Microprocessor Floating Point: Past, Present, and Future

After Three Generations, Microprocessor Floating-Point is Maturing

By Allen Samuels, Weitek Corp.

Last issue, the first article of this series explained the IEEE floating-point standard, which serves as the basis for nearly all microprocessor floating-point units. This article gives an overview of the four generations of microprocessor FPU implementations; future articles will delve into more implementation details.

The dramatic improvement in silicon processing technology has allowed microprocessor floating-point units (FPUs) to evolve greatly since their introduction in the late '70s. Advances in gate speed and architecture have allowed over a hundred-fold increase in performance during the 1980s.

How do you measure the floating-point (FP) performance of a microprocessor system? The computation unit's performance is relatively easy to characterize (see below), but this is not the entire story. The computation unit must be fed constantly with data to crunch: address, load, store and control operations frequently outnumber floating-point operations by two or three to one. (Actually, this is also true for many integer programs if you go to the trouble of separating pure computation from data movement and control operations. The commonly quoted MFLOPS excludes data and control management operations, however, whereas the commonly quoted MIPS includes them.) Also, FP programs are notorious cache-busters, placing high demands on the memory system. The moral of the story: a good FP unit needs a good system to support it.

Two numbers measure floating-point performance: *latency* and *bandwidth*. Latency is the length of time before you can use the output (result) of an operation as an input to a subsequent operation. Bandwidth is a measure of the time between operation starts (or, inversely, the rate at which operations can be started), and it takes into account the degree to which operations can be overlapped. For example, a moderately fast FPU might have an add latency (i.e., the time from when the operation starts to when the result is available) of 4 clock cycles, but the issue rate could be every two cycles. If one instruction depends on the result of the previous instruction, then performance is limited by the latency. If not, it is limited by the bandwidth.

Latency is not solely a property of the first instruction; it can be affected by the second instruction (the one waiting for the results of the first) as well. Therefore, to properly document latency, a matrix of first and second instructions is required. One example is the i860 floating-point unit, where the latency of a floating-point add operation is three cycles unless it is used as the "src1" input to a floating-point multiply, in which case the latency is four. (Since floating-point multiplication is commutative, and this restriction does not apply to the "src2" input, simply reversing the operands in the second instruction recovers the extra cycle.)

Bandwidth can also be highly instruction-dependent. A look at the rules for overlapping operations for the R4000 series shows very detailed cycle counts and pipeline diagrams. Getting maximum performance requires sophisticated software scheduling algorithms that keep track of the states of the CPU's functional units.

High bandwidth is most useful for problems that can be vectorized; i.e., each element can be computed independently. In general, higher bandwidth (smaller numbers of cycles between instructions) is good, provided that it doesn't cause an increase in latency. Signal processing is an example of an application where bandwidth is often more important than latency, since there is a continuous data stream. Bandwidth beyond the system's ability to keep the pipeline fed is wasted.

Microprocessor FPUs for desktop computers have gone through three distinct generations up to now: Paleolithic, Neolithic, and modern, as the following sections describe.

Paleolithic (8087, 80287, 80387, 68881, 32081)

The limited silicon technology of the late '70s and early '80s, combined with the smaller market potential, forced the first generation of microprocessor FPUs to look very much like their associated integer units (IUs): single chips with a small data path and gobs of microcode. Since pins on the CPU were at a premium, instructions and data were fetched by the CPU and retransmitted to the FPU over the main system bus via a complicated and slow protocol. Once received, instructions were decoded and executed using a small (typically 16-bit-wide), heavily microcoded data path. This lead to instruction latencies in the 10-100 cycle range in an era when a 32-bit integer add required as few as two

cycles. Due to the limited resources, no instruction overlap was allowed and, therefore, bandwidths were usually the same as latencies.

The dominance of the Whetstone benchmark is perhaps the clearest example of how a benchmark affects design decisions. Since Whetstone emphasizes trigonometric and transcendental functions, many FPU designers opted to devote large amounts of microcode space to these relatively unimportant operations—to the detriment of the performance of more critical add and multiply instructions.

As silicon technology advanced, CISC designers revised their CPUs but tended to continue with first-generation type FPUs. This was due to the poor instruction protocols, which could add as much as a dozen cycles before a computation could be initiated, diluting floating-point core performance. This is borne out by the Cyrix FPU for the 80386. It has a computational core that is as much as five times faster on add and multiply instructions than the core used with the Intel 80387. However, when used in the 80387 compatibility mode, it is able to deliver only a modest 20-40% improvement on the majority of benchmarks. (Interestingly, the performance delta of the Cyrix part is greatest—almost 2×—for the Whetstone benchmark, primarily due to the 7-10× boost in speed for the transcendental and trigonometric instructions.)

Cyrix offers a faster memory-mapped interface as an option, following the approach Weitek adopted for its add-on coprocessor chips. While this results in systems with substantially higher performance, the burden of incompatible software has retarded widespread acceptance.

Neolithic (Early RISC: Fujitsu/Cypress SPARC, R2000/3000, RS/6000, PA-RISC)

The second generation of microprocessor FPUs came with the first generation of RISC processors. The early RISCs were all multiple-chip implementations with off-chip primary caches. Most designers chose to put the FPU and private register file on a separate chip, virtually eliminating any protocol overhead by dedicating additional pins to the CPU/FPU interface. (One notable exception is Motorola's 88000, which was one of the first microprocessors to implement an on-chip floating-point unit. Intergraph's Clipper is another processor that had an on-chip FPU early in the game, but the latest implementation has a separate FP chip.)

Other architectural changes in this generation of microprocessors, such as instruction and data alignment restrictions, allow the FPU unit to communicate directly with memory for loads, stores and instruction fetches, eliminating a costly passage through the IU as required by many first-generation FPUs. (Ironically, the original 8087 FPU used exactly this type of arrangement. Its byte-wide bus and lack of virtual addressing allowed the 8087 to track the CPU instruction pipeline and to perform loads and stores directly to memory. The 286/287 and 386/387, with wider buses and sophisticated memory management, forced its abandonment.)

For this microprocessor generation, system cycle times were constrained by the use of an off-chip primary cache. At the same time, transistor budgets increased, and workstation users had the desire for fast floating-point and the willingness to pay for it. The result was some of the lowest FPU latencies (in clock cycles) ever seen in computer systems; typical systems of this generation have latencies of three or four cycles. The most extreme example is the RS/6000, with a two-cycle latency; even more startling, this is for a fused multiply-add, i.e., two cycles for two operations.

It was near the end of this generation that the dominant workstation suppliers formed SPEC to create a new series of system benchmarks. Naturally, they hoped to show that their chips were way ahead of a newly awakened and aggressive Intel. This was done by heavily weighting the benchmark suite toward the long suit of the current RISC machines: floating-point performance. In other words, the tail now wagged the dog. Comparisons of just the integer portions (SPECint) show a much smaller differential between the RISCs and the 486. (This situation is probably what has forced Intel to divulge that the P5 will have much higher floating-point performance.)

Modern (i486, 68040, 88110, R4000, SuperSPARC, Alpha)

As the march of technology continues, million-plus transistor budgets have become all the rage. Most processors are becoming superscalar or superpipelined to achieve higher performance levels. Both techniques tend to push designers toward single-chip implementations. IC packaging is the culprit here. Superpipelined systems, with their fast clock speeds, cannot afford the inter-chip delays associated with standard IC packaging. Superscalar systems have wide interfaces to instruction and data memories that require several hundred pins, boosting costs for systems using processors without on-chip caches.

Unlike the Neolithic CPUs, modern CPUs tend to have high clock rates, typically made possible by their on-chip caches, making it more difficult to pack all the operations required for a floating-point calculation into a small number of pipeline stages. At the same time, the

FPUs must now share the same die with the CPU, so there is pressure to keep the transistor count from increasing greatly. The net result is longer latencies, and, frequently, lower bandwidth. Often, due to decreased basic cycle times, actual FPU latencies (measured in ns) have held roughly constant, even though latencies in clock cycles have increased. You can see these effects when comparing the newly announced processors to their ancestors. (At first glance, SuperSPARC appears to be the exception, but this is illusory. Since it can execute two *dependent* integer operations in one cycle, at 40 MHz it can be considered as an 80-MHz non-superscalar integer unit with a six-cycle latency and two-cycle bandwidth FPU—a modest boost over the previous four-cycle latency of the SPARCstation 2 at 40-MHz.

The limited transistor budget—and the recognition of Whetstone's misplaced emphasis—probably caused Motorola to drop some of the transcendental and trigonometric instructions from the native instruction set of the 68040. (They are emulated via software traps to maintain software compatibility.) The extra transistors were used to speed up the basic floating-point performance, which is quite respectable. Intel couldn't do that in the i486 due to the primitive state of operating systems for the PC, resulting in a slow on-chip FPU with a lot of microcode.

For a recent example of how a limited transistor budget affects the FPU, compare SuperSPARC and Alpha. Both are implemented in 0.8-micron triple-metal processes with die sizes near the limit of the reticle. (SuperSPARC is BiCMOS, and Alpha has a 0.5-micron effective gate length, but both processes provide about the same number of potential transistor sites.) The SuperSPARC designers clearly were concerned about using inexpensive secondary cache technology, so they devoted large amounts of chip area to the 36-Kbyte primary cache, leaving enough space for only a 75-ns latency FPU (three cycles at 40 MHz). The Alpha designers rely on a high-speed secondary cache with a smaller primary cache. This allows space for a much faster 33-ns latency FPU (5 cycles at 150 MHz).

The Crystal Ball

The battle of the benchmarks will continue. The RISC workstation vendors are moving to single-chip superscalar and superpipelined CPUs with their limited FPUs, while Intel is talking about a P5 (586) with much better FPU performance than the 486.

As device geometries decrease, more and more transistors will allow FPU latencies to continue to decrease relative to integer latencies. Larger transistor budgets will permit latencies to decrease faster than process-driven gate speeds. Soon the law of decreasing returns will set in; Alpha is getting close to the limit for add and multiply, even though its divide performance could be considerably improved. Within the next few years, every desktop microprocessor will have an FPU with a 3 to 5 cycle latency for add and multiply, with radically shorter divide and square root times to be expected thereafter.

In the distant future (5+ years), when the silicon and compiler technologies have advanced to the point where high-performance machines issue three or four instructions every cycle and transistor budgets are dramatically increased, you will see single-chip processors with very fast, dual FPUs. (This trend is visible today in multichip, application-specific processors).

Since their introduction, microprocessor floating-point units have seen tremendous evolution in architecture and implementation. The painfully slow separate coprocessor chips of the early days have been replaced by dedicated, high-performance, floating-point data paths integrated on to the CPU die. The push for high integration, with the CPU, FPU, and cache all on the same chip, has limited the growth in FPU transistor budgets, but future generations will see another spurt of FPU performance increases. ♦

Mwave

Continue from page 11

brary, and development tools. IBM could match the generality of AT&T's solution while beating AT&T's price and performance by releasing the DSP Manager and real-time executive on TI's TMS320C31. Instead, IBM has chosen a less-general, lower-precision DSP core to minimize cost and allow integration of serial ports for peripherals. However, thanks to the Mwave DSP's efficient DMA, many signal processing cycles will be freed from the task of handling real-time I/O.

Although real-time video processing is a key ingredient of multimedia computing, it can be expensive. It is unclear how the Mwave standard will handle any form of real-time video in the future or how it will work with the Intel DVI chips IBM has been using in its multimedia computers. Due to high costs, products based on processors (e.g. iWARP) with the kind of power needed for full-featured multimedia computing cannot be successfully marketed to the masses at present. However, Mwave's dramatic improvement over currently dismal multimedia standards for the PC, along with the potential to support future higher-precision DSPs, may result in Mwave forming a long-lasting foundation. ♦