VOLUME 13, NUMBER 3
MARCH 8, 1999

MICROPROCESSOR REPORT

THE INSIDERS' GUIDE TO MICROPROCESSOR HARDWARE

Pentium III = Pentium II + SSE

Internet SSE Architecture Boosts Multimedia Performance

by Keith Diefendorff

The name, Pentium III, belies the processor's modest changes. Extrapolating from the differences between Pentium and Pentium II—out-of-order execution, long pipeline, backside L2 cache, and a split-transaction bus—one might have expected Pentium III to be a completely new microprocessor. But the reality is less meaty; Katmai, the first member of the Pentium III family, is essentially a Pentium II with an enhanced multimedia capability. Consumers, however, may not be so disappointed: Katmai's new features will enable improvements in multimedia performance large enough to spawn new applications that are simply intractable on the current generation of Pentium II processors.

Katmai was announced at 450 and 500 MHz, an 11% frequency boost over the 450-MHz Deschutes processor, the current top-of-the-line Pentium II. A 550-MHz version will come out in the second quarter and later this year the second Pentium III, Coppermine, will surface. That processor will be implemented in Intel's upcoming 0.18-micron P858 process (see MPR 1/25/99, p. 22), boosting its frequency to 600 MHz—and eventually to 733 MHz or more—while also making space for at least 256K of on-chip L2 cache.

The new processor uses the same pipeline, same caches, same bus, and same IC process as Deschutes. So, not surprisingly, it achieves the same per-cycle performance on existing software. All of Katmai's new features, save one, are focused on multimedia, as Figure 1 shows. The new features, including 70 new instructions (alias KNI) and a new memory-streaming architecture, are officially dubbed the "Internet streaming-SIMD extensions" (Internet SSE).

The Internet prefix is probably just a ploy by Intel marketeers to catch the wave of Internet enthusiasm and transfer its momentum to their chip. But, while SSE may have nothing to do with the Internet per se, content providers are finding it useful for building soft delivery mechanisms, like soft ADSL modems, and for achieving high data-compression ratios with techniques like 3D NURBS (nonuniform rational B-splines), which are impractical on Pentium II.

The only new feature—or misfeature, depending on your view—in Katmai not related directly to multimedia is the processor serial number (see MPR 2/15/99, p. 3). Contrary to some reports, Katmai does not implement a random-number generator, which would have been a far more useful feature than the serial number.

Streaming SIMD Increases Multimedia Capability

The SSE architecture is likely to be more significant to multimedia performance than MMX ever was. SSE represents a major improvement over MMX for processing video, sound, speech, and 3D graphics. As we pointed out earlier (see MPR 10/5/98, p. 1), Intel did about as well as anyone could expect in defining the architecture, given the x86 starting point (a hole).

What is disappointing about Katmai, however, is that it implements only half of SSE's 128-bit architectural width, double-cycling the existing 64-bit data paths. This approach limits Katmai to only half of the architecture's potential performance, leaving it with little advantage over the K6 III's

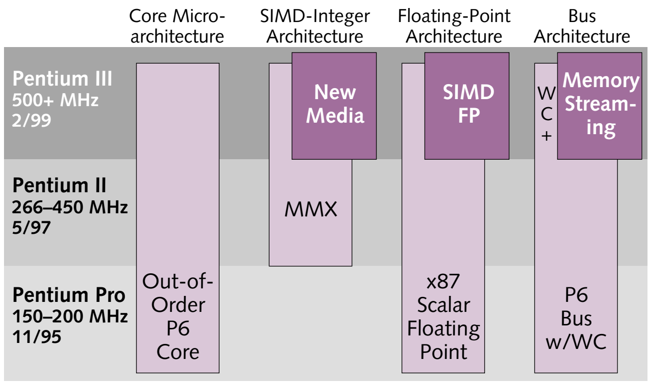
Core Micro-architecture
SIMD-Integer Architecture
Floating-Point Architecture
Bus Architecture
Pentium III 500+ MHz 2/99
Pentium II 266–450 MHz 5/97
Pentium Pro 150–200 MHz 11/95
Out-of-Order P6 Core
New Media
MMX
SIMD FP
x87 Scalar Floating Point
WC+
Memory Streaming
P6 Bus w/WC

Figure 1. The first Pentium III, Katmai, is based on the same core microarchitecture as Pentium Pro and Pentium II, adding SIMD floating point, new MMX instructions, and memory-streaming features, including prefetch and improved write combining (WC).

(see MPR 3/8/99, p. 22) implementation of 3DNow, which also delivers 4 FLOPS/cycle. Why Intel would sacrifice this much performance for a few measly mm^2 of silicon is a mystery.

The cost of SSE is indeed small. At 128 mm^2, Katmai is only 15 mm^2 larger than Deschutes. According to the MDR Cost Model, the additional silicon adds only 7% to the manufacturing cost of a Katmai module—bringing it to $75. But a full implementation of SSE might not have cost much more. The four-wide AltiVec SIMD-FP unit on Motorola's G4 (see MPR 11/16/98, p. 17), for example, would occupy only 8 mm^2 in Katmai's process; even if it were added on top of Katmai's 128 mm^2, it would add only $3 to the manufacturing cost.

Surprisingly, Katmai's 700,000 extra transistors do not add to its power consumption. To the contrary, improvements to the design actually reduced power dissipation by 7%, to 25 W (max at 450 MHz).

It Won't Fly Without Software

The new chip has the potential to significantly outperform Deschutes on everything multimedia. To realize that potential, however, Katmai must have software written expressly for its new features. Intel has seen to it that the obvious graphics APIs, such as Direct3D and OpenGL, are in place, but the most significant benefits will depend on applications being created or modified for the new features.

The last time Intel made a processor transition that depended on software (when it added MMX to Pentium), it failed to get a critical mass of applications ready in time, blunting the appeal of the new feature. Fortuitously, Pentium/MMX had larger L1 caches to power it through the transition. But such is not the case this time—if Intel wants its new chip to look good, it must get applications in place.

To this end, Intel has, over the past year, mounted a massive campaign—three times the size of any previous Intel software-enablement program—to assist software vendors. The $300 million launch effort was clearly successful, as more than 300 Pentium III–ready applications were shown at the preview event in February. Most of these applications used SSE to some extent, although some just use the serial number.

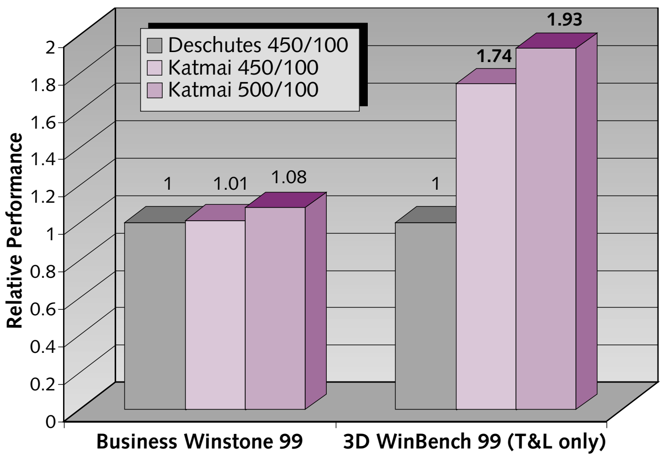

Figure 2. On the 3D WinBench 99 Transform and Lighting CPU Test (rendering nulled out), Intel says that Katmai is 74% faster than Deschutes (although we expect that the overall 3D WinBench speedup will probably be less than 30%). (Source: Intel)

Last summer, Intel decided to pull in the Katmai launch from June to February. The option to do so resulted from better-than-expected silicon, but the impetus was probably a desire to curb the growing software support for AMD's 3DNow and to preempt the announcement of the K6 III.

Although the schedule pull-in is good for consumers, it did foul Intel's 133-MHz-bus plans. Katmai was initially planned to sync up with the Camino chip set in June. But now, without Camino, Katmai is stuck with the BX chip set, limiting it to the same 100-MHz bus as Pentium II. The situation may be a blessing in disguise, as it allows Intel to introduce a 550-MHz Katmai rather than the 533-MHz part it would have been limited to with a 133-MHz bus. Sources now indicate that Camino will be delayed until September, presumably mating up with the 600-MHz Coppermine.

Seventy New Instructions

The SSE features in Katmai divide into two categories: new SIMD instructions and new memory-streaming features. The purpose of SIMD instructions is to increase the rate at which vectorizable data—the kind most common in multimedia applications—can be processed. SIMD instructions give the software a way to express the parallelism that is inherent in this type of data, so the hardware can easily process it in parallel.

SSE introduces new integer and new floating-point SIMD instructions. The integer instructions are really just extensions to the MMX instruction set. Intel, however, refers to these as "new-media instructions." The name makes them easier to explain to customers and avoids lending any credence to the notion that MMX had shortcomings, such as the lack of video-compression capability that we pointed out earlier (see MPR 3/5/96, p. 1).

The new-media instructions, listed in Table 1, will accelerate important multimedia tasks that were poorly served by MMX. For example, the PMAX and PMIN instructions, which are important to the Viterbi-search algorithm used in speech recognition, were notably absent in MMX. Average (PAVG) instructions were added to accelerate video decoding, and Sum of Absolute Differences (PSADBW) was added to speed motion-search in video encoding.

To save silicon, Intel used a clever trick to implement PSADBW. The instruction is issued as three µops: the first computes the differences (A_i–B_i) and carry-outs (C_i) of each of the byte elements in the two source operands; the second computes the absolute values ($|A_i–B_i|$) of the intermediate results; and the third sums the eight absolute values ($\Sigma_{i=0...7}|A_i–B_i|$). The trick was to use the Wallace tree in the SIMD multiplier to perform the final summation. With this approach, PSADBW added only 2% to the area of the SIMD integer unit.

Mnemonic	Description	SIMD	Scalar	Integer	FP	Port	Latency	Thruput
Floating-Point Instructions (use SIMD-FP registers)								
ADD	Add elements ($D_i = D_i + S_i$, i = 0…3)	•	•		•	1	4	2
SUB	Subtract elements ($D_i = D_i - S_i$)	•	•		•	1	4	2
MUL	Multiply elements ($D_i = D_i \times S_i$)	•	•		•	0	5	2
DIV	Divide elements ($D_i = D_i \div S_i$)	•	•		•	0	18–36	
SQRT	Square root of elements ($D_i = f(S_i)$)	•	•		•	0	29–58	
MAX	Maximum of elements ($D_i = f(S_i)$)	•	•		•	1	4	2
MIN	Minimum of elements ($D_i = f(S_i)$)	•	•		•	1	4	2
RSQRT	Reciprocal square-root estimate (12-bit accuracy) ($D_i = f(S_i)$)	•	•		•	1	2	2
RCP	Reciprocal estimate (12-bit accuracy) ($D_i = f(S_i)$)	•	•		•	1	2	2
CMP	Compare elements using eq, lt, le, unord, neq, nlt, or nle returning Boolean mask	•	•		•	1	4	2
COMISS/ UCOMISS	Ordered/unordered compare scalar element setting condition flags		•		•	1	1	1
ANDPS	Bitwise logical AND of elements	•			•	1	2	2
ANDNPS	Bitwise logical AND of elements w/complement of one operand	•			•	1	2	2
ORPS	Bitwise logical OR of elements	•			•	1	2	2
XORPS	Bitwise logical XOR of elements	•			•	1	2	2
New-Media Instructions (use MMX registers)								
PINSRW	Insert 16-bit value from general register into one of four elements (specified by immediate)	•		•		0,1	4	1
PEXTRW	Extract one of four elements (specified by immediate) to a general register	•		•		0,1	2	2
PMULHU	Multiply four 16-bit unsigned elements returning most significant 16 bits of each	•		•		1	3	1
PSHUFW	Full shuffle of 16-bit elements under control of 8-bit immediate mask	•		•		1	1	1
PMOVMSB	Move 8-bit mask composed of MSbs of byte elements to a general register	•		•		1	1	1
PAVGB	Average of byte elements ($D_i = -(D_i + S_i + 1)/2$, i = 0…7)	•		•		0,1	1	0.5
PAVGW	Average of 16-bit elements ($D_i = -(D_i + S_i + 1)/2$, i = 0…3)	•		•		0,1	1	0.5
PSADBW	Sum of absolute value of differences of 16-bit elements ($D_{1,0} = \Sigma_{i=0…3} \lvert D_i - S_i \rvert$)	•		•		0,1 *	5	2
PMINSW	Minimum of signed 16-bit elements	•		•		0,1	1	0.5
PMINUB	Minimum of unsigned byte elements	•		•		0,1	1	0.5
PMAXSW	Maximum of signed 16-bit elements	•		•		0,1	1	0.5
PMAXUB	Maximum of unsigned byte elements	•		•		0,1	1	0.5

Mnemonic	Description	SIMD	Scalar	Integer	FP	Port	Latency	Thruput
Type-Conversion Instructions (use MMX and SIMD-FP registers)								
CVTSS2SI	Convert FP scalar to integer in MMX register		•	•	•	1,2	3	1
CVTTSS2SI	Convert FP scalar to integer in MMX register		•	•	•	1,2	3	1
CVTSI2SS	Convert integer in MMX register to FP scalar		•	•	•	1,2	4	2
CVTPS2PI	Convert two FP elements to integers in MMX register	•		•	•	1	3	1
CVTTPS2PI	Convert two FP elements to integers in MMX register	•		•	•	1	3	1
CVTPI2PS	Convert two integers in MMX register to FP scalars	•		•	•	1	3	1
Data-Movement Instructions (use SIMD-FP registers)								
MOVA	Load/store/move an aligned 128-bit vector or 32-bit scalar (fault on misaligned)	•	•		•	2 3/4 0,1	4 1 4	2 1 2
MOVUPS	Load/store an unaligned 128-bit vector from/to memory	•			•	2 3/4	4 5	2 3
MOVLPS/ MOVHPS	Load/store 64-bit mem operand to/from low/high half of register	•			•	2 3/4	3	1
MOVLHPS/ MOVHLPS	Move lower/upper 64 bits of src reg into upper/lower 64 bits of dest reg	•			•	0,1	1	1
MOVMSKPS	Move 4-bit mask composed of MSbs of four elements to a general register	•			•	0	1	1
SHUFPS	Move any two elements from two source registers to dest register under control of 8-bit mask	•			•	1	2	2
UNPCKHPS	Interleave high two elements from two source regs to dest register	•			•	1	3	2
UNPCKLPS	Interleave low two elements from two source regs to dest register	•			•	1	3	2
State Save/Restore Instructions (use SIMD-FP registers)								
FXSAVE	Save registers to memory	•			•		µcode	
FXRSTORE	Load registers from memory	•			•		µcode	
STMXCSR	Save the SIMD-FP status and control register to memory		•	•			µcode	
LDMXCSR	Load the SIMD-FP status and control register from memory		•	•			µcode	
Streaming/Prefetching Instructions								
MOVMSKQ	Store MMX register to memory under byte mask	•		•		0,1 3/4	4	1
MOVNTQ	Store MMX register to aligned mem minimizing cache pollution	•		•		3/4	3	1
MOVNTPS	Store SIMD-FP register to aligned memory minimizing cache pollution	•			•	3/4	4	2
PREFETCH	Load cache line containing addressed datum into specified levels of the cache hierarchy	•	•	•	•	2	2	1
SFENCE	Ensure all prior stores are globally visible (flush WC buffers)					3/4	3	1

Table 1. The SSE architecture introduces new SIMD floating-point instructions, new SIMD integer instructions, and new memory-streaming instructions. The SIMD floating-point instructions operate on four-element vectors of IEEE-754 single-precision values in a new 128-bit eight-entry register file. The SIMD-FP instructions also have a scalar mode that operates on only the rightmost element of vectors. Intel calls the SIMD integer instructions new-media instructions. SIMD-FP multiplies are dispatched to port 0, allowing them to be issued in parallel with SIMD-FP adds, which are dispatched to port 1. Most of the new-media instructions can be dispatched to either port 0 or port 1 (shown as 0,1). *PSADBW requires three µops; the first two can execute on either port 0 or 1, while the third requires port 0. Load instructions use port 2, while store instructions require both ports 3 and 4 (shown as 3/4). Instruction latencies are shown in cycles and throughputs in cycles per instruction. In scalar mode, floating-point instruction latencies are one cycle shorter than shown. (Source: Intel)

SIMD-FP: the Main Attraction

The most significant new feature of Katmai is SIMD floating point. Initially, the feature will be used mainly to accelerate 3D graphics, as Figure 2 shows. But Intel expects Katmai's high SIMD-FP performance to spur the use of floating-point data in many other signal-processing algorithms as well, making the feature broadly applicable.

Architecturally, the SIMD-FP feature introduces a new register file containing eight 128-bit registers, each capable of holding a vector of four IEEE single-precision floating-point data elements. The SIMD-FP instructions perform arithmetic on the respective elements in two registers, returning a four-element result vector to one of the two registers. Thus, the architecture allows four single-precision floating-point operations to be carried out with a single instruction and, in a full implementation, to achieve a throughput of four multiply or four add operations per cycle.

To avoid completely redesigning the core microarchitecture, however, Intel had to shoehorn the new SIMD-FP architecture into the Deschutes core. Since Katmai is built in the same 0.25-micron process as Deschutes, it also had to implement the feature using as little silicon as possible. To achieve these goals, Intel implemented the 128-bit architecture by double-cycling the existing 64-bit data paths and by merging the SIMD-FP multiplier with the x87 scalar floating-point multiplier into a single unit, as Figure 3 shows.

To utilize the existing 64-bit data paths, Katmai issues each SIMD-FP instruction as two µops. To avoid the possibility of leaving the machine in an imprecise architectural state between µops, Intel devised a check-next-µop mechanism to hold off the register update from the first µop until it is determined that the second µop will complete without generating an exception (e.g., overflow). The check-next-µop feature has no effect on the execution pipeline but can block retirement for an extra cycle. This potential penalty is avoided when exceptions are disabled, which is the normal mode.

To partially compensate for implementing only half of SSE's architectural width, Katmai implements the SIMD-FP adder as a separate unit on the second dispatch port, as Figure 3 shows. This organization allows half of a SIMD multiply and half of an independent SIMD add to be issued together, as Figure 4 shows, bringing the peak throughput back to four floating-point operations per cycle—at least for code with an even distribution of multiplies and adds.

Although this situation is common in DSP algorithms, the adds are usually data dependent on the multiplies. Thus, for Katmai to realize its peak throughput, the code will have to be scheduled to cover the latencies. With only eight registers and a destructive two-operand (i.e., $R_D \leftarrow R_D$ op R_S) instruction format, however, such a code schedule will be difficult to achieve. To help with this problem, Katmai can issue two register-move instructions simultaneously.

The Katmai implementation of the SIMD-FP architecture adds new units for the shuffle instructions and reciprocal estimates. The new units increase the loading on the dispatch ports, which would have compromised frequency if not for the enormous effort Katmai's engineers put into tuning critical circuit paths. Apparently they were successful, as Katmai will actually be 50 MHz faster than Deschutes.

Although Katmai's implementation leaves some of the potential of the SSE architecture on the table, the tradeoff for silicon area is understandable. But the danger with Katmai's method is that the hardware has implemented a different model of parallelism than is implied by the architecture. This sets up a code-scheduling dilemma: Should the code be scheduled for Katmai to maximize near-term performance, or would it be better to schedule for the architecture in anticipation of a full implementation in a future processor? Sources indicate that Coppermine will use the same core as Katmai, so a full implementation of SSE is not likely until Willamette, which isn't expected until 2H00.

SIMD-FP Architecture Is Full IEEE-754

Pentium III's SIMD-FP instructions adhere to the IEEE-754 specification, including all four rounding modes and the maskable numeric exceptions. Unlike the x87 scalar FPU, which implements numeric exceptions in microcode, Katmai's SIMD-FP units implement all exceptions in hardware,

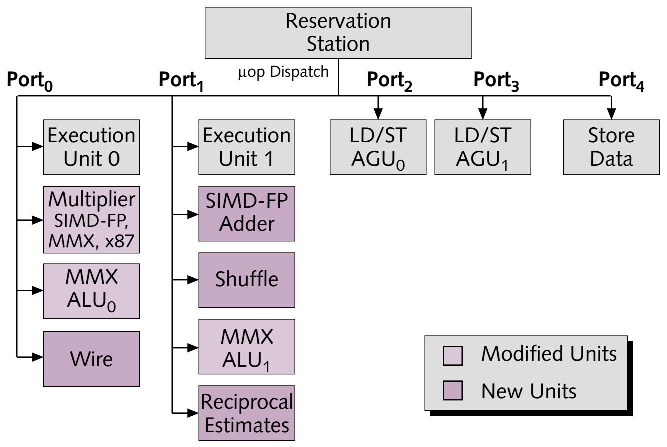

Figure 3. Katmai adds new execution units and modifies others while preserving the P6's basic five-port dispatch mechanism.

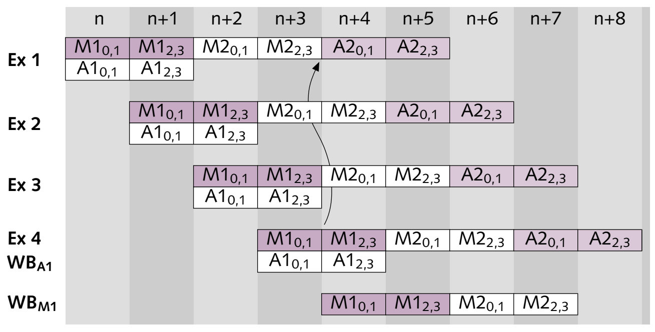

Figure 4. Katmai can execute a SIMD multiply (M1) and an independent add (A1) every two cycles. A dependent add (A2←M1) suffers a four-cycle delay.

except underflows and denormalized results. This is crucial to SIMD performance, since any exception within a four-element vector can trigger a microcode-recovery routine.

To prevent microcode processing from hampering performance, the architecture provides a mode to flush underflows to zero (FTZ). Although operation in this mode is not IEEE-754 compliant, it is acceptable for most 3D-graphics applications, and it will be useful in many other algorithms where IEEE accuracy is less important than raw speed.

A new control-and-status register (the MXCSR) is provided to report the IEEE exception flags and to control the FTZ mode, the directed-rounding modes, and the exception enables. Unmasked exception traps are delivered through interrupt 19, a new x86 interrupt assigned for that purpose.

More Than Just SIMD-FP Arithmetic

By performing operations on multiple data elements simultaneously, SIMD architectures impose difficulties not present with scalar machines. Means must be provided to make control-flow decisions on individual data elements, to perform operations on a subset of elements within a vector, and to reorganize the data so the SIMD units can be fully utilized.

For data-dependent decisions, the architecture provides instructions that compare elements in two SIMD-FP vectors according to one of the relations: equal, less-than, less-than-or-equal, unordered, not-equal, not-less-than, or not-less-than-or-equal. The result of a compare instruction is a four-element Boolean-result vector (true = 0xFFFFFFFF, false = 0x00000000) that can be manipulated with a set of SIMD logical instructions (AND, ANDN, OR, XOR) to mask elements within a vector. A MOVMSKPS instruction allows the most significant bit of each Boolean element to be deposited into an x86 general register. From there, control-flow decisions can be made on any of these four bits.

The need often arises to transpose data elements from a row to a column organization so the SIMD parallelism can be exploited. Consider the example of 3D coordinates organized as WZYX vectors in memory, as Figure 5 shows. To transpose these into vectors of like coordinates, the architecture provides MOVHPS, MOVLPS, and SHUFPS instructions. With these instructions, 3D transformations (without clipping) can be performed with only a 25% overhead, compared with data that has been preorganized and stored as $X_3X_2X_1X_0$. Additional arithmetic operations further amortize the overhead, typically into the 5–10% range.

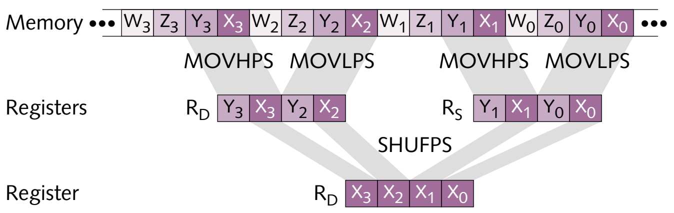
Memory ••• W_3 Z_3 Y_3 X_3 W_2 Z_2 Y_2 X_2 W_1 Z_1 Y_1 X_1 W_0 Z_0 Y_0 X_0 •••
MOVHPS MOVLPS MOVHPS MOVLPS
Registers
R_D Y_3 X_3 Y_2 X_2
R_S Y_1 X_1 Y_0 X_0
SHUFPS
Register
R_D X_3 X_2 X_1 X_0

Figure 5. With the MOVHPS, MOVLPS, and SHUFPS instructions, Pentium IIIs can transpose vectors with only a small overhead.

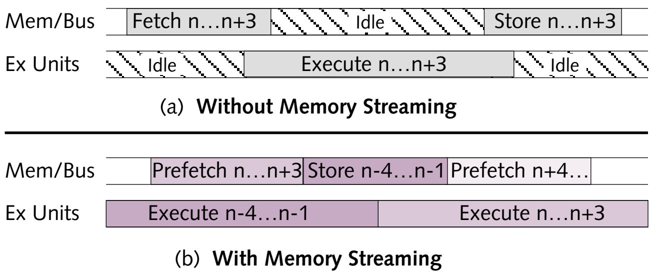

Figure 6. Prefetch instructions allow memory fetches to overlap execution, increasing bus and execution-unit utilization.

Memory Streaming Feeds Execution Engine

A downside of SIMD engines is that they can easily increase the processing rate to a level above the memory system's ability to supply data. When that happens, the execution units stall or become poorly utilized, thus limiting the SIMD speedup. Such would have been the case for Katmai, as the 3D-transform-and-lighting example in Figure 6(a) illustrates, had Intel done nothing to improve memory performance.

But Intel increased the throughput of the memory system and the P6 bus, as Figure 6(b) shows, with a set of features it collectively calls memory streaming. These features include prefetch instructions, streaming stores, and enhanced write combining (WC). As with the SIMD code itself, the effectiveness of these measures will depend on software's use of the new instructions and on a good code schedule.

Memory-streaming features have been used before in RISC processors. Although they have proved effective in many specific situations, they have suffered in general from three problems: they do not lend themselves to use with the random memory-access patterns found; they are notoriously difficult to apply effectively due to the dynamics of the run-time environment; and compilers have not been trained to use them, so they do not have broad applicability.

Multimedia applications, however, rarely have random access patterns, and Katmai's prefetch cachability should help control the dynamics. Rapid advances are also being made in data prediction, allowing compilers to use prefetch instructions more effectively than in the past.

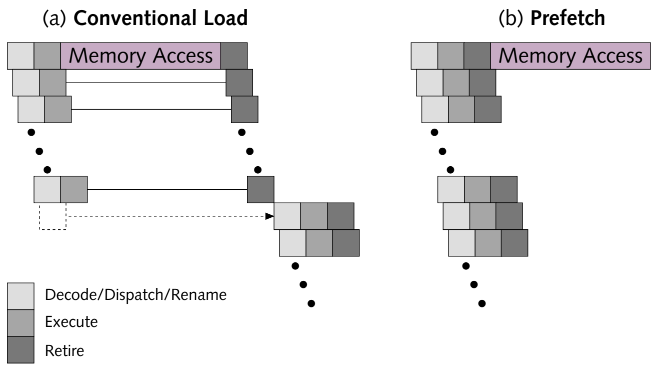

Figure 7. In-order retirement can cause completion resources to saturate following a load, stalling the pipeline (a). Prefetches retire ahead of the memory access, freeing completion resources (b).

Prefetch Hides Latency, Frees Resources

For multimedia streams, loads have shortcomings. Because they can fault (e.g., on a protection violation), loads cannot generally be hoisted (moved up in the code sequence) far enough to cover very much memory latency. When they are hoisted, they often cause the machine's in-order completion resources (e.g., reorder-buffer entries) to fill up, limiting the number of instructions that can be in flight, as Figure 7(a) shows. Furthermore, loads give no clue to the temporal locality of their data, so cache optimization is impossible.

Katmai's prefetch instructions address these shortcomings. Being only hints, prefetch instructions do not fault or modify architectural state. As a result, they can be hoisted arbitrarily far and retired before the memory access completes, as Figure 7(b) shows, freeing completion resources so more instructions can be dispatched. Plus, Katmai's prefetch instructions provide suggestions to the hardware regarding the cache level to which the addressed line should be allocated. Options include all cache levels, L1 only, all levels above L1, and cache bypass (fetching to a temporary read buffer).

Katmai also adds a streaming store, which, on a miss, bypasses the cache without allocating a new line. This mechanism allows software to avoid polluting the caches when it knows the data being stored will not be accessed again soon.

To improve the performance of streaming stores, Katmai enhanced the write-combining (WC) feature that has been in all P6-bus-based processors. Previous P6 processors used a single cache-line buffer (32 bytes) to collect sequential write-through stores so they could be pushed out as a single bus transaction, improving bus efficiency. Katmai increases to four the number of WC buffers, as Figure 8 shows, and also improves the buffer-management policies to increase their effectiveness. (Actually, Katmai doesn't add new buffers; it just uses Deschutes' existing load buffers for double duty.)

The WC buffer allocation and flush policies were improved to reduce the average buffer occupancy. In Deschutes, the WC buffer was not flushed until a new request forced it to be pushed out; Katmai, on the other hand, automatically empties the buffer as soon as it fills. Intel insists that the new WC structure is effective, citing simulations that show it rarely, if ever, degrades load performance but often substantially improves write-through performance. An SFENCE instruction was also added to give software a means of manually flushing the WC buffers when it is necessary to ensure that all prior stores are globally visible (e.g., for screen updates or for software-managed coherence).

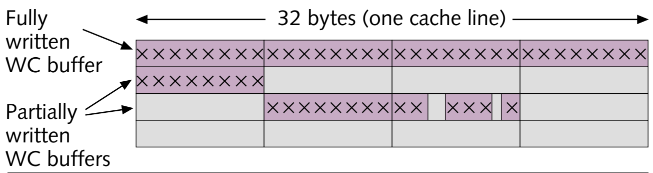

Figure 8. Katmai reuses Deschutes' load buffers for write combining. Intel says that, in practice, conflicts between the two uses rarely occur.

Not a New Core, But Not Chopped Liver Either

Intel's choice of name set the expectation of a new microprocessor core. Against that expectation, Katmai fails. But that goal may have been wrong anyway. A new core may have—with luck—gained a 50% speedup. While that speedup may have made for faster word processors and spreadsheets, it would hardly have enabled anything fundamentally new.

By focusing efforts on multimedia features instead of a new core, however, Katmai will accelerate—by several times—the critical underlying multimedia algorithms. In one speech application, training time improved 60×. Real-time MPEG-2 encoding, impossible on a Pentium II, is achievable on Katmai. Such speedups can, and will, enable new applications, a feat that no practical degree of superscalarity or frequency boost could have achieved.

From this perspective, it's hard to argue that Intel made a mistake. With SSE in Katmai, Intel has set the software stage for the big frequency and memory-bandwidth improvements that will come with Coppermine. But if the focus was on SSE, Intel should have gone all the way. By shoehorning SSE into the existing 64-bit Deschutes core, and by being so stingy with silicon, Intel sacrificed too much.

Unwillingness to revamp Deschutes' instruction decoder forced Intel to stick with the x86 instruction format, limiting to eight the number of registers that could be included in SSE, and constraining the instructions to a destructive two-operand format. Reuse of Deschutes' 64-bit data paths sacrificed nearly half of Katmai's multimedia-performance potential and set up a code-scheduling dilemma with long-term consequences—all for marginal silicon savings.

Not Enough Distance Between P III and K6 III

Although processor architects may find Katmai underwhelming, it is sure to be a commercial success. On competitive grounds, however, the chip may be somewhat of a tactical blunder. In too many ways, AMD's K6 III (see MPR 3/8/99, p. 22) is an even match for Katmai.

SSE does have advantages over 3DNow. SSE uses full IEEE-754 arithmetic, allowing it to serve scientific applications that 3DNow can't. And 3DNow lacks both SSE's new-media instructions, which boost video performance, and its memory-streaming features, which, along with the P6 bus, give Katmai substantially higher delivered bus bandwidth. SSE's new register file gives it a big theoretical advantage over 3DNow, although that advantage is diminished by the destructive two-operand instruction format.

Despite Katmai's half-wide implementation, SSE's largest advantage may still be its 128-bit architectural width. This width allows twice the parallelism to be expressed per instruction as with 3DNow—a fact that reduces the degree of loop unrolling needed to expose parallelism and puts less pressure on the register file, the instruction cache, and instruction issue and reordering resources. This headroom should allow Katmai to sustain higher throughput in practice, even though the K6 III may have the same peak rating.

Price & Availability

The Pentium III processor (Katmai) is available now at a quantity-1,000 list price of $696 for the 500-MHz version and $496 for the 450-MHz version. Both are offered in the SECC2 (Slot 1) module.

A Xeon version of Katmai, code-named Tanner, is expected to be announced in March, with delivery a month later. Intel has also disclosed that a 550-MHz Katmai will be available in 2Q99.

But while Katmai's advantages are not insignificant, they are also not overpowering. By not pulling out the stops on the architecture and on Katmai's implementation of it, Intel may have blown its chance to knock out 3DNow and to regain complete architectural control of the x86 platform.

Diffusion Key to Success

Katmai's technical advantages do not seem so compelling that they alone can overcome 3DNow's 15-million-unit installed-base head start. To complete that job, Intel must rely on its persuasive powers to divert software developers' attention from 3DNow. It must also—quickly—drive the SSE architecture across its product line and down into the Celeron space, where it can build a large enough installed base to attract software developers on its own.

As it normally does with new microprocessors, Intel introduced Pentium III at the top of its product line. The company has set the initial 1,000-unit list price of the 500-MHz part at $696; $496 for the 450-MHz version. At the same time, it dropped the Pentium II-450 price to $476, while the -400 and -300 fell to $264 and $192.

Thus, at the same frequency, Intel is charging only a $20 (4%) premium for SSE, indicating that it is serious about a rapid transition to SSE-based Pentium IIIs. We expect the SSE premium to disappear quickly and Pentium III to come rapidly down Intel's price curve, approaching $200 by year end. By that time, the 600-MHz Coppermine will have replaced Katmai at the higher price points.

The SSE architecture, however, will not make it into the Celeron line until 2000. This should happen in the first quarter, as we expect Intel to introduce a version of Coppermine, possibly with half the on-chip cache, into that space. Once the Celeron space is breached, SSE will serve as a powerful differentiator that could leave AMD in the lurch.

All Hopes Riding on SSE

Even if SSE bests 3DNow, Katmai will still not claim the desktop-multimedia high ground. Apple, whose desktop market share is once again on the rise, will this summer field G4-based platforms. The G4 processor provides a full implementation of Motorola's 128-bit AltiVec multimedia architecture (see MPR 5/11/98, p. 1). With 32 registers, four-operand nondestructive instructions, multiply-add, 32 × 16 bytewise permute, select, multistream prefetching, and a host of other powerful features, the G4 should outperform Katmai on all multimedia algorithms, regardless of frequency differential. Although PowerPC is not a direct competitor, the pesky Apple could sap some software development resources that Intel would prefer to have on SSE and Katmai.

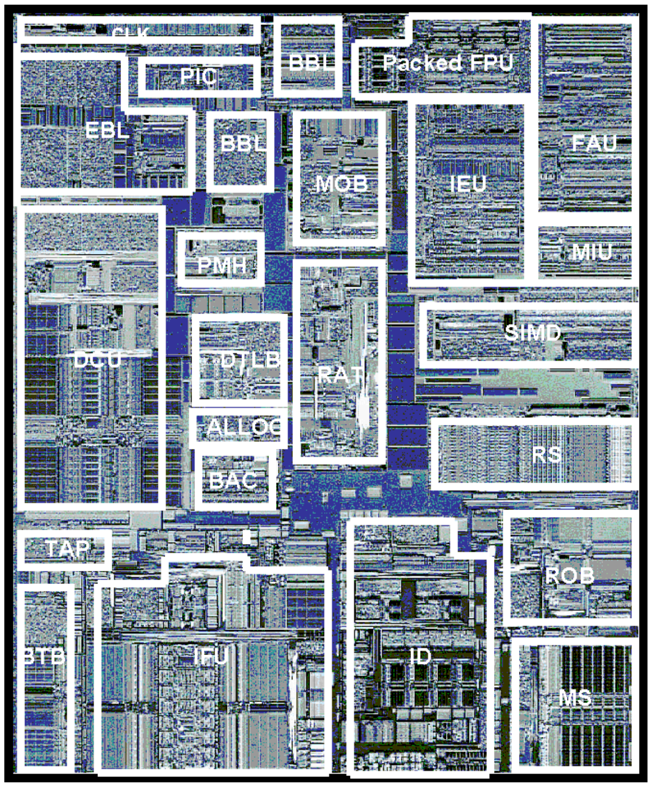

Figure 9. The first Pentium III processor, Katmai, implements 9.5 million transistors and measures 12.3 × 10.4 mm in 0.25-micron P856.5 with five layers of aluminum interconnect. (Source: Intel)

With the K6 III nipping at Katmai's heels, Intel must also contend with the K7 this summer. If AMD can execute, which is not yet a sure thing, the K7 could actually grab the performance lead on general applications, and possibly even 3D. Intel's secret wish that AMD will stumble getting K7 into production at frequency, as it did with K6, appears less likely to come true with each passing day. AMD is probably breathing a sigh of relief that Intel was not more aggressive with Katmai. For the first time, it is possible that AMD could have Intel surrounded. How this could happen is a question that Intel execs are undoubtedly contemplating.

But Intel can potentially blunt the impact of the K7 by getting quickly to a 0.18-micron process with Coppermine. This move should allow Intel to stay ahead of the K7 on frequency and within marketing distance of it on performance. If Intel can avoid missteps, Katmai and Coppermine should safely bridge the company to Willamette, with only a few scrapes and bruises along the way. If it makes the transition safely, we expect that the company will never again yield to the complacency that, in this round, has allowed a competitor to get so dangerously close. M