COMPAQ
Technology Brief

January 1999
ECG050/0199

Prepared by Workstations

Compaq Computer Corporation

Technology for Performance: Compaq Professional Workstation XP1000

Contents

Abstract: The Alpha based Compaq Professional Workstation XP1000 is the most powerful member of the Compaq Professional Workstation Line. Using the new Alpha 21264 processor, the XP1000 delivers the highest performance of any workstation, both on benchmarks and running technical applications. The Alpha processor is a full 64-bit processor, based on advanced RISC technology. It delivers exceptional computational power, especially on floating point computations, and has unprecedented memory bandwidth.

The Alpha 21264 processor is the third generation of Alpha processors. It builds on the performance heritage of Alpha by retaining the high clock speeds that are a hallmark of Alpha, starting at 500 MHz. The 21264 maintains full binary compatibility with earlier Alpha processors, while implementing *aggressive parallelism* to deliver roughly twice the performance of the predecessor 21164 processor running at the same speed.

Notice

Technology for Performance: Compaq Professional Workstation XP1000
Technology Brief prepared by Compaq Workstations Marketing Services

First Edition (January 1999)
Document Number ECG050/0199

Help us improve our technical communication. Let us know what you think about the technical information in this document. Your feedback is valuable and will help us structure future communications. Please send your comments to: Russ.Doty@digital.com

Introduction

The Alpha based Compaq Professional Workstation XP1000 is the most powerful member of the Compaq Professional Workstation Line. Using the new Alpha 21264 processor, the XP1000 delivers the highest performance of any workstation, both on benchmarks and running technical applications. The Alpha processor is a full 64-bit processor, based on advanced RISC technology. It delivers exceptional computational power, especially on floating point computations, and has unprecedented memory bandwidth.

The Alpha 21264 processor is the third generation of Alpha processors. It builds on the performance heritage of Alpha by retaining the high clock speeds that are a hallmark of Alpha, starting at 500 MHz. The 21264 maintains full binary compatibility with earlier Alpha processors, while implementing *aggressive parallelism* to deliver roughly twice the performance of the predecessor 21164 processor running at the same speed. The XP1000 is being initially delivered with a 500 MHz processor. Compaq anticipates upgrades to the 21264 processor that will allow it to run at speeds greater than 1,000 MHz (1 GHz) during the life of this product.

The XP1000 is a complete, balanced high performance system. It is engineered for performance at all levels, ranging from the Alpha 21264 processor through the large 4 MB L2 cache, high performance memory subsystem (delivering 2.6 GB/s bandwidth), dual independent 32/64 bit PCI buses, integrated Wide-Ultra SCSI disk subsystem, and PowerStorm graphics.

The XP1000 supports both Tru64 UNIX and Microsoft® Windows NT Workstation 4.0. Windows NT 4.0 is the standard version from Microsoft. Any Windows NT 4.0 CD can be used to install Windows NT 4.0 on an XP1000 system. Tru64 UNIX is a full 64-bit UNIX operating system. It is a mature, robust, proven OS that delivers the full performance and large memory capabilities of the Alpha processor.

The XP1000 is the highest end member of the Compaq Professional Workstation family. The XP1000 extends the Professional Workstation Line. It does not compete with or replace the other members of the line. The Professional Workstation family includes the Affordable Performance (AP) Line, the Scalable Performance (SP) Line, and the Extreme Performance (XP) Line. The Affordable Performance Line is based on the Intel Pentium® II processor and the Intel BX core logic chips, and offers the lowest cost Professional Workstations, with high performance and full Intel applications compatibility. The Scalable Performance Line uses the Intel Pentium II Xeon™ processor and the innovative Highly Parallel System Architecture (HPSA). It combines the highest performance Intel processors with the parallelism and bandwidth of HPSA to deliver the best performance available on industry standard workstation platforms. The Extreme Performance Line uses the Alpha 21264 processor and delivers the absolute highest performance for technical and scientific applications, a broad base of standard applications, and full support for both Tru64 UNIX and Windows NT operating systems.

Alpha 21264 Processor

Introduction

The Alpha microprocessor has been the performance leader since its introduction in 1992. An unequalled cycle time at the time, facilitated by a clean RISC architecture and leading edge design techniques, provided much of the performance difference. The 21264 (EV6) is the third generation superscalar Alpha microprocessor. (See [Dob92][Edm95] for descriptions of the prior two generations.) In this design, absolute performance leadership was again a project goal. The 21264 achieves this goal using a unique combination of advanced circuit and architectural techniques.

Detailed architectural and circuit analysis in the early stages of the 21264 project showed that more aggressive micro-architectural techniques were possible while continuing the leadership clock frequencies that have become an Alpha tradition. The 21264 shows that a clean RISC architecture not only allows for a very fast clock rate, currently up to 600 MHz, but it also allows for sophisticated micro-architectural techniques that maximize the number of instructions executed every cycle. This combination results in industry-leading performance levels for the third consecutive Alpha generation.

The 21264 is a superscalar microprocessor with out-of-order and speculative execution. Out-of-order execution implies that instructions can execute in an order that is different from the order that the instructions are fetched. In effect, instructions execute as soon as possible. This allows for faster execution since critical path computations are started and completed as soon as possible. In addition, the 21264 employs speculative execution to maximize performance. It speculatively fetches and executes instructions even though it may not know immediately whether the instruction will be on the final execution path. This is particularly useful, for instance, when the 21264 predicts branch directions and speculatively executes down the predicted path. The sophisticated branch prediction in the 21264 coupled with speculative and dynamic execution extracts the most instruction parallelism from applications. Branch prediction and speculative and dynamic execution are part of the hardware run-time environment; applications take advantage of these capabilities without modification or re-compilation.

The 21264 memory system is another enabler of the high performance levels of the 21264. On chip and off-chip caches provide for very low latency data access. In addition, many memory references can be serviced in parallel to all caches in the 21264 as well as to the off-chip memory system. This allows for very high bandwidth data access.

This paper describes many of the micro-architectural techniques used to achieve the high performance levels in the Alpha 21264 microprocessor. (Two other references to the 21264 are: [Gie97][Lei97].). Figure 1 shows a high-level overview of the 21264 pipeline. Stage 0 is the *instruction fetch* stage that provides four instructions per cycle from the instruction cache. Stage 1 assigns instructions to slots associated with the integer and floating-point queues. The *rename* (or *map*) stage (2) maps instruction “virtual” registers to internal “physical” registers and allocates new physical registers for instruction results. The *issue* (or *queue*) stage (3) maintains an inventory from which it dynamically selects to issue up to 6 instructions – this is where instruction issue reordering takes place. Stages 4 and higher constitute the instruction execution stages that support all arithmetic and memory operations. Each stage is described in more detail on the following page. This section was taken from [Kes98].

Instruction Fetch

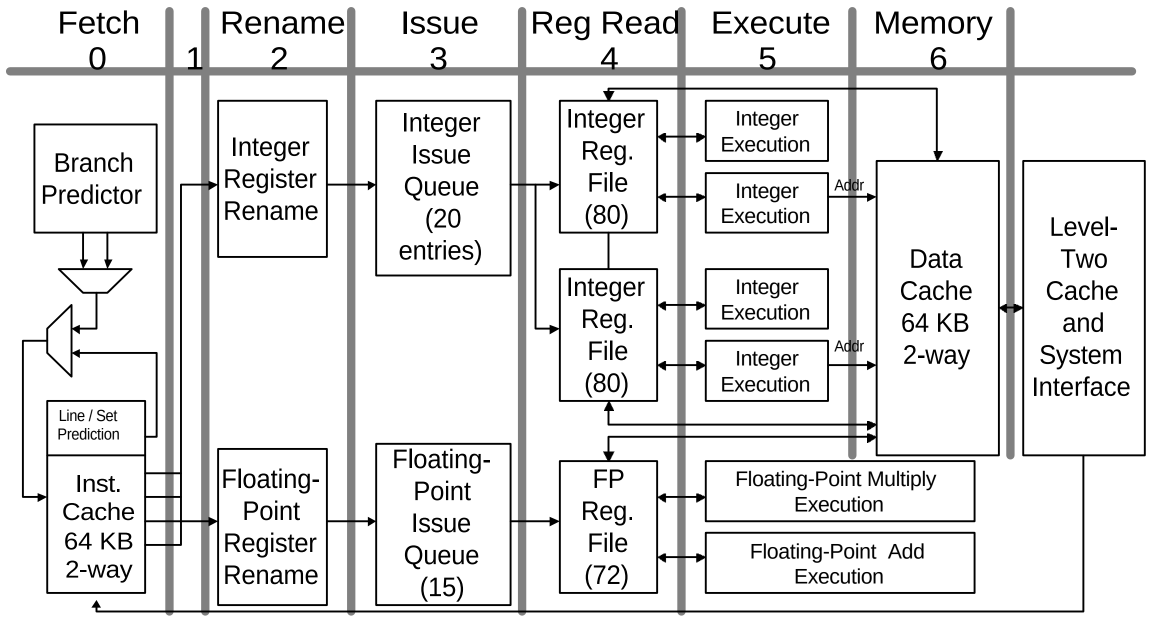
The instruction fetch stage is the beginning of the 21264 instruction pipeline. Four instructions are fetched each cycle to be delivered to the out-of-order execution engine. The 21264 uses many architectural techniques to provide maximum fetch efficiency. One important enabler is a large, 64K byte, two-way set-associative instruction cache. This offers much improved hit rates as compared to the 8K direct-mapped instruction cache used in the Alpha 21164.

Line and Set Prediction

The 21264 instruction cache implements two-way associativity via a line and set prediction technique that combines the speed advantage of a direct-mapped cache with the lower miss ratio of a two-way set-associative cache. Each fetch block of four instructions includes a line and set prediction. This prediction indicates from where to fetch the next block of four instructions, including which set (i.e., which of the two choices allowed by two-way associative cache) should be used. These predictors are loaded on cache fill and dynamically re-trained when they are in error. The mispredict cost is typically a single-cycle bubble to re-fetch the needed data. Line and set prediction is an important speed enhancement since the mispredict cost is so low and the line or set mispredictions are rare (the hit rates are typically 85% or higher in simulated applications and benchmarks).

In addition to the speed benefits of direct cache access, there are other benefits that come from line and set prediction. For example, frequently encountered predictable branches, such as loop terminators, will avoid the mis-fetch penalty often associated with a taken branch. The 21264 also trains the line predictor with the address of jumps that use direct register addressing. Code using DLL (dynamically linked library) routines will benefit after the line predictor is trained with the target.

Figure 1: Basic 21264 Pipeline

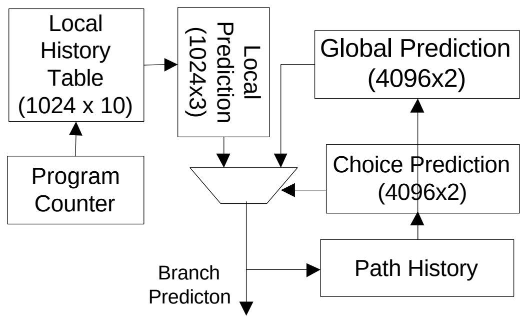


The 21264 line predictor does not train on every mispredict. There is a 2-bit hysteresis associated with each line that only enables training after the predictor has been in error several times recently. This avoids some unnecessary training and misprediction

Branch Prediction

Another important contributor to fetch efficiency is branch prediction. The 21264 speculative execution capabilities make branch prediction a more important contributor to overall performance than with previous microprocessor generations. Studies show that branch pattern behavior sometimes correlates with the execution of a single branch at a unique PC location (i.e.,

Figure 2: 21264 Tournament Branch Predictor


Local History Table (1024 x 10)
Local Prediction (1024x3)
Global Prediction (4096x2)
Program Counter
Choice Prediction (4096x2)
Path History
Branch Predicton

local history), and pattern behavior sometimes correlates with execution of all previous branches (i.e., global history). Both correlation techniques are needed to extract the maximum branch prediction efficiency.

The 21264 implements a sophisticated tournament branch prediction scheme that dynamically chooses between local and global history to predict the direction of a given branch [McF93]. The result is a branch predictor that produces a better prediction accuracy than larger tables of either individual method, 90 to100% on most simulated applications and benchmarks.

Figure 2 is a block-diagram of the 21264 tournament branch predictor. The local prediction path is on the left. The global prediction path and the chooser are on the right The local history table holds 10 bits of branch history for up to 1024 branches, indexed by the instruction address. The 21264 uses the 10-bit local history to pick from one of 1024 prediction counters. The local prediction is the most-significant bit of the prediction counter. After branches issue and retire, the 21264 inserts the true branch direction in the local history table and updates the referenced counter (using saturating addition) to train the correct prediction.

The local prediction is very useful, for example, with an alternating taken/not-taken sequence from a given branch. The local history of the branch eventually resolves to either 1010101010 or 0101010101 (the alternating pattern of zeroes and ones indicates the success or failure of the branch on alternate invocations). As the branch executes multiple times, it saturates the prediction counters corresponding to these local history values and makes the prediction correct. Any repeating pattern of 10 branch invocations can be trained this way.

The global predictor is a 4096 entry table of two-bit saturating counters that is indexed by the global, or path, history of the last twelve branches. The prediction is the most-significant bit of the indexed prediction counter. Global history is useful when the outcome of a branch can be inferred from the direction of previous branches. For example, if a first branch that checks for a value equal to ten succeeds, a second branch that checks for the same value to be even must also always succeed. The global history predictor can learn this pattern with repeated invocations of the two branches. Eventually, the global prediction counters with indices that have their lower-most bit set (indicating that the last branch was taken) will saturate at the correct value. The 21264 maintains global history with a silo of thirteen branch predictions and the 4096 prediction counters. The silo is backed up and corrected on a mispredict. The 21264 updates the referenced global prediction counter when the branch retires.

The 21264 updates the chooser when a branch instruction retires, just like the local and global prediction information. The chooser array is 4096 two-bit saturating counters. If the predictions of the local and global predictor differ, the 21264 updates the selected choice prediction entry to support the correct predictor.

The instruction fetcher forwards speculative instructions from the predicted path to the execution core after a branch prediction. The 21264 can speculate through up to 20 branches.

Register Renaming and Out-of-Order Issue

The 21264 offers out-of-order efficiencies with much higher clock speeds than competing designs. This speed, however, is not accomplished by the restriction of dynamic execution capabilities. The out-of-order issue logic in the 21264 receives four fetched instructions every cycle, renames and re-maps the registers to avoid unnecessary register dependencies, and queues the instructions until operands and/or functional units become available. It dynamically issues up to six instructions every cycle, four integer instructions and two floating-point instructions.

Register Renaming

Register renaming assigns a unique storage location with each write-reference to a register. The 21264 speculatively allocates a register to each register-result-producing instruction. The register only becomes part of the architectural register state when the instruction commits or retires. This allows the instruction to speculatively issue and deposit its result into the register file before the instruction is committed. Register renaming also eliminates write-after-write and write-after-read register dependencies, but preserves all the read-after-write register dependencies that are necessary for correct computation. Renaming extracts the maximum parallelism from an application since only necessary dependencies are retained and speculative execution is allowed in the instruction flow.

In addition to the 64 architectural (i.e., software-visible) registers, up to 41 integer and 41 floating point registers are available to hold speculative results prior to instruction retirement in a large 80 instruction in-flight window. This implies that up to 80 instructions can be in partial states of completion at any time, allowing for significant execution concurrency and latency hiding. (Particularly since the memory system can track an additional 32 in-flight loads and 32 in-flight stores.) The 21264 tracks outstanding unretired instructions (and their associated register map information) so that the machine architectural state can be preserved in the case of a mis-speculation.

Out-of-Order Issue Queues

The issue queue logic maintains a list of pending instructions. Each cycle the separate integer and floating-point queues select from these instructions as they become data-ready using register scoreboards based on the renamed register numbers. These scoreboards maintain the status of the renamed registers by tracking the progress of single-cycle, multiple-cycle, and variable cycle (i.e., memory load) instructions. When functional unit or load data results become available, the scoreboard unit notifies all instructions in the queue that require the register value. These dependent instructions can issue as soon as the bypass result becomes available from the functional unit or load. Each queue selects the oldest data-ready and functional-unit-ready instructions for execution each cycle. The 20-entry integer queue can issue four instructions, and the 15-entry floating-point queue can issue two instructions per cycle.

The 21264 cannot schedule each instruction to any of the four integer execution pipes. Rather, it statically assigns instructions to two of the four pipes, either upper or lower, before they enter the queue. The issue queue has two arbiters that dynamically issue the oldest two queued instructions each cycle within the upper and lower pipes, respectively. This static assignment to upper or lower fits well with the integer execution engine. (Some functional units do not exist in both the upper and lower pipelines, and the dynamic scheduling of the issue queue minimizes cross-cluster delays.)

The queues issue instructions speculatively. Since older instructions are given priority over newer instructions in the queue, speculative issues do not slow down older, less speculative issues. The queue is collapsing. An entry becomes immediately available once the instruction issues or is squashed due to mis-speculation.

Execution Engine

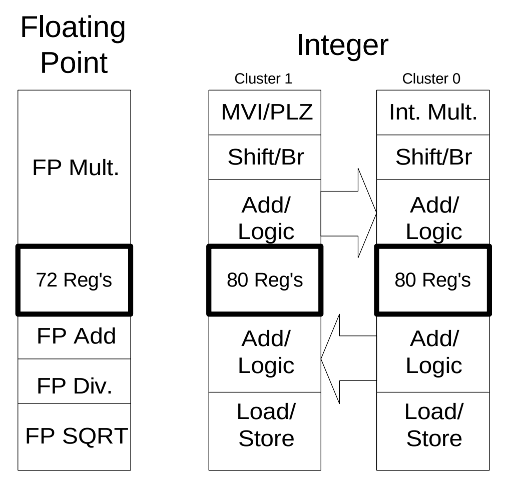
To support the high frequency goals of the project, the design of the integer register file was a particular challenge. Typically, all execution units require access to the register file, making it a single point of access and a potential bottleneck to processor performance. With as many as fourteen ports necessary to support four simultaneous instructions in addition to two outstanding load operations, it was clear that the register file would be large and an implementation challenge. Instead, the 21264 splits the file into two clusters that contain duplicates of the 80-entry register file. Two pipes access a single register file to form a cluster, and the two clusters are combined to support 4-way integer instruction execution.

The incremental cost of this design is an additional cycle of latency to broadcast results from each integer cluster to the other cluster. Performance simulation shows this cost to be small, a few percent or less performance difference from an idealized unclustered implementation with most applications. The integer issue queue dynamically schedules instructions to minimize the one cycle cross-cluster communication cost. An instruction can usually first issue on the same cluster that produces the result. This architecture provides much of the implementation simplicity and lower risk of a two-issue machine with the performance benefits of four-way integer issue. There are two floating-point execution pipes organized in a single cluster with a single 72-entry register file. Figure 3 shows the configuration.

This figure shows the four integer execution pipes (upper and lower for each cluster) and the two floating-point pipes in the 21264, together with the functional units in each.

The 21264 includes new functionality not present in prior Alpha microprocessors: a fully-pipelined integer multiply unit, an integer population count and leading or trailing zero count unit, a floating-point square-root functional unit, and instructions to move register values directly

Figure 3 21264 Execution Pipes



between floating-point and integer registers. It also provides more complete hardware support for the IEEE floating-point standard, including precise exceptions, NaN and infinity processing, and support for flushing denormal results to zero.

Memory System

The memory system in the 21264 is high-bandwidth, supporting many in-flight memory references and out-of-order operation. It receives up to two memory operations (loads or stores) from the integer execution pipes every cycle. This means that the on-chip 64 KB two-way set-associative data cache is referenced twice every cycle, delivering 16 bytes every cycle. In effect, the data cache operates at twice the frequency of the processor clock. Since the cache is double-pumped this way, there is full support for two memory references every cycle without conflict. The off-chip (level-two) cache provides a very fast backup store for the primary caches. This cache is direct-mapped, shared by both instructions and data, and can range from 1 to 16 MB. The off-chip clock-forwarded cache interface can support a peak data transfer rate of 16 bytes every 1.5 CPU cycles.

The latency of the virtual-indexed on-chip data cache is three cycles for integer loads and four cycles for floating-point loads. The latency to the physical-indexed off-chip cache is twelve cycles, depending on the speed of the cache. The 21264 supports many SRAM variants, including late-write synchronous, PC-style, and dual-data for very high frequency operation.

The 21264 also has a fast interface that allows the memory system surrounding the microprocessor to provide data quickly, typically from DRAM, upon a cache miss. The peak bandwidth of the clock-forwarded system interface is 8 bytes of data per 1.5 CPU cycles.

The 21264 memory core logics up to 32 in-flight loads, 32 in-flight stores, 8 in-flight (64-byte) cache block fills, and 8 cache victims. This allows a high degree of parallel memory system activity to the cache and system interface. It translates into high memory system performance, even with many cache misses. For example, Compaq has measured a 1 GB/s sustained memory bandwidth on the STREAMS benchmark [Str98].

Store or Load Memory Ordering

The 21264 memory system supports the full capabilities of the out-of-order execution core, yet maintains an in-order architectural memory model. This is a challenge, for example, when there are multiple loads and stores that reference the same address. It would be incorrect if a later load issued prior to an earlier store and, thus, did not return the value of the earlier store to the same address. This is a somewhat infrequent event, but it must be handled correctly. Unfortunately, the register rename logic cannot automatically handle this read-after-write memory dependency as it does other register dependencies because it does not know the memory address before the instruction issues. Instead, the memory system dynamically detects the problem case *after* the instructions issue (and the addresses are available).

The 21264 has hazard detection logic to recover from a mis-speculation that allowes a load to incorrectly issue before an earlier store to the same address. After the first time a load mis-speculates in this way, the 21264 trains the out-of-order execution core to avoid it on subsequent executions of the same load. It does this by setting a bit in a load wait table that is examined at fetch time. If the bit is set, the 21264 forces the issue point of the load to be delayed until all prior stores have issued, thereby avoiding all possible store and load order violations. This load wait table is periodically cleared to avoid unnecessary waits.

This example store and load order case shows how the 21264 memory system produces a result that is the same as in-order memory system execution while utilizing the performance advantages of out-of-order execution. Almost all of the major 21264 functional blocks are needed to implement this storeand load order solution: fetch, issue queue, and memory system. This implementation provides the highest performance for the normal case when there are no dependencies since loads can be issued ahead of earlier stores. It also dynamically adjusts to perform well in the less frequent case where a load should not be scheduled before a prior store.

Load Hit or Miss Prediction

There are mini-speculations within the 21264 speculative execution pipeline. To achieve the three-cycle integer load hit latency, it is necessary to speculatively issue consumers of integer load data before knowing if the load hit or missed in the on-chip data cache. The consumers that receive bypassed data from a load must issue the same cycle as the load reads the data cache tags, so it is impossible for the load hit or miss indication to stop the issue of the consumers. Furthermore, it takes another cycle after the data cache tag lookup to get the hit or miss indication to the issue queue. This means that consumers of the results produced by the consumers of the load data can also speculatively issue, even though the load may have actually missed!

The 21264 could rely on the general mechanisms available in the speculative execution engine to abort the speculatively executed consumers of the integer load data, but that requires a restart of the entire instruction pipeline. Given that load misses can be frequent in some applications, this technique would be too expensive. Instead, the 21264 has a mini-restart to handle this case. When consumers speculatively issue three cycles after a load that misses, two integer issue cycles (on all four integer pipes) are squashed and all integer instructions that issued during those two cycles are pulled back into the issue queue to be re-issued later. This means that both the consumer of the load data and the consumer of the consumer will be restarted and re-issued.

While this two-cycle window is less costly than a full restart of the processor pipeline, it still can be expensive for applications that have many integer load misses. Consequently, the 21264 predicts when loads will miss and does not speculatively issue the consumers of the load data in that case. The effective load latency is five cycles rather than the minimum three for an integer load hit that is incorrectly predicted to miss.

The 21264 load hit or miss predictor is the most-significant bit of a 4-bit counter that tracks the hit or miss behavior of recent loads. The saturating counter decrements by two on cycles when there is a load miss, otherwise it increments by one when there is a hit.

The 21264 treats floating-point loads differently than integer loads for load hit or miss prediction. Their latency is four cycles and there are no single-cycle operations, so there is enough time to resolve the exact instruction that used the load result.

Cache Prefetching and Management

The 21264 provides cache prefetch instructions that allow the compiler and/or assembly programmer to take full advantage of the parallelism and high-bandwidth capabilities of the memory system. These prefetches are particularly useful in applications that have loops that reference large arrays. In these and other cases where software can predict memory references, it can prefetch the associated (64-byte) cache blocks to overlap the cache miss time with other operations. Software prefetches can also eliminate unnecessary data reads and control cache-ability. The prefetch can be scheduled far in advance because the block is held in the cache until it is used.

Table 1 describes the cache prefetch and management instructions used in the 21264. The normal,

Table 1 21264 Cache Prefetch and Management Instructions

Instruction	Description
Normal Prefetch	The 21264 fetches the (64-byte) block into the (level one data and level 2) cache.
Prefetch with Modify Intent	The same as the normal prefetch except that the block is loaded into the cache in dirty state so that subsequent stores can immediately update the block.
Prefetchand Evict Next	The same as the normal prefetch except that the block will be evicted from the (level one) data cache as soon as there is another block loaded at the same cache index.
Write Hint 64	The 21264 obtains write access to the 64-byte block without reading the old contents of the block. The application typically intends to over-write the entire contents of the block.
Evict	The cache block is evicted from the caches.

modify-intent, and evict-next prefetches perform similar operations but are used in different specific circumstances. For each of them, the 21264 fills the block into the data cache if it was not already present in the cache. The write-hint 64 instruction is similar to a prefetch with modify intent except that the previous value of the block is not loaded. For example, this is very useful for zeroing out a contiguous region of memory. The evict instruction evicts the selected cache block from the cache.

Conclusion

The 21264 is the fastest microprocessor available. It reaches excellent performance levels using a combination of the expected high Alpha clock speeds together with many advanced micro-architectural techniques, including out-of-order and speculative execution with many in-flight instructions. The 21264 also includes a high-bandwidth memory system to quickly deliver data values to the execution core, providing robust performance for many applications, including those without cache locality.

21272 Core Logic Chipset

Overview

While most of the attention is given to the processor, the core logic chipset plays a critical role in the capabilities and performance of a system. The core logic chipset connects the processor to the outside world. It includes the memory controller, the interface to I/O subsystems such as PCI, and connection to mundane but critical components, such as keyboard and mouse.

Thus, the core logic chipset has the responsibility of feeding data to the processor. As modern processors have a voracious appetite for data, the demands on the core logic chipset have gone from bandwidths of tens of MB/s to hundreds of MB/s to the Alpha 21264 which requires GB/s of data.

The core logic chipset includes the circuitry necessary to directly control and interact with the memory chips and includes a complete implementation of the PCI bus. On older designs, the core logic chipset would also control the L2 cache. Modern processors, including the Alpha 21264, provide a fully independent dedicated cache bus directly from the processor. This backside bus is directly controlled by the processor.

Three approaches can be taken to deliver greater bandwidth:

- One is to use a faster bus, so that more data operations can be performed per second.
- The second is to use a wider bus, so that more data can be moved in a single operation.
- The third is to provide multiple buses that can operate in parallel.

An issue closely related to bandwidth is latency. Latency is the amount of time required to get the data. In many cases, latency has a greater actual effect on performance than bandwidth. This is especially true when working with small amounts of data. The time required to access the data is much smaller than the time required to transfer the data.

The 21272 (“Tsunami”) chipset is a high performance core logic chipset that is designed to work with the Alpha 21264 processor. It provides high bandwidth (2.6 GB/s), supports wide memory buses, supports multiple banks of memory, and provides two independent 64-bit PCI buses. The 21272 chipset provides a range of implementation options, allowing several tradeoffs between cost and performance, and, as implemented in the XP1000, it provides maximum performance.

21272 Architecture and Implementation

The 21272 is implemented as a chipset, rather than a single chip. This is due to the amount of logic involved, as well as the large number of I/O pins required to support the wide buses that are necessary to get the required bandwidth from industry standard parts and interfaces. Many systems today use designs that minimize the number of I/O pins and compensate with a fast clock rate. They also use a system bus design, where everything is connected to a single backbone bus. These designs trade performance for cost.

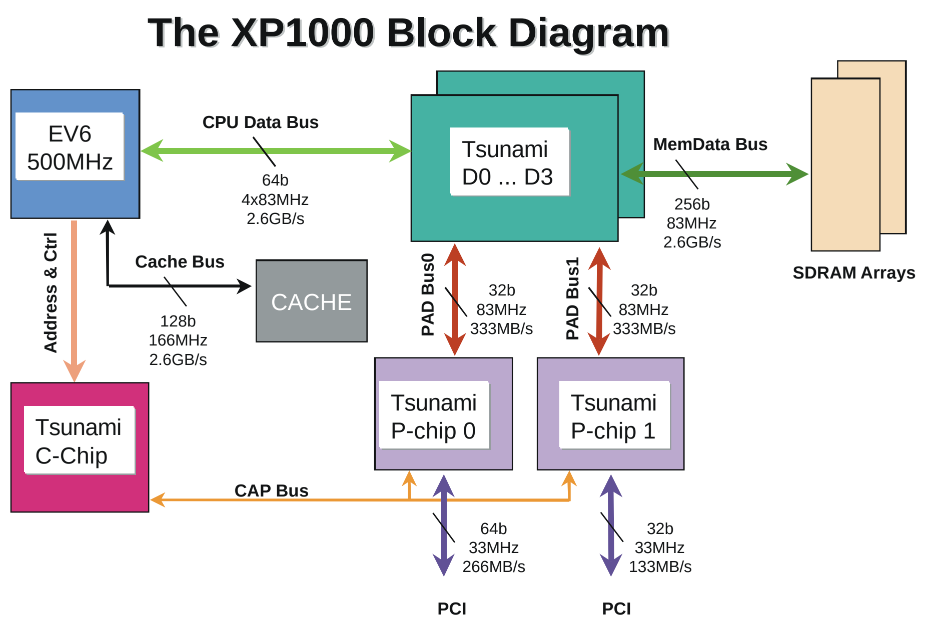
The 21272 uses a point to point, crossbar switch based design. This allows direct pairwise connection between two devices, and helps performance in two ways. First, since only two devices are connected to the bus, electrical design of the bus is simpler and the bus can be driven at higher clock rates than when multiple devices are connected to a single bus. A familiar example of this is the Accelerated Graphics Port (AGP), which supports higher clock rates than the similar PCI bus. Second, the aggregate bandwidth of the system improves as more devices are added. This is in direct contrast to a bus based design where a fixed bandwidth is shared by all devices.

The 21272 uses a variety of advanced techniques for performance. One of these is *clock forwarding*, where timing signals are sent in parallel with data signals, instead of data signals waiting for the clock signals. Since the clock signals are traveling down the same path as the data signals, the clock and data remain fully synchronized. Use of clock forwarding allows higher clock rates over longer distances, enabling system designers to build faster, more robust internal buses.

The 21272 is made of three unique chips:

- A controller chip or “Cchip”
- A data slice or “Dchip”
- A peripheral interface chip or “Pchip”

The Cchip chip issues commands and addresses to the Dchips and Pchips, which execute the data transfers. An implementation includes a single Cchip, two, four, or eight Dchips, and one or two Pchips. Since the XP1000 is designed for maximum performance, it uses a Cchip, four Dchips, and two Pchips.



Cchip

The Cchip is responsible for control of the I/O and memory subsystem. The Cchip issues commands and addresses to the Dchips and Pchips, which are then responsible for the actual data transfer.

The Cchip provides two system address ports, which allow it to support one or two Alpha 21264 processors. On the single processor XP1000, only a single system address port is used.

The Cchip provides four independent DRAM command and address ports, each of which can supply addresses for one memory array. Also provided is a Dchip control port. The XP1000 uses two DRAM ports, supporting two memory arrays.

The Cchip contains internal queues and buffers to receive, hold, and dispatch requests for the Dchips and Pchips. The Cchip includes multiple request queues and allocation logic to distribute operations across the memory arrays, as well as logic to ensure that ordering requirements for operations are maintained.

The Cchip is packaged in a 432-point Enhanced Super Ball Grid Array (ESBGA) package.

Dchip

Each Dchip contains multiple interfaces. These include memory bus data ports, CPU data ports, Pchip data ports, and a Cchip interface. In addition, the Dchip contains a set of queues and accumulators. These support DMA operations, PIO buffering, memory accumulation (to allow full bandwidth transfers from a pair of memory buses to a single CPU), and a write merge buffer (which holds memory data to be merged with Pchip data for DMA writes).

Each Dchip has two memory bus data ports. Each port is 36 bits wide, allowing for 4 bytes of data plus check bits. In some configurations, the two ports operate as a single 72-bit wide port.

The Dchip also includes 4 CPU data ports. Each port is 11 bits wide, allowing 8 data bits, 1 check bit, and a pair of forwarded clocks (one in each direction). Each port interfaces with one CPU using a clock-forwarding scheme that allows transfer of data every 3 ns (333 MHz). In some configurations the ports operate as a single 4-byte wide port. In the XP1000, the ports operate as 2-byte wide ports.

The Dchip receives all of its commands from the Cchip. The control from the Cchip consists of setting the switches within the Dchip to move data between ports or between ports and queues. All connections are possible except from one memory port to the other memory port.

The XP1000 uses four Dchips driving two memory arrays. Each memory array is a full 256 bit data path, transferring 32 bytes of data on each cycle. The memory arrays run at 83 MHz.

The XP1000 includes eight memory slots and uses industry standard PC100 SDRAM modules. Each SDRAM module is 72 bits wide (64 data bits plus 8 check bits). Four SDRAM modules fully populate a memory array and provide the 288 bit memory bus (256 bits of data plus 32 check bits). A memory array must be fully populated to function.

The XP1000 memory subsystem can be populated with four SDRAM modules for minimum cost, or with eight SDRAM modules for maximum performance and memory size. The XP1000 supports a minimum of 128 MB of memory by using four 32 MB memory modules and a maximum of 2 GB of memory using eight 256 MB memory modules.

The Dchip is packaged in a 304-point ESBGA package.

Pchip

The Pchip interfaces with both Cchip and Dchip and provides a 64-bit, 33 MHz, PCI 2.1 compliant interface. Each PCI bus provides 256 MB/s of I/O bandwidth. This 64-bit interface fully supports 32-bit PCI operation and all 32 bit PCI devices. The Cchip controls the Pchip. All data transfers to or from the Pchip are done through the Dchips.

The Pchip supports PIO, DMA and PCI To PCI (PTP) transfers, providing maximum performance and the ability to transfer data without CPU involvement. The Pchip provides a scatter-gather TLB, which supports direct-mapped and scatter-gather DMA memory access.

The XP1000 uses two Pchips, which provide two fully independent PCI buses. Each bus can communicate with the CPU, with main memory, or support direct communication between two devices on the PCI bus. Each Pchip supports a 64-bit PCI bus. One of the PCI buses is brought out to the PCI expansion slots as a 64-bit bus, the other is brought out to the PCI expansion slots as a 32-bit PCI bus.

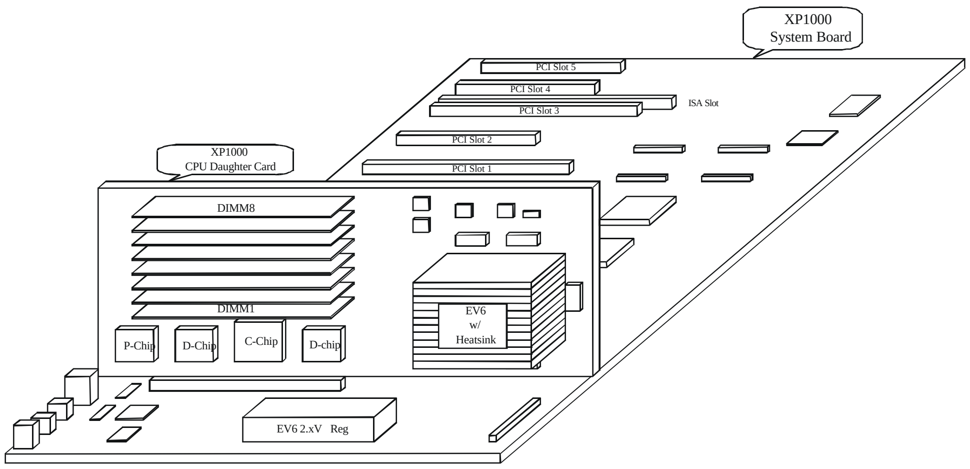
The Pchip is packaged in a 304-point ESBGA package.

XP1000 Architecture

Overview

The XP1000 melds the Alpha 21264 processor and the 21272 chipset with other components to deliver maximum performance with a full set of functional capabilities in a cost effective package. The XP1000 is a highly integrated design with ethernet networking, SCSI controller, and sound all packaged directly on the main board. The graphics card is the only device that utilizes a PCI slot in the base system. All other PCI slots are available for true expansion devices, rather than being consumed for basic system capabilities.

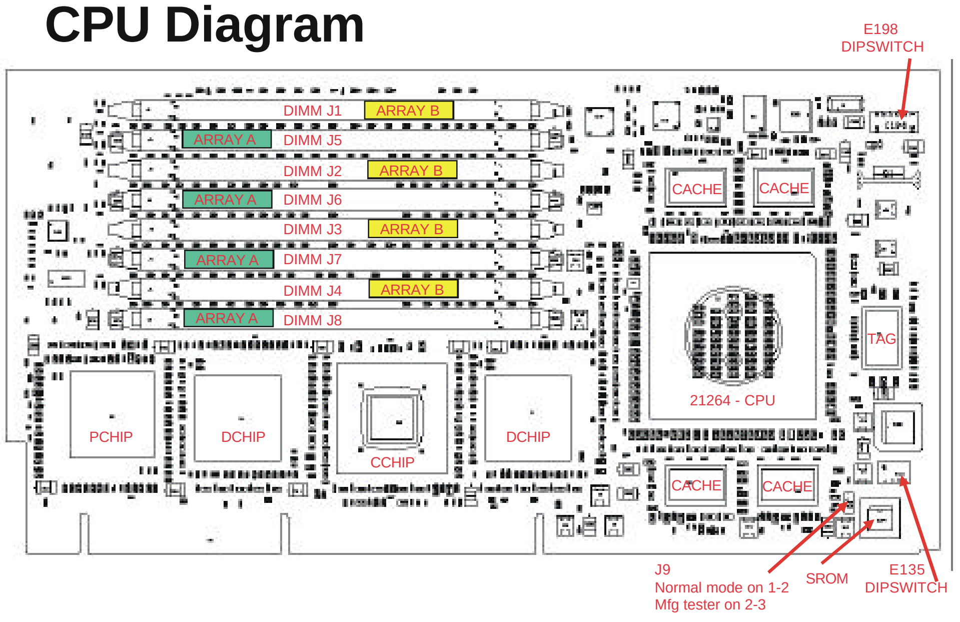
The XP1000 is built using a CPU daughter card module that attaches to a main system board. The CPU module contains the processor and system logic, while the main system board includes the I/O – embedded devices, PCI slots, and system I/O (mouse, keyboard, etc.).



CPU Module

The CPU module includes the Alpha 21264 processor, 4 MB of L2 cache, the 21272 core logic chipset, main memory, and the interface to the system board. The CPU module is a 12-inch x 7-inch board with components mounted on two sides. Virtually all of the components are high density, highly integrated parts using BGA, ESBGA or CPGA packaging. The CPU module relies on the system board for all external connections.

CPU Diagram



System Board

The system board supports all I/O. It contains the PCI and ISA expansion slots, embedded controllers for Ethernet, SCSI and sound, and standard system I/O: keyboard, mouse, diskette drive, serial ports, parallel port, and USB.

The system board also contains the voltage regulator, which provides 2.2 volts at 75 watts to power the 21264 processor.

The XP1000 has 5 PCI slots, driven by the two Pchips (Pchip 0 and Pchip1). Pchip 0 drives slots 1, 2 and 3. Slots 1 and 3 are 64 bit PCI slots, and slot 2 is configured with a 32 bit PCI connector.

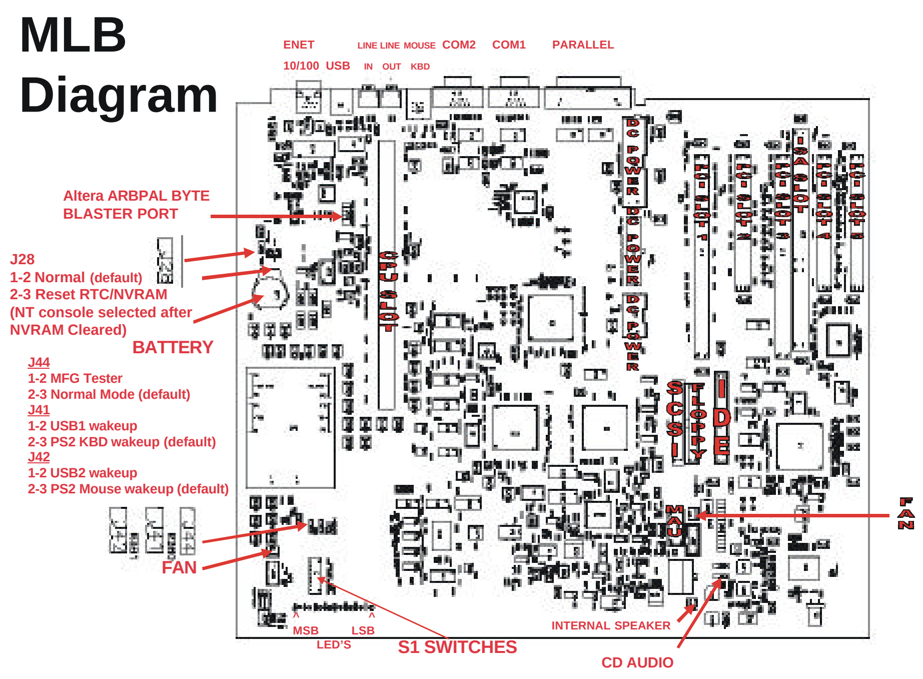
Pchip 0 also drives the Cypress 82C693UB “Southbridge” chip. The Southbridge is a multi-function chip that provides keyboard and mouse support, a two port USB hub, an Enhanced IDE port (used for CDROM), ISA bridge, real time clock and interrupt controller. A Super I/O chip on the ISA bus provides diskette drive, parallel port, and serial port support. An ESS 1887 sound chip on the ISA bus provides sound support.

Pchip 1 drives the embedded Ethernet controller, sound controller, SCSI controller, and a PCI to PCI bridge.

The Ethernet controller is an Intel 21143 high performance 10/100 Mb/s twisted pair Ethernet controller. It is a native 32 bit PCI device.

SCSI support is provided through a Qlogic 1040C Wide-Ultra SCSI controller. This controller provides a 40 MB/s interface to up to four internal storage devices. Due to cable length restrictions for Wide-Ultra SCSI, an optional PCI SCSI controller is required to support external devices.

Because of bus loading issues, a PCI to PCI (PTP) bridge is used to connect PCI slots 4 and 5 to Pchip 1. Use of the PTP bridge ensures signal integrity and that the Pchip drivers are not overloaded.

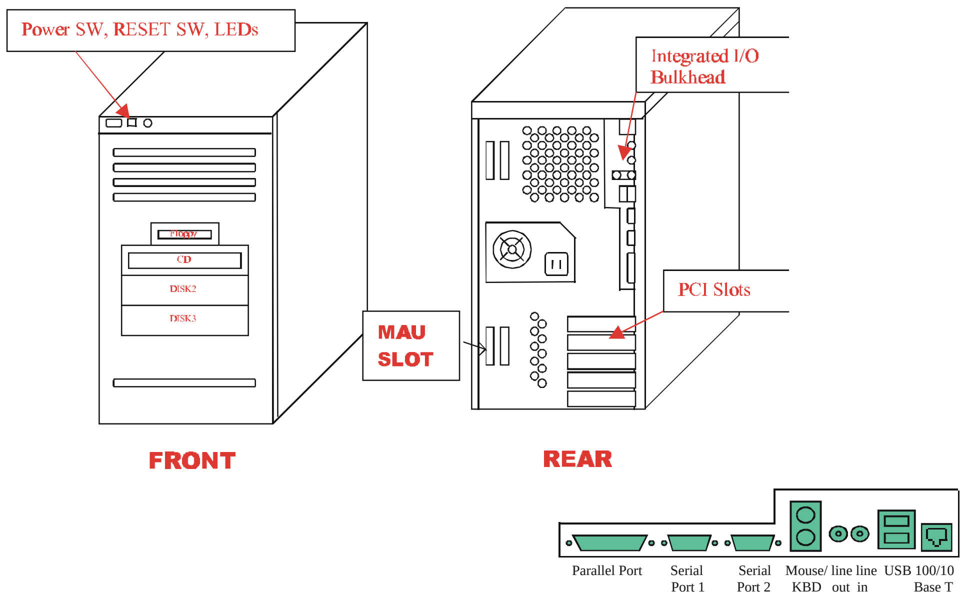


Physical Packaging

The XP1000 is packaged in a compact yet expandable mini-tower configuration. The system box contains the processor module and MLB, power supply, six storage bays, PCI card cage, and I/O ports. The system box is designed to provide power and cooling to support a fully loaded system, as evidenced by the ability to install four 10K rpm hard drives in the system.

Internal SCSI Hard Drives	Quantity Supported	Size	Controller
18.2-GB Wide-Ultra SCSI 10,000 rpm	1	1.6 in	Wide-Ultra SCSI
18.2-GB Wide-Ultra SCSI 7200 rpm	Up to 2	1.6 in	Wide-Ultra SCSI
9.1-GB Wide-Ultra SCSI 10,000 rpm	Up to 4	1 in	Wide-Ultra SCSI
9.1-GB Wide-Ultra SCSI 7200 rpm	Up to 4	1 in	Wide-Ultra SCSI
4.3-GB Wide-Ultra SCSI 7200 rpm	Up to 4	1 in	Wide-Ultra SCSI
4.3-GB Wide-Ultra SCSI 10,000 rpm	Up to 4	1 in	Wide-Ultra SCSI

The system box is designed for ease of access, including "no tool" access inside the box.



Software Development and Optimization

While one of the characteristics of Alpha 21264 is that it will run software developed for previous generations of Alpha processors, and run it faster than the previous processors, advances have been made in compilers and software development tools. Some of these improvements help all Alpha processors, and others are focused on the 21264.

Some of the enhancements are incorporated into the compilers, and are exploited simply by recompiling an application. New work has been done on Profile Directed Compilation, which allows information obtained from running a program to be fed back into the compilation process. And, finally, analysis tools are available that enable software engineers to understand the behavior of their software and make source code changes to improve performance.

Two of the biggest changes in the Alpha 21264 are dramatic increases in memory bandwidth and out of order instruction execution. Memory bandwidth, already a strength of Alpha, has been increased four to five times on the 21264, and is now measured in multiple GigaBytes per second.

On the other hand, latency (the time required to access data in memory) has not been improved as dramatically. This is because the fundamental memory technology, SDRAM, has not changed significantly. Several steps have been made to hardware to improve latency, including major enhancements to the cache subsystems.

Data Latencies			
In Register	Immediate	In Main Memory	34 cycles
In L1 Cache	2 cycles	In Virtual Memory	Millions of cycles
In L2 Cache	13 cycles		

The Alpha 21264 has also added new *prefetch* instructions. These instructions allow memory operations to be issued in advance, under software control, so that the data is available when needed. The key to benefitting from prefetch is knowing what data to fetch and when to start requesting it.

Prefetch is especially beneficial in floating point computation, which is extremely memory intensive. Consider a floating point multiplication, which involves four separate operations:

1. The first number is loaded from memory into a register.
2. The second number is loaded from memory into another register.
3. The two registers are multiplied together, and the result stored in a third register.
4. The register containing the result is written back into memory.

As can be seen, the memory load and store operations consume the most time, especially if the numbers have to be loaded from main memory!

A very common situation in numerically intensive computation is for the floating point operations to be combined into a loop, and the loop executed repeatedly. Compilers look for these loops, and perform a variety of optimizations on them. The new compiler adds prefetch operations to the code it generates, so that the data is available when it is needed. In addition to floating point numbers, other types of data that are used inside a loop can be prefetched.

Another new capability is *write hint*, which is commonly used in copy operations. While the Alpha processor does load or store operations that appear to operate directly on main memory, they actually work against the cache. Recall that the XP1000 actually reads and writes memory in chunks of 64 bytes (two cycles of its 256 bit wide memory bus). A memory write operation is actually done by reading 64 bytes of data from memory into cache, writing the changed value into the cache (typically one byte, four bytes, or eight bytes), and then writing the full 64 bytes from the cache back into main memory.

By using write hint, the software is able to tell the hardware that it is going to write an entire cache line, so there is no reason to read it in; it will just overwrite all of the data. For example, in a large copy operation, the system will read 64 bytes of source data into the cache and then write it back to a new memory location. The system will first read the data from the target location into cache, and then overwrite it with the source data and write it out to the target location. By using write hint, the system avoids the unneeded read of data from the target location. This has the effect of dramatically increasing the *effective* bandwidth of the system without requiring greater physical bandwidth.

The new compilers have been enhanced in other areas. These include improved register allocation, especially in the floating point registers, which allow more effective use of the register renaming capabilities of Alpha 21264. Considerable work has been done to ensure that out of order instruction execution can proceed effectively. An example of this occurs when several instructions change the values of a single byte (for example, when setting condition flags). Since several of these changes can be in the internal queues of the 21264 at the same time, significant stalls can occur while load or store synchronization is done. The new compilers recognize these conditions, and optimize the instructions to allow the 21264 to operate at full speed.

As has been previously discussed, the branch prediction done in the 21264 is very sophisticated. It would be even better, however, if the compiler could know which branches were *actually taken* when the program was executing.

This is exactly what is done with Profile Directed Compilation. Using this approach, the program is first compiled with special compiler switches to generate feedback. The compiler will *instrument* the program by inserting calls into the program that observe and report what the program does; for example, to report which branch is taken.

The program is then run. It can be run once or several times. It can be run with test data or with actual production data. The program can be a batch program or an interactive program. The data on how the program behaves is recorded into a *feedback file*. The program is then recompiled, using the feedback file. With the information from the actual program runs, the compiler can optimize for the actual behavior of the program.

The enhancements discussed so far are implemented in the part of the compiler that generates the actual machine instructions. This part of the compiler is shared by multiple languages, so all of these enhancements are incorporated into the "C", C++, and FORTRAN compilers.

Other tools are available to help programmers understand and improve the execution of their programs. These include DCPI for Windows NT and ATOM for UNIX. DCPI is a profiling tool that monitors the system and application and provides great detail on program execution, hot spots, and system resource usage.

ATOM is used to instrument programs and is a tool for building other analysis tools, such as Hiproff and 3rd degree (a memory checker).

SPIKE is another optimization tool that restructures already compiled applications for improved performance.

More details on these tools, as well as the actual software, can be obtained from http://www.windows.digital.com/nttools/

References

[Dob92] D. Dobberpuhl, et. al., “A 200 MHz 64-bit Dual Issue CMOS Microprocessor,” IEEE Journal of Solid State Circuits, Vol. 27, No. 11, November 1992, pp. 1555-1567.
[Edm95] J. Edmondson, et. al., “Superscalar Instruction Execution in the 21164 Alpha Microprocessor,” IEEE Micro, Vol. 15, No. 2, April 1995.
[Gie97] B. Gieseke, et. al., “A 600 MHz Superscalar RISC Microprocessor With Out-of-Order Execution,” IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers, Feb. 1997, pp. 176-177.
[Lei97] Daniel Leibholz and Rahul Razdan, “The Alpha 21264: A 500 MHz Out-of-Order Execution Microprocessor,” Proceedings of IEEE COMPCON ’97, pp. 28-36.
[McF93] S. McFarling, “Combining Branch Predictors,” Technical Note TN-36, Digital Equipment Corporation Western Research Laboratory, June 1993.
<www.research.digital.com/wrl/techreports/abstracts/TN-36.html>
[Str98] http://www.cs.virginia.edu/stream
[Kes98}R. E. Kessler, et. Al., “The Alpha 21264 Microprocessor Architecture”, technical paper, Compaq Computer Corporation