Cahners SPECIAL ISSUE COVERING THE 1999 MICROPROCESSOR FORUM

MICROPROCESSOR REPORT

VOLUME 13, NUMBER 13
OCTOBER 6, 1999

THE INSIDERS' GUIDE TO MICROPROCESSOR HARDWARE

Merced Shows Innovative Design

Static, Dynamic Elements Work in Synergy With Compiler

By Linley Gwennap

At this week's Microprocessor Forum, Intel unwrapped the Merced microarchitecture, showing how IA-64's EPIC design results in hardware that is both simpler and more powerful than traditional RISC or CISC processors. Gone are the complex instruction reorder buffers and register alias tables found in modern superscalar processors. In their place are more registers, more function units, and more branch predictors. These trade-offs eliminate unneeded complexity while leaving some dynamic structures in the hardware to handle events the compiler can't easily predict.

Merced microarchitecture manager Harsh Sharangpani described Merced as a six-wide machine, fetching and executing two bundles, or six instructions, per cycle at its peak rate. The processor uses a 10-stage pipeline to achieve high clock speeds, although Sharangpani declined to specify the target clock speed. IA-64 features such as predication, speculation, and register rotation are implemented with simple hardware structures. Dynamic structures, such as a decoupled fetch unit, nonblocking caches, and register scoreboarding, avoid pipeline stalls due to level-one (L1) cache misses.

The tighter coupling between the compiler and the processor improves hardware efficiency compared with traditional RISC or x86 designs. For example, the compiler has more control over branch prediction, allowing the processor to focus only on those branches that require dynamic prediction. Since all modern compilers perform instruction scheduling, allowing the compiler to communicate that information directly to the processor eliminates redundant scheduling circuitry. Ultimately, IA-64 gives the compiler more flexibility in scheduling instructions, increasing potential performance as well as the compiler's complexity.

Merced is no longer just a paper design. Intel and its system partners are currently validating first silicon, which has booted four operating systems and several key applications. The company says the processor is on track for mid-2000 production, with systems appearing in 2H00.

Six-Issue EPIC Processor

As Figure 1 shows, Merced can fetch and issue six instructions per cycle to a pool of function units that includes four integer units, two FPUs, and three branch units. Two of the integer units can also handle load/store instructions. Additional operations can be achieved using the SIMD integer and FP capabilities, the pointer post-increment feature of the load and store instructions, and the loop-counter update in special branch instructions (see MPR 5/31/99, p. 1).

The ability to handle up to three branches per cycle is unique among announced server processors; in fact, most can handle only one. In IA-64, branch-prediction instructions consume some of the branch slots, but there should be more than enough slots remaining to efficiently process branch-rich commercial server code.

The processor really shines on floating-point code, as each load/store unit can start a pair of double-precision

Continued on page 6

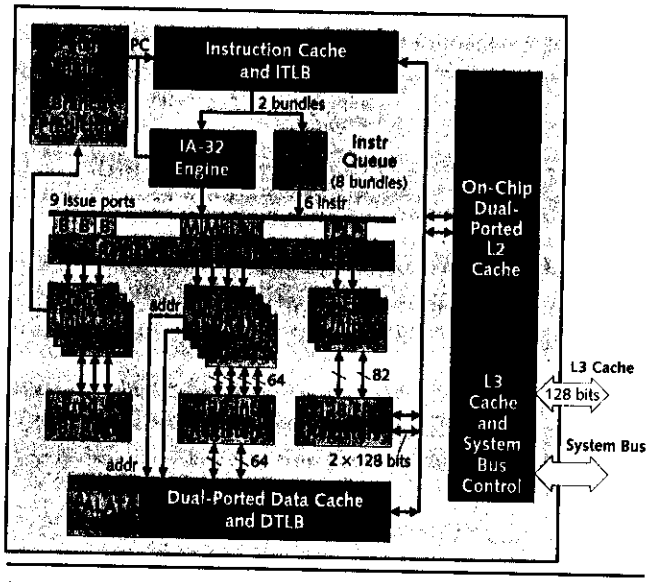


Figure 1. Merced can fetch and issue up to six instructions per cycle on nine issue ports: B = branch, M = memory, I = integer, F = floating point.

Inside: Power4 · Athlon 700 · StarCore SC140 · SH-5 · Cradle · 810E/820

Merced

Continued from page 1

(DP) loads per cycle using the LDFPD instruction, and each FPU can launch two DP operations using the FMA instruction. Thus, Merced can fetch four DP operands and execute four DP operations per cycle and still have two instruction slots left over to handle integer arithmetic or branches.

Most integer instructions execute in a single cycle. The dual FPUs are fully pipelined for all operations, but execution takes multiple cycles. Intel did not disclose the latency of Merced's FPU, but presumably it is no worse than that of Pentium III, which requires three cycles for an add or a multiply. As previously disclosed (see MPR 10/26/98, p. 16), each of Merced's FPUs can execute one DP MAC or two SP MACs.

The chip includes small L1 instruction and data caches backed by a larger on-chip L2 cache. Intel did not disclose details of the memory subsystem, such as the size and organization of these caches or the TLBs. We expect the L1 caches to be about 16K, allowing them to maintain an access time of about 1.5 cycles at a high clock speed. The L2 cache is probably about 256K and can supply 256 bits per cycle to the L1 caches. Both the data cache and the L2 cache can service two loads per cycle. FP loads and stores bypass the data cache and go directly to the L2 cache, as Figure 1 shows. Merced connects to an external L3 cache through a 128-bit bus operating at the full CPU speed and delivering at least 10 GBytes/s.

In addition to the backside L3 cache bus, Merced includes a frontside bus that connects to memory, I/O, and other processors, a design similar to that used by Pentium III today. Although Intel did not disclose the width or speed of this bus, we expect it to provide 2–3 GBytes/s of sustainable bandwidth, more than twice that of the current 100-MHz Xeon bus. This bus will support up to four Merced processors, using Intel's forthcoming 460GX chip set (see MPR 9/13/99, p. 4) as well as chip sets being developed by Intel partners.

Ten-Stage In-Order Pipeline

Figure 2 shows the Merced pipeline. At 10 stages, it is two to three stages shorter than Pentium III's pipeline (see MPR 2/16/95, p. 9); much of this advantage, however, is due to the complexities of the x86 architecture. Merced's pipeline is actually two to three stages longer than that of the Alpha 21264 (see MPR 10/28/96, p. 11), a high-speed RISC processor.

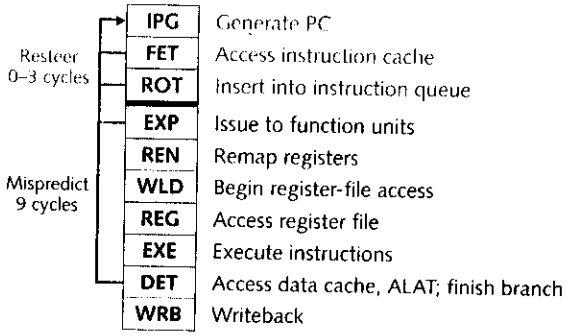


Figure 2. Merced's 10-stage pipeline features an independent three-stage front end and a straightforward execution engine.

Like many modern processors, Merced uses an instruction queue to decouple the instruction-fetch pipeline from the execution pipeline. This decoupling allows the front end to continue fetching instructions, even when the execution engine stalls; conversely, execution can continue using queued instructions if a branch causes a pipeline bubble, which Intel calls a resteer. The queue holds eight bundles, or 24 instructions, enough to cover resteers but not enough for a full branch misprediction.

The front end of the pipeline consists of three stages. After calculating the fetch address in the first stage, the processor accesses the instruction cache in the FET stage, and instructions flow into the queue in the third stage. After spending zero or more cycles in the queue, instructions are issued in the fourth stage. Register remapping occurs in the next stage, and accessing the large (128-entry) register files requires the two stages that follow. Finally, instructions execute in the EXE stage, with DET available to access the data cache or to complete branches.

Merced does not reorder instructions as out-of-order RISC or CISC processors do. But due to the different latencies of integer math, loads, FP math, and cache misses, instructions can finish executing out of order. The processor employs a register scoreboard to determine if a target register has been updated. As long as no instruction requires the result of a multicycle operation, the pipeline continues flowing normally. It stalls only if an instruction attempts to access a register that the scoreboard indicates is unavailable.

CPU, Compiler Both Predict Branches

With a branch misprediction penalty of nine cycles, Merced, like most modern processors, must avoid mispredicted branches at all costs. One of the strengths of IA-64 is that it can eliminate many branches using predication, completely avoiding the possibility of a misprediction. But Intel didn't ignore the branches that aren't eliminated; to accurately predict these branches, Merced employs a variety of hardware and software techniques that go well beyond the methods implemented in current processors.

First, Merced implements a branch predict instruction, BRP, that the compiler can use to help the processor more accurately predict branches and prefetch target instructions. The BRP instruction provides the address of an upcoming branch instruction, its predicted target address, and the "importance" of the branch, as well as other prediction aids.

The implementation of BRP varies from processor to processor. Merced implements four target address registers (TARs) that hold the targets of the most recent BRP instructions of high importance. Each TAR also holds the address of the branch instruction. When the program counter matches one of these branch addresses, the corresponding target address is fed to the instruction cache on the next cycle.

Thus, up to four individual branches can achieve zero-cycle execution using this mechanism.

The second opportunity to redirect, or resteer, the fetch stream occurs in the FET stage, as Figure 3 shows. Here, Merced employs three more conventional prediction mechanisms—an 8-entry return stack buffer (RSB) to predict subroutine returns, a 512-entry branch history table (BHT), and a 64-entry branch target address cache (BTAC)—but uses them in a unique fashion.

Unlike standard RISC and CISC architectures, IA-64 can provide a static-prediction "hint" to indicate that easy-to-predict branches should not be placed in the BHT. Thus, although the Merced BHT is much smaller than the ones in the 21264, AMD's Athlon (see MPR 8/23/99, p. 1), and other leading microprocessors, it may achieve similar effectiveness by focusing only on those branches that require dynamic prediction.

The four-way set-associative BHT uses a two-level PAs algorithm (see MPR 3/27/95, p. 17). Each entry, selected using the branch address, tracks the four most recent occurrences of that branch. This 4-bit value then indexes one of 128 pattern tables (one per set). The 16 entries in each pattern table use the standard 2-bit saturating up/down counter to predict the branch direction. The total storage required for this BHT is about 20 Kbits.

A second smaller BHT handles multiway branches. This 64-entry structure uses the same two-level algorithm but keeps three history registers per bundle entry. It does a find-first-taken selection to provide the first predicted taken address or indicate that none is predicted taken.

The compiler can place addresses directly in the BTAC using BRP instructions. A branch that hits in the BTAC or in the RSB immediately routes its target address back to the front of the pipeline, creating a single-cycle bubble in the fetch stream. As long as the fetch stream is ahead of the execution engine, this bubble will not stall the processor. If the BHT predicts the branch to be taken but the smaller BTAC does not contain the target address, it must be computed later by one of two branch address calculators (BAC).

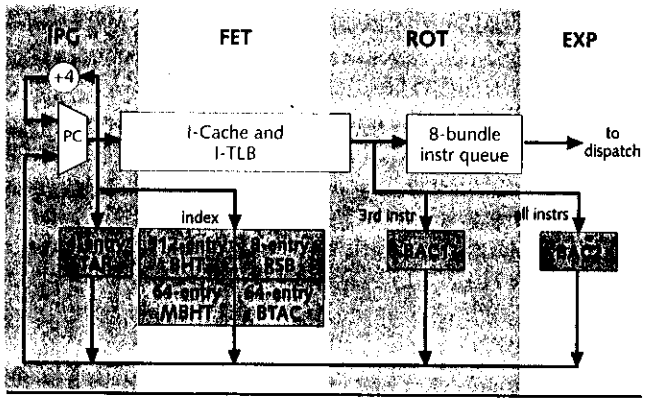


Figure 3. The front end of the pipeline contains several mechanisms that predict branches and redirect the fetch stream with minimal delays.

A third resteer occurs in two situations. BAC1 can compute the target address of a branch in the third slot of either bundle. Most templates place a branch in the third slot, so BAC1 will handle most branches. If the branch direction was not predicted by the BHT, the BAC will use the static prediction encoded in the branch instruction. BAC1 also contains logic that tracks the loop count (LC) register and overrides the TAR resteer when LC indicates a loop exit. Either of these cases causes a two-cycle bubble. Finally, BAC2 can calculate the target address of a branch in any slot; if this unit is used, it causes a three-cycle bubble.

In most cases, these bubbles will not stall the execution pipeline, although the instruction buffer must be nearly full to cover the rare three-cycle bubble. Other than the effect of these bubbles, branches will stall the execution pipeline only if mispredicted; that is, if the final result of all resteers is found to be incorrect once the branch condition is finally evaluated in the DET stage.

Simple Instruction Issue

Because it does no reordering, Merced can assign instructions to function units as soon as they exit the queue, in the EXP stage. The queue emits two bundles (six instructions) at once. The template fields in these bundles indicate the type of each instruction (integer, memory, FP, etc.). A standard instruction set would require a full crossbar to issue any six arbitrary instructions to a set of function units, consuming die area and potentially extending the cycle time. IA-64, in contrast, allows a limited number of templates (see MPR 5/31/99, p. 1); for example, memory instructions are never in the third slot, whereas FP instructions are never in the first slot. As Figure 4 shows, each function unit must choose among no more than three instructions, simplifying the muxes and instruction routing.

The templates also indicate the end of each instruction group, marked by a "stop." The compiler specifies these

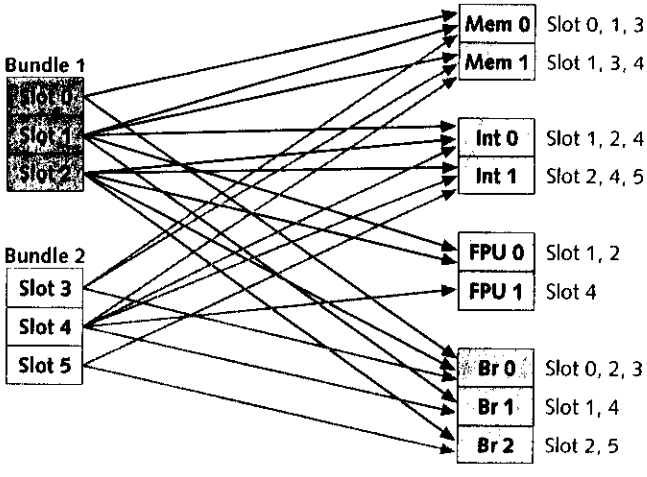
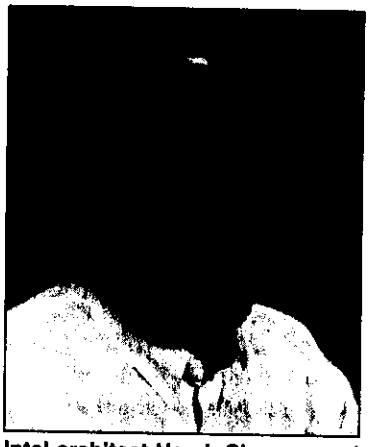


Figure 4. Merced dispatches up to six instructions per cycle from two bundles with three slots each. Because certain instruction types are required to be in certain slots, the dispatch task is simplified.

groups such that they contain no data dependencies. If the template contains a stop, the processor simply holds subsequent instructions until the next cycle. Thus, the hardware does not check for dependencies during the issue process. Traditional superscalar processors have extensive logic to check for such dependencies, and this logic grows exponentially worse as the issue width increases.

All instructions from both templates can be issued, as long as there is no stop (except at the end of the second bundle) and no resource is oversubscribed. The latter occurs if there are more than two memory instructions, for example. Any instructions not issued on the first cycle will be issued in a subsequent cycle; although this situation does not stall the execution pipeline, it permits the instruction buffer to fill.

After the processor issues instructions, it must fetch their operands. To support IA-64's register frames and register rotation (see MPR 3/8/99, p. 16), Merced remaps the register addresses using small 7-bit adders in the REN stage. In WLD, the remapped addresses flow to the register file, which is accessed in REG. Despite the simpler register-renaming hardware, this process takes three cycles in Merced but only two cycles in the 21264 or even Pentium III. Some of this increase is due to the longer access time of the large multiported register file, but the nearly empty WLD stage looks suspiciously like a late addition to the pipeline, needed to meet the clock-speed target.



Intel architect Harsh Sharangpani described Merced's EPIC microarchitecture at the Forum.

Merced includes a register stack engine (RSE) to automatically handle any spills and fills required when register frames exceed the size of the physical register file. In these situations, the RSE stalls the machine to issue the necessary save or restore requests using the two memory ports. In future IA-64 designs, the RSE could take advantage of unused memory slots, issuing loads or stores before the registers are needed, but the Merced team opted for a simpler design.

Predication and Speculation

All IA-64 instructions are executed conditionally, based on the contents of one of 64 1-bit predicate registers (see MPR 10/27/97, p. 1). Because of its large supply of function units, Merced simply executes all instructions through EXE, canceling their results in DET if the predicate turns out to be false. Canceling instructions earlier would eliminate some resource conflicts but would require an early access to the predicate registers that could wreak havoc with the pipeline. Instead, Merced reads the predicate value along with the other operands and passes it along to the retirement logic.

The designers did make an effort to avoid stalling the pipeline for an instruction that will later be canceled. The pipeline will normally stall if an instruction requires data that is not yet available, but if that instruction is predicated with a false value, the stall is completely avoided as long as the predicate is precomputed in the register file. If the predicate must be bypassed from the previous EXE stage, the stall will be only one cycle, or two cycles if the predicate comes from a floating-point compare (FCMP).

IA-64 forms conditional branches by predicating the basic branch instruction. The architecture allows the compiler to group a branch with an instruction that generates its predicate—one of the few data dependencies permitted within an instruction group. To handle this case, Merced does not determine the final direction of the branch until the DET stage, one cycle after the compare instruction executes in EXE. In the DET stage, the processor analyzes up to three branch conditions and chooses the correct target address. In the rare case that this address does not match the address predicted by the fetch unit, a misprediction occurs.

Speculative loads (LD.S) are easily implemented in hardware; exceptions set the NaT bit (see MPR 3/8/99, p. 16) and are propagated through subsequent calculations. In these calculations, the processor treats the NaT bit much like a 65th data bit. The check (CHK.S) instruction is simply a conditional branch based on the NaT value.

An advanced load (LD.A) inserts its address into the ALAT (advanced load address table). The DET stage checks subsequent store addresses against the ALAT and removes the entry if it finds a match. A later LD.C or CHK.A instruction checks the ALAT; if the entry is not found, the processor takes a microtrap and reexecutes the load, causing a several-cycle delay as the pipeline is flushed.

But in most cases, the check instruction will succeed without delay. To avoid overhead, Merced can execute check instructions in the same cycle as an instruction using the data; if the check fails, the "use" instruction is canceled.

Merced's ALAT has 32 entries and is indexed by the 7-bit register ID. It stores only a subset of the physical address tag, which reduces storage requirements but causes some false matches. The odds of a false match are low, and the penalty is simply reexecuting the load.

RISC and x86 designs can reorder loads and stores dynamically and check for conflicts, performing the equivalent of an advanced load. The ALAT provides a simpler hardware mechanism to handle this case. Some PowerPC and SPARC processors use hardware- or software-controlled prefetching instead of speculative loads to help cover load latency. IA-64's speculative loads have the advantage of bringing data all the way into the register file, avoiding load-use interlocks, and should better avoid prefetching of unneeded data. Merced implements these features with a minimal amount of complexity.

Designed for High Reliability

Because Merced is designed for use in highly reliable servers, it has several features to improve reliability, availability, and serviceability (RAS). Many processors offer parity protection for on-chip memory structures, but Merced goes one step further with full ECC on the on-chip L2 cache data, allowing the chip to correct single-bit errors and detect multiple-bit errors. Parity is sufficient for small structures, such as the TLBs and the instruction cache, that never contain modified data and can simply be reloaded if necessary.

The external L3 cache is also ECC-protected, including the tags. To reduce overhead, the state bits are not fully protected except for the M (modified) state, which ensures modified data will not be lost. Both the backside L3 bus and the frontside memory bus are covered by ECC.

Intel designed the frontside ECC algorithm to detect errors in four consecutive bits, which would be caused if a single ×4 DRAM chip fails. Instead of taking a fault on a frontside ECC error, the "poisoned" data is held in the processor until it is used. This helps identify the affected process, and in some cases, the data may never be needed.

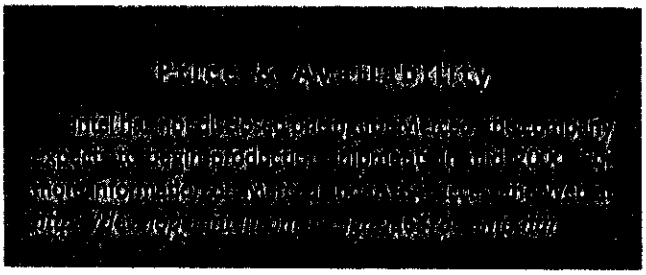
Using these mechanisms, a Merced system can tolerate many hardware errors. Errors corrected by ECC cause only a few cycles of delay and are transparent to software (although the processor optionally generates a low-priority interrupt for logging purposes). In many cases, a parity error can be corrected by a firmware routine that flushes and reloads the affected entry. If the failure has caused data loss, the processor notifies the operating system, which can terminate the affected process and restart it from a previous checkpoint. Only if the OS itself is affected will the entire system, or a single node in a large system, go offline.

Some Details Not Yet Disclosed

Sharangpani's presentation shed little new light on Merced's IA-32 portion. As disclosed earlier, IA-32 code and data share the same caches and execute in the same function units. When in IA-32 mode, the processor fetches x86 instructions into a separate decoding and scheduling unit that reorders the instructions and executes them using the native execution core. We expect the IA-32 decoding and scheduling unit to be similar to Pentium III's front end.

Intel has already fabricated Merced in its 0.18-micron P858 process. Considering the six-wide native CPU core, the complex IA-32 front end, and the on-chip L2 cache, we expect the die size to be large, around 300 mm^2 or so. Merced will use a thermally enhanced plastic package similar to Pentium III's 528-pin LGA. We expect it to require about 700 contacts, allowing for the wider (128-bit) backside cache bus and extra pins for power and ground to handle the high power dissipation, which we expect to exceed 60 W, including L3 cache. Intel will ship the CPU with up to 4M of L3 cache in a module (see MPR 3/8/99, p. 16).

The company remains mum on any numerical description of Merced's performance, or even its clock speed. Sources



indicate the chip is designed to achieve speeds of around 800 MHz, impressive by today's standards but likely to fall behind the speed of the 0.18-micron 21264 and even the 0.18-micron Pentium III in 2H00. Thus, to outperform these and other competing processors, Merced must execute more instructions per cycle. Its EPIC architecture should help in this regard, but it may not be enough.

Head to Head With 21264

The 21264 will be Merced's toughest competition on the performance front. Compaq expects that chip to exceed 1 GHz by mid-2000, giving it as much as a 25% frequency advantage over Merced. Except for its single branch unit, the Alpha chip has a comparable set of function units. The 21264 is limited to four instructions per cycle, but Merced's two extra two issue slots will probably provide little advantage on most applications. It remains to be seen how the 21264's aggressive hardware reordering matches up against the compiler-driven scheduling of Merced. We expect a close race on single-thread workstation benchmarks such as SPEC95.

Merced could have more of an advantage on large server applications. On these applications, the 21264 is hampered by its modest 128K of on-chip cache and a maximum of 4.0 GBytes/s of backside cache bandwidth. Although those numbers were impressive when the 21264 was announced three years ago, Merced will have more on-chip cache and as much as three times the backside-cache bandwidth. The 21264 tries to make up for this with 2.7 GBytes/s of frontside bandwidth, but Merced should come close to this number as well. Intel claims a four-processor Merced system will outperform a four-way server with 1.1-GHz 21264 processors on transaction-processing benchmarks.

Other competing processors are likely to fall well behind Merced's performance. Before Merced ships, Sun should be shipping UltraSparc-3 systems at 600 MHz or so, but even that new processor is a simple in-order RISC chip that won't match Merced's workstation benchmarks. Sun currently holds a system-level performance advantage with its 64-way SPARC servers, but SGI, IBM, and others expect to take advantage of Merced's improved scalability and ship systems as large or larger.

At clock speeds of 550 MHz or below, HP's PA-8600, the MIPS R14000, and IBM's Power3 are simply too slow to challenge Merced's performance on most applications. Both AMD's Athlon and Intel's own Foster may approach Merced's scores in small commercial servers, but they lack

the 64-bit addressing, RAS features, and scalability of the IA-64 chip.

Merced provides another key advantage over its RISC competitors: x86 compatibility. Current Xeon users can gradually move their applications to IA-64 and take advantage of its performance rather than making a hard switch to a new instruction set. Merced is the first processor to combine leading-edge native performance with hardware x86 compatibility, an impressive engineering achievement that no company is better suited to deliver than Intel.

Merced Shows EPIC Advantages

These disclosures show for the first time how IA-64 fundamentally alters the balance between the processor and the compiler. Trapped within architectural walls built more than a decade ago, other vendors must cram more and more complexity into their processors to increase performance. Most of this complexity is not for calculating results, but rather to analyze instruction flows and to perform the bookkeeping required to translate from the original instruction order to a new order and back again.

As the Merced design shows, IA-64 results in simpler hardware for issuing instructions and renaming registers, despite the wide issue width of the machine. But the IA-64 chip also includes dynamic hardware where necessary: for example, for branch prediction and for avoiding load stalls.

In theory, IA-64 designs should deliver better performance with a smaller die size than competing RISC and CISC parts. The inclusion of IA-32 compatibility, however, may nullify much of the potential die-size advantage, at least in the first few IA-64 chips. Evaluation of any cost and performance claims awaits more information from Intel, but we expect the Merced design to provide a per-clock performance advantage over both Pentium III and the 21264. If Merced falls short, it is likely to be in the area of clock speed.

Clock speed should not be a problem for McKinley, the Merced follow-on that Intel says will achieve speeds in excess of 1 GHz in the same 0.18-micron process as Merced. McKinley, due in late 2001, is likely to give Intel a sizable performance lead on most CPU benchmarks, while Merced looks like it will merely match the performance of the industry leaders.

Even matching the leaders, however, will be a big step forward for Intel in the workstation and server markets. The combination of strong native performance and full x86 compatibility has won the backing of virtually every significant workstation and server vendor except Sun. We expect these vendors to roll out a variety of Merced systems, starting in 2H00.

Processor vendors such as Compaq and IBM (see page 11) still have a few tricks up their sleeves, and they aren't giving up in the performance race. To slow the IA-64 juggernaut, these vendors can't just be as good as Intel—they have to be better. That could be tough. Merced looks to be a solid starting point for IA-64, and Intel will keep raising the bar from there. M

Most Significant Bits

Continued from page 5

■ K8 Loses Architect

AMD's next-generation K8 project has lost a key player. Following the recent departure of president and COO Atiq Raza (see MPR 8/2/99, p. 4), Jim Keller, coarchitect of the K8, has left the company. Keller, who was previously the architect of the Alpha 21264, was hired away from Digital by Raza specifically for the K8 design.

Keller's departure is especially disquieting for AMD coming shortly after the loss of AMD Fellow and K6 architect Greg Favor to startup Siara. Although Favor wasn't working directly on the K8, AMD suddenly finds itself short two top architects. AMD Fellow and IC-process expert Don Drapper has also reportedly resigned.

Although the loses will have an impact on the K8 project, AMD says it will be minor. According to AMD, the K8 definition is already complete, and the remainder of the team remains intact. Fred Weber, coarchitect of the K8 along with Keller and Dirk Meyer, has assumed responsibility for the K8 project. Keller's high-speed logic-design experience will be sorely missed, but luckily AMD still has Meyer, chief architect of Athlon, and previously coarchitect of the 21264 with Keller at Digital, to fill that gap. AMD insists that the K8 schedule will not be affected and the chip remains on track for 2001 delivery.

The reasons for Keller's departure are not known, but it may have been precipitated by the departure of Raza, who was highly respected by the engineers and was considered by most to be AMD's best hope for success against Intel. Assuming that AMD can prevent further defections, Keller's loss may not be too serious. Athlon is looking good and still has the potential to return AMD to profitability in the near term. Meyer's design team in Austin remains solid, and if he can coax enough out of Athlon to stay within marketing distance of Willamette, he could buy enough time to make up for any delay Keller's departure may cause. —*K.D.*

■ Clarification

The article "Mercy, Mercy, Merced," which appeared in the previous issue of *Microprocessor Report*, should have been identified as a Guest Viewpoint. From time to time we publish viewpoint articles as a way of exposing our readers to a variety of opinions and to provoke debate on timely issues. Guest viewpoints, however, do not necessarily represent the opinions of MicroDesign Resources analysts or the editors of *Microprocessor Report*. —*Editor* M