■ MDR ANALYSIS

Design Concepts for Merced

Forecasting the Inner Workings of the Decade's Most Anticipated Processor

by Linley Gwennap

A few issues ago (see **1017VP.PDF***), we delved into the software architecture of Merced. This piece looks at how that architecture might be implemented in hardware and what performance might result.*

Let me start by saying we know nothing about Merced except the few stale tidbits Intel and HP have publicly disclosed *(see* **080801.PDF***)*. This article is completely untainted by any proprietary knowledge about the processor because, frankly, neither Intel nor HP has chosen to divulge any, under nondisclosure agreements or otherwise. The few hapless design engineers we've managed to capture have perished on the rack without revealing any secrets.

On the other hand, the flavor of this top-secret project has emerged from conversations with various sources. More important, even Intel and HP must work within the laws of physics and economics, in this case the inexorable trends associated with semiconductor fabrication. Combining all this information leads to a deeper understanding of what Merced might be, and that will have to suffice until the companies choose to disclose more.

Process Details Already Revealed

It's almost certain Merced will begin life in Intel's 0.25-micron P856 process *(see* **101203.PDF***)*. We believe the chip is currently scheduled to reach production in 1H99, and Intel's next-generation 0.18-micron process won't be ready until late 1999 or early 2000, so P856 will be Intel's leading-edge process at Merced's debut. By 1H99, Intel will have tweaked the transistors in P856 to optimize their speed, but the vendor typically keeps the metal layers constant within a process generation, fixing the die size.

Intel has already revealed many details about P856, which will enter production later this year with a version of the P55C design (code-named Tillamook), followed by a P6 processor, Deschutes. The most relevant information for system designers is the core operating voltage, a tiny 1.8 V. We expect that, like the P6, Merced will use GTL+ or some other low-voltage signaling method for its system interface.

We project the die size of the initial Merced to be about 300 mm^2, about the same as the first P6. For the first twenty years of the microprocessor era, die sizes rose 15% per year, driven mainly by improvements in optical technology. Since then, this growth has slowed dramatically. Although optical technology continues to improve, the defect densities of current manufacturing processes cause poor yield with very large chips. Unfortunately, defect rates are improving slowly at best.

Intel has another good reason to avoid chips much larger than 300 mm^2. For Merced to have the option of following the P6 family into high-volume production, it must reach the sweet spot of about 100 mm^2 in two process shrinks. Intel aims for a 40% area reduction from each process shrink, so chips that start much beyond 300 mm^2 would not reach the sweet spot, increasing Intel's overall manufacturing costs and decreasing margins.

Eight-Way Processor Core

The workings of the Merced processor core are more difficult to discern. Our best guess is that the processor will be able to dispatch eight instructions at once. Fewer would be too like today's high-end processors, most of which can dispatch four instructions per cycle. More than eight would be difficult for the compiler to schedule in a useful fashion.

Merced has long been rumored to have a VLIW core; building an eight-way superscalar RISC processor using traditional out-of-order techniques would be monstrously complex. While seeking the advantages of VLIW, Merced is unlikely to repeat the mistakes of the original VLIW processors from Multiflow and Cydrome *(see* **080205.PDF***)*—in part because some of the key designers of these pioneering efforts participated in the specification of IA-64, the Merced instruction set.

One criticism of VLIW is code expansion: an eight-way VLIW processor might have a 256-bit instruction word, causing severe code bloat if the compiler can't fill all eight slots with useful instructions. Several recent VLIW designs aim to solve this problem by allowing more flexible instruction partitioning. For example, TI's 'C6201 DSP *(see* **110204.PDF***)* uses an 8-bit encoding scheme to identify groups of instructions that can be executed simultaneously without dependencies; each group can be as small as one instruction or as large as eight instructions. Merced is likely to use a similar scheme to allow concise encoding for programs with limited instruction-level parallelism.

Another problem with classic VLIW processors was a complete lack of hardware interlocks, forcing the compiler to plan for worst-case instruction timing. Newer VLIW designs have shown that register scoreboarding and other techniques can be implemented in hardware to simplify the compiler's task. Although this extra hardware adds overhead compared with a "pure" VLIW design, it is far less than the overhead of an out-of-order RISC or CISC processor.

A final concern regarding classic VLIW designs is scalability; in these designs, programs must be recompiled for each new processor generation. With a more modern approach, the hardware can hide issues of instruction latency and even the number of function units through simple scheduling hardware. The compiler's main task is to create groups of nondependent instructions and communicate this grouping information to the hardware. If the compiler creates a particularly large group, a high-end implementation might execute that group in a single cycle while a low-end processor might take two or more cycles; the same binary, however, would run on both processors.

In addition to this flexible instruction grouping, IA-64 is likely to include several other features beyond the capabilities of current RISC processors. These could include prefetch instructions, predicated execution, and a register file with at least 128 registers *(see* **101003.PDF***)*. Unless the number of registers grows tremendously, none of these changes is likely to significantly increase the size of the VLIW core.

x86 Compatibility Adds Transistors

An eight-way VLIW core could be relatively small. Consider that both Digital's 21164 and Sun's UltraSparc have about two million logic transistors. Both are four-way superscalar 64-bit RISC processors with normal pipeline interlocks but little or no out-of-order overhead. Based on these two data points, an eight-way VLIW processor might have about four million transistors, even with a larger register file and a fairly robust set of interlocks and scheduling hardware. A more stripped-down VLIW design would have fewer transistors.

For Merced, we must consider the issue of x86 compatibility. There are at least two possible approaches. To minimize the impact on the transistor count and potentially on cycle time, Merced could implement two levels of compatibility. For executing BIOS and similar code, the chip could include a scalar x86 integer core (e.g., a 486). This unit, running at up to 200 MHz, would provide more than adequate performance for low-level code while consuming about 500,000 transistors.

To deliver competitive performance on x86 applications, the chip could use an emulation/translation mechanism similar to Digital's FX!32 *(see* **100302.PDF***)*. To improve efficiency, the function units in the VLIW core might have some special condition-code logic, and the chip might have an x86 segmentation unit. This extra logic, including the 486 core, should add less than a million transistors to the chip and have minimal impact on cycle time.

A second approach would be to add an x86-to-IA-64 translation engine to the chip, making Merced look much like a P6 with the native instruction set exposed. This design would eliminate the messiness of software emulation and translation. The x86 front end would be very complex, however, because in addition to translating instructions to IA-64 encodings, it must also do dependency checking and instruction reordering to achieve reasonable performance.

Adding an x86 front end of similar complexity to the P6's would cost about two million transistors. A higher-performance front end that takes better advantage of the eight-way core might require three million. Furthermore, if not handled carefully, executing all x86 features in hardware could increase the cycle time, slowing performance in native mode as well.

This decision boils down to whether the x86 translation should be done in hardware or by software. The hardware method adds more transistors, could impact native performance, and probably delivers worse x86 performance. Software translation, however, requires changes to all target operating systems. Since only Unix and Windows NT are likely to support Merced in the foreseeable future, OS support for emulation may not be a problem.

Lots of On-Chip Cache Likely

So let's say the Merced processor core contains about five million transistors: four million for the native core and another million for x86 support. Based on the projected transistor density of P856, this core would consume roughly 100 mm^2. Allowing 15% of the 300-mm^2 die for the pad ring, this would leave half the die for cache memory. Given the SRAM density of P856, roughly 512K of cache would fit into this area of the die. This amount of cache is not unreasonable: most high-end processors designed for 0.25-micron processes, such as UltraSparc-3 and the MIPS H1, will probably have at least 256K of on-chip cache, and the PA-8500 will have 1.5M *(see* **1103MSB.PDF***)*.

A 512K on-chip cache would be a big step up from the paltry 32K found on the Klamath chip. Intel has historically gotten away with smaller cache sizes than RISC vendors, in part because x86 programs have better code density and use smaller data structures than RISC programs. But this advantage will be nullified, and probably reversed, for IA-64. Even with the variable instruction-grouping mechanism mentioned above, IA-64 binaries are likely to be significantly bigger than RISC binaries. In addition, Merced is likely to operate at much higher clock speeds than current Intel processors, requiring large on-chip caches to compensate for the relatively slow external memory.

Even with plenty of on-chip cache, Merced will require a high-performance system interface to achieve optimal performance. As Merced is likely to be used in PCs as well as expensive workstations, the chip must also allow for lower-cost system components. We expect Merced to resemble the 21264 *(see* **101402.PDF***)* in its system interface, in particular, a 128-bit cache interface that operates at speeds ranging from roughly 133 MHz to 300 MHz as well as a 64-bit system bus that operates across a similar speed range. Intel is likely to sell Merced only on a daughtercard that contains the high-speed cache interface.

The 21264 package has 588 pins. Assuming Merced fits in a 600-pin PBGA, the MDR Cost Model estimates the manufacturing cost of the 300-mm^2 chip to be only $130.

Stellar Performance Projected

Estimating Merced's clock speed is difficult, since it is very dependent on the type of pipeline used. As a lower bound, Intel's P6-family processors should reach 400 MHz using the same process as Merced. Removing the complexities of the x86 architecture should allow Merced to run significantly faster. As an upper bound, Digital's Alpha processors should reach 900 MHz in a comparable 0.25-micron process.

To achieve such high clock speeds, Digital uses some unique circuit and process designs that increase cost and limit volume production; Intel is unlikely to accept such tradeoffs. Picking a number within the range, let's guess that Merced will debut at 600 MHz.

The per-cycle SPECmark performance of a VLIW core will not be that different from the performance of a comparable RISC core, because in both cases instruction scheduling is almost entirely in the hands of the compiler. HP's PA-8000, for example, achieves 0.06 SPECint95 (base) per MHz on a four-way superscalar core. Merced should have no problems matching this throughput, which at 600 MHz would equate to 35 SPECint95. Assuming the eight-way core provides a 20–30% advantage in throughput pushes performance into the 40–45 SPECint95 range.

Floating-point performance depends heavily on both the number of transistors devoted to floating-point units and on the bandwidth of the external system interface. Given HP's participation in the project and the PC market's increasing focus on 3D performance, Merced is likely to place a strong emphasis on floating point. The PA-8000 achieves 0.10 SPECfp95 (base) per MHz, and floating-point code could gain as much as 50% from the extra four dispatch slots in Merced. This gain would put Merced's performance at 80–90 SPECfp95.

Assuming the chip uses some sort of emulation for x86 applications, the question is the efficiency of the emulator. Digital claims 50–70% efficiency for FX!32 running on a standard RISC core. With some emulation hooks in the Merced core, the Intel chip should easily match this efficiency. Based on the native performance estimates above, this efficiency would give Merced x86 performance similar to or better than that of a 400-MHz Willamette, the fastest x86 chip expected in the Merced time frame.

The killer is what comes next: Merced's performance could rise by 50% or more within a year of the initial launch. By 1H00, a shrink to 0.18-micron CMOS could push clock speeds to nearly 1 GHz, with performance exceeding 60 SPECint95 and 120 SPECfp95. The shrink will also trim the die size to less than 200 mm^2, bringing the manufacturing cost below $100 and positioning the chip for volume deployment.

If well executed, Merced could be both the fastest x86 processor and the fastest native-mode processor available when it debuts. Digital's 21264 in a 0.25-micron shrink will be the closest challenger in the latter category. We can now only wait impatiently to discover whether Merced turns out as expected. MR