Cahners
MICROPROCESSOR REPORT
www.MPRonline.com
THE INSIDER'S GUIDE TO MICROPROCESSOR HARDWARE

TRANSMETA BREAKS X86 LOW-POWER BARRIER

VLIW Chips Use Hardware-Assisted x86 Emulation

By Tom R. Halfhill {2/14/00-01}

Like moths drawn to a flame, semiconductor startups seem to find the bright but dangerous glow of the x86 market irresistible. Never mind that companies as resourceful as AMD, Centaur, Cyrix, IBM, National Semiconductor, and Rise have all charred their wings in the fires of competition with Intel. More than 120 million x86 chips were sold in the profitable PC market last year, casting off a warmth that lures newly hatched companies from the darkness.

The latest newcomer to emerge from its cocoon is Transmeta. After nearly five years of unprecedented secrecy, the Santa Clara–based startup finally unveiled its pair of x86-compatible Crusoe processors at a widely covered media event near Silicon Valley last month. The event received the same sort of overhyped coverage the U.S. Air Force might attract by flinging open the gates to Area 51. A large crowd of mainstream and business journalists were dazzled by marketing claims about "revolutionary" microprocessors that are "part hardware, part software" and that rely on something called "code morphing" to achieve x86 compatibility at amazingly low power levels.

What's behind the hype? Transmeta announced a pair of VLIW chips that have special hardware and software for emulating other instruction sets, and they can also dynamically scale their voltage and core frequencies to conserve power. Revolutionary may be an overstatement, but they are definitely different.

A Fresh Approach to the x86

Transmeta could have followed the well-trodden path of designing chips that clone the devilish x86 architecture entirely in hardware, but that's a technically difficult and legally hazardous endeavor. (Ask AMD and Cyrix.) Instead, the company created a new VLIW architecture with a software envelope that translates x86 binaries into native code at run time.

While some companies have used the term "emulation" to describe the binary-translation process, Transmeta founder Dave Ditzel shuns that term, preferring to describe his company's method of converting x86 instructions into VLIW instructions as "code morphing" or simply "translation." Sometimes this process is called dynamic binary recompilation. Transmeta's code-morphing software certainly is more advanced than old-fashioned emulators, which slowly convert one type of binary executable into another by translating one instruction at a time. Although Transmeta's code-morphing software begins translating a program that way, it gradually identifies sections of frequently executed code in the nonnative program, dynamically recompiles those sections into optimized native instructions, and then caches the native code for reuse. Those techniques greatly improve performance.

Other modern emulation packages use similar techniques, however. Examples include FWB's SoftWindows for the Macintosh and for Unix, Connectix's Virtual PC for the Mac, FX!32 for Alpha, and Sun's HotSpot, a Java just-in-time (JIT) compiler. For the purposes of this article we will sometimes refer to any process of translating one type of binary executable into another at run time as "emulation," on the grounds that optimizing and caching the translated code are performance-enhancing techniques that do not alter the fundamental character of what's going on.

One thing that sets Transmeta's Crusoe processors apart is special hardware to assist emulation—although this too is not a completely novel approach. In 1992, International Meta Systems (IMS) announced the 3250, a microprocessor designed to emulate the x86, 68K, and 6502 architectures by using customizable microcode, among other techniques (see *MPR 5/6/92-03*, "Microcode Engine Offers Enhanced Emulation"). But the 3250 never reached the market, and IMS went up in smoke.

For reasons that are perhaps more legal than technical, Ditzel says the special hardware in Crusoe chips isn't specifically for x86 emulation. Strictly speaking, this is true. The special features should boost the performance of any nonnative executables that Transmeta targets for translation, as well as the performance of native VLIW software. Transmeta has even demonstrated Crusoe processors running Java programs by translating Java bytecodes into native VLIW code. As we'll explain later, though, it's probably more than just a coincidence that Crusoe chips have 80-bit-wide floating-point registers, the ability to perform partial-register writes, and native support for the same data types and single-instruction multiple-data (SIMD) operations found in Intel's MMX extensions. Those features are eerily reminiscent of the x86 architecture, and they will improve Crusoe's ability to run any x86 code the chips should happen to encounter.

Transmeta's most important accomplishment is combining the concept of dynamic binary recompilation with the inherent efficiency and parallelism of VLIW. The result is a pair of x86-compatible processors unlike any others on the market. Beyond that, Crusoe chips appear to achieve three additional milestones:

- Thanks to a unique hardware/software technology called LongRun, the higher-end version of Crusoe can dynamically vary its voltage and clock frequency by monitoring the changing demands of application programs. The processor can scale its performance and power consumption up or down in small increments to conserve power. This innovative technique should greatly improve battery life in mobile systems.
- Crusoe chips appear to set a new standard for low power consumption among x86-compatible processors. Transmeta claims the typical power consumption of its fastest

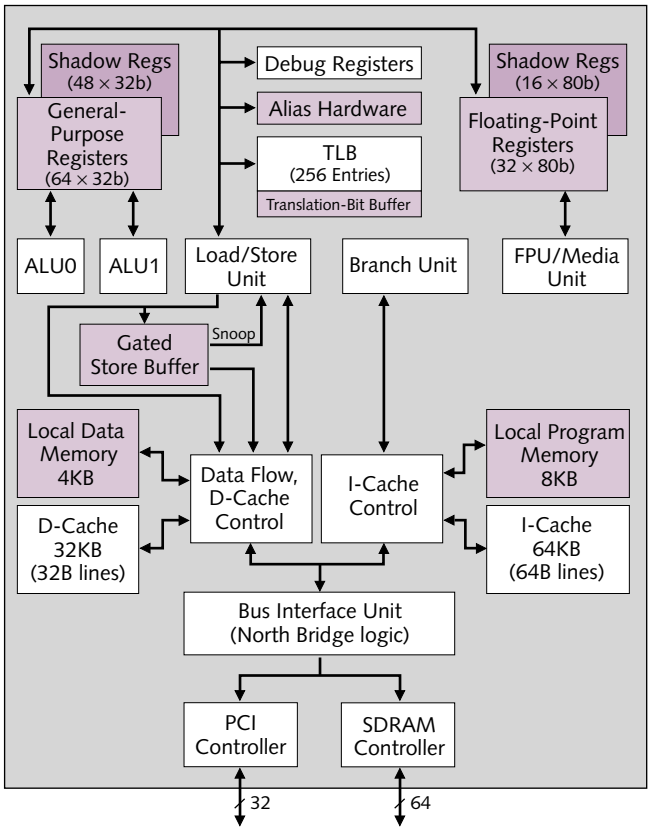

Figure 1. Transmeta's Crusoe TM3120 processor has special features (highlighted in purple) that improve software performance, particularly when translating nonnative code into native VLIW instructions.

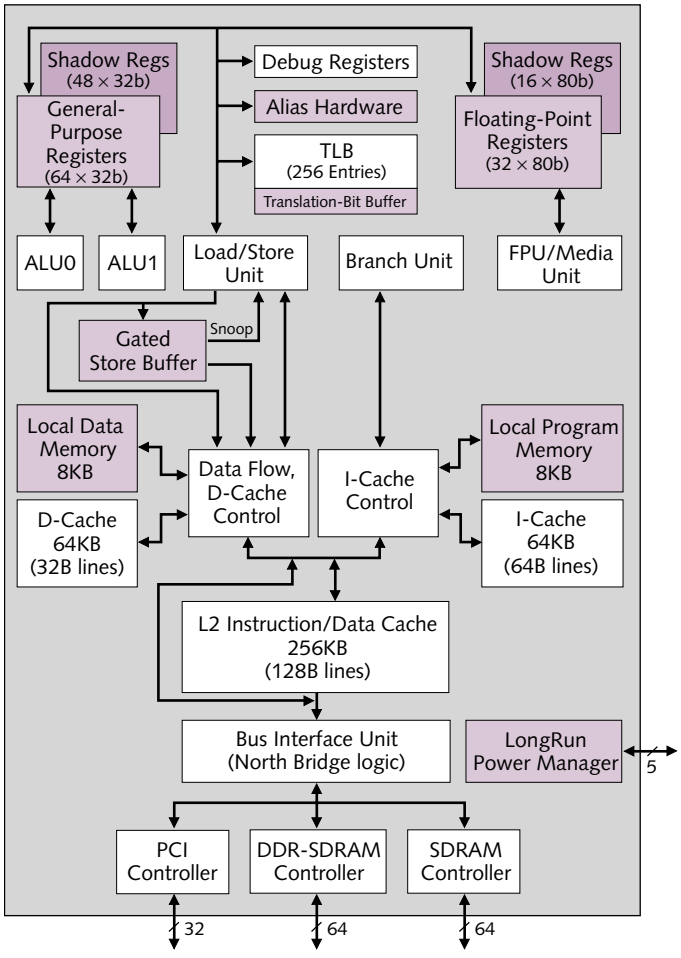

Figure 2. Transmeta's Crusoe TM5400 has some features not found in the lower-end TM3120: an on-chip L2 cache, a DDR-SDRAM controller, LongRun power-management hardware, larger caches, and a slightly different instruction set.

700-MHz chip is only 1–2 W, which is significantly less than the 14–21 W consumed by a 650-MHz Mobile Pentium III. These claims haven't been independently verified yet, but if they're remotely accurate, Transmeta can exploit a key vulnerability of the hand-me-down desktop chips that Intel and AMD sell into the mobile market.

- Crusoe processors appear to sacrifice much less performance to emulation than other software translators. According to Ditzel there's actually no translation overhead at all—Crusoe chips are slower than similarly clocked x86 chips merely because Transmeta optimized the cores for low power consumption, not performance. The company says a 700-MHz Crusoe is about 70% as fast as a 700-MHz Pentium III. If that claim proves accurate, Crusoe's unusual approach to x86 compatibility subtracts less performance than might be expected.

In short, Transmeta doesn't need revolutionary technology or media hype to succeed. Low power consumption and adequate performance should be enough to secure Transmeta a profitable future in the competitive x86 market—if the fledgling company can deliver everything it promises.

Two Chips for Two Markets

Transmeta has announced two versions of Crusoe: the TM3120, which will be available in speed grades of 333, 366, and 400 MHz, and the TM5400, which will be available in speed grades ranging from 500 to 700 MHz. Both chips are sampling now, and the TM3120 is in production. The TM5400, a later design that improves on the TM3120, is scheduled for production in 2Q00. Prices range from $65 to $89 for the TM3120, and from $119 to $329 for the TM5400.

IBM Microelectronics will manufacture both processors as part of a foundry arrangement that gives Transmeta valuable patent protection, because IBM has a comprehensive patent cross-licensing agreement with Intel. IBM is packaging the chips in 474-contact BGAs. Because the TM3120 and TM5400 have different bus structures, the TM3120 has about 80 unused pads in this package, but Transmeta says the common pinout makes it easier for customers to design boards that work with either chip. The packages can be soldered onto the motherboard or mounted in a new type of socket from FormFactor/Tyco Electronics. The socket works with BGA and LGA chips and has a lower profile than the soldered MBGA (micro-BGA) packages favored by Intel.

Transmeta designed the two Crusoe chips for quite different markets. The TM3120 is for mobile Internet appliances, while the TM5400 is for Windows-compatible notebook PCs. The latter market is well established, and OEMs should welcome a new x86-compatible CPU that consumes only a few watts. The market for mobile Internet appliances, while nebulous today, is expected to be a fast-growing market in the future.

The TM3120, seen in Figure 1, is the lower-end chip. It has 96K of primary cache, divided into a 64K instruction cache and a 32K data cache, both eight-way set-associative. There is no on-chip secondary cache. The TM3120 has an integrated PCI controller, an SDRAM controller, and other components of a traditional north bridge; therefore Transmeta won't have to rely on other companies to provide compatible core logic, and the processor gains the power-consumption and performance efficiencies of an on-chip memory controller. IBM is building the TM3120 in its 0.22-micron copper CMOS-7S process. This process allows up to six metal layers, but the TM3120 uses only five. The chip's die area is 77 mm^2.

The TM5400, seen in Figure 2, is the higher-end model. It has 128K of primary cache, evenly divided into an 8-way set-associative instruction cache and a 16-way set-associative data cache. (The TM5400's data cache has twice the set-associativity of the TM3120's data cache in order to keep each set at 4K, which matches the page size of the x86.) The TM5400 also has a four-way set-associative 256K secondary cache on chip, which should improve performance over the TM3120. Like its fellow Crusoe chip, the TM5400 integrates a PCI controller, an SDRAM controller, and north-bridge functions, but it also supports double-data-rate (DDR) SDRAM on a separate bus. IBM is building the TM5400 in its 0.18-micron copper CMOS-8S process. This process allows up to seven metal layers, but the TM5400 uses only five. The chip's die area is 73 mm^2.

One significant difference between the TM3120 and the TM5400 is that the latter chip has the LongRun voltage/frequency-scaling hardware, which allows the 700-MHz TM5400 to consume only a little more power

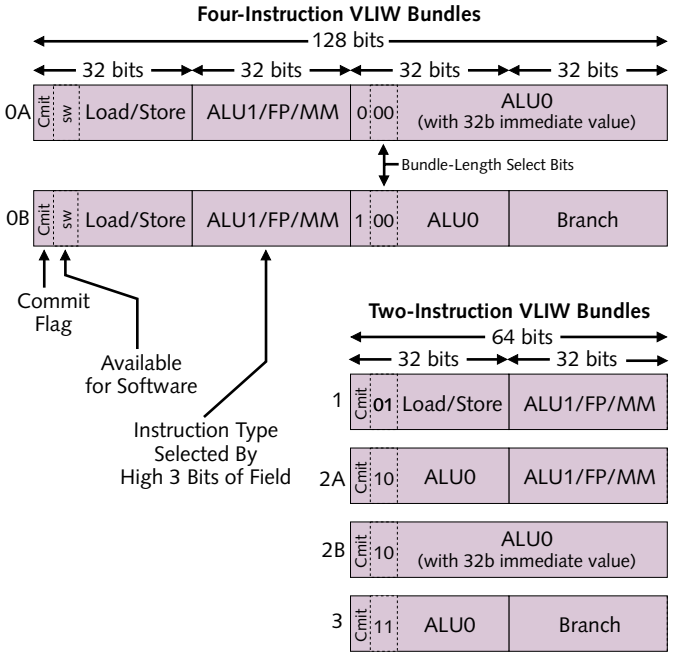

Figure 3. There are six possible types of VLIW bundles. They vary by length and by the arrangement of subinstructions within the bundles.

than the 400-MHz TM3120. It's likely that all future Crusoe processors will incorporate LongRun.

In most other ways, the two chips are similar. Both are VLIW machines that execute 64- or 128-bit instruction bundles, with each bundle containing two or four 32-bit subinstructions, as shown in Figure 3. Transmeta refers to the VLIW bundles as "molecules" and to the subinstructions as "atoms." If the translation software can't fill all the subinstruction slots in a molecule with useful atoms, it pads the empty slots with NOPs (null operations). By allowing two types of molecules—one with two atoms and one with four atoms—Transmeta ensures that no molecule ever has to carry more than one NOP. This minimizes the space and fetch bandwidth wasted by NOPs, which are the Achilles' heel of traditional VLIW architectures.

Both Crusoe chips have two integer ALUs, a load/store unit, a branch unit, and an FPU/multimedia unit, allowing them to execute up to four VLIW subinstructions per cycle. Their execution pipelines are identical, as shown in Figure 4. Both chips also have 64 general-purpose registers (GPRs) that are 32 bits wide and 32 floating-point registers (FPRs) that are 80 bits wide. Although the FPU/multimedia unit can handle the same data types as Intel's MMX instructions, Crusoe chips don't have the new 128-bit registers defined by Intel's SSE (Streaming SIMD Extensions). Transmeta says Crusoe could emulate SSE-type instructions and registers, but there's not enough software support for SSE to justify the effort at this time. Crusoe doesn't support AMD's 3DNow! extensions, either.

The architectural features of Crusoe chips are of little importance to software developers, because nobody except Transmeta writes software for the native architecture. To operating systems, development tools, and applications, the chips look like regular x86 processors. Transmeta has no plans to develop native VLIW applications or to encourage others to do so. In fact, Transmeta discourages native software development, because the company wants the freedom to change the architecture from one processor generation to another, or even from one chip to another in the same generation.

Indeed, the TM5400's instruction set has some improvements over that of the earlier TM3120, so their VLIW binaries are not 100% compatible. Future Crusoe processors may introduce completely different instruction sets or architectures. It doesn't matter to users or to software developers, because the only software that runs natively is Transmeta's own code-morphing software, which runs transparently below the operating system.

The first publicly announced Transmeta customer was S3, which is working on a "Web pad" appliance based on a 400-MHz TM3120. The prototype has a 10.4-inch color LCD screen, a hard disk drive, and wireless network connectivity. It weighs 2–3 pounds with a rechargeable battery and runs Mobile Linux, a special version of Linux designed by Transmeta. (Interestingly, Transmeta says it hired Linux creator Linus Torvalds primarily for his programming prowess and that his work on Mobile Linux was a relatively minor part of his job.)

S3 plans to introduce the Web pad later this year at a retail price of $500–$1,000, although it could cost less if subsidized by an Internet service provider. S3's chief technology officer, Andy Wolfe, says his engineers chose the Crusoe processor over alternative x86 chips because of its low power consumption, which extends battery life and eliminates the need for a cooling fan. Some embedded RISC chips met the same power and performance requirements, but they aren't x86 compatible, so they can't run the growing number of Web-browser plug-ins that Wolfe considers vital for the product.

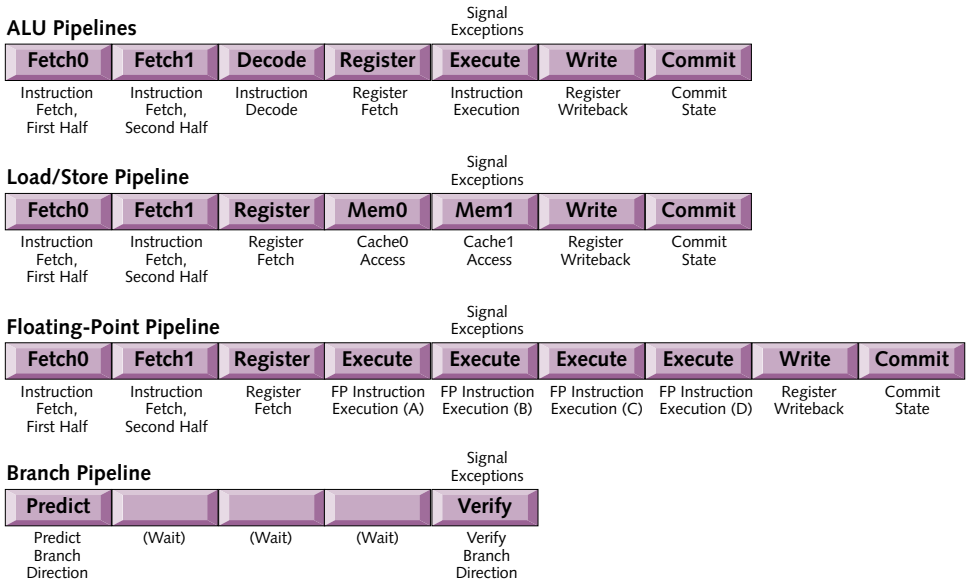

Figure 4. Crusoe processors have shorter pipelines than most x86-compatible processors.

LongRun Extends Battery Life

The TM5400's LongRun feature is one of the most innovative technologies introduced by Transmeta. To our knowledge, no other microprocessor can conserve power by scaling its voltage and clock frequency in response to the variable demands of software. Intel's SpeedStep—announced the day before Transmeta's coming-out

party—is crude in comparison. It merely steps down the CPU clock and supply voltage when a user unplugs a mobile computer from AC power, and it has only one voltage/frequency step (from 650 or 600 MHz to 500 MHz).

LongRun can scale the CPU's voltage in as many as 32 steps, though in practice 5–7 steps are sufficient to achieve most of the benefits, according to Transmeta engineers. There are individually controllable, codependent ranges for voltage and frequency. In the current version of the TM5400, voltage can vary from 1.1 V to 1.6 V, and frequency can vary from 200 MHz to 700 MHz in increments of 33 MHz. Transmeta's software controls the scaling through a five-pin interface that adjusts an off-chip voltage regulator, as seen in Figure 2.

When the LongRun software detects a change in the CPU load, it signals the chip to adjust the voltage and frequency up or down. (It's not exactly clear how LongRun measures CPU loads; even an idle loop can make a CPU seem very busy. This function may be related to x86 emulation, because Transmeta's code-morphing software constantly monitors software execution to control binary recompilation, as explained later in this article.) If the CPU needs to handle a heavier load, LongRun tells the chip to start ramping up its voltage. When the voltage stabilizes at the higher level, the chip scales up its clock frequency.

If the LongRun software determines that the CPU can save power by running more slowly, the chip starts scaling down its frequency. When the phase-lock loop (PLL) locks onto the lower clock rate, LongRun reduces the voltage. By always keeping the clock frequency within the limits required by the voltage, LongRun avoids any clock skewing or other undesirable effects.

LongRun never needs more than one frequency step to reach a different target. To scale from 600 to 700 MHz, for instance, LongRun doesn't have to take three 33-MHz steps. Instead, it raises the voltage to 1.6 V in multiple steps, then boosts the frequency to 700 MHz in one big jump. This avoids the latencies of resetting the frequency multiple times.

One concern is that LongRun might not react quickly enough to accommodate the fast-changing demands of some programs. When the computer is playing MPEG-compressed video, for example, a transition from a relatively static frame to an action-filled frame might overwhelm a CPU that's comfortably loafing at a low clock speed. MPEG compression works by saving the differences between frames, and the frames are only 1/30 of a second apart. The CPU load would vastly increase after a sudden transition from an Al Gore speech to a car chase.

Not to worry, says Transmeta. The company claims its software can detect a change in the CPU load in about half a microsecond, and LongRun can scale the voltage up or down in less than 20 microseconds per step. The worst-case scenario of a full swing from 1.1 V to 1.6 V and from 200 to 700 MHz takes only 280 microseconds.

Furthermore, Transmeta says, the CPU doesn't stall during the swing, as a mobile Pentium III does during a SpeedStep adjustment. The processor keeps executing instructions, stalling only while the PLL relocks onto the new frequency. That doesn't take longer than 20 microseconds in the worst case, and Transmeta's engineers say they've never observed a relock taking longer than 10 microseconds.

LongRun isn't the only reason that Crusoe processors appear to consume much less power than comparable x86 chips. The TM3120 doesn't have LongRun, yet its power consumption is impressive too. The simplicity of Transmeta's VLIW architecture is evidently a larger factor. Some embedded RISC chips are even more efficient—Intel's second-generation StrongARM, which will ship in 2H00, is expected to consume only 450 mW (typical) at 600 MHz. But when x86 compatibility matters, Crusoe is the front-runner, and LongRun is a genuine innovation that gives Crusoe an extra edge.

Technology Drives Transmeta's Strategy

Intel and AMD are more vulnerable in the mobile market because their cores are not primarily designed for low power consumption. Typically, Intel and AMD repackage their desktop processors as mobile processors when shrinking process technology and aggressive power management reduce power consumption into the range considered acceptable for notebook PCs. Transmeta's cores are better suited for mobile applications because their relatively simple VLIW architectures are more efficient than the x86 architecture. To counter Transmeta's claimed power-consumption advantage, Intel and AMD would have to introduce new x86 cores specifically designed for low power—a costly undertaking that probably only Intel could justify. VIA's future WinChips might give Crusoe some competition, but we don't know enough about their power consumption to draw any conclusions.

Pursuing mobile PCs instead of the larger desktop market also relieves some pressure on Transmeta. As other companies have discovered, performance is paramount in the desktop segment. An x86 vendor must match or exceed Intel's clock frequencies to maintain profitable selling prices, and Intel has enormous advantages in engineering resources, fabrication technology, and manufacturing capacity. Even a company as large as AMD has trouble keeping up.

Transmeta doesn't rule out introducing future processors aimed at the desktop or even the server markets. Ditzel says there's no inherent penalty for translation, so a future Transmeta processor with code-morphing software could compete with native x86 chips on raw performance. If his claim is true, then Transmeta may indeed have advanced the art of emulation beyond anything seen before.

Traditionally, the overhead of emulating another CPU architecture in software is highly variable. In terms of clock cycles, it's generally about 10 to 1. By using such techniques

as caching and optimized recompilation, modern emulators can reduce that ratio to 4 to 1 in some cases. Transmeta says its 700-MHz TM5400 delivers about the same performance as a 500-MHz Pentium III on industry-standard application-level benchmarks, which is significantly better than 4 to 1. S3's Andy Wolfe says a 400-MHz TM3120 runs software on Mobile Linux about as fast as a 333- or 400-MHz Intel Celeron runs comparable software on Windows 98—although he acknowledges that Linux, not just Transmeta's code-morphing software, is a major factor.

For reasons explained below, Transmeta's performance claims are difficult to verify independently because of the way modern emulators work. The important point is that Transmeta's current processors are better suited to mobile and embedded applications, where power conservation matters more than raw performance.

The keys to a low-power design are minimizing the amount of switching logic and shrinking the die. Transmeta's choice of a VLIW-based architecture makes sense, because VLIW can, theoretically, extract lots of instruction-level parallelism from program code without the complex control logic that burdens superscalar RISC and CISC processors. Crusoe chips are admirably small; see the die photos in Figure 5 and Figure 6. (Transmeta hasn't released transistor counts for either chip.) The inherent efficiency of VLIW has made it the darling of CPU architects, superseding RISC as the design pattern for most new architectures. In recent years, VLIW variants have been announced by Analog Devices, Equator, Fujitsu, Hewlett-Packard, Intel, Philips, StarCore, Sun, and Texas Instruments. All were inspired by the pioneering work of Cydrome, Culler, and especially Multiflow in the 1980s (see *MPR 2/24/94-05*, "VLIW: The Wave of the Future?").

Transmeta's focus on the x86 makes sense, too—at least for Crusoe chips aimed at the PC market, where x86-based Windows software reigns supreme. In the embedded market, the x86 has been trampled by RISC architectures, such as ARM, MIPS, and SuperH. But that's largely because RISC chips are more power-efficient than x86 chips with comparable performance, and also because there's no dominant x86 software base.

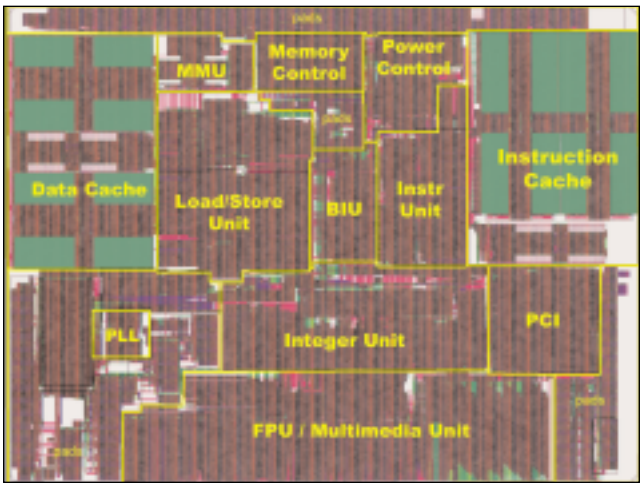

Figure 5. The Crusoe TM3120's die size is 77 mm². Note the PCI controller and memory controller; the north bridge is also on chip.

Crusoe could alter that picture. The chips appear to be competitive with the performance/power-consumption ratios of many embedded RISC chips. And because they are x86-compatible, they can run Web-browser plug-ins and other software that would require more extensive porting to run on RISC processors. That could be an important advantage for newfangled information appliances like S3's Web pad.

In that regard, Transmeta's strategy is similar to National Semiconductor's. Last year, National announced the x86-compatible Geode SC1400 for the embedded market and also demonstrated a prototype Web pad (see *MPR 8/2/99-03*, "National Unveils 'Appliance On a Chip'"). The SC1400 is more highly integrated than Crusoe. It incorporates not only the north-bridge logic, PCI controller, and SDRAM controller, like Crusoe, but also a VGA-graphics accelerator, an MPEG audio/video processor, a video-overlay processor with NTSC and PAL outputs, an IDE interface, a three-port USB interface, and numerous UARTs and general-purpose I/O ports.

Transmeta's TM3120 should be faster than the SC1400, which is built around an old 486-class Cyrix core that runs at 233–266 MHz and delivers about the same performance as a fast Pentium. Power-consumption comparisons are difficult because National hasn't publicly released those figures for the SC1400, and also because Crusoe processors need additional logic to match the higher integration of the SC1400. But Crusoe has one big advantage: a deep-sleep mode that consumes less than 20 mW. The SC1400 conspicuously lacks a deep-sleep mode. National needs to upgrade its Geode chips to stay competitive with upstarts like Transmeta.

Decoding x86 Binaries in Software

Transmeta's claim that Crusoe is "part hardware, part software" isn't merely hype. The code-morphing software performs the x86-instruction decoding that all other x86 processors implement in hardware. The "decoding," in this case, actually involves translating x86 instructions into native VLIW instructions. Although competitors will probably attack this unusual approach to x86 compatibility, there's no technical reason that it can't it work as reliably as traditional approaches.

Current Intel and AMD x86 processors convert x86 instructions into RISC-like micro-ops that are simpler and easier to handle in a superscalar microarchitecture. (In contrast, Cyrix and Centaur cores execute x86 instructions directly.) The micro-op translation adds at least one pipeline stage and requires the decoder to call a microcode routine to

translate some of the most complex x86 instructions. Implementing the equivalent of that front-end translation in software saves Transmeta a great deal of control logic and simplifies the design of its chips. It also allows Transmeta to patch some bugs in software. (The engineers fixed a timing problem in the TM5400 in this manner.) Some x86 chips, such as Pentium III, allow some patches to microcode, but these patches are very limited in comparison.

Transmeta's software translation is a little more like the Motorola 68K emulation built into PowerPC-based Macs since 1994. The Mac OS translates ("decodes") 68K instructions into PowerPC native instructions at run time, allowing Power Macs to transparently run older Mac software. This translation works extremely well, and 68K programs actually run faster under emulation on almost all Power Macs than they do on the fastest 68K Macs (thanks to higher clock frequencies). FWB's SoftWindows and Connectix's Virtual PC for the Mac translate x86 binaries into PowerPC instructions, and those translators work well too. Software emulation is a proven concept, dating back at least as far as 1964, when IBM's new System/360 provided emulation for IBM's older 1401 computers.

What's new about Transmeta's approach is that translation isn't merely an alternative to native execution—it's the whole strategy. Crusoe does for microprocessors what Java does for software: it interposes an abstraction layer that hides internal details from the outside world. Just as a Java programmer can write code without needing any knowledge about the underlying operating system or CPU, x86 programmers can continue writing software without needing any knowledge about a Crusoe system's VLIW architecture or code-morphing software.

x86-Flavored VLIW

Crusoe processors have several features that assist the code-morphing software, and some of the features are unmistakably x86-specific. One example is the register files, which have 160 physical registers. These include 64 GPRs with 48 shadow registers and 32 FPRs with 16 shadow registers. The GPRs are 32 bits wide and support partial-register writes, just like real x86 registers. The FPRs are 80 bits wide, so they can directly support x86 extended-precision floating-point operations. (Compaq's FX!32 emulator for Alpha requires extra steps to support 80-bit math in x86 programs because Alpha's FPRs, like those in most RISC processors, are only 64 bits wide.)

The translation software uses the shadow registers to checkpoint the speculative state of the CPU while programs alter the contents of the working registers. This allows Crusoe to execute instructions speculatively and out of order. If an exception occurs, the processor can roll execution back to the most recently committed state of the machine by copying the contents of the shadow registers into the working registers. This also preserves the precise exception model of the x86 (such as it is).

VLIW bundles can include a one-bit "commit" flag that tells the processor to commit the working state of the GPRs and FPRs by copying their contents to the shadow registers. Typically, the translation software commits the results of a block of code before proceeding to the next block of code. In no case will the processor commit results until it has resolved all conditions that might trigger an exception.

One example of a condition that might cause an exception is a sequence of instructions that contains a memory load or store operation. The translation software can reorder loads and stores, moving them higher in the instruction stream to hide memory latencies. Crusoe processors have special alias hardware (see Figures 1 and 2) that watches for subsequent instructions that access memory locations whose data is already loaded in a register. If an instruction tries to access one of those memory locations, the alias hardware raises an exception, and the processor rolls back to the last committed state. The translator then continues forward from that point without using such aggressive optimizations.

The translator can also eliminate redundant loads. If no instructions alter the contents of a memory location between loads, the translator detects the redundancy, skips the unnecessary loads, and uses the data from the first load. This can happen frequently in architectures with as few registers as the x86.

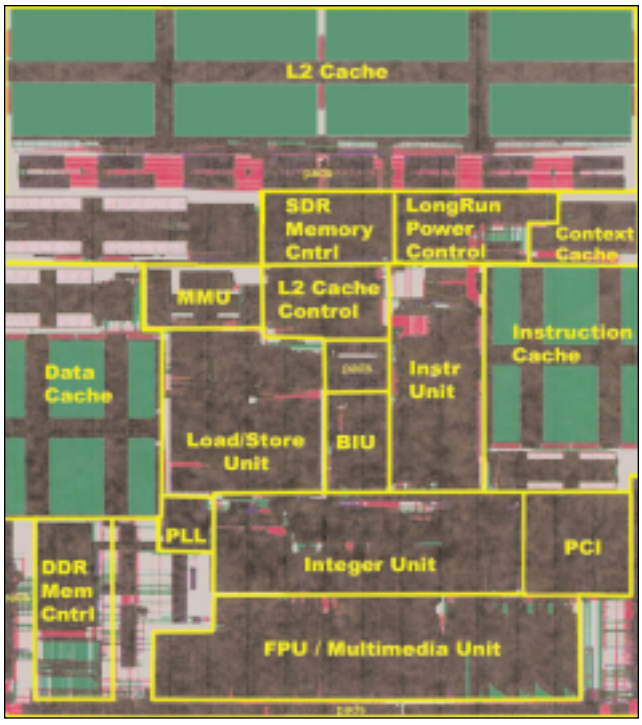

Figure 6. The Crusoe TM5400's die size is 73 mm^2. In addition to a PCI controller, memory controller, and north bridge, it also has 256K of on-chip L2 cache.

Another mechanism that protects the state of the machine is a gated store buffer, also shown in Figures 1 and 2. The processor temporarily holds the results of all store operations in this 32-entry buffer until an instruction commits the next block of results. At that point, the gate opens and the processor writes the results back to memory.

To solve one thorny problem that rankles x86 architects—self-modifying code—the MMU has a special translation-bit buffer in the TLB. When the translator converts a block of nonnative code into native instructions, it write-protects the memory containing the nonnative code by setting a bit in the translation buffer. If a program tries to overwrite the protected memory, the processor can invalidate the translated code, allow the program to change the original code in memory, and then retranslate the modified code.

Crusoe processors also have special caches, independent of the primary instruction and data caches, to hold critical pieces of the translation software's VLIW code and data structures in fast on-chip SRAM. The TM5400 has 8K of local program memory and 8K of local data memory; the TM3120 has 8K of local program memory and 4K of local data memory. The translation software manages these caches directly, so they're not flushed by misses on the regular primary caches.

Parlez-Vous x86?

In other respects, Transmeta uses techniques similar to those in other modern emulators and JIT compilers. The translator dynamically analyzes the run-time behavior of a program, interprets code that the program executes infrequently, recompiles and caches code that the program executes often, and applies common compiler optimizations whenever practical.

When a Crusoe system boots, it immediately reserves a 16M block of main memory, though Transmeta says 8M is acceptable for systems with less RAM. Next, the translator loads from ROM into the reserved memory, occupying about 512K. The rest of the 16M block is set aside as a translation cache, where the translator stores recompiled code for later use. Enlarging this cache beyond 16M contributes little to performance with the kind of software that users tend to run on notebook computers and information appliances, according to Transmeta.

The reserved memory is invisible to the BIOS, the operating system, and the application programs. From the user's point of view, that memory never existed—so a system with 64M of RAM would appear to have only 48M. OEMs determine the size of this memory block when they configure a system at the factory, and normally it's immutable. Some emulators, such as SoftWindows, allow users to change the size of the translation cache to tweak performance. But those emulators run on top of the host operating system, not beneath it as Transmeta's does.

In practice, Transmeta's code-morphing software works much like a Java JIT compiler, and especially like Sun's HotSpot JIT compiler. The translator starts out conservatively, interpreting x86 instructions one at a time without necessarily storing the converted code in the translation cache. There's nothing to gain by caching a program's initialization routine, for instance, because that code executes only once. (An exception might be a large initialization loop that could benefit from caching.) Straight interpretation requires at least 12 clock cycles per x86 instruction.

As the x86 program continues to run, the translator monitors the program's behavior, noting which code is frequently executed. In effect, the translator performs the same analysis as the code profilers used by programmers to identify critical sections in their programs—except the translator does it at run time, gathering information from real users, based on their actual working patterns.

When the translator identifies a frequently executed section of code, it schedules that code for compilation and perhaps for optimization. There are several degrees of possible optimization. Like Sun's HotSpot, the translator estimates how much time it needs to optimize the code and balances that time against the execution time it's likely to save. If the equation makes sense, the translator recompiles the x86 code into VLIW code and stores it in the translation cache. One difference between Transmeta's translator and Sun's HotSpot is that Transmeta can optimize one or more basic blocks of code within subroutines, whereas HotSpot optimizes only whole subroutines (Java methods).

Many of the possible compiler optimizations the translator applies are familiar: loop unrolling, common subexpression elimination, loop-invariant code removal, and so on. Some are x86-specific: the translator can skip instructions that redundantly set x86 condition codes. And some are VLIW-specific: the translator can combine multiple x86 basic blocks into a single VLIW basic block by removing unnecessary branches.

The Devil in the Details

Some code expansion is inevitable when converting CISC binaries into VLIW. The code can expand in two ways: by increasing the number of instructions required to do the same work and by translating the compact x86 instructions into longer VLIW equivalents.

Transmeta's technical white paper on compiler optimization shows an example of 20 x86 instructions collapsing into 10 VLIW instructions. But the 10 VLIW instructions are really VLIW bundles (molecules) that each contain two or four RISC-like subinstructions (atoms). Actually, the translator needs 23 VLIW subinstructions to do the work of these 20 x86 instructions.

Moreover, the example doesn't show that some VLIW bundles contain NOPs wherever the translator couldn't fill slots with useful subinstructions. After restoring the missing NOPs, the translated VLIW code contains a total of 32 subinstructions—50% more than the original x86 code, as shown in Figure 7.

The actual code expansion is worse than that, because subinstructions are always 32 bits long. In the x86 architecture, instructions can vary in length from 8 to 120 bits, and the average length is usually estimated at 16–24 bits. So converting x86 code into VLIW may expand the code 33–100%, even if the instruction ratio is the same. If the white-paper example is typical, add another 50% for the extra subinstructions and NOPs required to do the same work as the x86 code.

Transmeta says the code expansion seems less drastic if the subinstructions are compared to the RISC-like micro-ops that Intel and AMD processors use internally, after the decoders break down x86 instructions into their component operations. This is true, because most of the time, x86 instructions fracture into multiple micro-ops, just as they translate into multiple VLIW subinstructions. That's a fair comparison for the purpose of explaining instruction execution, but it doesn't address the important issues of code expansion. The Intel and AMD micro-ops exist only inside the processor's pipelines; they don't occupy space in memory and caches, and they don't consume I/O bandwidth. In contrast, Transmeta's VLIW subinstructions take up space in main memory (in the translation cache) and in the instruction caches, and the processor has to fetch the subinstructions over the I/O bus, even if they're do-nothing NOPs.

The degree of code expansion caused by translation is not a serious flaw, but it does reduce the effective sizes of the caches in comparison with those of other x86 processors. A 64K instruction cache in a Crusoe chip isn't quite as large as it seems, and neither is the 16M translation cache. If code expansion causes the processor to flush the translation cache more often to make room for newly recompiled code, it's more difficult to amortize the clock cycles spent on recompilation and optimization. This depends on software-usage patterns—switching among multiple applications is less efficient than using one application at a time. Of course, this affects the caches of all CPUs, but it's even more true for Crusoe.

Enlarging the translation cache is one solution, but this cache is already pretty large. Subtracting 16M of RAM from main memory is acceptable for notebook PCs with at least 64M, but it could be troublesome for low-end information appliances with less memory. Paging some of the recompiled code to disk is another possibility, but losing clock cycles to a page fault might be worse than recompiling the x86 code again. Fortunately, RAM is getting cheaper all the time. On balance, users will probably consider the "lost" RAM a worthwhile tradeoff for additional battery life. Still, it demonstrates once more that there's no such thing as a free lunch. Transmeta's technology doesn't rewrite the rules that all CPUs must obey.

Transmeta also faces some of the criticism aimed at other VLIW architectures, particularly with regard to its code-morphing software. The performance of VLIW processors depends heavily on the scheduling efficiency of their compilers, because they don't have the dynamic reordering hardware found in superscalar RISC processors. The compiler is responsible for scheduling instructions.

When they designed IA-64, Intel and Hewlett-Packard started with the back end of Multiflow's VLIW compiler, and they have have been refining it for five years. Transmeta started from scratch, presumably without a proven back end. Moreover, an IA-64 compiler starts with source code and works statically—a programmer can let the compiler churn overnight, if necessary, to extract the most parallelism. Transmeta's VLIW compiler must translate and optimize an x86 binary file while the program is

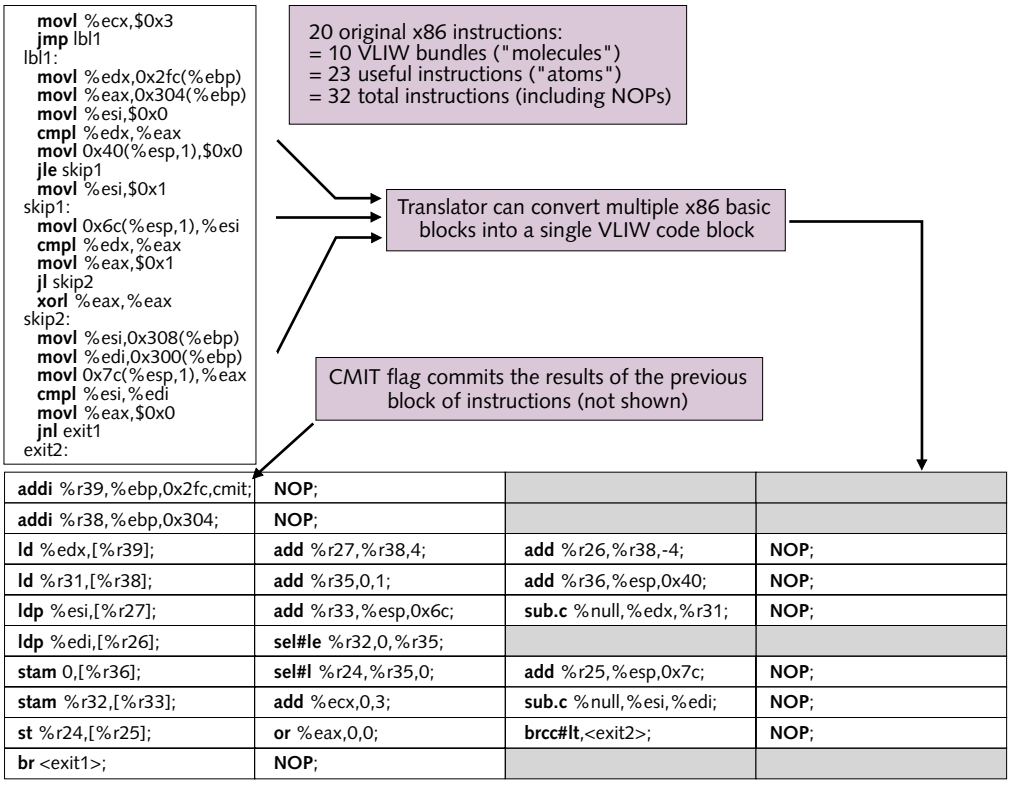

Figure 7. This example, adapted from a Transmeta white paper, shows how the translation software recompiles x86 code into native VLIW code.

Price & Availability

Transmeta's Crusoe TM3120 is in production now and will be available at 333, 366, and 400 MHz. The Crusoe TM5400 is sampling now and is scheduled for production in 2Q00; it will be available at speed grades ranging from 500 to 700 MHz. Prices range from $65 to $89 for the TM3120 chips, and from $119 to $329 for the TM5400 chips. For more information, go to *www.transmeta.com*.

running. The x86 code has already been compiled once, and it may contain optimizations for specific x86 microarchitectures that have nothing to do with Crusoe's microarchitecture.

One factor in Transmeta's favor is that the translator monitors actual usage patterns at run time—dynamic compilers get better feedback than static compilers. Some other software emulators and Java JIT compilers use these same techniques to great effect, so Transmeta hasn't wandered into unexplored territory. But it's worth noting that the quality of Transmeta's run-time translation will greatly influence the x86 compatibility and performance of Crusoe processors. And users will have to be satisfied with Transmeta's translation software, because it will be available only from Transmeta.

Waiting for Real-World Results

It's difficult to judge Transmeta's accomplishments, because after stripping away the hype, many unconfirmed claims remain. At this writing, there are no independent test results of Crusoe's compatibility, performance, software stability, or power consumption. Indeed, such results will be hard to come by, because the very nature of Crusoe's technology defies conventional benchmarking.

Consider the problem of measuring performance. Benchmark programs that rely too heavily on repetitive loops will overestimate the speed of Crusoe's x86 execution, because the translator will quickly recognize the program's behavior, recompile the loops with optimizations, and execute native code out of the translation cache. The results will be accurate only for users who perform the same small set of repetitive tasks all day. (OK, there are some of you out there.)

Conversely, benchmark programs, such as Winstone, that drive real applications with automated scripts may underestimate Crusoe's performance. The scripts run through a series of tasks and switch among applications at much higher speeds than any real user would, so the translator rarely gets a chance to amortize the cost of compiling and optimizing the code.

Battery-life tests have similar flaws. Unless they mirror real-world usage very closely, they won't yield results that average users can expect from Crusoe-based systems. New benchmark suites must be written to take all these factors into account, as Transmeta pointed out during its public announcement.

We're not too worried about the accuracy of Crusoe's x86 emulation. There are companies that do nothing but test for x86 compatibility, so any bugs should be discovered and fixed early. Also, emulation isn't the experimental technology it used to be. Java JIT compilers and Windows emulators have significantly advanced the art in recent years. A little searching on the Web will turn up dozens of emulators for all kinds of platforms, including emulators for some otherwise-dead machines such as the Apple II, Atari 2600, and Commodore 64. If Transmeta had built an x86 processor in the currently fashionable manner—that is, with hardware decoders that convert x86 instructions into native micro-ops—the designers would have written similar translation code, anyway, except they would have written it in Verilog instead of in C or assembly language. Even if Transmeta's code morphing does have some glitches, they're easily patched in software.

One cause for worry is Transmeta's pricing strategy, which appears to bet heavily that Crusoe-based notebook PCs will deliver enough additional battery life to offset a price/performance gap relative to Intel's mobile processors. Currently, a top-of-the-line 650-MHz mobile Pentium III costs $637—almost twice as much as the 700-MHz TM5400, which costs $329. But direct comparisons are invalid, because neither chip is as fast as its clock speeds imply. With SpeedStep, the 650-MHz mobile Pentium III actually runs at only 500 MHz on battery power. If we accept Transmeta's claim that the 700-MHz TM5400 is as fast as Pentium III at 500 MHz, then Crusoe has a worse price/performance ratio. A 500-MHz mobile Pentium III without SpeedStep currently costs only $294, about 10% less than the 700-MHz TM5400. The 650-MHz Pentium III with SpeedStep costs a lot more, but it delivers at least 30% more performance on AC power than the TM5400. These price/performance ratios are likely to change in Intel's favor as time goes by, because Intel will probably cut prices once or twice before Transmeta ships the TM5400 in 2Q00.

Apparently, Transmeta is betting that Crusoe will deliver enough extra battery life to offset the price/performance disadvantage and the user's loss of 16M of system memory (for the translation cache). Transmeta promises to make "all-day computing" a reality. Even if the TM5400 consumes an order of magnitude less power than Pentium III, users won't get an order of magnitude more battery life, because the CPU in a notebook PC doesn't consume 100% of the system's power. Estimates of the CPU's share range from 20% to 75%, with the rest consumed by the LCD screen, disk drive, and other components. (When making these comparisons, remember that Crusoe integrates a north bridge, whereas Intel and AMD processors require extra chips for those functions.) We feel confident that a Crusoe-based notebook PC will

run longer on battery power, but if it's only a couple of hours longer, it might not be enough to make Transmeta's pricing strategy work.

But then, competing against Intel is always a risky gambit. Previous attempts to find chinks in Intel's armor—such as Rise's low-power processors and Centaur's simplified WinChips—have proved time and again that Intel is a formidable foe. In the embedded market, Crusoe faces similarly tough competition against RISC chips, but x86 compatibility is a major differentiation that will win some customers for that reason alone.

As we learn more about Transmeta's technology and get samples of the chips for hands-on testing, we'll follow up with more detailed analysis. Until then, if Transmeta can fulfill all its promises, we think Crusoe chips will find profitable niches in the PC-notebook and embedded markets.