slice.

The system designer can link the system to an expansion bus, such as the Industry Standard Architecture (ISA) bus or the Extended Industry Standard Architecture (EISA) bus, by using a PCI-to-ISA bridge or a PCI-to-EISA bridge. The Intel 82378IB and 82375EB bridges, for example, are available in the market for the ISA and the EISA buses, respectively.[4]

Partitioning Alternatives

As a result of our customer visits, we found that the following features were important for cost-focused systems. The features, which affect the partitioning, are listed in descending order of importance.

- Low cost for the chip set
- Low chip count
- Parity protection on memory
- Inexpensive memory subsystem

The following features were identified as important for performance-oriented, server-type systems (in descending order of importance).

- High memory bandwidth
- Chip set cost
- Low chip count
- ECC-protected memory (This is a requirement in a server system.)

During the feasibility stages, we decided to support a 128-bit secondary cache data path and not offer optional support for a 64-bit cache data path. We felt that a system based on the DECchip 21066 microprocessor, which supports a 64-bit cache interface, would meet the cost and performance needs in this segment of the market.[5] Keeping in mind the importance of time-to-market, we decided that the added flexibility in system design alternatives was not worth the additional design and verification time required to incorporate this feature.

We decided to provide an option between 64-bit-wide memory and 128-bit-wide memory. The wider memory data path provides higher memory bandwidth but at an additional cost. The minimum memory that the system can support with a 128-bit-wide memory data path is double that supported by a 64-bit memory data path. Memory upgrades are also more expensive. For example, with 4-megabyte (MB) single in-line memory modules (SIMMs), the minimum memory supported by a 64-bit memory data path is 8 MB (two SIMMs); with a 128-bit memory data path, it is 16 MB. Memory increments with a 64-bit data path are 8 MB each, and with a 128-bit data path are 16 MB each. We decided that the performance of the 64-bit memory data path was sufficient for a cost-focused system; however, for memory-intensive applications in the server market, 128-bit-wide memory was necessary.

One alternative we explored could have provided all the features of a cost-focused system in a chip set of three chips, using two identical 208-pin data path slices and one 240-pin controller that provided the PCI bridge, cache controller, and memory controller functions. This configuration, however, would have been restricted to 64-bit memory width and parity protection on memory. Thus it would not have met two of the four desirable features of a high-performance system.

The partitioning we chose allowed us to satisfy the requirements of both cost-focused and performance-oriented systems. By splitting the design into three unique chips: a data slice, a cache/memory controller, and a PCI bridge, we met the requirements of a cost-focused system with the 4-chip configuration. All 4 chips are 208-pin packages, costing roughly the same as the 3-chip alternative. This partitioning scheme allowed us to support a 128-bit-wide data path to memory and ECC protection with the addition of 2 data slices at relatively low incremental cost. Thus it met the requirements of a performance-focused system. We could not support ECC with the 64-bit-wide memory due to pin-count constraints, but we felt that this trade-off was reasonable given that cost was more important than ECC-protected memory in this market. This partitioning scheme had the added advantage of presenting a single load on the PCI local bus, as opposed to the two loads presented by the 3-chip configuration described above.

Another alternative was to provide a 4-chip configuration with

128-bit-wide, ECC-protected memory. This would have required the data slices to be of higher pin count and therefore higher cost, thus penalizing the cost-focused implementation.

PCI Local Bus Interface

The PCI local bus is a high-performance bus intended for use as an interconnect mechanism between highly integrated peripheral controller components, peripheral add-in boards, and processor/memory subsystems. Interfacing the DECchip 21064 family of CPUs to the PCI local bus opens up the Alpha AXP architecture to what promises to be an industry-standard, plug-and-play interconnect for PCs. The PCI bridge provides a fully compliant host interface to the PCI local bus. This section describes some features of the PCI bridge.

The PCI bridge includes a rich set of DMA transaction buffers that allows it to perform burst transfers of up to 64 bytes in length with no wait states between transfers. We optimized our design for naturally aligned bursts of 32 bytes and 64 bytes because this would eliminate the need for a large address counter and because we discovered through research that most PCI devices in development would not perform DMA bursts longer than 64 bytes.

DMA Write Buffering. We chose a DMA write buffer size of four cache blocks. This size would allow for two PCI peripheral devices to alternate bursts of 64 bytes each, thus maximizing use of PCI bandwidth. We organized the DMA write buffer as four cache block entries (four addresses) to simplify the cache/memory interface. In addition, this would allow the data buffers to be used efficiently whenever 32-byte bursts were in use.

DMA Read Buffering. We designed the DMA read buffer to be able to store a fetch cache block and a prefetch cache block. As with the DMA write buffer, the DMA read buffer is organized to allow for efficient operation during both 64-byte and 32-byte bursts. Prefetching is performed only if either the initiating PCI command type or a programmable enable bit indicates that the prefetch data will likely be used. This allows the system designer to combine 32-byte and 64-byte devices without sacrificing cache/memory bandwidth. To minimize typical DMA read latency while maintaining a coherent view of memory from the PCI, we designed the capability for DMA read transactions to bypass DMA write transactions, which are queued in the DMA write buffer, as long as the DMA read address does not conflict with any of the valid DMA write addresses. Because most DMA read addresses are not expected to conflict, typical DMA read latency does not suffer as a result of the relatively deep DMA write buffer.

Scatter Gather Address Mapping (S/G Mapping). The PCI bridge

provides the ability to map virtual PCI addresses to physical locations in main memory. Because each 8-kilobyte (kB) page can be mapped to an arbitrary physical page in main memory, a virtual address range that spans one or more contiguous pages can be mapped to pages that are physically scattered in main memory, thus the name S/G mapping. Using this mechanism, software designers can efficiently manage memory while performing multiple-page DMA transfers.

Although our inclusion of S/G mapping offers efficiency benefits to software designers, it also presented us with design challenges in the areas of performance and cost goals. The PCI bridge performs address translation by using incoming PCI physical addresses to index into a lookup table. Each incoming PCI transaction requires the PCI bridge to perform an address translation. A simple implementation might store the entire lookup table in local static random-access memory (RAM). To avoid use of this costly component and corresponding chip set pin allocations, our designers opted to store the lookup table in main memory. To minimize the performance impact of storing the table in main memory, the designers incorporated an on-chip translation lookaside buffer (TLB) for storing the eight most recently used translations. To keep things simple, we implemented a circular TLB replacement algorithm.

PCI Byte Access Support. To successfully incorporate Alpha AXP CPUs into PC environments, we required collaboration across the corporation. Digital engineers defined a software/hardware mechanism that allows the 32-bit/64-bit Alpha AXP architecture to coexist with components on the PCI local bus that require arbitrary byte access granularity. This mechanism requires that low-order address bits be used to encode byte lane validity. Implementing this mechanism reduces the density of I/O registers in the address space and conveys byte lane validity information through the address itself.

I/O write performance in this address space suffers because each CPU-initiated I/O transaction can convey only up to 64 bits (a quadword) of data and byte lane validity information. To allow for full utilization of the DECchip 21064 microprocessor's 32-byte internal write buffer during I/O writes to devices that do not require byte granularity, the chip set designers implemented an address range that does not perform byte lane decoding. In this space, up to 32 bytes can be transferred from the CPU and burst onto the PCI in a single transaction. This allows for efficient bandwidth utilization during writes to I/O devices that exhibit memory-like interfaces, such as video adapters with directly accessible frame buffers.

Guaranteed Access Time. Systems that support EISA or ISA expansion buses must be able to provide a guaranteed maximum read latency from EISA/ISA peripherals to main memory (2.5

microseconds for EISA, 2.1 microseconds for ISA). This requirement presented a challenge for us during our design. In the worst case, a simple memory read request from an EISA/ISA peripheral can result in significant latency due to our use of deep DMA write buffering and S/G mapping. Although our decision to allow DMA reads to bypass DMA writes provides systems with a typically low latency, this feature does not avoid worst-case high latency. To meet the EISA/ISA worst-case requirements, we included in our design PCI sideband signals and cache/memory arbitration sequences that allow for guaranteed main memory access time. When guaranteed access time is required, the EISA/ISA bridge must signal the PCI bridge by asserting a PCI sideband signal. In response, the PCI bridge will flush its DMA write buffers, hold ownership of the cache/memory, and signal readiness to the EISA/ISA bridge. When the EISA/ISA transaction starts, this sequence guarantees that the path to main memory is clear and will therefore have guaranteed access time.

Memory Controller

The memory controller supports up to eight banks of dynamic random-access memory (DRAM) and one bank of dual-port video random-access memory (VRAM). Each memory bank can be selectively programmed to enable two subbanks, which allows the memory controller to support double-sided SIMMs that have two row address strobe (RAS) lines per bank. The memory controller thus has the flexibility to support system memory sizes of 8 MB to 4 gigabytes (GB) of DRAM and 1 MB to 8 MB of VRAM. System designers can choose to implement memory by banks of individual DRAMs or SIMMs, either on board or across connectors. The memory controller is able to support a wide range of DRAM sizes and speeds across multiple banks in a system, by providing separate programmable bank base address, configuration, and timing registers on a per-bank basis.

We designed the memory controller for system flexibility by supporting fully programmable memory timing with 15-nanosecond (ns) granularity. This programmability supports SIMM speeds ranging from 100 ns down to 50 ns. Each memory bank's timing is programmed through registers that consist of DRAM timing parameters to control counters. Some examples of programmable timing parameters used to control the memory interface are "row address setup," "read CAS width," and "CAS precharge." As the memory controller sequences through a memory transaction, these programmed counters control the exact timing of RAS, column address strobe (CAS), the DRAM address bits, and write enables. At the same time, the memory controller sends commands from the cache/memory controller chip to the data slice chips to control the clock edge for sending and receiving memory data on DRAM writes and reads, respectively.

One customer is currently using one of the banks in combination with medium-scale integration (MSI) components to interface to a

very slow memory bus that supports flash read-only memories ROMs, nonvolatile RAM, and light-emitting diodes (LEDs). Since the original design was not done with a very slow memory interface in mind, this demonstrates that the chip set provides flexible, programmable timing functionality independent of the system.

The memory controller allows the system designer to build an inexpensive graphics subsystem using a video frame buffer on the memory data bus, and a low-cost video controller on an expansion bus like the ISA bus. The system designer can achieve competitive graphics performance by using the processing power of the CPU for graphics computations and the existing high-bandwidth memory data path for transferring data between the graphics computation engine (the CPU) and the frame buffer. The interface between the memory controller and the video controller is very economical: only two control signals are required to time the transfer of screen data from the random-access memory of the VRAM to the serial-access memory of the VRAM. The video controller is responsible for transferring the data from the serial memory of the VRAM to the screen.

Although we designed the memory controller to be flexible, we also included features that improved performance. Two such features are optimizations to reduce memory read latency and selective support for use of page mode between memory transactions.

To minimize memory read latency, the memory controller prioritizes reads above writes pending in the memory write buffer. For a CPU memory read, the memory controller waits six system cycles after the last read data before servicing a pending write, unless the memory write buffer is full. At least six system cycles occur between the time the memory controller latches the last read data from the DRAMs and the time a subsequent read request could be issued by the DECchip 21064 processor. Because memory write transactions take longer than six cycles to complete, our choice to delay the execution of a pending write allows read latency to be reduced for the following read. Waiting six system cycles after a read is a significant performance improvement for successive reads with cache victims because every read is accompanied by a write.

We also chose to improve performance by selectively determining which memory transactions would benefit most by staying in page mode. The memory controller stays in page mode after a DMA read burst and between successive memory writes. Page mode is not supported between CPU memory read transactions since the RAS precharge time can typically be hidden between successive CPU read requests.

Cache Controller

The secondary cache interface logic is partitioned across the

cache/memory controller chip and the data slice chips. The cache/memory controller chip contains the address path and control logic, and the data slice chips provide buffering for four cache lines of data to and from memory. We designed the cache controller to be system independent and flexible so that it could be designed into a wide range of systems.

The chip set supports a direct-mapped, write-back secondary cache with a data width of 128 bits and a cache line fixed at 32 bytes. The chip set allows the system designer to choose a secondary cache size ranging from 128 kB to 16 MB, as determined by software configuration. The speed of the cache RAMs must be fast enough to support the chip set's read access time of one system cycle. Writes to the cache can be programmed to take one or two system cycles. The write enables can be programmed to have a half-cycle or full-cycle pulse width when writing the cache during fill cycles. This feature was added to give the system designer flexibility in meeting SRAM write-enable specifications with various system cycle times.

Another feature added to the cache controller to provide flexibility is the support of an optional allocation policy on CPU writes. The write-back secondary cache is always allocated on CPU memory read misses. The option to allocate the cache on CPU memory write cache misses is programmable and can be disabled by software during system initialization. We chose to provide this option since disabling cache write allocation can allow higher memory write bandwidth. This feature can be used by system designers to determine whether particular applications have better performance when secondary cache write allocation is disabled.

The cache controller provides arbitration between the CPU and the PCI bridge chip for secondary cache ownership. The arbitration policy is programmable and varies the level of control the PCI bridge has in keeping the ownership of the secondary cache during DMA transactions.

Although we designed the cache controller for system flexibility, we also included features that would give it performance advantages. One such feature is the memory write buffer. The cache controller uses the memory write buffer to store four cache lines of data for cache victims, DMA writes, CPU-noncacheable writes, and CPU-cacheable writes when write allocate mode is disabled. The buffer is organized as first in, first out (FIFO) on cache-line boundaries. Successive writes to the same cache line are not merged into the buffer because the CPU chip write buffer performs this function. The cache controller allows CPU and DMA reads to bypass the write buffer as long as the read address does not conflict with any of the write addresses. The memory write buffer improves performance by allowing timely acknowledgment of write transactions. Read bypassing of the write buffer improves performance by reducing memory read latency.

Global Design Considerations

This section briefly discusses some of the decisions concerning silicon technology, packaging technology, and internal clocking of the chip sets.

Silicon Technology. The design team chose to use an externally supplied gate-array process that offered quick time-to-market and low cost. Most chips designed in the Semiconductor Engineering Group are manufactured using Digital's proprietary complementary metal-oxide semiconductor (CMOS) processes, which emphasize high speed and high integration. Our chips' performance and complexity -- 30-ns cycle time, approximately 35,000 gates per chip -- did not require these capabilities. Gate-array technology offered shorter design times and quicker turnaround times than Digital's custom silicon technology.

Packaging Technology. When choosing a package, the design team considered issues of package and system cost, design partitioning, and heat produced by power dissipation. Some of these issues are discussed in the Partitioning Alternatives section.

We chose to put all three chips in 208-pin plastic quad flat packages (PQFPs). The 208-pin PQFP is one of the most popular low-cost, medium pin-count, surface-mount packages. One drawback of PQFPs, however, is their low limit on power dissipation. To ensure a junction temperature of 85 degrees Celsius with 100 linear feet per minute of airflow, the power dissipation must be limited to 1.5 watts (W). The power dissipation of the data slice is about 1.7 W, resulting in a junction temperature approaching 100 degrees Celsius. We verified that reliability was not an issue at a junction of 100 degrees Celsius. However, we had to ensure that the chip timing worked at a junction temperature of 100 degrees Celsius, as opposed to the 85 degrees Celsius we would normally use. We could not use this approach on the PCI bridge chip because the additional timing optimization required would have adversely affected the schedule. We had to take special measures in the design to keep the power dissipation within the 1.5-W limit. We implemented conditional clock nets for large blocks of registers that are loaded infrequently, such as the CSRs and the TLB.

Internal Clocking. To achieve the shortest possible cross chip set latencies, we implemented a four-phase clock system. A four-phase system allows data to be transferred from one section of the chip set to another in less than a full clock cycle if logic delays are sufficiently small.

In contrast to approaches based on latch designs, which can offer

lower gate-count implementations, we chose to use mostly edge-triggered devices. We viewed this as an opportunity to simplify the design analysis and timing verification process by keeping the number of timing reference points to four clock edges.

To further simplify the clocking system, the designers chose to make the PCI clock and the cache/memory clock synchronous to each other. This approach avoids the need for synchronizers (and corresponding synchronizer delays) between clock domains; it also reduces the number of clock speed combinations to be verified. Although the synchronous approach does not allow the system designer to decouple the PCI clock speed from the cache/memory clock speed, we felt that the added complexity and verification effort required to support asynchronous clocks would not be worth the small degree of flexibility that would be gained from such a design.

Functional Verification

Given the short design schedule and the requirement that first-pass prototypes be highly functional for customers, the team adopted a strategy of pseudorandom testing at the architectural level of the chip set as a whole. We felt that this strategy would test more of the design more quickly and would find more subtle and complex bugs than a testing methodology focused on the gate/register level of each separate chip.

The DECSIM simulation environment included models for the three chips, a DECchip 21064 bus functional model (BFM), a PCI BFM, a cache model, a memory model, and some "demon" models that could be programmed to pseudorandomly generate events such as the assertion of the video port inputs or the injection of errors. We developed SEGUE templates and used them in a variety of exercisers to generate DECSIM scripts pseudorandomly.[6]

To keep the testing environment from being overly complicated, we allowed users to pseudorandomly configure only those aspects of the design that significantly altered the operation of the control logic. Many configurable aspects of the chip set and simulation environment (for example, the PCI S/G map) were not varied in the exercisers and were tested with simple focused tests.

In addition to programming BFMs to read back and check data, we built a variety of checkers into the simulation environment to verify correct operation of RAM control timing, PCI protocol, tristate bus control, PCI transaction generation, data cache invalidate control on the DECchip 21064 CPU, and many other functions. At the end of every exerciser run, the secondary cache and memory were checked for coherence and correct error protection.

The verification efforts of the team resulted in the removal of over 200 functional bugs, ranging from simple bugs to quite complex and subtle bugs, prior to the fabrication of first-pass prototypes. We found no "show stopper" bugs in the core functions required for first-pass prototype chips, and we used simple work-arounds for the few bugs that we did find in the first-pass design.

THE EB64+ EVALUATION KIT

This section of the paper discusses the development of the EB64+ evaluation kit. After presenting the project's goals and the overview of the kit, it discusses some of the module design issues that were addressed during the design of the EB64+ module. This section concludes with performance results of benchmarks run on the EB64+ system.

Project Goals

The first and most important goal of the EB64+ evaluation kit project was to provide a sample design for customers using the DECchip 21064 microprocessor and the DECchip 21071 and the DECchip 21072 chip sets. Another major goal was to provide an evaluation and development platform that used standard PC components. These two goals would enable a customer to evaluate their design trade-offs quickly and to complete their system design faster and with a better chance of success.

Secondary goals were to provide a development and debug environment for the core chip set and to provide a high-performance benchmarking system for the microprocessor and core chip set. The EB64+ kit also serves as a platform for hardware and software development for PCI I/O devices.

EB64+ Kit Overview

Figure 2 shows a block diagram of the EB64+ module, a full-size PC (12.0 inch by 13.0 inch) mother board. The major components on the module are given below:

- DECchip 21064 microprocessor (150 megahertz [MHz] to 275 MHz)
- Secondary cache (512 kB, 1 MB, or 2 MB)
- Secondary cache address buffer
- Interrupt/configuration programmable array logic (PAL) device
- Serial ROM interface for the microprocessor

- System clock generator: oscillator, phase-locked loop (PLL), clock buffers
- Core logic chip set
- Two secondary cache control PALs
- PCI bus peripherals: embedded small computer system interface (SCSI) and Ethernet
- PCI bus arbiter
- Intel 82378IB bridge between the PCI and ISA buses
- Three ISA expansion slots
- Eight slots of standard 36-bit memory SIMMs
- Memory control signal buffers

[Figure 2 (Block Diagram of the EB64+ Module) is not available in ASCII format.]

Secondary Cache Size and Speed

The DECchip 21064 processor has programmable secondary cache read and write access times with a granularity equal to the processor clock cycle time. For instance, if the read access time is 25 ns, the programmed value for a 150-MHz processor (6.6-ns cycle time) would be 4, and the programmed value for a 200-MHz processor (5-ns cycle time) would be 5.

One of the more difficult decisions for any system designer is to determine the optimal cache size and speed in terms of cost and performance. The EB64+ module supports various cache size and speed options in order to allow a system designer to evaluate the difference between a large, slow cache and a small, fast cache. The trade-off here is usually between lower cost for the 512-kB cache and higher performance for the 2-MB cache. The 2-MB cache uses four 128K by 9 SRAMs and twelve 128K by 8 SRAMs for the data store, and the 512-kB cache uses four 32K by 9 SRAMs and twelve 32K by 8 SRAMs.

We decided to share data RAM footprints between the 32K by 8 SRAMs and the 128K by 8 SRAMs, thus allowing the system designer to build two different modules: one with a 512-kB cache and the other with a 2-MB cache. The designer can evaluate the speed-to-size trade-off by using faster SRAMs for the smaller cache and slower SRAMs for the larger cache. The system designer can choose to evaluate the effect of varying the cache size from 512 kB, to 1 MB, to 2 MB, without varying the cache speed, by configuring jumpers to disable portions of the 2-MB cache on an

EB64+ module.

System Clocking Design

System clocking for the EB64+ module presented a challenge in two different areas. The first area was the high-frequency input clocks needed by the DECchip 21064 microprocessor. The input clocks operate at twice the frequency of the DECchip 21064 CPU, requiring a 300- to 550-MHz oscillator for the EB64+ module. Initially, an emitter-coupled logic (ECL) output oscillator was used for this purpose. The main drawback to this solution was the cost, which is in the $40 to $50 range. The other disadvantage was the long lead time and nonrecurring engineering charges associated with unique oscillator frequencies.

By working closely with a vendor of gallium arsenide (GaAs) devices, we were able to provide an alternative in the $8 to $18 range. The device consists of a low-frequency oscillator and a PLL that multiplies the low-frequency oscillator to provide the high-frequency input that the processor requires. For example, a 300-MHz frequency clock is generated using a 30-MHz oscillator connected to a PLL that multiplies this by 10 to provide the 300-MHz input. Since lower frequency oscillators are produced by more vendors, the lead times and nonrecurring engineering charges for unique frequencies are minimal when compared to the ECL output oscillators.

Generating the clocks for the other system components was quite challenging. The core logic chip set, PCI devices, and the cache control PALs together require three types of clock signals: the first clock is in phase with the processor's sysClkOut clock signal; another clock is 90 degrees phase shifted from the first; and a third clock has twice the frequency of and is in phase with the first. The frequency of sysClkOut is an integral divisor (between 2 and 17) of the processor's internal clock frequency. Some divisors may result in a sysClkOut duty cycle that is not 50 percent. A PLL is used to generate both the phase-shifted and the double-frequency clock. It also guarantees a 50 percent duty cycle, which is required for the PCI clock.

Figure 3 illustrates how the EB64+ module generates the three system clocks from the processor's sysClkOut signal. In addition to the PLL, the system clock generator uses low-skew clock buffers to drive the final device loads of the system. One output of the clock buffers is used to provide the feedback to the PLL. This causes the overall delay from sysClkOut to the system clock to be close to zero.

[Figure 3 (EB64+ System Clock Distribution) is not available in ASCII format.]

Design Evolution

As noted previously, the EB64+ kit was developed to provide an example design to external customers as well as provide a debug and development platform for the core logic chip set. The focus of the design evolved during the project.

Initially, we included several features on the EB64+ module to support the various modes of the chip set. As the design progressed, an updated version of the EB64+ module was developed. The final version focused more on being a sample design than a debug and development platform for the chip set. Some of the features that fell into this category are listed below.

- Initially, the EB64+ module supported both the 64-bit and 128-bit memory on the same module with configuration jumpers. This design affected performance because 64 bits of the cache data bus were routed to two data slice chips. The final version of the EB64+ module supports only 128-bit memory. This change allowed us to reduce the cache read access time on the DECchip 21064 processor by 3 ns, thus reducing the programmed 2-MB cache read access time for a 200-MHz DECchip 21064 processor from 7 cycles to 6 cycles.

- Certain modes of the chip set were controlled by configuration jumpers initially. These have been redefined to support additional cache sizes and speeds to support a wider range of evaluation and benchmarking.

Performance

Figures 4 and 5 show the results of the BYTE magazine portable CPU/floating-point unit (FPU) benchmarks run on an EB64+ system running the Windows NT operating system. The EB64+ system has a 128-bit memory subsystem with 70-ns (RAS access time) DRAMs. The 150-MHz, 166-MHz, and 200-MHz benchmarks were run using a DECchip 21064 microprocessor with a 512-kB cache with a 28-ns read access time. The 275-MHz benchmark was run on a DECchip 21064A microprocessor with a 2-MB cache with a 35-ns read access time. The benchmarks for the DECchip 21066 processor were run on an EB66 system with a 256-kB cache. The figures show the performance relative to other Windows NT systems available in the market today. The benchmark data for the Intel486 DX2-66 and Pentium 60-MHz chips and for the MIPS Computer Systems' R4400SC processors was taken from the Alpha AXP Personal Computer Performance Brief--Windows NT.[7]

[Figure 4 (BYTE Portable CPU/FPU Benchmarks) is not available in ASCII format.]

[Figure 5 (EB64+ System Performance Benchmarks) is not available in ASCII format.]

Table 1 compares the bandwidths on an EB64+ system using the two possible chip set configurations, a 200-MHz processor, and 70-ns DRAMs.

Table 1 Comparison between a 64-bit Memory Data Path and a 128-bit Memory Data Path

Transaction Type	64-bit Memory 4-chip Configuration	128-bit Memory 6-chip Configuration
CPU Memory Writes:		
Write with secondary cache allocate	133 MB/s	133 MB/s
Write with no allocate	133 MB/s	267 MB/s
CPU Memory Read: Bandwidth	76 MB/s	107 MB/s
I/O Write:		
8 bytes	38 MB/s	38 MB/s
32 bytes (PCI dense memory space)	82 MB/s	82 MB/s
I/O Read:		
8 bytes	22 MB/s	22 MB/s
DMA Write:		
64-byte PCI burst	119 MB/s	119 MB/s
32-byte burst	107 MB/s	107 MB/s
DMA Read:		
Cache miss, 64-byte burst	55 MB/s	65 MB/s
Cache miss, 32-byte burst	41 MB/s	48 MB/s
Cache hit, 64-byte burst	74 MB/s	74 MB/s
Cache hit, 32-byte burst	51 MB/s	51 MB/s

SUMMARY

The DECchip 21071 and the DECchip 21072 chip sets and the EB64+ evaluation kit met their project goals by helping to proliferate the Alpha AXP architecture in the PC market. Several customers, as well as some groups within Digital, use the chip sets in their systems today. Many of these customers and internal groups have used the EB64+ platform as a basis for their designs and as a means of initiating their software development while they were developing their hardware. The EB64+ platform has also been used to develop device drivers for several PCI devices developed by Digital.

ACKNOWLEDGMENTS

The authors would like to acknowledge the efforts of the following people. The projects would not have been successful without them. Aaron Bauch, Dick Bissen, Mike Blake, Gregg Bouchard, Mike Callander, Derrick Dacosta, Paul Dziekowicz, Greg Fitzgerald, Avi Godbole, Mike Goulet, Shaheed Haque, Franklin Hooker, Dave Ives, John Jakubowski, Mike Kagen, Elias Kazan, Don MacKinnon, Mike Martino, Mark Matulaitis, Kevin McCarthy, John Murphy, Mike Napier, Victor Peng, Eric Rasmussen, Tracy Richardson, Mark Riggs, George Rzeznik, Debbie Salois, Raghu Shankar, Will Sherwood, Jai Singh, Wilson Snyder, Hemendra Talesara, Tom Walthall, Juanita Wickham, Mary Woodcome, Marco Zamora, and Beth Zeranski.

REFERENCES

1. DECchip 21064-AA Microprocessor Hardware Reference Manual (Maynard, MA: Digital Equipment Corporation, Order No. EC-N0079-72, 1992).

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

3. PCI Local Bus Specification, Revision 2.0 (Hillsboro, OR: PCI Special Interest Group, Order No. 281446-001, April 1993).

4. 82420/82430 PCIset ISA and EISA Bridges (Santa Clara, CA: Intel Corporation, 1993).

5. DECchip 21066 and DECchip 21068 Hardware Reference Manual (Maynard, MA: Digital Equipment Corporation, Order No. EC-N2681-72, 1994).

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

7. Alpha AXP Personal Computer Performance Brief--Windows NT, 2d ed. (Maynard, MA: Digital Equipment Corporation, Order No. EC-N2685-10, January 1994).

TRADEMARKS

Alpha AXP and DECchip are trademarks of Digital Equipment Corporation.

BYTE is a registered trademark of McGraw-Hill, Inc.

Intel, Intel486, and Pentium are trademarks of Intel Corporation.

MIPS is a trademark of MIPS Computer Systems, Inc.

PAL is a registered trademark of Advanced Micro Devices, Inc.

Windows NT is a registered trademark of Microsoft Corporation.

BIOGRAPHIES

Samyojita A. Nadkarni Sam Nadkarni is the program manager for CPU core logic chip sets in the Semiconductor Engineering Group. She was the leader of the DECchip 21071 development project. Prior to that, Sam led the development of the memory controller chip used in the VAX 4000 Models 400, 500, and 600 systems. She also worked on memory controller/bus adapter chips for the VAX 4000 Model 300 and MicroVAX 3500 systems. Sam joined Digital in 1985 and holds a Bachelor of Technology (1983) from the Indian Institute of Technology and an M.S. (1985) from Rensselaer Polytechnic Institute.

Walker Anderson A principal engineer in the Semiconductor Engineering Group, Walker Anderson is currently the manager of graphics and multimedia chip verification. He was the verification team leader for the NVAX chip and for the DECchip 21071/21072 chip sets as well as a co-leader of the verification team for a future Alpha AXP chip. Before joining Digital in 1988, Walker was a diagnostic and testability engineer in a CPU development group at Data General Corporation for eight years. He holds a B.S.E.E. (1980) from Cornell University and an M.B.A. (1985) from Boston University.

Lauren M. Carlson A senior hardware engineer in the Semiconductor Engineering Group, Lauren Carlson is currently working on the design of a core logic chip set for a new microprocessor. Prior to this, she worked on the design of the cache/memory controller of the DECchip 21071 chip set and completed the hardware functional verification of the chip set on the EB64+ evaluation board. Lauren has also contributed to the design of the I/O controller and system module of the VAXstation 4000 Model 90. Lauren holds a patent on gate array design. She has a B.S.E.E. from Worcester Polytechnic Institute (1986) and joined Digital in 1987.

David Kravitz David Kravitz received a B.S.E.E. from the Massachusetts Institute of Technology. Upon joining Digital in 1985, he worked on the cache control and processor chips for the VAX 6000 Models 400 and 500 systems in Hudson, Massachusetts, and a Cluster Interconnect (CI) chip in Jerusalem, Israel. As a senior hardware engineer in the Semiconductor Engineering Group, David designed the data path chip for the DECchip 21071 and DECchip 21072 chip sets. He is currently working on a low-cost microprocessor.

Mitchell O. Norcross Senior engineer Mitch Norcross is currently

project leader for a second-generation core logic chip set for the DECchip 21064. Since joining Digital in 1986, Mitch has contributed to the design of several ASICs and systems, including the DECchip 21072 chip set, the VAXstation 4000 Model 90, and Digital's first fault-tolerant VAX system, the VAXft 3000. He received a B.E. in electrical engineering (1985) and an M.S. in computer engineering (1987), both from Manhattan College. Mitch holds two patents related to fault-tolerant system design.

Thomas M. Wenners Thomas Wenners is a principal hardware engineer in the Semiconductor Engineering Group. He is the project leader responsible for various high-performance mother boards for Alpha AXP PCs. In addition, he is involved with issues concerning high-speed clocking in Alpha AXP chips. Tom's previous work includes the module design of the VAX 6000 Model 600 and VAX 4000 Model 90, as well as module design and signal integrity support on ESB products. Tom joined Digital in 1985. He received a B.S.E.E. (cum laude, 1985) and an M.S.E.E. (1990) from Northeastern University.