The Second-generation Processor Module for AlphaServer 2100 Systems

by

Nitin D. Godiwala and Barry A. Maskas

ABSTRACT

The second-generation KN470 processor module for AlphaServer 2100 systems performs significantly better than the first-generation KN460 module and was designed to be swap-compatible as an upgrade. The KN470 processor module derives its performance improvements from the enhanced architecture of Digital's new Alpha 21164 microprocessor, the synchronous design of the third-level cache and system interface, the implementation of a duplicate tag of the third-level cache, and the implementation of a write-invalidate cache coherence protocol for the multiprocessor system bus. Additional design features such as read-miss pipelining, system bus grant parking, hidden coherence transactions to the duplicate tag, and Alpha 21164 microprocessor write transactions to the system bus back-off and replay were combined to produce a higher performance processor module. The scope of the project required implementing functionality in system components such as the memory, the backplane, the system bus arbiter, and the I/O bridge, which shipped one year ahead of the KN470 module.

INTRODUCTION

The second-generation KN470 processor module for AlphaServer 2100 systems achieves a higher performance than the first-generation KN460 module while maintaining compatibility with the AlphaServer 2100 system environment. This paper describes the processor module project and the resulting design. Topics discussed are the elements that contribute to the compatibility and to the higher performance: coherence protocol, system bus protocol, system bus arbitration, system interface and shared data, and clocking. Some key design trade-offs are described. The paper concludes with a performance summary that presents measured attributes of the higher performance KN470 processor in the context of the AlphaServer 2100 family.

When the AlphaServer 2100 product family was being defined in late 1992, the processor module performance-over-time roadmap projected three performance variations based on increasing the clock rate of the Alpha 21064 microprocessor.[1] These modules were to be compatible with Digital's mid-range multiprocessor system bus and would support enhanced functionality such as

direct-mapped I/O, up to four microprocessors, an I/O bridge to 32-bit Peripheral Component Interconnect (PCI) and Extended Industry Standard Architecture (EISA) buses, and an I/O expansion option module with an I/O bridge to a 64-bit PCI bus.[2] Two members of the DEC 4000 processor design team were assigned to deliver this first-generation processor module. At this time, there was no goal to develop a second-generation processor module. Therefore, the remainder of the team designed the arbiter chip and the enhancements required in the processor-module system interface chips and at the same time contributed to the Alpha 21164 microprocessor development effort.

Goals for contributions to the Alpha 21164 microprocessor development effort were partitioned into short- and longer-term goals. A short-term goal was to define a system for the new Alpha microprocessor.[3] The related longer-term goal was to ensure that the Alpha 21164 microprocessor could operate in that defined system. An architectural study resulted in a proposal and a project plan to develop a second-generation processor module that extended the performance and longevity of the AlphaServer 2100 family. In addition, the remainder of the team made requests of the Alpha 21164 microprocessor team to incorporate specific legacy-related AlphaServer 2100 functions such as support for 32-byte cache blocks, control of I/O address space read merging, and completion of memory barriers on the Alpha 21164 microprocessor. The business management team accepted the proposal and the project plan. The Alpha 21164 microprocessor team agreed to support the functionality requests. The design team staffing was completed by March 1993, and detailed design work began in May 1993. The design team's goal was to have a processor module ready to accept the Alpha 21164 microprocessor for installation when the microprocessor first became available. The team met this goal.

Since the first- and second-generation processor modules would operate in the same enclosure and with the same power supply, the size and shape (i.e., form factor), cooling demands, and power consumption of the new module had to be compatible with those of the first-generation module. Because of the presence of an on-chip, write-back second-level cache and an estimated longer access time to that cache from the system bus, the Alpha 21164 microprocessor architecture adopted an invalidate-on-write cache coherence protocol. The Alpha 21064 microprocessor supported an off-chip, write-back second-level cache that has a faster access time from the system bus. This faster access time enabled the implementation of a good-performing update-on-write cache coherence protocol. Support of these snooping, multiprocessor system bus coherence protocols required enhancements to the system bus transaction types. This resulted in minor logic changes to the memory interface chips and to the I/O bridge chip.[4,5] These changes were defined and implemented in time for the first-generation system power-on. Hence, the system components, the I/O bridge chip, the memory modules, and the system bus and backplane are compatible with the first- and

second-generation processors. This basic difference in the system bus coherence protocols prevented the system from supporting the coexistence of the first- and second-generation processor modules because such a configuration has asymmetric attributes. Alpha operating system software does not support asymmetric multiprocessing; symmetry is assumed.

Another project goal was to maintain the AlphaServer 2100 family's position among the industry's leading high-performance server systems. This goal was achieved by exploiting the Alpha 21164 microprocessor's performance through the design of the processor module's third-level cache, by implementing a full-duplicate tag of this cache, and by implementing a synchronous clocking scheme. Combining the processor design attributes with a pipelined read transaction of a faster read-access system memory module enabled the team to achieve the project's goal of designing a higher performance processor module and multiprocessor system.

OVERVIEW OF THE PROCESSOR MODULE

The KN470 processor module provides an operational environment for the Alpha 21164 microprocessor. This environment, which is similar to that of the first-generation KN460 processor module environment, includes the following:

- Alpha 21164 microprocessor---a superscalar, superpipelined implementation of the Alpha architecture with low average cycles per instruction because of its four-instruction issue
- B-cache---a module or third-level write-back cache
- System interface---two application-specific integrated circuit (ASIC) chips that interface the Alpha 21164 microprocessor, B-cache, and duplicate tag store to the system bus
- Duplicate tag store---a tag store of the third-level write-back cache
- System bus clock repeater that provides system bus synchronous clocks to the module
- System bus arbiter that determines which system bus node can access the system bus
- Serial control bus subsystem that includes clock and reset control circuitry, a microcontroller with a serial interface, serial read-only memory with power-on firmware bits, and nonvolatile memory for processor configuration parameters

Figure 1 shows a block diagram of the KN470 processor module.

[Figure 1 (Block Diagram of the KN470 Processor Module) is not available in ASCII format.]

The Alpha 21164 microprocessor is organized with an on-chip 8-kilobyte (KB) primary instruction cache and an 8-KB write-through data cache, which are referred to as first-level caches. In addition, a 96-KB, second-level, three-way, set-associative write-back cache is implemented on the chip.

The module design includes a B-cache or third-level cache to minimize the miss penalty and to be configurable through the use of various densities of similarly packaged static random-access memory (RAM) chips. Such a design enabled final product definition late in the verification process based on static RAM costs and delivered performance from the B-cache. The size of the B-cache is either 1, 2, 4, 8, or 16 megabytes (MB). Each B-cache entry stores 32 bytes of data and the associated tag bits and is called a cache block. To facilitate read-fill data and victim-write data exchange with the system interface, the Alpha 21164 microprocessor and the system interface share the B-cache data port. The B-cache is controlled by the Alpha 21164 microprocessor, which has its second-level cache configured to operate in 32-byte instead of 64-byte cache block mode. This 32-byte mode of operation for the second-level cache was the most complex request made of the Alpha 21164 microprocessor design team. However, this design element was required to achieve the AlphaServer 2100 compatibility goal.

The system interface is a common boundary between the system bus, the Alpha 21164 microprocessor, and the third-level cache. The system interface provides the protocol and circuitry for the Alpha 21164 microprocessor to read or write devices connected to the system bus. Conversely, the system interface provides the protocol and circuitry for the processor module to respond to read or write transactions from the system bus. The system interface comprises two identical bit slices of an ASIC. The ASICs operate as even and odd slices, based on a mode-select pin on the module. The system interface selects the operating mode of the arbiter chip. It also encodes the system bus transaction type as read or write and then supplies a control signal to the arbiter chip. The arbiter chip must know the present system bus transaction type to remain synchronized with the system bus events and to know when to sample new requests for the system bus.

The module maintains a duplicate copy of tag control bits of each B-cache block in the duplicate tag store. The duplicate tag store is controlled by the system interface and is time multiplexed between system bus requests and Alpha 21164 microprocessor requests. This ability to pipeline transactions to the duplicate tag store from the Alpha 21164 microprocessor and the system bus allowed the Alpha 21164 microprocessor's requests to fill

predictable time slots in parallel to the system bus transactions, hidden from the system bus. This is called cycle-stealing because the coherence transactions requested by the Alpha 21164 microprocessor do not require arbitration for or use of the system bus cycles. Cycle-stealing provided more useful system bus bandwidth while at the same time reduced the Alpha 21164 microprocessor latency for coherence transactions to the duplicate tag store.

The clock repeater chips generate complementary metal-oxide semiconductor (CMOS)--level clocks from positive emitter-coupled logic (PECL)--driven backplane clocks. These CMOS-level clocks are skew regulated and distributed to the module's components. The Alpha 21164 microprocessor has digital-lock-loop circuitry, which aligns the Alpha 21164 microprocessor's interface clock to the reference clocks that run to all other module components. This scheme is basically the synchronous clocking scheme.

The module includes the system bus arbiter chip. The decision to locate this chip on the processor instead of the backplane stemmed from concerns over compatibility between the first- and second-generation processor arbitration algorithms supported by this chip. The system bus arbiter chip was designed and fabricated for the first-generation processor and included the functionality of the second-generation processor. The chip design was completed prior to the design of the KN470 processor module's system interface. To minimize the chance of design error, the team performed extensive simulations to help the project realize a full-function, second-pass chip for use in the first- and second-generation processor modules.

The KN470 processor module implements the system bus reset control and serial control bus subsystems, with minor modifications, that were designed for the first-generation processor module.

CACHE COHERENCE PROTOCOL

To improve the in-system performance of the Alpha 21164 microprocessor and its write-invalidate cache coherence protocol, the KN470 module implements a duplicate tag store of the B-cache. The Alpha 21164 microprocessor has two levels of on-chip cache that are maintained as a subset of the B-cache. This discussion assumes that the first- and second-level caches remain coherent with the B-cache and duplicate tag store. Operations performed by the Alpha 21164 microprocessor and system interface maintain the B-cache and duplicate tag store subset rule for the on-chip caches. The duplicate tag store and the B-cache each keep three control bits to maintain coherence with the on-chip caches and also with system memory and other module caches. The three control bits are called valid (V), shared (S), and dirty (D). A combination of control bits makes up a state of a cache block. The five possible cache block states are as follows:

1. VSD = 000 A cache block is either empty or removed from the B-cache and hence invalid.

2. VSD = 100 A cache block is the only cached copy in the system.

3. VSD = 101 A cache block is valid, and this copy has been modified more recently than the copy in memory.

4. VSD = 110 A valid cache block may also be in another cache. This processor must write or broadcast write modifications of this block to the system bus.

5. VSD = 111 A valid cache block may also be in another cache, and the copy of this block has been modified more recently than the copy in memory.

Cache state transitions are synchronized to system bus transaction cycles because the system bus is the common point of coherence and coherence conflict resolution.

When the Alpha 21164 microprocessor requests a transaction for a cache block to be read or filled from system memory into the B-cache and on-chip caches, the cache block state is set to VSD = 100 in the duplicate tag store, the B-cache, and the on-chip caches. The first-level instruction and write-through data caches must maintain only a valid bit. The second-level write-back cache must maintain the VSD bits consistent with the B-cache and the duplicate tag store.

If an Alpha 21164 microprocessor's read transaction request is of the type intent-to-modify, then the cache block state makes a direct transition to VSD = 101. A block in the valid state of VSD = 100 will make a transition to VSD = 101 when an Alpha 21164 microprocessor's request to modify the cache state has reached the point of coherence, i.e., the system bus. However, the duplicate tag store is maintaining coherence with the system bus, so this request must find a nonconflicting cycle to effect the state transition. If another processor reads the same block before this processor's request has reached the point of coherence, then the cache block state makes a transition to VSD = 110. In this case, the system interface updates the shared state to VSD = 110 for the read with intent-to-modify transaction before the block is modified. Because the block is shared, this processor's request to modify the block must now also become a broadcast write transaction to the system bus. Once the modified block is written to the system bus, the next state transition is to VSD = 100. The broadcast write transaction is sometimes referred to as an unsharing transaction.

Once a cache block state is valid, i.e., VSD = 100, it can be invalidated or set to the state VSD = 000 from the system bus by another processor's read with intent-to-modify or by a write

transaction to that block. The Alpha 21164 microprocessor does not allow an update of write data from the system bus but instead invalidates the block. Invalidation requires the duplicate tag store, the B-cache, and the on-chip caches to clear their V state. Implementation of system bus write transactions that cause block invalidation is required to support the cache coherence protocol of the Alpha 21164 microprocessor.

By filtering out system bus transactions that do not alter the coherence states of the Alpha 21164 microprocessor, the duplicate tag store serves to minimize the frequency with which the system bus transactions interrupt the microprocessor operations. Without the duplicate tag store filtering, the Alpha 21164 microprocessor would have to be interrupted on every system bus transaction, thus limiting the system performance.

SYSTEM BUS PROTOCOL

The KN470 processor module incorporates both an enhanced system bus protocol and a system bus arbiter that minimizes the arbitration latency.

Enhancement of Transactions

For the first-generation AlphaServer 2100 processor, the system bus protocol is the same as the one implemented in the DEC 4000 system. This system bus protocol is a snooping bus protocol in which all bus participants are required to monitor system bus transactions and to keep their cached copy of memory coherent. For the second-generation processor, the designers enhanced the DEC 4000 system bus protocol to support the write-invalidate cache coherence protocol of the Alpha 21164 microprocessor.

The DEC 4000 system bus protocol supports four types of transactions: read, write, exchange, and no operation. The exchange transaction performs a victim-write transaction to one memory location and a read transaction of another memory location. The two transactions are separated by the common lower 18 bits of address. The exchange transaction combines read transactions and victim-write transactions into one transaction, sharing the address cycle of the system bus. The exchange transaction is used to evict modified cache blocks from the caches back to system memory to allow a replacement block with a different tag to be allocated.

To support the second-generation processor's write-invalidate coherence protocol, transactions were added to the first-generation system bus protocol. These added transactions were needed to signal other processors and the I/O bridge chip to invalidate a block when a block was being read for the purpose of being modified. The exclusive-read and exclusive-exchange transactions were added to the four first-generation transaction

types. The exclusive-read transaction is the read transaction that also causes cache invalidation by a bystander processor module and the I/O bridge chip of the block being read. The exclusive-exchange transaction is the exchange transaction that also causes cache invalidation by a bystander processor module and the I/O bridge chip of the block being read.

The KN470 module implemented the exclusive transaction types to establish private ownership of a block. Establishing private ownership to a previously shared block enables write transactions to complete without having to broadcast write transactions back to the system bus. This occurs because the block is invalidated by bystanders who were sharing the block.

The enhancements of the system bus transaction types did not affect the memory module. The implementation of the exclusive indication signal was such that memory would decode a read or exchange transaction and not know of the exclusive signaling. Because the I/O bridge chip caches translation addresses for direct memory access of devices on the PCI or EISA buses, minor modifications were designed into the I/O bridge chip to support these enhanced commands.

Minimization of Arbitration Latency

The system bus arbiter implemented a bus grant parking or pregrant signaling scheme that minimized the arbitration timing overhead. This scheme combined with the pipelining of the read-miss commands from the Alpha 21164 microprocessor enabled the system bus interface to use the available memory bandwidth.

The arbiter for the first-generation processor followed the protocol used in the DEC 4000 system. The arbiter samples the requests and then issues the grants according to round-robin arbitration rules. The arbitration rules allow processor modules to have fair access to the system bus. The elapsed time from when a processor makes a system bus request to the arbiter until it receives a grant is referred to as the arbitration cycle or arbitration overhead. The arbitration overhead increases the memory and direct-mapped I/O access latency, as well as the cache-miss penalty. Typically, the arbitration overhead for each processor appears low in a multiprocessor system in which bus utilization is extremely high. The appearance of low arbitration overhead results from the time the system bus waits to finish a transaction before the arbiter can issue the next grant. However, the arbitration overhead may be as high as 20 percent of the transaction time in a system configuration in which one processor module is consuming the available grants from the arbiter.

The arbiter used by the second-generation processor pregrants or parks a grant to the processor module whenever the system bus goes idle. This feature eliminates arbitration overhead. The result is a lower miss penalty and an ability to sustain a

continuous stream of read transactions when the bus is not utilized by other system bus nodes. This arbiter enhancement does not cost additional arbitration overhead for other requests because the cost of unparking a grant was eliminated through the signaling protocol. This signaling protocol enabled the pregranted signal to be negated at the same time a new grant signal is asserted.

The Alpha 21164 microprocessor is capable of pending read-miss requests to the system interface. These read transaction requests sometimes have an associated victim that must be displaced by the requested read data. By pipelining these requests in relation to the system bus grants, a continuous stream of back-to-back system bus read or exchange transaction requests can flow because of the parked grant. Since the Alpha 21164 microprocessor is capable of continued execution while miss requests are pended, the processor designers had to carefully schedule the use of the B-cache. The fill data coming from system memory and the Alpha 21164 microprocessor are in contention for use of the B-cache. The system interface minimizes the time that the B-cache is allocated to accept the fill data while maintaining the flow of commands into the read miss transaction pipeline from the Alpha 21164 microprocessor. By allowing the microprocessor to have access to the B-cache before and after each fill, a continuous flow of transactions was realized. The continuous flow of transactions uses the available system bus bandwidth.

HANDLING OF SHARED DATA

A shared-database environment in which write transactions are prominent uses the system bus exclusive transaction types to establish ownership of the cache blocks. These transaction types minimize the system bus bandwidth usage by avoiding write broadcast transactions of modified blocks.

In a multiprocessor environment, a block that is valid in more than one cache is called a shared block. The coherence state of a shared block is VSD = 110. The following example summarizes the problem associated with a write transaction to a shared cache block in a system bus protocol without the exclusive transaction types.

Processor A has a modified but unshared cache block with state VSD = 101. Processor B wants to write the cache block that Processor A has modified. Processor B issues a read transaction on the system bus and then must immediately follow the read transaction with a write broadcast transaction of the modified data. The write broadcast transaction must be issued by Processor B because the read transaction was shared. At the end of the two bus transactions that it issued, Processor B's cache block state will be VSD = 101. Processor A has invalidated its cache block. Thus, two bus transactions were required from Processor B to write the modified cache block. With fair arbitration, however,

Processor B may not have access to the system bus after the read transaction. The write transaction may be blocked, thus creating other coherence situations. If two or more processors in a system are trying to write the same block, Processor B may not get access to the system bus to complete the write transaction. The system is potentially in deadlock.

The system bus protocol implemented by the KN470 enables the write transaction to complete but requires only one system bus exclusive-read transaction. In response to the Alpha 21164 microprocessor's request to modify a cache block, the processor initiates an exclusive-read transaction on the system bus. Other processor modules responding to this exclusive-read transaction provide the data if their block is dirty, but regardless of the dirty state, they also invalidate their cache block. The invalidation eliminates the shared state. If no other processor module has a dirty block, the data is returned from the system memory. The processor module that is issuing an exclusive-read transaction sets its cache block state to VSD = 101 as it fills. The write transaction that is pending in the processor can complete without broadcasting a write transaction to the system bus.

A system bus that does not support the exclusive transaction types requires a shared write transaction to a block to be decomposed into two system bus transactions. This can result in system bus bandwidth saturation. A system bus that supports the exclusive transaction types requires only one system bus transaction. In a shared-data environment in which write transactions to shared data are the prominent cause of cache misses, support for the exclusive transaction types helps preserve bus bandwidth. Also, the deadlock scenario presented above does not exist. The KN470 processor write transactions to a cached block consume only one system bus transaction and can always complete. The invalidate window does not exist during the time it takes for the write transaction to complete.

The system bus protocol implemented by the KN470 module allows forward progress during shared write transactions in the system. However, system software is expected to avoid repetitive write transactions to blocks that are shared without some higher level ownership protocol. Write transactions, if issued to a shared block by several processors, consume bus bandwidth and trigger false invalidations for bystanders. This may hinder forward progress and affect system performance.

SUPPORT OF AN INTERLOCK MECHANISM

The system interface implements an address lock register as specified in the Alpha Architecture Reference Manual to support software synchronization operations.[6] The address lock register in the system interface has a signal that reflects the state of a valid bit to the Alpha 21164 microprocessor. The microprocessor

manages the lock address register in the system interface based on sampling this signal during fill transactions from the system bus.

The Alpha 21164 microprocessor has an internal lock register that is maintained consistent with the lock register in the system interface, which is referred to as the external lock register. The external lock register is a backup copy of the Alpha 21164 microprocessor's lock register and is used only when instruction stream prefetching causes the locked address to be evicted from the B-cache. The execution of a load with lock instruction by an Alpha 21164 microprocessor results in a transaction that sets both internal and external lock flags and lock address registers.

The external lock flag is cleared by the system interface if the lock address matches the system bus address of either a write transaction or an exclusive transaction. The internal lock flag is cleared by the Alpha 21164 microprocessor due to system bus probe transactions from the write or exclusive transaction to a valid cache block.

The lock address resolution is a single-aligned 32-byte block and is consistent with the size of cache blocks in this system. The Alpha 21164 microprocessor has 64-byte internal lock register resolution. Since the address of a load to memory and the corresponding store to memory must both be within the same 16-byte aligned region, the difference in the resolution of the internal and the external lock registers was determined to be insignificant to performance.[6]

THE KN470 MODULE AND SYSTEM BUS CLOCKING

The KN470 module implements a low-cost synchronous clocking scheme. The scheme exploits the system bus clocking to run the Alpha 21164 microprocessor synchronous to the system bus. This scheme compensates for the half-cycle correction phase of the Alpha 21164 microprocessor's digital lock loop (DLL).

The AlphaServer 2100 system interconnect has an edge-to-edge clock architecture, and it implements an edge-to-edge data transfer scheme. The microprocessor has an internal DLL that synchronizes to a reference clock supplied by the clock repeater chip. Instead of trying to precisely control the clock skew across four different chips, data valid windows are set around the edge-to-edge data transfer clock edges to avoid setup or hold-time issues. This simpler clocking scheme takes advantage of the four delivered clock edges per cycle from the clock repeater chips. It also enables a simpler synchronous boundary between the Alpha 21164 microprocessor and the system interface. The synchronous clocking improves data transfer rates, lowers the miss penalty, and improves the pipeline efficiency among the components of the system.

Figure 2 shows the clocking scheme that is implemented on the KN470 module. The Alpha 21164 microprocessor accepts a differential clock at twice the desired internal clock frequency. The oscillator for the processor runs at 6, 7, 8, or 9 times the 41.66 megahertz (MHz) system bus clock frequency. The DLL subtracts one half of an internal clock cycle to maintain phase alignment with the system bus reference clock. This DLL scheme assumes that the internal clock frequency runs slightly faster than the system bus clock frequency. Given these scaling rates, the interface between the Alpha 21164 microprocessor and the system interface are locked at the system bus clock rate.

[Figure 2 (KN470 Clocking Scheme) is not available in ASCII format.]

The AlphaServer 2100 backplane distributes PECL-level system bus clocks PHI1 L and PHI1 H, and PHI3 L and PHI3 H differentially to each module on the system bus. Each module receives, terminates, and capacitively couples the clock signals into PECL-to-CMOS--level converters to provide four edges per system bus clock cycle. This level conversion is completed in the clock repeater chips. System bus handshake and data transfers occur from clock edge to clock edge and thus form a primary clock in the system. The remaining three edges in a clocking cycle are secondary clocks. The clock repeater chip, a custom CMOS clock chip, provides module-to-module clock skew of less than 1 nanosecond (ns) and is implemented to provide skew-regulated clock copies to be consumed by components on the module. The skew regulation is maintained by the repeater chip through the use of a feedback path or replica loop of the primary clock path. The KN470 module uses this clock repeater chip to generate the references for synchronous clocking from a central point.

Components on the module are clocked by outputs from the clock repeater chips. The clock repeater chips generate six copies of the primary clock TPHI1 H. TPHI1 H clocks are distributed as follows: one copy to the Alpha 21164 microprocessor, two copies to each of the two system interface ASICs, and one copy to the system bus arbiter chip. The Alpha 21164 microprocessor uses its copy of the primary clock as a reference clock for its DLL. The data transfers between the microprocessor and the system interface are edge-to-edge transfers and are referenced to the primary clock. The clock repeater chip generates three secondary clocks: TPHI1 L, TPHI3 H, and TPHI3 L. The clock-edge relationships among these four clocks are specified such that each clock edge is 90 degrees out of phase with the other two clock edges. The relationships among the different clock phases are shown in Figure 3 for the case of the Alpha 21164 oscillator with a frequency six times that of the system bus clock. The system interface uses all three secondary clocks for on-chip data transfers, whereas the arbiter chip uses one secondary clock, TPHI1 L.

[Figure 3 (Relationships among Different Clock Phases) is not available in ASCII format.]

This synchronous clocking scheme works well if the driver turn-on and turn-off times are extremely fast for all components. However, the technologies selected could not guarantee such speed. The Alpha 21164 microprocessor driver turn-on and turn-off times are fast, but the ASICs have slow turn-on and turn-off times. To compensate for the fast and slow driver characteristics, the edge-to-edge clocking scheme required a modification. The Alpha 21164 microprocessor uses its copy of TPHI1 H as the reference clock edge to align its SYSCLK1/2 H--generated interface output clocks. Though SYSCLK1/2 H does not physically connect to the system interface, the Alpha 21164 microprocessor uses the internal copy of the SYSCLK1/2 H edge to either drive data or receive data. The system interface uses its copy of the reference clock as the data receive edge for signaling from the Alpha 21164 microprocessor. To drive the data to both the microprocessor and the B-cache, the system interface uses the TPHI3 L secondary clock, which is phase-delayed 90 degrees from the primary clock TPHI1 H.

The above clocking scheme achieves single-clock, edge-to-edge data transfer rates without imposing overly strict constraints on clock routing and layout. The scheme can withstand larger than 1 ns of clock skew and compensates for the Alpha 21164 microprocessor's DLL half-cycle correction between the reference clock and SYSCLK1/2 H.

DESIGN TRADE-OFFS

The KN470 module design achieved aggressive schedule goals and achieved lower cost by means of the bit-slice design of the system interface. Also, the higher performance goal was realized while keeping the design complexity at a moderate level.

The bit-slice design of the system interface was motivated by the organization of the Alpha 21164 microprocessor's 64-bit error-correcting code--protected data bus. This forced at least a 64-bit slice organization. Other organizations were found to have too many pins or would have encountered system bus signal integrity problems because of long stubs and additional loads. The decision to also include the address and control functions was further motivated by the project's human resource constraints and its spending constraints. Designing one ASIC as a slice to implement the 128-bit-wide system interface was found to be the best choice.

The system interface controls the address and data paths between the Alpha 21164 microprocessor and the system bus. The system interface does not stall the system bus on transactions that require cache state changes in the B-cache. Instead, the interface posts a pended request to the processor for changing

the cache state of the B-cache. The system interface stalls the system bus when the processor has not acknowledged a previously pended request and the present transaction on the system bus needs a cache state change request. At the cost of increased complexity, the design could have been implemented such that the system bus would not stall in the absence of acknowledgments of previously pended requests. This level of complexity avoided the more complex issues of managing a queue of block invalidate, set block to shared, and read block transaction requests.

The KN470 module design implements a scheme of write transaction back-off or replay that exploits the transaction replay queue of the Alpha 21164 microprocessor. This replay functionality helps the system interface handle cache state changes when simultaneous requests to write to the system bus and to invalidate from the system bus are made to the same cache block. The designers simplified the cache coherence management and logic design by avoiding the use of a pended write transaction in the system interface, which would have required a one-block write cache.

A write transaction from the Alpha 21164 microprocessor to the system bus is not considered complete until the system bus is granted. This nonpended scheme for write transactions enables write transaction replay from the Alpha 21164 microprocessor and avoids the requirement for the system interface to preserve logic states if a system bus transaction takes precedence. When the system bus transaction takes precedence, the system interface removes the arbitration request, signals the Alpha 21164 microprocessor to replay the write transaction, and flushes all states associated with the write transaction. The Alpha 21164 microprocessor must determine whether the write transaction has been affected by the change in its cache state and then decide to replay the write transaction or to perform another transaction such as a read transaction to revalidate the block.

Removing a system bus request from the arbiter chip rather than converting the write transaction to a no-operation transaction avoided a livelock condition. The livelock condition could have resulted from the system interface's completion of a no-operation transaction and re-requesting the system bus to complete the write transaction. While waiting for the grant to this second arbitration request, the system bus could force the Alpha 21164 microprocessor to replay the write again. In addition to avoiding the livelock condition, the replay scheme has the additional benefit of conserving bandwidth by not issuing no-operation transactions while the system bus interface is waiting for the Alpha 21164 microprocessor to replay the write transaction.

Removing a system bus request in response to other bus transactions reduces the probability of a timely completion of the write transaction from the Alpha 21164 microprocessor. More complex design approaches increase the probability that the write transaction will complete, but they do not guarantee the completion. This is a result of the uncertain time for a response

from the Alpha 21164 microprocessor to replay the write transaction in relation to the next system bus grant. The designers chose the simpler implementation to reduce logic design complexity and verification time.

PERFORMANCE OF AlphaServer 2100 SYSTEMS WITH KN470 MODULES

To validate the improved performance goal of the KN470 processor module in AlphaServer 2100 systems running Digital UNIX (formerly DEC OSF/1) version 3.2B, project engineers measured several industry-standard benchmarks. A brief description of each benchmark follows. Table 1 lists the benchmarks that were run on an AlphaServer 2100 Model 5/250 system, the number of processor modules in a configuration for each benchmark, the measured estimates or unaudited results of the benchmark, and the performance gain. Performance gain is reported as a ratio of the KN470 result to the top-performing, first-generation KN460 result. The ratios demonstrate that the KN470 processor module achieves the primary project goal by providing more performance to AlphaServer 2100 systems than the first-generation KN460 processor.

Table 1 Performance Data for an AlphaServer 2100 System That Incorporates the KN470 Processor Module

Benchmark	Number of Processor Modules per Configuration	AlphaServer 2100 Model 5/250	Performance Gain Expressed As a Ratio of Model 5/250 Performance to Model 4/275 Performance
SPEC CINT92			
SPECint92	1	277	1.4
SPECrate_int92	4	24,996	1.4
SPEC CFP92			
SPECfp92	1	410	1.4
SPECrate_fp92	4	37,926	1.4
AIM Suite III Benchmark Suite Performance (Estimated)			
Performance Rating	2	396	
Maximum User Loads		2,400	1.4
Performance Rating	4	719	
Maximum User Loads		3,100	1.3
LINPACK (MFLOPS)			
1000 X 1000	4	1,022	1.6
McCalpin			
copy	2	171	1.28
scale	2	169	1.27
sum	2	162	1.25
triad	2	162	1.27

The AlphaServer 2100 Model 5/250 system uses the KN470 processor module that incorporates the Alpha 21164 microprocessor operating at 250 MHz with a 4-MB B-cache. The AlphaServer 2100 Model 4/275 system uses the KN460 processor module with the Alpha 21064 microprocessor operating at 275 MHz with a 4-MB B-cache. The AlphaServer 2100 system remained fixed as the processor models were swapped for these performance measurements.

The Standard Performance Evaluation Corporation (SPEC) was formed to identify and create objective sets of applications-oriented tests, which can serve as common reference points. SPEC CINT92 is a good base indicator of CPU performance in a commercial environment. This benchmark is the geometric mean of ratios by which the six benchmarks in this suite exceed the performance of the reference machine. SPEC CFP92 may be used to compare floating-point intensive environments, typically engineering and scientific applications. SPEC CFP92 is the geometric mean of ratios by which the 14 benchmarks in this suite exceed the performance of the reference machine. SPEC Homogeneous Capacity Method benchmarks test multiprocessor efficiency. They provide a fair measure for the processing capacity of a system, namely, how much work the system can perform in a given amount of time. The SPECrate is a capacity measure. It is not a measure of how fast a system can perform any task but of how many of those tasks the system completes within an arbitrary time interval.

Developed by AIM Technology, the AIM Suite III Benchmark Suite was designed to measure, evaluate, and predict UNIX multiuser system performance. The benchmark suite uses 33 functional tests, and these tests can be grouped to reflect the computing activities of various types of applications. The AIM Performance Ratings identify the maximum performance of the system under optimum usage of CPU, floating-point, and disk caching. At a system's peak performance, an increase in the workload will cause a deterioration in performance. The AIM Maximum User Load Rating identifies system capacity under heavy multitasking loads, where disk performance also becomes a significant factor. Throughput is the total amount of work the system processes, measured in jobs per minute. Maximum throughput is the point at which the system is able to process the most jobs per minute.

The LINPACK benchmark is a linear equation solver written in FORTRAN. LINPACK programs consist of floating-point additions and multiplications of matrices. The LINPACK 1000 X 1000 solves a 1,000-by-1,000 matrix of simultaneous linear equations. The result is a measure of the execution rate in millions of floating-point operations per second (MFLOPS).

The McCalpin benchmark is a public domain set of programs that measures the effective memory bandwidth available to each processor in MB per second. The four parts of this benchmark, which are shown in Figure 4, perform a double-precision operation

j times, where j increments 2 million times. Often, the four numbers are averaged to show an effective memory bandwidth rating for the configuration.

Figure 4 The Four Parts of the McCalpin Benchmark

```
copy c(j) = a(j)                 ;copy a to c
scale b(j) = 3.0 * c(j)          ;multiply c times 3, store
                                 ; result in b
sum  c(j) = a(j) + b(j)          ;add a to b and store in c
triad a(j) = b(j) + 3.0 * c(j)   ;multiply c times 3, add to
                                 ; b, store result in a
```

Table 2 shows estimated AIM Suite III Benchmark Suite performance scaling for AlphaServer configurations of one to four processor modules. These results validate improvements in the ability of KN470 processor modules to scale in multiprocessor configurations.

Table 2 AIM Suite III Benchmark Suite Performance Scaling (Estimated)

	AlphaServer 2100 System			
Number of Processor Modules	1	2	3	4
Maximum Throughput Jobs/Minute	2,178	3,882	5,249	7,047
Model 5/250 Scaling	1.0	1.8	2.4	3.2
Maximum Throughput Jobs/Minute	1,451	2,229	2,998	3,587
Model 4/275 Scaling	1.0	1.5	2.1	2.5

SUMMARY

The implementation of the write-invalidate coherence protocol combined with synchronous clocking, a duplicate tag store, and pipelining cache-miss requests led to a more efficient use of the system bus bandwidth. A higher complexity design could have been realized but only at the risk of missing schedule deadlines. The KN470 processor development project achieved the goals of AlphaServer 2100 compatibility and performance improvement that were established early in the project.

ACKNOWLEDGMENTS

The development of this new generation of processor and its integration into the AlphaServer 2100 family required the outstanding dedication and contributions from many individuals. The authors wish to extend a large thank you to Steve Holmes for believing in and supporting this project. The authors also wish to acknowledge the key contributors to the project. The core design team of Chet Pawlowski, Jim Padgett, Judy Weiss Prescott, and Gary Zeltser devoted long hours from the concept development through the empirical verification and the manufacturing startup of this processor module and system model. The verification team of Abdollah Ataie, Erik Debriae, Norbert Eng, Jeff Metzger, Don Caley, Dick Beaven, and Ginny Lamere proved the design's integrity. Andy Ebert provided the ASIC test strategy and applications support. Peter Woods and Traci Post were responsible for the serial control subsystem and diagnostics. Stephen Shirron made the OpenVMS system boot on the new machine, and Kevin Peterson and Harold Buckingham provided the firmware and console bits and consultation regarding software issues. Janet Walsh and Jeff Kerrigan contributed operations support. Steve Brooks, Rich Freiss, and Dean Gagne developed the operating system software that supports this system. Simon Steely, Zarka Cvetanovic, and John Shakshober carried out performance analysis and validations. John Edmondson, Pete Bannon, Anil Jain, and Paul Rubinfeld designed the KN470-specific functionality in the Alpha 21164 microprocessor.

REFERENCES

1. F. Hayes, "Design of the AlphaServer Multiprocessor Server Systems," Digital Technical Journal, vol. 6, no. 3 (Summer 1994): 8-19.

2. B. Maskas, S. Shirron, and N. Warchol, "Design and Performance of the DEC 4000 AXP Departmental Server Computing Systems," Digital Technical Journal, vol. 4, no. 4 (Special issue 1992): 82-99.

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

4. J. Hennessy and D. Patterson, Computer Architecture: A Quantitative Approach (San Mateo, Calif.: Morgan Kaufmann, 1990): 467-474.

5. A. Russo, "The AlphaServer 2100 I/O Subsystem," Digital Technical Journal, vol. 6, no. 3 (Summer 1994): 20-28.

6. R. Sites, ed., Alpha Architecture Reference Manual (Burlington, Mass.: Digital Press, Order No. EY-l520E-DP, 1992).

BIOGRAPHIES

Nitin D. Godiwala

Nitin Godiwala is a principal engineer in Digital's Server Product Development Group. His area of expertise is digital system architecture and pipelined machines. As a contributor to the AlphaServer 2100 server product, he was the project leader and principal architect of the arbitration ASIC and the system interface ASICs for the Alpha 21064 and 21164 microprocessor-based processor modules. In previous work, he was a principal architect and designer of the system interface ASIC for the DEC 4000 processor module. Before coming to Digital in 1986, Nitin worked for Analogic Corp., Gould Modicon, and Honeywell Inc. He received a B.E. from Bombay University and an M.S. in computer and electrical engineering from the University of Wisconsin, Madison. He holds four patents and has eight patents pending.

Barry A. Maskas

A consultant engineer in Digital's Server Product Development Group, Barry Maskas was the project leader responsible for the development of the Alpha 21164 microprocessor-based AlphaServer 2100 and 2000 processors and systems. He is currently involved in further Alpha-based server system development work. In earlier work, Barry was the project leader and architect for the DEC 4000 system bus, backplane, and processor modules and for the architecture and development of custom VLSI peripheral chip sets for VAX 4000 and MicroVAX systems. He was also co-designer of the MicroVAX II processor and memory modules. Barry joined Digital in 1979 after receiving a B.S.E.E. from Pennsylvania State University. He holds eight patents.

TRADEMARKS

==========

==========