

1. INTRODUCTION

Mobile computing has been the buzzword for quite a long time. Mobile computing devices like laptops, webslates & notebook PCs are becoming common nowadays. The heart of every PC whether a desktop or mobile PC is the microprocessor. Several microprocessors are available in the market for desktop PCs from companies like Intel, AMD, Cyrix etc. The mobile computing market has never had a microprocessor specifically designed for it. The microprocessors used in mobile PCs are optimized versions of the desktop PC microprocessor. Mobile computing makes very different demands on processors than desktop computing, yet up until now, mobile x86 platforms have simply made do with the same old processors originally designed for desktops. Those processors consume lots of power, and they get very hot. When you're on the go, a power-hungry processor means you have to pay a price: run out of power before you've finished, run more slowly and lose application performance, or run through the airport with pounds of extra batteries. A hot processor also needs fans to cool it; making the resulting mobile computer bigger, clunkier and noisier. A newly designed microprocessor with low power consumption will still be rejected by the market if the performance is poor. So any attempt in this regard must have a proper 'performance-power' balance to ensure commercial success. A newly designed microprocessor must be fully x86 compatible that is they should run x86 applications just like conventional x86 microprocessors since most of the presently available software's have been designed to work on x86 platform.

Crusoe is the new microprocessor which has been designed specially for the mobile computing market. It has been designed after considering the above mentioned constraints. This microprocessor was developed by a small Silicon Valley startup company called Transmeta Corp. after five years of secret toil at an expenditure of \$100 million. The concept of Crusoe is well understood from the simple sketch of the processor architecture, called 'amoeba'. In this concept, the x86-architecture is an ill-defined amoeba containing features like segmentation, ASCII arithmetic, variable-length instructions etc. The amoeba explained how a traditional microprocessor was, in their design, to be divided up into hardware and software.

Thus Crusoe was conceptualized as a hybrid microprocessor that is it has a software part and a hardware part with the software layer surrounding the hardware unit. The role of software is to act as an emulator to translate x86 binaries into native code at run time. Crusoe is a 128-bit microprocessor fabricated using the CMOS process. The chip's design is based on a technique called VLIW to ensure design simplicity and high performance. Besides this it also uses Transmeta's two patented technologies, namely, Code Morphing Software and Longrun Power Management. It is a highly integrated processor available in different versions for different market segments.

Technology Perspective

The Transmeta designers have decoupled the x86 instruction set architecture (ISA) from the underlying processor hardware, which allows this hardware to be very different from a conventional x86 implementation. For the same reason, the underlying hardware can be changed radically without affecting legacy x86 software: each new CPU design only requires a new version of the Code Morphing software to translate x86 instructions to the new CPU's native instruction set. For the initial Transmeta products, models TM3120 and TM5400, the hardware designers opted for minimal space and power. By eliminating roughly three quarters of the logic transistors that would be required for an all-hardware design of similar performance, the designers have likewise reduced power requirements and die size. However, future hardware designs can emphasize different factors and accordingly use different implementation techniques. Finally, the Code Morphing software which resides in standard Flash ROMs itself offers opportunities to improve performance without altering the underlying hardware.

CRUSOE-“AMOEBA”

Hardware: high-MHz, small disc size, non-x86, VLIW processor

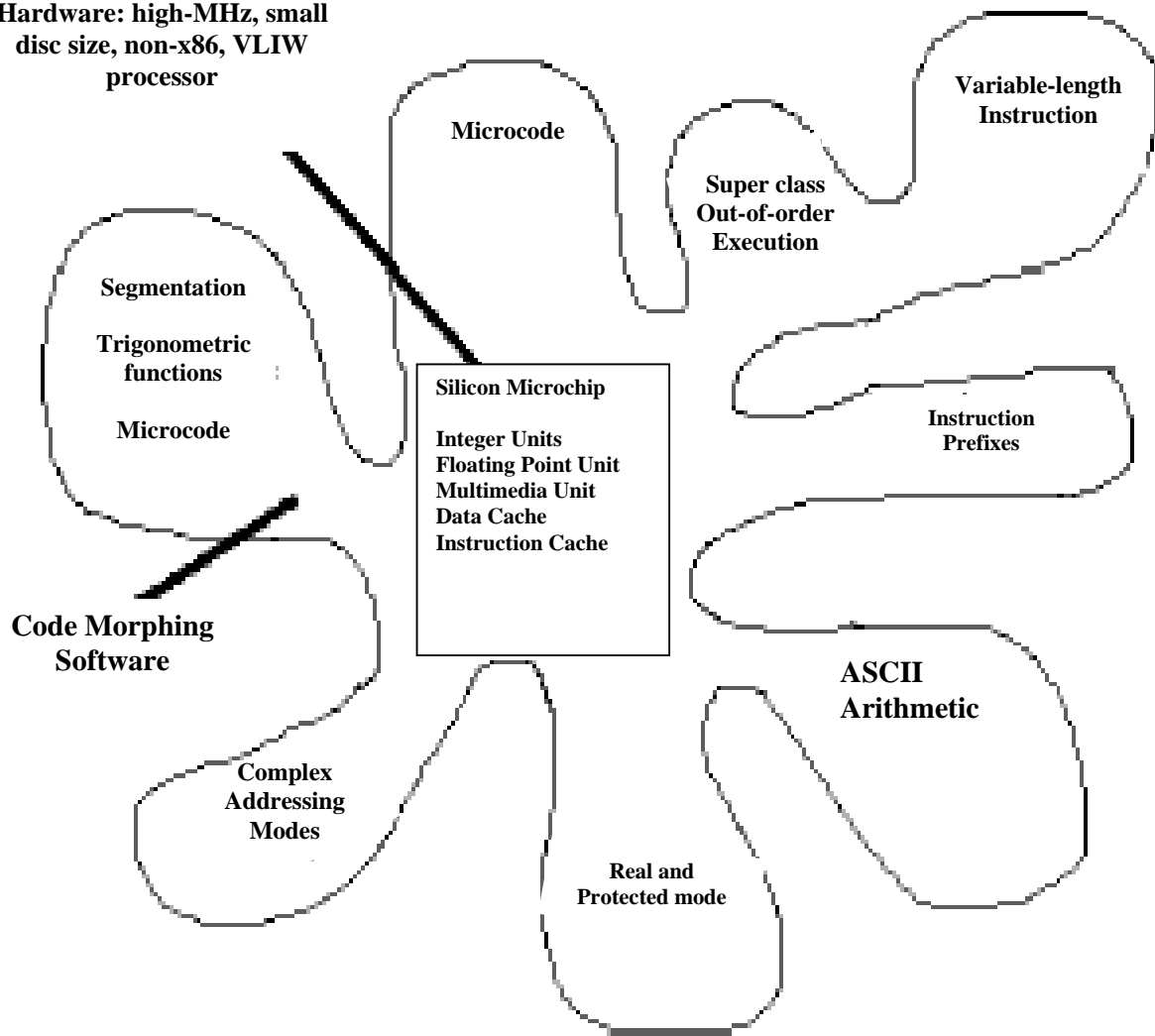


Figure 1: Crusoe- “AMOEBA”

2. CRUSOE PROCESSOR VLIW HARDWARE

2.1 Basic principles of VLIW Architecture

VLIW stands for Very Long Instruction Word. VLIW is a method that combines multiple standard instructions into one long instruction word. This word contains instructions that can be executed at the same time on separate chips or different parts of the same chip. This provides explicit parallelism. VLIW architectures are a suitable alternative for exploiting instruction-level parallelism (ILP) in programs, that is, for executing more than one basic (primitive) instruction at a time. By using VLIW you enable the compiler, not the chip to determine which instructions can be run concurrently. This is an advantage because the compiler knows more information about the program than the chip does by the time the code gets to the chip. These processors contain multiple functional units, fetch from the instruction cache a Very-Long Instruction Word containing several primitive instructions, and dispatch the entire VLIW for parallel execution. These capabilities are exploited by compilers which generate code that has grouped together independent primitive instructions executable in parallel. The processors have relatively simple control logic because they do not perform any dynamic scheduling or reordering of operations (as is the case in most contemporary superscalar processors). Trace scheduling is an important technique in VLIW processing. Trace scheduling is when the compiler processes the code and determines which path is used the most frequently traveled. The compiler then optimizes this path. The basic blocks that compose the path are separated from the other basic blocks. The path is then optimized and rejoined with the other basic blocks. The rejoining includes special split and rejoin blocks that help align the converted code with the original code.

Dynamic scheduling is another important method when compiling VLIW code. The process, called split-issue splits the code into two phases, phase one and phase two. This allows for multiple instructions to execute at the same time. Thus, instructions that have certain delays associated with them can be run concurrently, and out-of-order execution is possible. Hardware support is needed to implement this technique and requires delay buffers and temporary variable space in the hardware. The temporary variable space is needed to store results when they come in. The results computed in phase two are stored in temporary variables and

are loaded into the appropriate phase one register when they are needed. VLIW has been described as *a natural successor to RISC*, because it moves complexity

from the hardware to the compiler, allowing simpler, faster processors. The objective of VLIW is to eliminate the complicated instruction scheduling and parallel dispatch that occurs in most modern microprocessors. In theory, a VLIW processor should be faster and less expensive than a comparable RISC chip.

The instruction set for a VLIW architecture tends to consist of simple instructions (RISC like). The compiler must assemble many primitive operations into a single "instruction word" such that the multiple functional units are kept busy, which requires enough instruction-level parallelism (ILP) in a code sequence to fill the available operation slots. Such parallelism is uncovered by the compiler through scheduling code speculatively across basic blocks, performing software pipelining, reducing the number of operations executed, among others.

2.2 VLIW in Crusoe Microprocessor

With the Code Morphing software handling x86 compatibility, Transmeta hardware designers created a very simple, high-performance, VLIW engine with two integer units, a floating point unit, a memory (load/store) unit, and a branch unit. A Crusoe processor long instruction word, called a molecule, can be 64 bits or 128 bits long and contain up to four RISC-like instructions, called atoms. All atoms within a molecule are executed in parallel, and the molecule format directly determines how atoms get routed to functional units; this greatly simplifies the decode and dispatch hardware. Figure 2 shows a sample 128-bit molecule and the straightforward mapping from atom slots to functional units. Molecules are executed in order, so there is no complex out-of-order hardware. To keep the processor running at full speed, molecules are packed as fully as possible with atoms. In a later section, we describe how the Code Morphing software accomplishes this.

128-bit Molecule

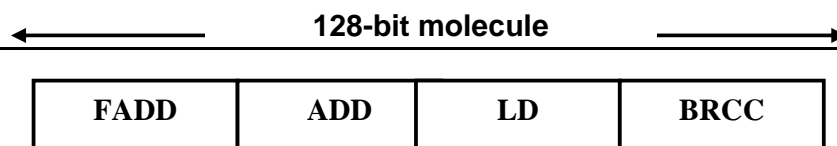


Figure 2: A molecule can contain up to four atoms, which are executed in parallel.

The integer register file has 64 registers, %r0 through %r63. By convention, the Code Morphing software allocates some of these to hold x86 state while others contain state internal to the system, or can be used as temporary registers, e.g., for register renaming in software. In the assembly code examples in this paper, we write one molecule per line, with atoms separated by semicolons. The destination register of an atom is specified first; a “.c” opcode suffix designates an operation that sets the condition codes. Where a register holds x86 state, we use the x86 name for that register (e.g., %eax instead of the less descriptive %r0). Superscalar out-of-order x86 processors, such as the Pentium II and Pentium III processors, also have multiple functional units that can execute RISC-like operations (micro-ops) in parallel. Figure 2 depicts the hardware these designs use to translate x86 instructions into micro-ops and schedule (dispatch) the micro-ops to make best use of the functional units. Since the dispatch unit reorders the micro-ops as required to keep the functional units busy, a separate piece of hardware, the in-order retire unit, is needed to effectively reconstruct the order of the original x86 instructions, and ensure that they take effect in proper order. Clearly, this type of processor hardware is much more complex than the Crusoe processor’s simple VLIW engine.

X86 instructions

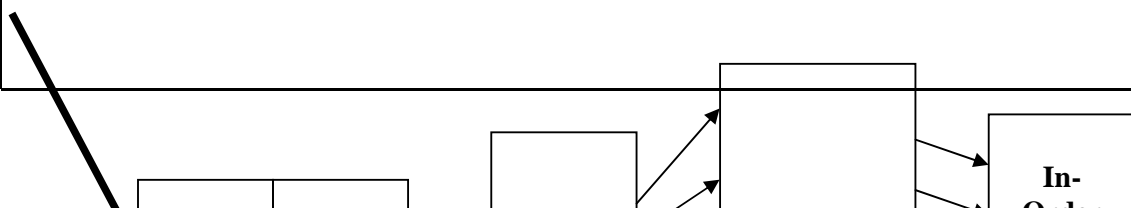


Figure 3: Conventional superscalar out-of-order CPUs use hardware to create and dispatch micro-ops that can execute in parallel

Because the x86 instruction set is quite complex, the decoding and dispatching hardware requires large quantities of power-hungry logic transistors; the chip dissipates heat in rough proportion to their numbers.

3. CODE MORPHING SOFTWARE

The Code Morphing software is fundamentally a dynamic translation system, a program that compiles instructions for one instruction set architecture (in this case, the x86 target ISA) into instructions for another ISA (the VLIW host ISA). The Code Morphing software resides in a ROM and is the first program to start executing when the processor boots. The Code Morphing Software supports ISA, and is the only thing x86 code sees; the only program written directly for the VLIW engine is the Code Morphing software itself. Figure 5 shows the relationship between x86 codes, the Code Morphing software, and a Crusoe processor. Because the Code Morphing software insulates x86 programs—including a PC's BIOS and operating system—from the hardware engine's native instruction set, that native instruction set can be changed arbitrarily without affecting any x86 software at all. The only program that needs to be ported is the Code Morphing

software itself, and that work is done once for each architectural change, by Transmeta. The feasibility of this concept has already been demonstrated: the native ISA of the model TM5400 is an enhancement (neither forward nor backward

compatible) of the model TM3120's ISA and therefore runs a different version of Code Morphing software. The processors are different because they are aimed at different segments of the mobile market: the model TM3120 is aimed at Internet appliances and ultra-light mobile PCs, while the model TM5400 supports high performance, full-featured 3-4lb. mobile PCs. Coincidentally, hiding the chip's ISA behind a software layer also avoids a problem that has in the past hampered the acceptance of VLIW machines. A traditional VLIW exposes details of the processor pipeline to the compiler; hence any change to that pipeline would require all existing binaries to be recompiled to make them run on the new hardware. Note that even traditional x86 processors suffer from a related problem: while old applications will run correctly on a new processor, they usually need to be recompiled to take full advantage of the new processor implementation. This is not a problem on Crusoe processors, since in effect, the Code Morphing software always transparently "recompiles" and optimizes the x86 code it is running. The flexibility of the software-translation approach comes at a price: the processor has to dedicate some of its cycles to running the Code Morphing software, cycles that a conventional x86 processor could use to execute application code. To deliver good practical system performance, Transmeta has carefully designed the Code Morphing software for maximum efficiency and low overhead.

3.1 Decoding and Scheduling

Conventional x86 superscalar processors fetch x86 binary instructions from memory and decode them into micro-operations, which are then reordered by out-of-order dispatch hardware and fed to the functional units for parallel execution.

In contrast (besides being a software rather than a hardware solution), Code Morphing can translate an entire group of x86 instructions at once, creating a translation, whereas a superscalar x86 translates single instructions in isolation. Moreover, while a traditional x86 translates each x86 instruction every time it is executed, Transmeta's software translates instructions once, saving the resulting translation in a translation cache. The next time the (now translated) x86 code is

executed, the system skips the translation step and directly executes the existing optimized translation. Implementing the translation step in software as opposed to hardware opens up new opportunities and challenges. Since an out-of-order processor has to translate and schedule instructions every time they execute, it

must do so very quickly. This seriously limits the kinds of transformations it can perform. The Code Morphing approach, on the other hand, can amortize the cost of translation over many executions, allowing it to use much more sophisticated translation and scheduling algorithms. Likewise, the amount of power consumed for the translation process is amortized, as opposed to having to pay it on every execution. Finally, the translation software can optimize the generated code and potentially reduce the number of instructions executed in a translation. In other words, Code Morphing can speed up execution while at the same time reducing power!

3.2 Caching

The translation cache, along with the Code Morphing code, resides in a separate memory space that is inaccessible to x86 code. (For better performance, the Code Morphing software copies itself from ROM to DRAM at initialization time.) The size of this memory space can be set at boot time, or the operating system can make the size adjustable. As with all caching, the Code Morphing software's technique of reusing translations takes advantage of "locality of reference". Specifically, the translation system exploits the high repeat rates (the number of times a translated block is executed on average) seen in real-life applications. After a block has been translated once, repeated execution "hits" in the translation cache and the hardware can then execute the optimized translation at full speed. Some benchmark programs attempt to exercise a large set of features in a small amount of time, with little repetition—a pattern that differs significantly from normal usage.

(When was the last time you used every other feature of Microsoft Word exactly once, over a period of a minute?) The overhead of Code Morphing translation is obviously more evident in those benchmarks. Furthermore, as an application executes, Code Morphing "learns" more about the program and improves it so it

will execute faster and faster. Today's benchmarks have not been written with a processor in mind that gets faster over time, and may "charge" Code Morphing for the learning phase without waiting for the payback. As a result, some benchmarks do not accurately predict the performance of Crusoe processors. On typical applications, due to their high repeat rates, Code Morphing has the opportunity to

optimize execution and amortize any initial translation overhead. As an example, consider a multimedia application such as playing a DVD—before the first video frame has been drawn; the DVD decoder will have been fully translated and optimized, incurring no further overhead during the playing time of the DVD. In summary, we find that the Crusoe processor's approach of caching translations delivers excellent performance in real-life situations.

3.3 Filtering

It is well known that in typical applications, a very small fraction of the applications code (often less than 10%, sometimes as little as 1%) accounts for more than 95% of execution time. Therefore, the translation system needs to choose carefully how much effort to spend on translating and optimizing a given piece of x86 code. Obviously, we want to lavish the optimizer's full attention on the most frequently executed code but not waste it on code that executes only once. The Code Morphing software includes in its arsenal a wide choice of execution modes for x86 code, ranging from interpretation (which has no translation overhead at all, but executes x86 code more slowly), through translation using very simple-minded code generation, all the way to highly optimized code (which takes longest to generate, but which runs fastest once translated). A sophisticated set of heuristics helps choose among these execution modes based on dynamic feedback information gathered during actual execution of the code.

3.4 Prediction and Path Selection

One of the many ways in which the Code Morphing software can gather feedback about the x86 programs is to instrument translations: the translator adds code whose sole purpose is to collect information such as block execution frequencies,

or branch history. This data can be used later to decide when and what to optimize and translate. For example, if a given conditional x86 branch is highly biased (e.g. usually taken), the system can likewise bias its optimizations to favor the most frequently taken path. Alternatively, for more balanced branches (taken as often as not, for example), the translator can decide to speculatively execute code from both paths and select the correct result later. Analogously, knowing how often a

piece of x86 code is executed helps decide how much to try to optimize that code. It would be extremely difficult to make similar decisions in a traditional hardware-only x86 implementation. Current Intel and AMD x86 processors convert x86 instructions into RISC-like micro-ops that are simpler and easier to handle in a superscalar micro architecture. (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 fronted 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. 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.

4. LONGRUN POWER MANAGEMENT

Although the Code Morphing software's primary responsibility is ensuring x86 compatibility, it also provides interfaces to capabilities available only in Crusoe processor models. Longrun power management is one example—a facility in the TM5400 model that can further minimize that processor's already low power consumption. In a mobile setting, most conventional x86 CPUs regulate their power consumption by rapidly alternating between running the processor at full

speed and (in effect) turning the processor off. Different performance levels can be obtained by varying the on/off ratio (the “duty cycle”). However, with this approach, the processor may be shut off just when a time-critical application needs it. The result may be glitches, such as dropped frames during movie playback, that are perceptible (and annoying) to a user. In contrast, the TM5400 can adjust its power consumption without turning itself off—instead, it can adjust its clock frequency on the fly. It does so extremely quickly, and without requiring an operating system reboot or having to go through a slow sequence of suspending to and restarting from RAM. As a result, software can continuously monitor the demands on the processor and dynamically pick just the right clock speed (and hence power consumption) needed to run the application—no more and no less. Since the switching happens so quickly, it is not noticeable to the user.

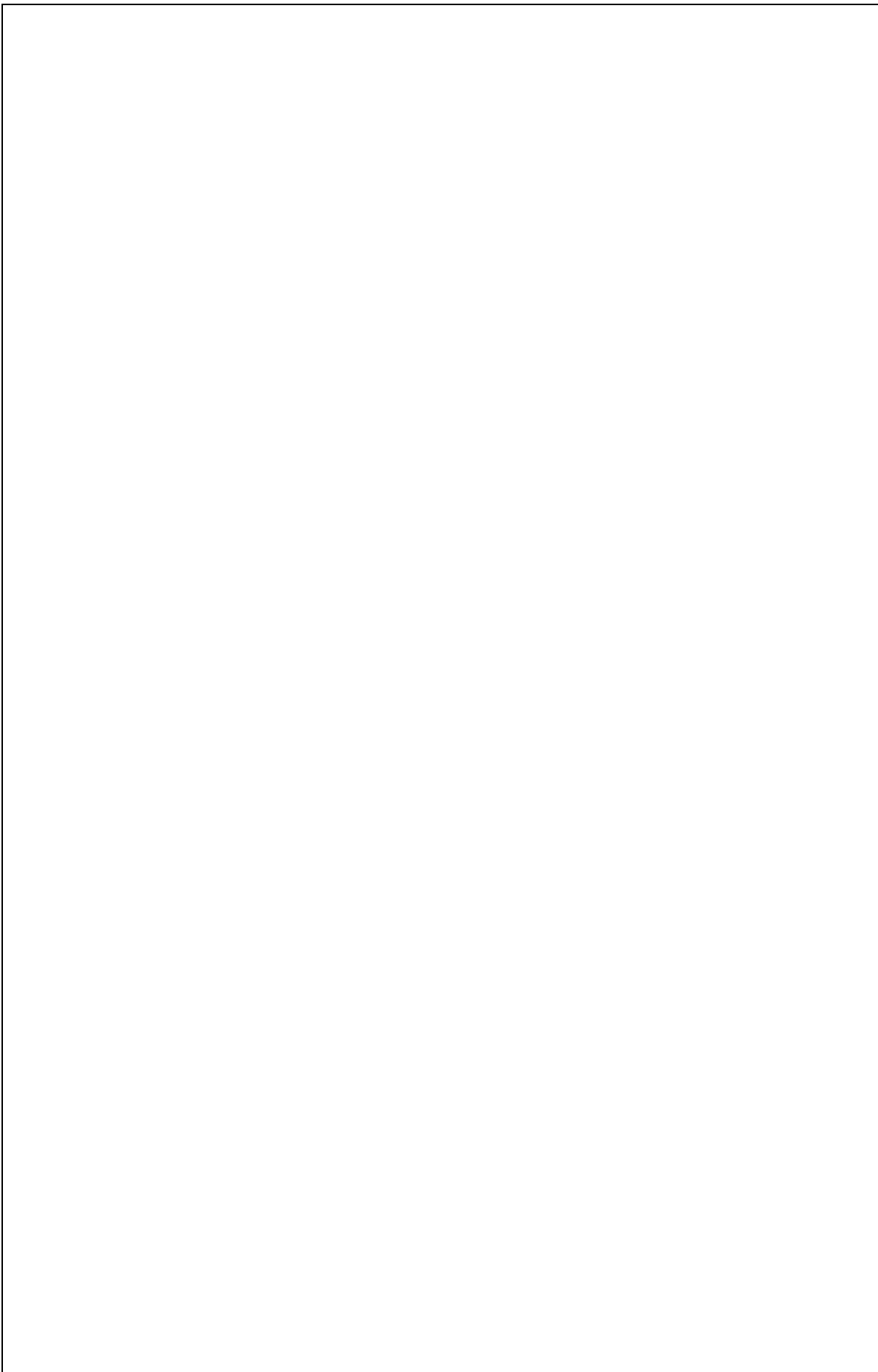
4.1 Longrun extends Battery Life

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. 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. When

the Longrun software detects a change in the CPU load, it signals the chip to adjust the voltage and frequency up or down. 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, and 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 a speech to a car chase. But Crusoe Longrun 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, the CPU doesn't stall during the swing. 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. Longrun is a genuine innovation that gives Crusoe an extra edge. Finally, the Code Morphing software can also adjust the Crusoe processor's voltage on the fly (since

at a lower operating frequency, a lower voltage can be used). Because power varies linearly with clock speed and by the square of the voltage, adjusting both can produce cubic reductions in power consumption whereas conventional CPU's can adjust power only linearly. For example, assume an application program only requires 90% of the processor's speed. On a conventional processor, throttling back the processor speed by 10% cuts power by 10%, whereas under the same conditions, Longrun power management can reduce power by almost 30%—a noticeable advantage!



CRUSOE PROCESSOR HARDWARE ARCHITECTURE

5. CRUSOE PROCESSOR ARCHITECTURE

The Crusoe microprocessor is available in the market in the following versions: TM3120, TM3200, TM5400 and TM5600. The basic architecture of all the above models is same except for some minor changes since various models have been introduced for different segments of the mobile computing market. The following architectural description has taken Crusoe TM5400 as reference. The Crusoe Processor incorporates integer and floating point execution units, separate instruction and data caches, a level-2 write-back cache, memory management unit, and multimedia instructions. In addition to these traditional processor features, the device integrates a DDR SDRAM memory controller, SDR SDRAM

memory controller, PCI bus controller and serial ROM interface controller. These additional units are usually part of the core system logic that surrounds the microprocessor. The VLIW processor, in combination with Code Morphing software and the additional system core logic units, allow the Crusoe Processor to provide a highly integrated, ultra-low power, high performance platform solution for the x86 mobile market.

5.1 Processor Core

The Crusoe Processor core architecture is relatively simple by conventional standards. It is based on a Very Long Instruction Word (VLIW) 128-bit instruction set. Within this VLIW architecture, the control logic of the processor is kept very simple and software is used to control the scheduling of instructions. This allows a simplified and very straightforward hardware implementation with an in-order 7-stage integer pipeline and a 10-stage floating point pipeline. By streamlining the processor hardware and reducing the control logic transistor count, the performance-to-power consumption ratio can be greatly improved over traditional x86 architectures. The Crusoe Processor includes a 8-way set-associative Level 1 (L1) instruction cache, and a 16-way set associative L1 data cache. It also includes an integrated Level 2 (L2) write-back cache for improved effective memory bandwidth and enhanced performance. This cache architecture assures maximum internal memory bandwidth for performance intensive mobile

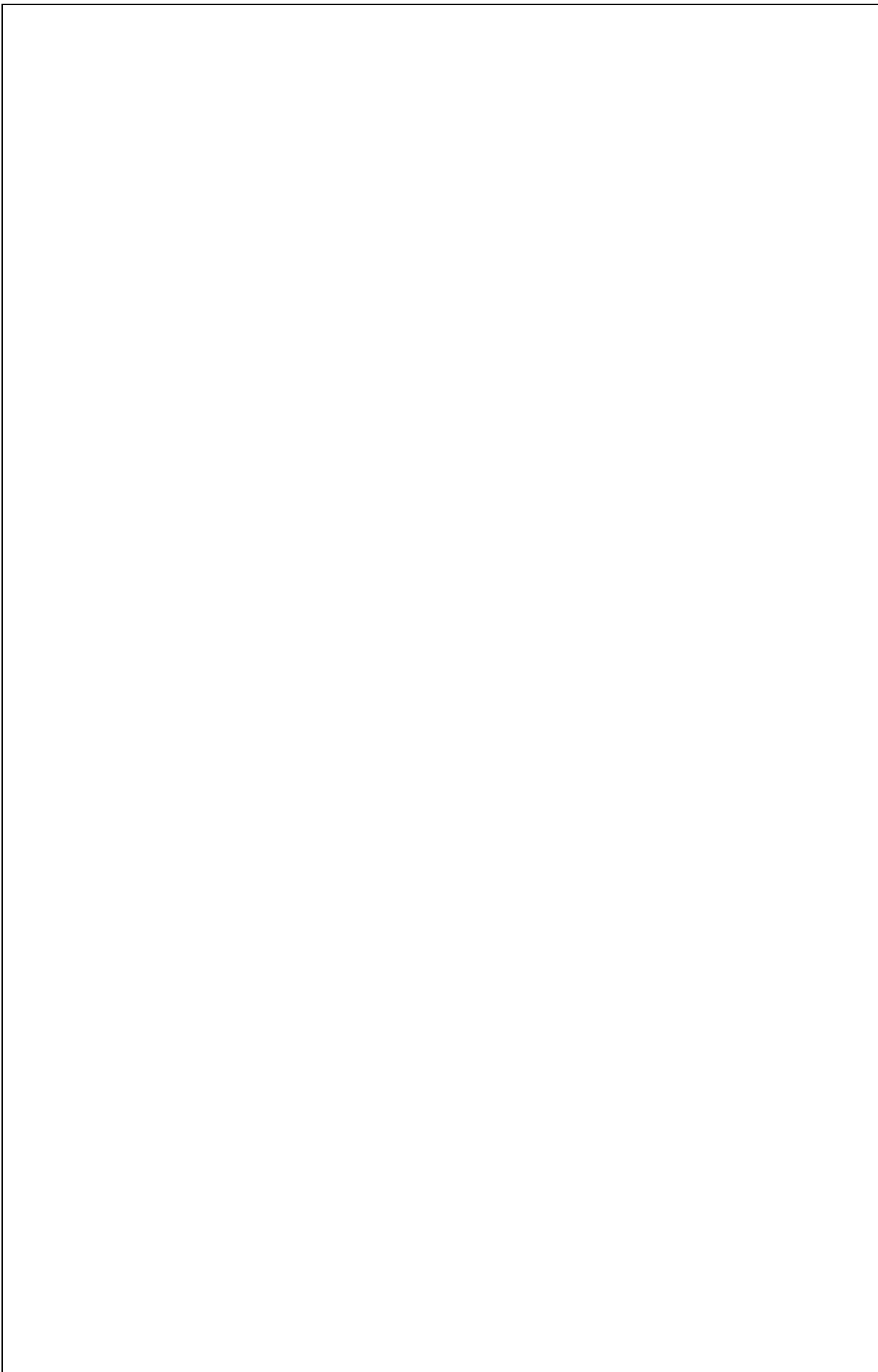
applications, while maintaining the same low-power implementation that provides a superior performance-to-power consumption ratio relative to previous x86 implementations. Other than having execution hardware for logical, arithmetic, shift, and floating point instructions, as in conventional processors, the Crusoe Processor has very distinctive features from traditional x86 designs. To ease the translation process from x86 to the core VLIW instruction set, the hardware generates the same condition codes as conventional x86 processors and operates on the same 80-bit floating point numbers. Also, the Translation Look-aside Buffer (TLB) has the same protection bits and address mapping as x86 processors. The software component of this solution is used to emulate all other features of the x86 architecture. The software that converts x86 programs into the core VLIW instructions is the Code Morphing software.

5.2 Integrated DDR SDRAM Memory Controller

The DDR SDRAM interface is the highest performance memory interface available on the Crusoe Processor. The DDR SDRAM controller supports only Double Data Rate (DDR) SDRAM and transfers data at a rate that is twice the clock frequency of the interface. This feature is absent in the Crusoe processor model TM 3200. The DDR SDRAM controller supports up to four banks, the equivalent of two Dual line Memory Modules (DIMM's), of DDR SDRAM using a 64-bit wide inter-face. The DDR SDRAM memory can be populated with 64M-bit, 128M-bit, or 256M-bit devices. The frequency setting for the DDR SDRAM interface is initialized during the power-on boot sequence.

5.3 Integrated SDR SDRAM Memory Controller

The SDR SDRAM memory controller supports up to four banks, equivalent to two Small Outline Dual In-line Memory Modules (SO-DIMMS), of Single Data Rate (SDR) SDRAM that can be configured as 64-bit or 72-bit SO-DIMM's. This SO-DIMM's can be populated with 64M-bit, 128M-bit or 256M-bit devices. All SO-DIMM's must use the same frequency SDRAM's, but there are no restrictions on mixing different SO- DIMM configurations into each SO-DIMM slot. The frequency setting for the SDR SDRAM interface is initialized during the power-on boot sequence.



5.4 Integrated PCI Controller

The Crusoe Processor includes a PCI bus controller that is PCI 2.1 compliant. The PCI bus is 32 bits wide, operates at 33 MHz, and is compatible with 3.3V signal levels. It is not 5V tolerant, however. The PCI controller on provides a PCI host bridge, the PCI bus arbiter, and a DMA controller.

5.5 Serial ROM Interface

The Crusoe Processor serial ROM interface is a five-pin interface used to read data from a serial flash ROM. The flash ROM is 1M-byte in size and provides non-volatile storage for the Code Morphing software. During the boot process, the Code Morphing code is copied from the ROM to the Code Morphing memory space in SDRAM. Once transferred, the Code Morphing code requires 8 to 16M-bytes of memory space. The portion of SDRAM space reserved for Code Morphing software is not visible to x86 code. Transmeta supplies programming information for the flash ROM device. This interface may also be used for in-system reprogramming of the flash ROM.

6. FEATURES OF VARIOUS CRUSOE PROCESSORS

6.1 Crusoe Processor Model TM3200 Features

- ➔ VLIW processor and x86 Code Morphing software provide x86-compatible mobile platform solution.
- ➔ Processor core operates at 366 and 400 MHz.
- ➔ Integrated 64K-byte instruction cache and 32K-byte data cache.
- ➔ Integrated north bridge core logic features facilitate compact system designs.
- ➔ SDR SDRAM memory controller with 66-133 MHz, 3.3V interface.
- ➔ PCI (Peripheral Component Interface) bus controller (PCI 2.1 compliant) with 33 MHz, 3.3V interface.
- ➔ Advanced power management features and very-low power operation extend mobile battery life
- ➔ Full System Management Mode (SMM) support.
- ➔ Compact 474-pin ceramic BGA (Ball Grid Array) package.

6.2 Crusoe Processor Model TM5400

- ➔ VLIW processor and x86 Code Morphing software provide x86-compatible mobile platform solution.
- ➔ Processor core operates at 500-700 MHz.
- ➔ Integrated 64K-byte L1 instruction cache, 64K-byte L1 data cache, and 256K-byte L2 write-back cache.

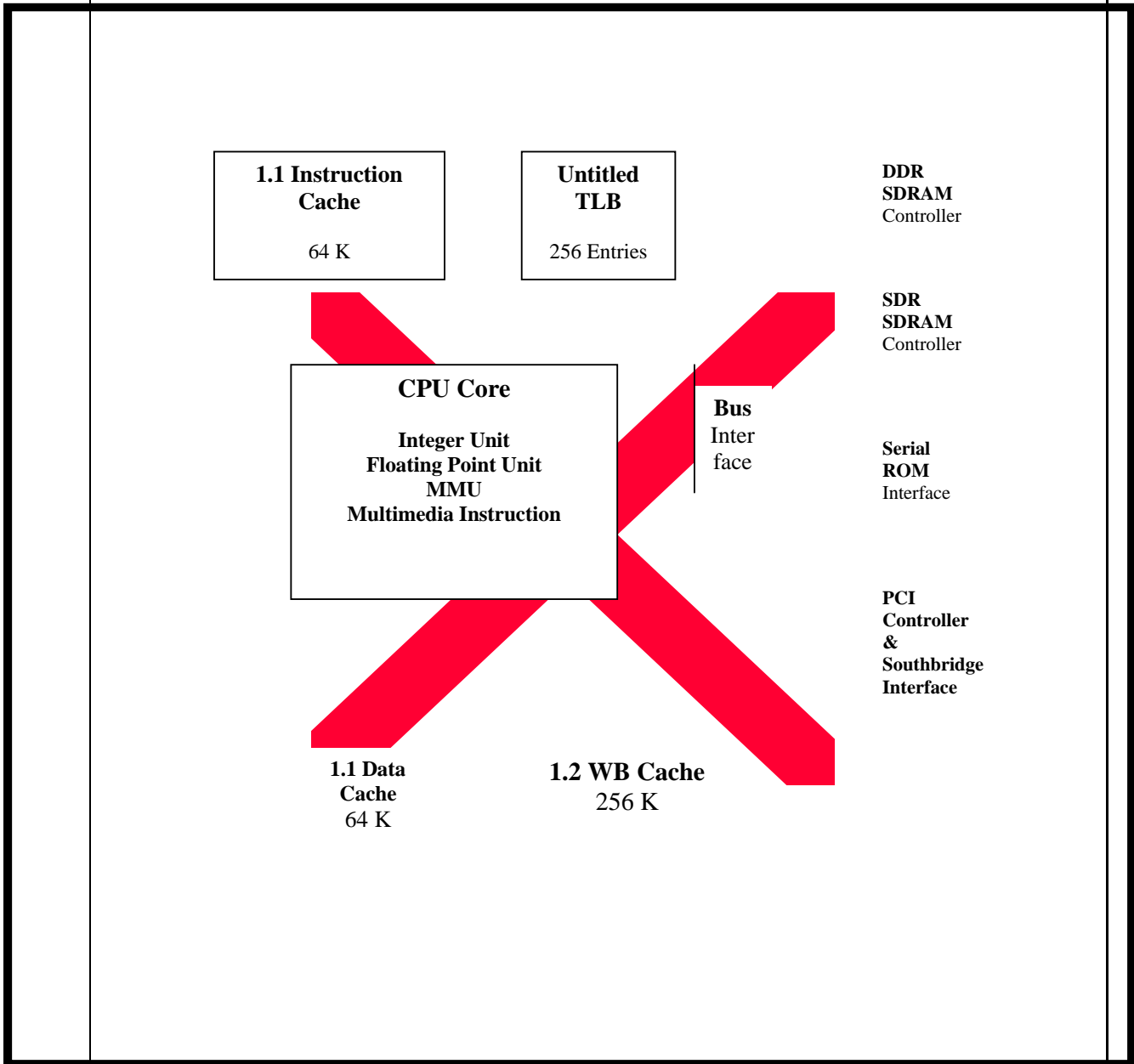


Figure 5: Crusoe processor Architecture – Model TM5400

- ➔ Integrated north bridge core logic features facilitate compact system designs.
- ➔ DDR SDRAM memory controller with 100-133 MHz, 2.5V interface.
- ➔ SDR SDRAM memory controller with 66-133 MHz, 3.3V interface.
- ➔ PCI bus controller (PCI 2.1 compliant) with 33 MHz, 3.3V interface.
- ➔ Longrun advanced power management with ultra-low power operation extends mobile battery life! 1-2 W @ 500-700 MHz, 1.2-1.6V running typical multimedia applications! 50 mW in deep sleep
- ➔ Full System Management Mode (SMM) support.
- ➔ Compact 474-pin ceramic BGA package.

6.3 Crusoe Processor Model TM5600 Features

- ➔ VLIW processor and x86 Code Morphing software provide x86-compatible mobile platform solution.
- ➔ Processor core operates at 500-700 MHz.
- ➔ Integrated 64K-byte L1 instruction cache, 64K-byte L1 data cache, and 512K-byte L2 write-back cache Integrated north bridge core logic features facilitate compact system designs DDR SDRAM memory controller with 100-133 MHz, 2.5V interface SDR SDRAM memory controller with 66-133 MHz, 3.3V interface PCI bus controller (PCI 2.1 compliant) with 33 MHz, 3.3V interface
- ➔ Longrun advanced power management with ultra-low power operation extends mobile battery life! 1-2 W @ 500-700 MHz, 1.2-1.6V running typical multimedia applications! 100 mW in deep sleep
- ➔ Full System Management Mode (SMM) support
- ➔ Compact 474-pin ceramic BGA package reprogramming of the flash ROM.

7. CONCLUSION

In 1995, Transmeta set out to expand the reach of microprocessors into new markets by dramatically changing the way microprocessors are designed. The initial market is mobile computing, in which complex power-hungry processors have forced users to give up either battery running time or performance. The Crusoe processor solutions have been designed for lightweight (two to four pound) mobile computers and Internet access devices such as handhelds and web pads. They can give these devices PC capabilities and unplugged running times of up to a day. To design the Crusoe processor chips, the Transmeta engineers did not resort to exotic fabrication processes. Instead they rethought the fundamentals of microprocessor design. Rather than “throwing hardware” at design problems, they chose an innovative approach that employs a unique combination of hardware and software. Using software to decompose complex instructions into simple atoms and to schedule and optimize the atoms for parallel execution saves millions of logic transistors and cuts power consumption on the order of 60–70% over conventional approaches—while at the same time enabling aggressive code optimization techniques that are simply not feasible in traditional x86 implementations. Transmeta’s Code Morphing software and fast VLIW hardware, working together, achieve low power consumption without sacrificing high performance for real-world applications. Although the model TM3120 and model TM5400 are impressive first efforts, the significance of the Transmeta approach to microprocessor design is likely to become more apparent over the next several years. The technology is young and offers more freedom to innovate (both hardware and software) than conventional hardware-only designs. Nor is the approach limited to low-power designs or to x86-compatible processors. Freed to render their ideas in a combination of hardware and software, and to evolve hardware without breaking legacy code, Transmeta microprocessor designers may produce one surprise after another in the coming years.

GLOSSARY

Atom - a component of a VLIW instruction word, or molecule. Conceptually, an atom is similar to an instruction for a RISC processor. A key difference is that a VLIW processor does not execute individual atoms by themselves but always as part of an entire molecule.

Binary instruction - generally speaking, an instruction for a computer, as encoded in a format understandable by the computer (as opposed to being readable by humans).

Cache - Generally a small chunk of fast memory that sits between either 1) a smaller, faster chunk of memory and a bigger, slower chunk of memory, or 2) a processor and a bigger, slower chunk of memory. This is to provide a bridge from something that's comparatively very fast to something that's comparatively slow.

Chipset - usually refers to Northbridge and Southbridge chips used in building a computer motherboard.

CISC - Intel x86 chips are CISC chips because of the complexity of the instruction set. On the other side of the coin, you have RISC chips that use a reduced instruction set. RISC chips split big operations into lots of simple, tiny instructions that are processed very quickly.

CMOS - the predominant integrated circuit technology used in semiconductors. Compared to its alternatives, CMOS consumes less power while enabling high speeds.

Code - what computer scientists call the collection of computer instructions that constitute an application program separate from the data.

Code Morphing software - Code Morphing software translates the ones and zeros of the application program instructions into a more efficiently executed set of

instructions for the Crusoe processor.

Commit atom - an atom specific to the Crusoe VLIW engine that commits a set of pending changes (to memory or to registers) in a single operation.

Compiler - software that translates a program written in a high-level programming language (COBOL, C, and others) into binary instructions.

CPU (Central Processing Unit, also called "processor") - the active part of the computer where all computation (such as addition) is performed. Today, the CPUs of almost all computers are contained on a single chip.

Crusoe - for consumers, the Crusoe name will mean long battery life, full compatibility, and high performance. The Crusoe brand stands for a family of smart microprocessors from Transmeta intended for use in Mobile Internet Computers.

DDR memory - a new type of SDRAM memory that is both higher performance and lower power than standard SDRAM, making it appropriate for the latest portable devices. "DDR" is short for "Double Data Rate," referring to the high speed at which this memory transfers data.

Full Internet experience - the ability to get access to all the data and details found on the World Wide Web, in the full form intended by the web site. This can only be done with a computer that is fully Internet compatible

Gated store buffer - a hardware mechanism in the Crusoe VLIW engine that holds writes to memory until those writes are either committed (via a commit atom) or discarded (via a rollback atom). The "gated" refers to the notion of the write operations being held behind a fence until they are ready to be released.

Information appliance - a computing device designed specifically for accessing the Internet - "browsing the web" - but not designed for running PC applications.

Instruction set architecture - in essence, the programmer-visible part of how a processor works, comprising the instructions that the processor implements, the set of registers available, and so on.

Internet appliance, Internet device - any computer device used to access information on the Internet.

Internet compatible - a computer that is x86 PC compatible and can also be compatible with the rich data types found on the World Wide Web, such as Macromedia Flash, RealAudio, and various streaming video formats. Often compatibility with web data comes from loading a plugin for an internet browser. Compatibility is often a combination of the PC system architecture, the operating system (such as Microsoft Windows or Linux), and the ability to execute x86-plugin applications.

Load-and-protect atom - an atom that loads a data value from a memory location, while at the same time "protecting" that memory location from being overwritten. The Code Morphing software uses load-and-protect atoms to generate faster code.

LongRun - LongRun is a power management technique found in the Crusoe processor model TM5400. LongRun works by monitoring the precise performance level needed by an application, and then by dynamically adjusting the Crusoe processor's operating speed and voltage to match that need. With LongRun, Crusoe can make adjustments while the application is running, thereby making the most efficient use of power, for the longest battery life.

Low-power processor - a low-power processor is a microprocessor that consumes less than five watts when running applications, and in cases where the

processor does not need to be actively running, has a sleep mode that consumes less than 25 milliWatts of power.

MHz (megahertz) - a measure of frequency, commonly used to describe the speed of electronic devices. A device said to run at 600 MHz, for example, performs some operation (such as executing an instruction) 600 million times per second.

Micro-ops - a term commonly used to refer to operations performed internally by out-of-order superscalar processors; typically, one or more micro-ops are performed for each instruction that the processor executes.

Mobile Internet Computer - a computer that uses a Mobile Internet Processor to achieve compatibility with the x86 PC system architecture, long battery life, and high performance.

Mobile Internet Processor - a microprocessor that simultaneously has x86 compatibility, low power, and sufficient performance to run demanding web-based applications such as streaming video.

Mobile Linux - a version of Linux created by Transmeta™ and distributed to its OEM customers for mobile Internet devices such as web slates and handheld clients that do not contain a hard drive. The principal enhancements for Mobile Linux are in the areas of power management and in the reduction of the memory footprint.

Molecule - a VLIW instruction word containing multiple atoms, which the hardware executes in parallel. The current Crusoe processors support 64- and 128-bit-long molecules containing two to four atoms each. When the hardware processes a molecule, it executes all atoms in the molecule in parallel, meaning the current Crusoe processors can perform up to four operations per cycle.

Northbridge - the interface between the CPU and memory, and the interface between the CPU and Input/Output busses. In prior years, the Northbridge was a separate chip. With the Crusoe processor, the DRAM controllers to memory and the PCI Input/Output busses have been integrated onto the same piece of silicon as the CPU.

OEM - Original Equipment Manufacturer - This acronym is used to denote equipment that is sold to other companies or resellers for integration into systems. For example, a chip manufacturer may sell an OEM chip in bulk quantities and no manual or cables with the promise that it will go into full systems.

Optimization - a process whereby a program is made to run faster by removing edundant instructions, reordering instructions, or replacing "slow" instruction sequences with faster ones.

Out-of-order machine - a processor architecture that internally executes instructions in an order different from that in which they occur in the program being executed. This is the traditional, hardware-intensive approach to speeding up old programs on new processors.

PCI bus - a hardware interface that connects a processor to user input/output devices (for example, graphics controller, USB controller, LANs, or modems). The Crusoe processor model TM3200 and Crusoe processor model TM5400 include on-chip PCI controllers.

Plugin - a term for a small computer program, usually x86 compatible, that can extend the functionality of a web browser, such as Netscape or Microsoft Internet Explorer.

Register - fast temporary storage areas located in the processor. All computation (such as addition) in a processor is typically done inside registers.

RISC (Reduced Instruction Set Computer) - a computer architecture that reduces chip complexity by using simpler instructions than a CISC (Complex Instruction Set Computer). An x86 processor is generally regarded as a CISC.

Rollback atom - an atom specific to the Crusoe VLIW hardware that discards a set of pending changes (to memory or to registers) in a single operation. In effect, the opposite of a commit atom.

ROM (Read Only Memory) - a memory chip that persistently stores instructions and data, even when power is turned off. A Flash ROM is a type of ROM that can actually be reprogrammed with different contents.

SDRAM (Synchronous DRAM) - a type of dynamic RAM memory chip that has been widely used in the latter part of the 1990s.

Shadow register - a mechanism in the Crusoe VLIW engine that enables fast commit and rollback operations. To perform a commit operation, the processor copies each normal, or "working", register into its corresponding shadow register. For a rollback, the copy is done in the reverse direction. In effect, each shadow register contains a "backup" copy of its corresponding working register, allowing the working register to be overwritten with (possibly speculative) data.

Smart Microprocessor - a microprocessor that is capable of learning about an application while it is running and is able to use that knowledge to improve behavior. Crusoe uses learning to both improve the performance and to lower the power of applications as they run.

Software optimized execution - the ability to use software to improve the characteristics of a computer program while it is running.

Southbridge - a chip that interfaces between the PCI Input/Output bus and the many individual interfaces on a computer, such as keyboard, mouse, and hard disk.

Speculation - the notion of performing an operation speculatively, that is, before knowing whether the result of the operation will actually be needed. The Crusoe

processor supports speculation via an innovative combination of hardware and software.

Superscalar - a CPU architecture that can execute more than one instruction in one clock cycle. Unlike a VLIW processor, a superscalar processor must first

analyze (at not insignificant hardware cost) all incoming instructions to decide whether they can indeed be legally executed in the same cycle.

TM3200 - the Crusoe processor targeted for Internet access devices like web slates and handheld computers.

TM5400 - the Crusoe processor model that is targeted at ultra-light (less than four pounds) mobile PCs.

Transistor - A three-terminal semiconductor device for amplification, switching, and detection, typically containing two rectifying junctions and operating so that the current between one pair of terminals controls the current between the other pair, one terminal being common to input and output.

Translation - in the context of Crusoe processors, a sequence of VLIW molecules that perform the same function as a given sequence of x86 instructions.

Translation cache - in a Code Morphing system, an area of memory that holds previously made translations ready for execution.

VLIW (Very Long Instruction Word) - a CPU architecture in which the processor executes long instruction words (molecules) consisting of several instructions (atoms) each. Unlike superscalar or out-of-order processors, a VLIW does not need to analyze whether instructions can be executed in parallel, since the molecule already explicitly encodes that information. This reduces hardware cost and enables higher processor speeds.

Web slate - a type of Internet appliance that consists of a large LCD screen and a touchpad for user input.

x86 (also 80x86) - refers to the Intel 8086 CPU family used in PCs, which includes the Pentium family. The majority of PC software applications is written specifically for the x86 microprocessor architecture.

x86 compatible - a term commonly used in the computer industry to signify the format and interfaces for a computer software application that will execute correctly on a platform originally known as an IBM-compatible PC.

REFERENCE

- **JOURNALS**

- **IEEE SPECTRUM**, May 2000, "Transmeta's Magic Show"
- **CHIP**, June 2000
- **PC Magazine**, November 2000, "The Mobile Edge"

- **WEBSITES**

- <http://www.transmeta.com>
- <http://www.mdronline.com>
- <http://www.techextreme.com>
- <http://www.ibm.com>
- <http://www.ieee.org>
- <http://www.mit.edu>
- <http://www.eetimes.com>
- <http://www.zdnet.com>