OBLIQUE PERSPECTIVE

Is Intel Sandbagging on Speed?

So Just How Fast Will the P6 *Really* Run?

by John Wharton, Applications Research

So Intel went and sharpened its pencils, recalibrated its 'scopes, and ran some more tests, and discovered—lo and behold!—the maximum clock frequency of its upcoming P6 processor might actually be a smidgen or two higher than the company first thought *(see **091001.PDF**)*. It seems early devices won't be limited to 133 MHz after all, but will actually do up to 150! This 200-SPECint92 beast may actually deliver 220!

Hunh.

I can't say I'm surprised. I've been telling my clients for months that the P6 was clearly designed to run much, much faster than Intel initially let on; rumors of 220-SPECint92 performance were first floated as far back as February. But I think even the 12% boost Intel just discovered is grossly conservative. Indications are that the current P6 design, *using the current P6 process,* could someday run at 266 MHz or more.

Subdivided Pipeline Stages

For starters, compare the design of the current P6 with that of the 0.6-micron Pentium, née P54C. (Curiously, the process geometry Intel calls "0.6 micron" has a gate length equivalent to other vendors' 0.5-micron processes.) A major difference in philosophy is clear. The P6 may be the first processor yet to earn the oft-used appellation "superpipelined": whereas the Pentium pipeline is five stages long, the P6 breaks execution of even the simplest instructions into a 14-stage ordeal *(see **090202.PDF**)*. Granted, several of these stages perform new functions, like renaming registers and retiring results, but for some aspects of the two designs that didn't change, the P6 performs significantly less work in each stage.

For example, both Pentium and the P6 contain an 8K instruction cache, but whereas Pentium can translate a virtual address, access the cache, compare tags, and return data all in a single clock cycle, the P6 allocates 2.5 cycles to do the same work. Ditto with data: again, the Pentium and P6 data caches each contain 8K bytes, but P6 accesses take three cycles versus one.

Likewise with instruction decoding. Pentium can examine a block of instruction bytes, determine the starting address and length of the first two instructions, decode each, and decide whether both may be issued at the same time, all in a single cycle. The P6 apparently devotes considerable extra logic to instruction decoding, with a massively replicated "speculative predecoder array" that cracks instruction formats beginning at every possible byte boundary of the prefetch buffer, which should actually accelerate the decoding process. Even so, the P6 design allots 2.5 CPU cycles to perform decoding tasks Pentium performs in one.

Stated differently, a P6 running at more than twice the core frequency of a Pentium would actually have more time to access the same-sized caches and to decode the same instructions. The 0.6-micron Pentium has been shipping at speeds up to 100 MHz since mid '94; doubling its 100-MHz clock would yield 200 MHz, not the 150 MHz Intel now claims for the P6.

Moreover, Intel demonstrated a P54C system running at 150 MHz at the ISSCC conference in February of 1994. Granted, this may have been a specially binned, specially tuned, specially chilled oddity; on the other hand, both the P54C design and the process on which it is built were quite young in 1994, and each has since had 18 months to mature. Plus, the P6 may have tricks up its voltage-tuning sleeve (see below). The point is that the ISSCC demo proved it's within the realm of technical feasibility for a 0.6-micron Pentium-like pipeline to run at 150 MHz. 2×150 MHz = 300 MHz, not 150.

Unnecessarily Precise Clock Distribution

At the sub-microarchitecture level, the P6 is different as well. Conventional microprocessors contain timing circuitry at a central location, with powerful drivers that distribute the signal to all the various points at which it's needed. The P6, in contrast, puts clock drivers at 80 separate locations on the die, each of which is tuned to match the specific load and timing needs of the functional units it controls. As a result, the worst-case clock skew is less than 250 ps across the entire chip.

Whatever time is lost to clock skew comes right off the top of the minimum clock period, so skew should be kept as low as practical—*but no lower:* past a certain point, the returns from further improvements diminish. One rule of the design engineer's thumb is that skew should be 5% to 10% of the total period. Reducing skew from 5% to 4%, for instance, would yield an effective interstage clocking time of 96% of the total period, an insignificant improvement over the original 95%.

Now consider the P6: at 150 MHz, a 250-ps skew corresponds to just 3.75% of the clock period. The P6 would have to run between 200 and 400 MHz in order for 250 ps of skew to fall within the 5–10% guidelines.

 © 1995 MicroDesign Resources

Locally Distributed Control Logic

But it may be that total edge-to-edge clock skew isn't even crucial for this chip. The P6 greatly reduces the need to synchronize signal timing across great expanses of silicon. In more conventional microprocessor designs, a central module generates control signals that affect other blocks throughout the chip; if an integer unit, FPU, data cache, and bus interface all respond to timing signals from central control logic, for example, all four functional units plus the control block had better be locked tightly onto the same core clock.

But the P6 CPU is divided into four largely autonomous machines—an in-order front end, an out-of-order dataflow engine, an in-order back end, and a bus interface unit—with the functional blocks that make up each machine grouped into geographically separate regions. Data generally passes between these regions via holding stations and queues for synchronization, and the control signals for each region are derived locally from information passed between the blocks. (Indeed, control logic and data storage functions are often replicated within the functional blocks themselves; see below.)

Moreover, comparing the P6 block diagram to the device floorplan shows that each functional block generally talks just to its physically adjacent neighbors, with relatively short buses and unidirectional data flow. Seldom do signals pass from one block to another, get processed, and return in a single cycle. These factors all relax the need for tight device-wide timing constraints; all imply a design that was truly built for speed.

Massive Local Data Replication

Early microprocessor designs pooled each resource in a central location, with data transferred to nether regions of the chip as needed. The 486 has just one register file and cache; in Pentium, both execution pipelines access the same files, so the same register may have to be read through up to four ports at once. The need for multiple access ports impedes register access time.

In contrast, the P6 design tends to replicate data—sometimes massively—within each functional unit that may eventually need it. Micro-operations (uops) are replicated within both the reservation station (RS) and the reorder buffer. Within the RS, 40 separate registers hold operands for the execution units; all 40 may at times contain the same value. This ensures that when an internal uop is eventually ready to execute, its data will all be *right there where it's needed* and not have to be retrieved from a register file or data bus elsewhere.

Even more remarkably, there appear to be (Intel hasn't released the details) 40 more registers within the RS that hold the six-bit indexes of the temporary-register values for which each uop might be waiting, plus some 120 or so six-bit comparators that constantly monitor the execution-unit result buses. When register EAX (e.g.) is eventually returned after a particularly glacial load operation, all 40 operand-holding registers could in theory glom onto it at once.

All these registers and comparators appear to be located within the RS itself, such that comparisons are made, control signals generated, and data captured using physically adjacent circuitry. This is not your parents' microprocessor; this is a chip designed to run much, much faster than the conventional state of the art.

Unnecessary Clock Configuration Options

The internal CPU clock of a 486DX2 is designed to run at twice the frequency of the bus clock; a DX4 runs three times faster. In the P6, however, the clock multiplier factor is configurable at run time, with even the earliest devices able to multiply the bus clock by factors up to four. The local bus of the P6 runs at 66 MHz, which Intel says will accommodate a fully populated multiprocessor array with four CPUs, two main-memory systems, and two I/O subsystems. Quadrupling a 66-MHz bus clock would produce a core frequency of 266 MHz, not 150. (Well, 264, actually, but with microprocessors, multiples of 66 traditionally round up.)

Why would Intel allow the P6 to be configured to run at 266 MHz if it thought the chip could handle just half that? At 150 MHz, the 4× multiplier could only be used if the local bus were slowed to 37.5 MHz, but there's little reason for a P6 bus ever to run below 60 or 66 MHz. Indeed, with a somewhat less fully populated uniprocessor system, it's far more likely that the enhanced Gunning transceiver logic (GTL+) protocol on which the bus is based could hum along nicely at 100 MHz or more; with a 100-MHz local bus, even the 2× frequency multiplier would boost the clock to 200 MHz, not 150.

Frequency-Independent L2 Caches

The P6 design partitions the CPU and a second-level cache (L2C) onto two separate die, the better to reduce production costs and increase marketing flexibility *(see **0906VP.PDF**)*. This has the side benefit of ensuring ready sockets for future products that allow faster execution. Were the L2 cache to be located off-package—as it is with most conventional system designs—its performance would be limited by the speed of whatever SRAM devices were present on the motherboard. Cranking up the CPU clock to 200 or 266 MHz (to pick a number) wouldn't do much good if the L2-cache bus then had to be slowed back down. But by teaming a sufficiently fast L2C in the same package as the P6 CPU, Intel can ensure that faster products could drop readily into the socket of any existing motherboard design.

This would clearly benefit Intel; since the days of the iAPX432 "micromainframe," the company has pursued schemes to sell multiple CPUs for every mother-

 © 1995 MicroDesign Resources

board shipped. End-user upgrades are a good way to do this. But end users tend not to replace existing CPUs unless they can get at least 50% to 100% more speed. When introducing a new product, then, it may be strategically advantageous to start slow in order to leave as much frequency headroom as possible. The P6 system partitioning will let Intel crank the clock 'way up while remaining compatible with all existing sockets.

Post-Production Voltage Tweaking

As mentioned above, at least some 0.6-micron P54C devices can run at 150 MHz already, but doing so requires carefully tuning the power supply and cooling the CPU. For any given chip, the higher the voltage, the faster (and hotter) it runs, until at some point its oxides punch through or its package melts down. Conventional microprocessors must thus be designed to accommodate variations in process oxide thickness, system supply voltage, and heat dissipation, yet support the rated frequency under worst-worst-worst case conditions. With special tuning and tweaking, then, standard chips can generally run considerably faster than specified.

But the P6 is already designed for post-production tweaking. Four pins on every P6 package are configured at assembly time to indicate the voltage at which the chip should be run; digitally programmable regulators on the motherboard read this code and adjust themselves accordingly. This capability might be used to give the P6 a 10% to 20% edge over conventional designs by letting each processor be tuned to operate at the maximum safe voltage for the particular die it contains.

(More intriguingly, this scheme may give Intel a hardware governor on processor operation: each chip could have its preferred Vcc setting downgraded to the lowest value for which it could still meet its rated frequency. This would also benefit end users, since chips would then not dissipate any more power than what's needed to meet their guaranteed frequency specs.)

Technical Caveats

The whole history of the microcomputer industry is one of vendors promising more than they can deliver, then backing off when the products are introduced. Chips always speed up as they move to smaller processes, but here's a design that may run faster than Intel says with the process technology it's on. Smaller processes should let the P6 run proportionately faster still.

Suppose, for the sake of discussion, the P6 could already run considerably faster than the 150 MHz Intel now admits. Why wouldn't Intel just say so? Maybe technical issues are involved. Maybe the design's not quite there yet; early silicon of a revolutionary new CPU often serves as a proof-of-concept demonstration vehicle, rather than a be-all and end-all product. Maybe a few (or a few hundred) relatively slow nodes currently limit the P6 to "just" 150 MHz. If so, as they are identified and fixed, these bottlenecks may go away.

Or maybe the CPU chip would indeed be able to run faster today, except that the level-2 cache chip currently limits the clock. The L2C chip runs at the same frequency as the CPU core and is designed to retrieve data in three clock cycles, the same as the on-core caches. Intel has never been strong in SRAMs; it's not unreasonable to assume that a 256K L2C would be hard-pressed to match the speed of an 8K cache built on the same 0.6-micron process. Intel is known to be moving the L2C design aggressively to a 0.35-micron process in order to increase its capacity; if cache speed is currently a bottleneck, perhaps a smaller L2C with a tighter process geometry would let the clock run faster still.

Would Intel Lie to You?

But it's more likely—and more fun—to assume that performance is all a marketing game. Pentium is still the king of the hill; recently published evaluations show that the fastest systems based on NexGen Nx586 devices consistently underperform 100-MHz Pentia, even on integer-only code, to say nothing of Intel's 120- and 133-MHz parts. The FPU version of the Nx586 and anticipated onslaughts from the Cyrix M1 and AMD K5 have thus far failed to materialize. And all the various RISC factions, most recently spearheaded by the IBM/Motorola PowerPC, have thus far failed to put much of a dent into the x86-compatible marketplace.

Maybe Intel hasn't claimed the P6 is any faster than it has because it doesn't have to. Too slow a product would underwhelm the market, but an excessively fast P6 could cannibalize Intel's own Pentium sales and spur its competitors to ever-greater levels of inventiveness and achievement. Intel is willing to kill off its own young provided it can do so by selling newer parts instead, but the company surely wants not to Osborne itself by inducing users to stop buying Pentia several months before P6-based systems are ready to take their place.

The goal of any flagship product is to get the market's attention and draw the competitors' fire. At 200 SPECint92—or 220, or 250—the P6 has already succeeded in catching the industry eye. Let others claim to match P6 performance, or discover bottlenecks in existing 16-bit code; Intel may then "discover" another 12%, or 25%, or whatever it takes to stay on top. It may not be until Pentium comes under serious fire from the competition—next year, maybe, or maybe in '97—before we learn what today's P6 design can really do. ♦

John Wharton (jwharton@netcom.com) is the editor and primary author of The Complete x86: The Definitive Guide to 386, 486, and Pentium-Class Microprocessors. *Contact MicroDesign Resources for ordering information on this and other Technical Library reports.*

 © 1995 MicroDesign Resources