Intel's P6 Uses Decoupled Superscalar Design

Next Generation of x86 Integrates L2 Cache in Package with CPU

by Linley Gwennap

Intel's forthcoming P6 processor (see cover story) is designed to outperform all other x86 CPUs by a significant margin. Although it shares some design techniques with competitors such as AMD's K5, NexGen's Nx586, and Cyrix's M1, the new Intel chip has several important advantages over these competitors.

The P6's deep pipeline eliminates the cache-access bottlenecks that restrict its competitors to clock speeds of about 100 MHz. The new CPU is designed to run at 133 MHz in its initial 0.5-micron BiCMOS implementation; a 0.35-micron version, due next year, could push the speed as high as 200 MHz.

In addition, the Intel design uses a closely coupled secondary cache to speed memory accesses, a critical issue for high-frequency CPUs. Intel will combine the P6 CPU and a 256K cache chip into a single PGA package, reducing the time needed for data to move from the cache to the processor.

Like some of its competitors, the P6 translates x86 instructions into simple, fixed-length instructions that Intel calls micro-operations or uops (pronounced "you-ops"). These uops are then executed in a decoupled superscalar core capable of register renaming and out-of-order execution. Intel has given the name "dynamic execution" to this particular combination of features, which is neither new nor unique, but highly effective in increasing x86 performance.

The P6 also implements a new system bus with increased bandwidth compared to the Pentium bus. The new bus is capable of supporting up to four P6 processors with no glue logic, reducing the cost of developing and building multiprocessor systems. This feature set makes the new processor particularly attractive for servers; it will also be used in high-end desktop PCs and, eventually, in mainstream PC products.

Not Your Grandfather's Pentium

While Pentium's microarchitecture carries a distinct legacy from the 486, it is hard to find a trace of Pentium in the P6. The P6 team threw out most of the design techniques used by the 486 and Pentium and started from a blank piece of paper to build a high-performance x86-compatible processor.

The result is a microarchitecture that is quite radical compared with Intel's previous x86 designs, but one that draws from the same bag of tricks as competitors' x86 chips. To this mix, the P6 adds high-performance cache and bus designs that allow even large programs to make good use of the superscalar CPU core.

As Figure 1 (see page 10) shows, the P6 can be divided into two portions: the in-order and out-of-order sections. Instructions start in order but can be executed out of order. Results flow to the reorder buffer (ROB), which puts them back into the correct order. Like AMD's K5 (see MPR 10/24/94, p. 1), the P6 uses the ROB to hold results that are generated by speculative and out-of-order instructions; if it turns out that these instructions should not have been executed, their results can be flushed from the ROB before they are committed.

The performance increase over Pentium comes largely from the out-of-order execution engine. In Pentium, if an instruction takes several cycles to execute, due to a cache miss or other long-latency operation, the entire processor stalls until that instruction can proceed. In the same situation, the P6 will continue to execute subsequent instructions, coming back to the stalled instruction once it is ready to execute. Intel estimates that the P6, by avoiding stalls, delivers 1.5 SPECint92 per MHz, about 40% better than Pentium.

x86 Instructions Translate to Micro-ops

The P6 CPU includes an 8K instruction cache that is similar in structure to Pentium's. On each cycle, it can deliver 16 aligned bytes into the instruction byte queue. Unlike Pentium, the P6 cache cannot fetch an unaligned cache line, throttling the decode process when poorly aligned branch targets are encountered. Any hiccups in the fetch stream, however, are generally hidden by the deep queues in the execution engine.

The instruction bytes are fed into three instruction decoders. The first decoder, at the front of the queue, can handle any x86 instruction; the others are restricted to only simple (e.g., register-to-register) instructions. Instructions are always decoded in program order, so if an instruction cannot be handled by a restricted decoder, neither that instruction nor any subsequent ones can be decoded on that cycle; the complex instruction will eventually reach the front of the queue and be decoded by the general decoder.

Assuming that instruction bytes are available, at least one x86 instruction will be decoded per cycle, but more than one will be decoded only if the second (and third) instructions fall into the "restricted" category. Intel refused to list these instructions, but they do not include any that operate on memory. Thus, the P6's ability to execute more than one x86 instruction per cycle relies

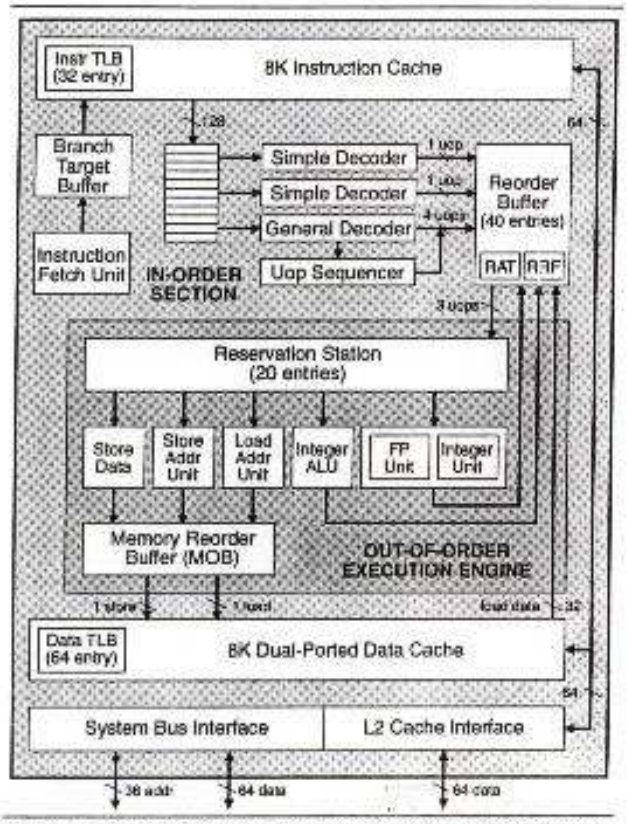

Figure 1. The P6 combines an in-order front end with a decoupled superscalar execution engine that can process RISC-like micro-ops speculatively and out of order.

on avoiding long sequences of complex instructions or instructions that operate on memory.

The decoders translate x86 instructions into uops. P6 uops have a fixed length of 118 bits, using a regular structure to encode an operation, two sources, and a destination. The source and destination fields are each wide enough to contain a 32-bit operand. Like RISC instructions, uops use a load/store model; x86 instructions that operate on memory must be broken into a load uop, an ALU uop, and possibly a store uop.

The restricted decoders can produce only one uop per cycle (and thus accept only instructions that translate into a single uop). The generalized decoder is capable of generating up to four uops per cycle. Instructions that require more than four uops are handled by a uop sequencer that generates the requisite series of uops over two or more cycles. Because of x86 constructs such as the string instructions, a single instruction can produce a very long sequence of uops. Many x86 instructions, on the other hand, translate into a single uop, and the average is 1.5–2.0 uops per instruction, according to Intel.

The uops then pass through the reorder buffer. The ROB must log each uop so it can later be retired in program order. Each of the 40 ROB entries also has room to store the result of a load or calculation uop along with the condition codes that could be changed by that uop. As uops execute, they write their results to the ROB.

Closely associated with the ROB is the register alias table (RAT). As uops are logged in the ROB, the RAT determines if their source operands should be taken from the real register file (RRF) or from the ROB. The latter case occurs if the destination register of a previous instruction in the ROB matches the source register; if so, that source register number is replaced by a pointer to the appropriate ROB entry. The RAT is also updated with the destination register of each uop.

In this way, the P6 implements register renaming. When uops are executed, they read their data from either the register file or the ROB, as needed. Renaming, a technique discussed in detail when Cyrix introduced its M1 design (see MPR 10/25/93, p. 1), helps break one of the major bottlenecks of the x86 instruction set: the small number of general-purpose registers. The ROB provides the P6 with 40 registers that can hold the contents of any integer or FP register, reducing the number of stalls due to register conflicts.

Out-of-Order Engine Drives Performance

Up to three uops can be renamed and logged in the ROB on each cycle; these three uops then flow into the reservation station (RS). This section of the chip holds up to 20 uops in a single structure. (The RS can be smaller than the ROB because the ROB must track uops that have executed but are not retired.) The uops wait in the reservation station until their source operands are all available. Due to the register renaming, only a single ROB entry (per source) must be checked to determine if the needed value is available. Any uops with all operands available are marked as ready.

Each cycle, up to five uops can be dispatched: two calculations, a load, a store address, and a store data. A store requires a second uop to carry the store data, which, in the case of floating-point data, must be converted before writing it to memory.

There are some restrictions to pairing calculation uops due to the arrangement of the read ports of the reservation station. As Figure 1 shows, only one uop per cycle can be dispatched to the main arithmetic unit, which consists of the floating-point unit and a complete integer unit, which has an ALU, shifter, multiplier, and divider. The second calculation uop goes to the secondary ALU and must be either an integer-ALU (no shifts, multiplies, or divides) or a branch-target-address calculation. Simplifying the second calculation unit reduces the die area with little impact on performance.

In many situations, more uops will be ready to execute than there are function units and read ports. When this happens, the dispatch logic prioritizes the available uops according to a complex set of rules. Intel declines to

discuss these rules but notes that older uops are given priority over newer ones, speeding the resolution of chains of dependent operations.

The integer units handle most calculations in a single cycle, but integer multiply and divide take longer. The P6 FPU executes adds in three cycles and requires five cycles for multiply operations. FP add and multiply are pipelined and can be executed in parallel with long-latency operations. Table 1 shows the cycle times for long-latency integer and floating-point operations.

The FXCH (floating-point exchange) instruction is not handled by the FPU. This instruction swaps the top of the FP register stack with another register in the stack and is frequently used in x86 floating-point applications, since many x86 FP instructions access only the top of the stack. The P6 handles FXCH entirely in the reorder buffer by treating it as a renaming of two registers. Thus, FXCH uops enter the ROB but not the reservation station and never enter the function units.

The dual address-generation units each contain a four-input adder that combines all possible x86 address components (segment, base, index, and immediate) in a single cycle. As in Pentium, each unit also contains a second four-input adder to perform a segment-limit check in parallel. The resulting address (along with, for a store, source data) is then placed in the memory reorder buffer (MOB, in P6-ese) to await availability of the data cache.

The reorder buffer can accept three results per cycle: one from the main arithmetic unit, one from the secondary ALU, and one from a load uop. Because of long-latency operations, two or more function units within the main arithmetic unit can generate results in a single cycle. In this case, the function units must arbitrate to write to the ROB.

After a uop writes its result to the ROB (or, in the case of a store, to the MOB), it is eligible to be retired. The ROB will retire up to three uops per cycle, always in program order, by writing their results to the register file. Uops are never retired until all previous uops have completed successfully. If any exception or error occurs, uncommitted results in the ROB can be flushed, resetting the CPU to its proper state, as if the instructions had been executed in order up to the point of the error.

Nonblocking Caches Reduce Stalls

A key to the P6's performance is its cache subsystem. The primary data cache, at 8K, is relatively small for a processor of this generation, but the fast level-two cache helps alleviate the lower hit rate of the primary cache. If an access misses the data cache, the cache can continue servicing requests while waiting for the miss data to be returned. This technique, called a nonblocking cache or hit-under-miss, has been used for years by PA-RISC processors to avoid stalls.

The P6 takes advantage of the nonblocking cache with its memory reorder buffer. If the access at the front of the MOB misses, subsequent accesses will continue to execute. Since these accesses can execute out of order, the P6 must take care to avoid incorrect program execution. For example, loads can arbitrarily pass loads, but stores must always be executed in order. Loads can pass stores only if the two addresses are verified to be different. Intel would not reveal the size of the MOB or other details of its function.

	P6		Pentium	
	Throughput	Latency	Throughput	Latency
Integer multiply	1 cycle	4 cycles	4–8 cycles	7–14 cycles
Integer divide	12–36 cyc	12–36 cyc	42–84 cyc	42–84 cyc
FP add	1 cycle	3 cycles	1 cycle	3 cycles
FP multiply	2 cycle	5 cycles	1 cycle	3 cycles
FP divide	18–38 cyc	18–38 cyc	39 cycles	39 cycles
FP sq root	29–69 cyc	29–69 cyc	70–140 cyc	70–140 cyc

Table 1. P6 floating-point latencies are similar to Pentium's, but integer arithmetic is much faster. Latencies are the same for single, double, and extended precision except where ranges are shown.

The P6 data cache can process one load and one store per cycle as long as they access different banks. The cache is divided into four interleaved banks, half as many as Pentium's data cache. AMD's K5 also implements a four-bank data cache, and that company says that there is little benefit to an eight-bank design.

The P6 cache cannot handle two loads at once, in stark contrast to processors such as the K5, M1, and even Pentium. Typical x86 code generates a large number of memory references due to the limited register set, and more of these references will be loads than stores. Intel points out that the entire processor is designed for one load per cycle—even the decoders cannot produce more than one load uop per cycle—and that its simulations show this capability is adequate to attain the desired performance level.

The data cache has a latency of three cycles (including address generation) but is fully pipelined, producing one result per cycle. The unified level-two (L2) cache has a latency of three cycles and, like the data cache, is pipelined and nonblocking. The fast L2 latency is achieved by implementing the L2 cache as a single chip and combining it with the CPU in a single package, as Figure 2 (see page 12) shows.

Address translation occurs in parallel with the data cache access. If the access misses the data cache, the translated (real) address is sent to the L2 cache. The latency of a load that misses the data cache but hits in the L2 is six cycles, assuming that the L2 cache is not busy returning data from an instruction cache miss or an earlier data cache miss.

The P6 CPU contains a complete L2 cache controller and is Intel's first x86 processor with a dedicated L2 cache bus. Both NexGen's 586 and the R4000 use this design style to increase cache-bus bandwidth while

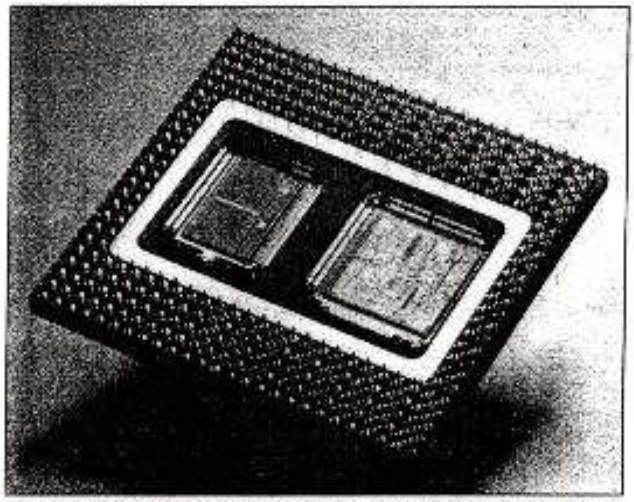

Figure 2. The P6 CPU and L2 cache are combined in a single 387-pin PGA package that measures 7.75 × 6.25 cm (2.66" × 2.46").

allowing the separate system bus to operate at a lower, more manageable speed.

The L2 cache uses the same 32-byte line size as the on-chip caches. It returns 64 bits of data at a time, taking four cycles to refill a cache line. The cache always returns the requested word within the first transfer, getting the critical data back to the processor as quickly as possible. Table 2 shows other cache parameters.

This combination of nonblocking caches with a fast L2 cache provides the P6 with better performance on memory accesses than its x86 competitors. The processor stalls less often and has relatively quick access to 256K of memory. The K5, by contrast, has 24K of on-chip cache but will take longer to access its secondary cache.

Deep Pipeline Speeds Clock Rate

Another advantage that the P6 has over its x86 competitors is a higher clock rate. Intel achieves this feat by deeply pipelining the chip. Figure 3 shows the P6 pipeline, which consists of 12 stages. This pipeline represents the case of an instruction that flows through the CPU as quickly as possible. It is more likely that the instruction will be stalled in the reservation station for some number of cycles, a delay represented by the thick black band. The second black band represents another potential delay: completed instructions can spend several cycles waiting in the ROB before retirement.

	Instruction	Data	Level Two
Cache Size	8K	8K	256K
Line Size	32 bytes	32 bytes	32 bytes
Throughput / Latency	1 / 3 cycles	1 / 3 cycles	1 / 3 cycles
Nonblocking?	yes	yes	yes
Associativity	four-way	two-way	four-way
Access Width	128 bits	64 bits	64 bits
Number of Ports	one	two	one
Number of Banks	n/a	four	n/a
Indexed	virtual	virtual	physical
Tagged	physical	physical	physical
TLB Entries	32 entries	64 entries	—
TLB Associativity	fully	fully	—
TLB Number of Ports	one	two	—

Table 2. The fast nonblocking L2 cache is fully pipelined and helps make up for the higher miss rate of its small primary caches.

In the first stage, the next fetch address is calculated by accessing the branch target buffer (BTB). If there is a hit in the BTB, the fetch stream is redirected to the indicated location. Otherwise, the processor continues to the next sequential address.

The instruction cache access is spread across two and one-half cycles. The K5, in contrast, must calculate the next address and read from the instruction cache in a single cycle. By allowing multiple pipeline stages for the cache access, Intel removes this task from the critical timing path and allows the P6 clock to run faster.

Instructions are then fed to the decoders. The complex problem of decoding variable-length x86 instructions is allocated two and one-half cycles as well. Part of the problem in a superscalar x86 processor is identifying the starting point of the second and subsequent instructions in a group. The K5 includes predecode information in its instruction cache to hasten this process, but the P6 does not, to avoid both instruction-cache bloat and the bottleneck of predecoding instructions as they are read from the L2 cache.

The decoding issue is easier for P6 because the second and third decoders handle only simple instructions. The restricted decoders can wait for the general decoder to identify the length of the first instruction and still have time to handle the remaining instructions, assuming that they are simple ones. If they are not simple, the restricted decoders must pass them on to the general decoder in the next cycle.

At the end of stage 6, the x86 instructions have been fully decoded and translated into uops. These uops have their registers renamed in stage 7. In stage 8, the renamed uops are written to the reservation station. If the operands for a particular uop are available, and if no other uops have priority for the needed function unit, that uop is dispatched. Otherwise, the uop will wait for its operands to become available.

It takes one cycle (stage 9) for the reservation station to decide which uops can be dispatched. The P6 implements operand bypassing, so a result can be used on the immediately following cycle. The reservation station will attempt to have the corresponding uop arrive at the function unit at the same time as the necessary data.

Simple integer uops can execute in a single cycle (stage 10). Some integer uops and all FP uops take several cycles at this point. Load and store uops generate their address in one cycle and are written to the MOB; if the MOB is empty, a load will go directly to the data

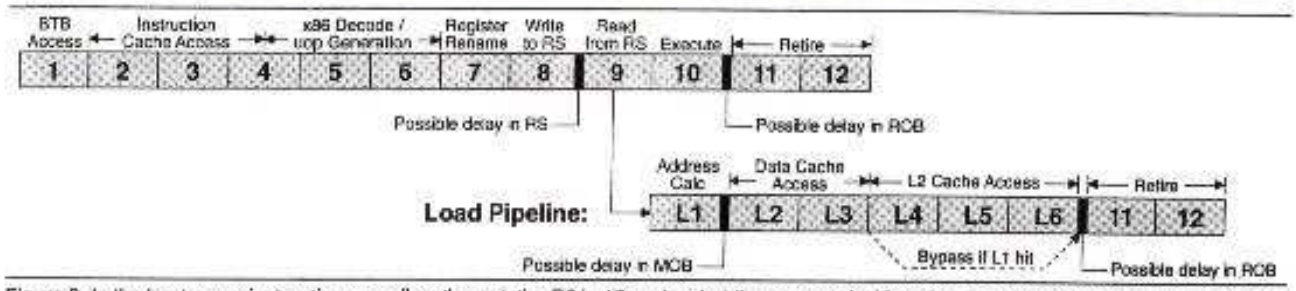

Figure 3. In the best case, instructions can flow through the P6 in 12 cycles, but the average is 18 cycles due to delays in the reservation station (RS) or the reorder buffer (ROB). Load instructions take longer and can also be delayed in the memory reorder buffer (MOB).

cache, but it will take an additional three cycles (assuming a cache hit) before the loaded data is available for use. The address-generation unit is probably a critical timing path in the P6, limiting the processor to 133 MHz; RISC processors can typically cycle their simpler ALUs at 200 MHz or better in 0.5-micron technology.

Once the uop executes, it writes its result to the ROB. If all previous uops have been retired, it takes one cycle to retire a uop. If previous instructions are still pending, however, it may take several cycles before the uop is retired. This delay does not impact performance.

Branch Prediction Accuracy Is Critical

The deep pipeline creates extraordinary branch penalties. The outcome of a conditional branch is not known until stage 10. The minimum penalty for a mispredicted branch is 11 cycles. Intel estimates that, on average, a uop spends four cycles in the reservation station, so a mispredicted branch will typically cause a 15-cycle penalty. If the branch spends an unusually long time in the reservation station, the penalty could be even worse.

Thus, the P6 designers spent a lot of effort to reduce the number of mispredicted branches. Like Pentium, the P6 uses a branch target buffer that retains both branch-history information and the predicted target of the branch. This table is accessed by the current program counter (PC). If the PC hits in the BTB, a branch is predicted to the target address indicated by the BTB entry; there is no delay for correctly predicted taken branches.

The BTB has 512 entries organized in a four-way set-associative cache. This size is twice that of Pentium, improving the hit rate. The P6 rejects the commonly used Smith algorithm, which maintains four states using two bits, in favor of the more recent Yeh method[1]. This adaptive algorithm uses four bits of branch history and can recognize and predict repeatable sequences of branches, for example, taken–taken–not taken. We estimate that the P6 BTB will deliver close to 90% accuracy on programs such as the SPECint92 suite.

A second type of misprediction occurs if a branch misses in the BTB. This situation is not detected until the instruction is fully decoded in stage 6. The branch is predicted to be taken if the offset is negative (indicating a likely loop), and the target address, if available, is used to redirect the fetch stream. At that point, however, seven cycles have been wasted fetching and decoding instructions that are unlikely to be needed. In this case, the long fetch-and-decode pipeline saps performance.

Forward branches that miss the BTB are predicted to be not taken, so the sequential path continues to be fetched with no delay. Branches that miss the BTB are mispredicted more often than those that hit and are subject to the same mispredicted branch penalties.

Conditional Move Added

The P6 instruction set is nearly identical to Pentium's and, in fact, to that of the 386. The most significant addition is a conditional move (CMOV) instruction. This instruction, which has been added to several RISC instruction sets recently, helps avoid costly mispredictions by eliminating branches. The instruction copies the contents of one register into another only if a particular condition flag is set, replacing a test-and-branch sequence.

Like all current Intel processors, the P6 implements system-management mode. The new CPU also supports all the features in Pentium's secret Appendix H. Intel says that these features—including CPU ID, large page sizes, and virtual-8086 extensions—will be fully documented when the P6 is released. The P6 also implements performance counters similar to those described in Appendix H, but it uses a new instruction that allows them to be accessed more easily.

It has been widely speculated that the P6 would include new instructions to speed NSP (native signal processing) applications. While the P6's improved integer multiplier will assist NSP, multimedia extensions such as those in Sun's UltraSparc (see MPR 12/5/94, p. 15) would have an even bigger effect. Intel, however, denies that the P6 has any such extensions. This oversight could give RISC processors like UltraSparc a significant performance advantage when executing increasingly popular multimedia software.

P6 Bus Allows Glueless MP

The P6 system bus is completely redesigned from the Pentium bus. Both use 64 bits of data and operate at

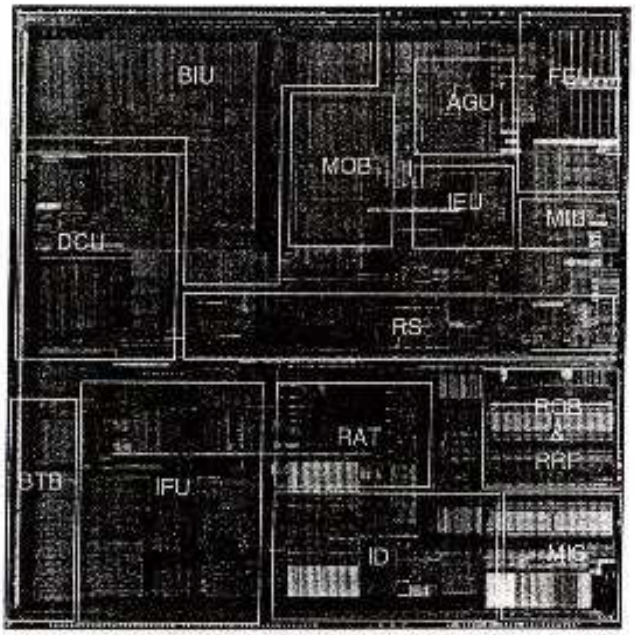

Figure 4. The P6 CPU measures 17.5 × 17.5 mm when fabricated in a 0.5-micron four-layer-metal BiCMOS process.

a maximum of 66 MHz, but the P6 can sustain much higher bandwidth because it uses a split-transaction protocol. When Pentium reads from its bus, it sends the address, waits for the result, and reads the returned data. This sequence ties up the bus for essentially the entire transaction. The P6 bus, in contrast, continues to conduct transactions while waiting for results. This overlapping of transactions greatly improves overall bus utilization.

All arbitration is conducted on separate signals on the P6 bus in parallel with data transmission. Addresses are sent on their own 36-bit bus, so the bus is capable of sustaining the full 528-Mbyte/s bandwidth for an indefinite period. Since the P6 has a private cache bus, the entire system-bus bandwidth can be devoted to memory and I/O accesses. (Intel will disclose the details of the P6 system bus at a later date.)

A split-transaction bus is ideal for a multiprocessor (MP) system. Intel has designed the P6 bus to support up to four processors without any glue logic; that is, the processor pins can be wired directly to each other with only a single chip set to support several CPUs. Like Pentium, the P6 includes Intel's advanced priority interrupt controller (APIC), simplifying multiprocessor designs. The company believes that the system bus has enough bandwidth to support four P6 processors with little performance degradation. Larger MP systems can be built from clusters of four processors each.

To operate at 66 MHz with up to eight devices (four processors along with two memory controllers and two I/O bridges), the P6 bus operates at modified GTL (Gunning transceiver logic) signal levels. This lower signal voltage (1.5 V) reduces settling time in a complex electrical environment. Intel will roll out the first P6 chip sets along with the processor, but other vendors are expected to produce compatible chip sets as well.

With its integrated cache and APIC, the P6 module offers an easy way to upgrade an MP system by plugging in a new processor. This feature will be most useful in servers, which often sell in MP configurations today. Ultimately, the P6 will be used in multiprocessor PCs, a market being seeded today by the APIC-enabled P54C.

Another Big, Power-Hungry CPU

As always, there is no free lunch: high performance comes at a price. The P6 CPU requires 5.5 million transistors, of which about 4.5 million are for logic and 1.0 million are in the 16K of cache. Even in a 0.5-micron four-layer-metal BiCMOS process, this circuitry requires 306 mm^2 of silicon, as Figure 4 shows. To keep this in perspective, however, the die size is only 4% bigger than that of the original Pentium. As with Pentium, a shrink to the next process will make the P6 much smaller and easier to manufacture.

The cache chip consumes 202 mm^2 and is built in the same process as the CPU, as it must operate at the same clock rate. It contains tags and data storage for 256K of cache and requires 15 million transistors.

Although the P6 uses the same nominal IC process as the P54C Pentium, it operates from a core voltage of 2.9 V rather than Pentium's 3.3 V. This change could indicate some minor process tweaks to reduce transistor size or gate-oxide thickness; such tweaks can improve performance but reduce the allowable supply voltage.

The lower voltage has the side effect of reducing the power consumption; even so, the P6 CPU has a preliminary power rating of 15 W maximum and 12 W typical. With the 256K L2 cache chip, the total power is expected to be 20 W maximum and 15 W typical. This power rating is slightly greater than that of the original P5 Pentium and is quite reasonable compared with next-generation RISC processors, which start at 30 W. The greater surface area of the P6 package allows it to use shorter heat sinks than the notoriously hot P5.

Intel investigated many types of multichip modules (MCMs) before settling on a simple two-cavity PGA, eschewing more complicated (and costly) flip-chip options. The package is similar to a standard PGA with extra ceramic layers to route the 64-bit cache bus between the CPU and cache chips. The die are attached using standard wire bonding; no unusual substrates are required.

The pin arrangement, shown previously in Figure 2, is asymmetric: interstitial pins surround the CPU, which drives all the signals that leave the module, but not the cache chip. We estimate that this 387-pin dual-cavity package costs Intel about $50 in volume. The MPR Cost Model computes the overall manufacturing cost of the P6

module to be roughly $350. This cost is greater than that of competitive chips, but the entire L2 cache is included. In a 0.35-micron process, the cost of the P6 could drop to $150 in 1997.

Improving System Performance

Intel designed the P6 to achieve high performance at the system level, not just within the CPU core. Thus, the team placed significant emphasis on the cache subsystem and the system bus as well as the CPU. The nonblocking caches, closely coupled L2 cache, and split-transaction bus exemplify this emphasis. These features put the P6 a step beyond competitive x86 processors.

In contrast, AMD's K5 appears to be a P6-class CPU core trapped in a Pentium pinout. The business decision to target the Pentium interface leaves the K5 with a single bus for both cache and I/O traffic, hampering performance on programs that overflow the K5's small on-chip caches. This bottleneck will become more severe as AMD increases the K5's clock speed to 150 MHz or higher, since the secondary cache will continue to be restricted to 66 MHz or less.

	Intel P6	MIPS R10000	AMD K5	Cyrix M1	NexGen Nx586	Intel Pentium
Clock Speed	133 MHz	200 MHz	100 MHz	100 MHz	93 MHz	100 MHz
Cache Size (I/D)	8K/8K	32K/32K	16K/8K	16K	16K/16K	8K/8K
Dispatch Rate	3 instr	4 instr	2–3 instr	2 instr	1 instr	2 instr
Function Units	5 units	5 units	7 units	2 units	4 units	3 units
Predecode Bits	none	4 per 32	5 per 8	none	none	none
Branch History	512 × 4	512 × 2	1,024 × 1	256 × 2	2,048 × 2	256 × 2
Out of Order	40 instr	32 instr	16 instr	limited	14 instr	none
Rename Regs	40 regs	64 regs	16 regs	32 regs	22 regs	none
L2 Cache Bus?	yes	yes	no	no	yes	no
Glueless MP?	yes	yes	no	no	no	no
IC Process	0.5µ 4M BICMOS	0.5µ 4M CMOS	0.5µ 3M CMOS	0.65µ 3M CMOS	0.65µ 4M CMOS	0.5µ 4M BiCMOS
Logic Transistors	4.5 million	2.3 million	2.4 million	2.1 million	1.6 million	2.4 million
Total Transistors	5.5 million	5.9 million	4.3 million	3.0 million	3.5 million	3.3 million
Package Type	387-pin MCM-C	527-pin CPGA	296-pin CPGA	296-pin CPGA	463-pin CPGA	296-pin CPGA
Die Size	306 mm²	298 mm²	225 mm²*	394 mm²	196 mm²	163 mm²
Est Mfg Cost	$350*†	$320*	$170*	$340*	$200*	$120*
Power (max)	20 W†	30 W	12 W*	10 W	16 W	10 W
Availability	3Q95*	4Q95	3Q95*	3Q95*	3Q94	2Q94
SPECint92 (est)	200 int	>300 int	130 int	120 int*	110 int*	113 int
SPECfp92 (est)	200 fp*	>600 fp	75 fp*	70 fp*	n/a	82 fp

Table 3. The P6 feature set stacks up well against top x86 competitors and the R10000, a similar RISC implementation. The key differences are clock speed and performance. (Source: vendors except *MDR estimates) †includes L2 cache chip

Intel's deeper pipeline should give the P6 a clock-speed advantage over the K5 in comparable manufacturing processes. While the deeper pipeline also increases pipeline penalties, the P6 has much more sophisticated branch prediction and a larger reorder buffer, allowing it to outperform the K5 on a clock-for-clock basis as well. The K5 will, of course, be less expensive and probably consume less power than the P6.

Cyrix's M1 shares the same system-interface constraints as the K5. Furthermore, its static two-pipeline design is less efficient than the decoupled design of the P6 (and K5). Although Cyrix has access to leading-edge manufacturing technology from IBM, it isn't clear that the company has the resources to quickly move its CPU to the latest processes, keeping pace with Intel and AMD. Cyrix must start from scratch to develop a decoupled P6-class CPU, a process that will take years.

The P6 is similar to several of the next-generation RISC processors, in particular the MIPS R10000 (see MPR 10/24/94, p. 18). Table 3 compares these designs. The R10000 is a four-way superscalar processor, while the P6 is three-way superscalar. The MIPS chip can execute up to 32 instructions out of order, a few less than the P6. Both have a dedicated L2 cache bus and support a high-bandwidth MP system bus.

The P6's CISC handicap shows in two places. Despite the similar microarchitectures, the P6 requires nearly twice as many logic transistors as the MIPS chip; the extra logic handles x86 decode, uop translation, and the foibles of the x86 instruction set. Since both chips have similar die size and transistor budgets, the R10000 is able to include four times as much on-chip cache as the P6, improving performance on many programs.

Second, the first P6 will run at 133 MHz, while the R10000 is expected to achieve 200 MHz using a similar manufacturing process. To come even this close in clock speed, Intel uses a very deep pipeline, a concept that MIPS tried and rejected for the R10000. The deeper pipeline has greater branch penalties, sapping performance. And, of course, the higher clock speed gives the R10000 an intrinsic performance advantage. As a result, the MIPS chip should achieve at least 50% better integer performance than the P6.

Working within the constraints of the x86 instruction set is always a challenge. The P6 takes a huge step beyond the static Pentium architecture, applying a decoupled superscalar engine to the performance problem. Although this design works around many of the bottlenecks of the x86 instruction set, it doesn't match the performance of a pure RISC chip. Compared with other x86 processors, the P6 is clearly the best of class and sets a new standard for other vendors to match. ♦

[1] Tse-Yu Yeh and Yale Patt, "Two-Level Adaptive Training Branch Prediction," *24th International Symposium on Microarchitecture* (Nov. 1991), pp. 51–61.