The Intel® Pentium® M Processor: Microarchitecture and Performance

Simcha Gochman, Mobile Platforms Group, Intel Corporation
Ronny Ronen, Microprocessor Research Labs, Intel Corporation
Ittai Anati, Mobile Platforms Group, Intel Corporation
Ariel Berkovits, Mobile Platforms Group, Intel Corporation
Tsvika Kurts, Mobile Platforms Group, Intel Corporation
Alon Naveh, Mobile Platforms Group, Intel Corporation
Ali Saeed, Mobile Platforms Group, Intel Corporation
Zeev Sperber, Mobile Platforms Group, Intel Corporation
Robert C. Valentine, Mobile Platforms Group, Intel Corporation

Index words: Pentium® M processor, microarchitecture, power-aware design, branch prediction, instruction fusion, processor bus, SpeedStep® technology

ABSTRACT

The Intel® Pentium® M processor is a key component of Intel® Centrino™ mobile technology platform. It is Intel's first microprocessor designed specifically for mobility. It provides outstanding mobile performance[1] and its dynamic power management enables energy saving for longer battery life.

Designing a mobile processor calls for different power/ performance tradeoffs than designing a traditional high-performance processor. In this paper we explain the design philosophy that was adopted by the Intel Pentium M processor's architects to achieve best performance at given power and thermal constraints.

We present an overview of the Intel Pentium M processor's major advanced power-aware performance features including the innovative branch predictor, the dedicated stack manager, the micro-operation fusion, and the Intel Pentium M processor bus.

We next describe the Pentium M processor's Enhanced Intel SpeedStep® technology that allows significant reduction in energy consumption without compromising performance.

We conclude with demonstrating the superior performance and power-awareness of the Pentium M processor by comparing it with other mobile processors on a variety of known industry benchmarks.

INTRODUCTION

The distinction between *Mobile* and *Desktop* computing segments is not new. There are several vectors in which these segments differ, two of which are relevant to our discussion: power dissipation and battery life [1].

- *Power, Power Density, and Thermal.* The overall dissipated power, as well as the power dissipated by the chip per unit area, are important factors. Power generates heat. In order to keep transistors within their allowed operating temperature range, the generated heat has to be dissipated from its source in a cost-effective manner. These constraints limit the processor's *peak power* consumption. Peak power consumption limits apply both to desktops and mobile computers. However, the mobile computer's

®Intel Pentium M is a trademark of Intel Corporation or its subsidiaries in the United States and other countries.

®Intel Centrino is a trademark of Intel Corporation or its subsidiaries in the United States and other countries.

[1] System performance, battery life and functionality will vary depending on your specific hardware and software configurations.

®SpeedStep is a trademark of Intel Corporation or its subsidiaries in the United States and other countries.

smaller form-factor and lighter weight decrease the mobile processor's power budget[2].

- *Battery Life.* Batteries are designed to support a certain Watts x Hours[3]. The higher the average power is, the shorter the time that a battery can operate. This constraint limits the processor's *average power* consumption, which is a crucial factor for mobile computers but less relevant for desktop computers.

Until not so long ago, the size of the mobile market was significantly smaller than that of the desktop market, causing mobile PC designers to retrofit processors designed to address the desktop market. Desktop processors were designed to achieve the highest performance possible for all user profiles with little consideration of power consumption. Meeting the more restrictive peak and average power constraints of mobile PCs involved a compromise. New processors were adapted to the mobile market by either operating them at a lower voltage and frequency, hence compromising performance, or by delaying the implementation of their mobile version to the next-generation process technology, hence losing time-to-market.

The increased demand for mobile PCs, combined with the growing gap between desktop and mobile peak and average power budgets, made it impractical to continue the trend of using desktop processors for the mobile market. Mobile users expect to have close-to-desktop performance even with the more restricted power and thermal environment. Addressing this need called for a processor designed with the mobile environment in mind. This is where the Intel Pentium M processor comes in.

The Pentium M processor is Intel's new flagship power-aware microprocessor. Upon introduction (in March 2003) its highest performing version ran at 1.6 GHz @1.47V[4], its Low Voltage version ran at 1.1 GHz @1.18V, and its Ultra Low Voltage version ran at 900 MHz @1.0V. It follows a new design approach with the goal of delivering breakthrough performance at a lower power budget as well as minimizing the processor's average power for extending battery life. The Intel Pentium M processor includes several new innovative features that enable it to meet its design goals.

This new processor has 77 million transistors implemented on Intel's 0.13µ CMOS process, with six levels of copper interconnect. Its die size is 84 mm^2 and its peak power consumption is 24.5 watts at 1.6 GHz. Its 3.2 GB/second processor bus helps to provide the high data bandwidths needed for today's and tomorrow's demanding applications. It fully implements the IA32 instruction set architecture [2], including Streaming SIMD Extension (SSE) and Streaming SIMD Extension 2 (SSE2) targeted for multimedia, content creation, scientific, and engineering applications.

We begin with an overview of the Intel Pentium M processor design philosophy. Then we examine in depth the major innovative power-aware and energy-aware features of this processor. We conclude by comparing the performance and power-awareness of the Pentium M processor with those of other mobile processors.

POWER-AWARENESS PHILOSOPHY AND STRATEGY

Design tradeoffs for the mobile market are rather complicated and involve several challenges:

- Optimizing the design for maximum performance and extended battery life. The challenge lies in how to balance between these conflicting goals.
- Trading performance for power. Performance features–whether increasing instruction-level parallelism (ILP) or speeding up frequency–usually increase power consumption. Power-saving features usually decrease performance. The challenge lies in figuring out how much power one can afford to lose in order to implement a performance feature.

Setting a clear direction for tradeoffs between higher performance and longer battery life and between performance and power was essential for converging the definition of the Intel Pentium M processor.

Higher Performance vs. Longer Battery Life

Early in the design it was realized that the average power consumption for typical usage of the Intel Pentium M processor is only a small portion of the whole platform power consumption–less than 10%. This low average power is mainly due to the ability of the processor to enter lower power states in idle periods and to the Enhanced Intel SpeedStep technology, which significantly reduces power in periods of low processor activity. The majority of the power in the platform is

[2] Currently (2003), typical desktop processor peak power consumption is about 100W. Typical mobile processor peak power is about 30W.

[3] Currently (2003), typical battery capacity is 24-72WxH. A typical Pentium M platform uses 48WxH batteries. A typical platform's power is 13W, of which the processor consumes about 1W on average. Smaller platforms use 24WxH to save weight.

[4] 1.47V is the highest operating voltage of the 0.13µ CMOS process on which the processor is implemented.

consumed by other components: the LCD, the hard disk, the memory system, networking components, etc. Under these circumstances, it was clear that the potential gain in system battery life by further reducing the processor's average power would be small[5]. With that in mind, we decided to optimize the design for the highest performance within power and thermal constraints, when the processor is active, and to focus on battery life when it is idle.

It should be noted that the above may change in the future. We expect high-performing processors' average power to grow due to more complex processor logic and higher static power and new demanding workloads. We expect platforms to become smaller, simpler, and more efficient hence consuming less power. Therefore, the portion of the processor's average power as part of the overall platform power consumption will increase.

Trading Performance For Power

The tradeoffs are different if we optimize for higher performance or for longer battery life. For higher performing mobile processors (and in fact, now, in all high-performing processors) the criterion is *"Maximizing performance at given thermal constraints."* For longer battery life the criterion is "*Minimizing energy per task."* Below, we explain what each criterion actually means.

Maximizing Performance at Given Thermal Constraints

The processor's thermal map depends on the power consumption, the local power density at various points on the die, the cooling mechanism, and more. At the early stages of the Intel Pentium M processor's microarchitecture definition, when thermal information was not available, we replaced the criterion *"Maximizing performance at given thermal constraints"* with: *"Maximizing performance at a given power envelope*" [6].

According to this criterion, a microarchitectural feature that gains performance or saves power should be better than simply using voltage/frequency scaling. For a given working point of core voltage V_0 and Frequency F_0 the power consumption of a processor is given by

$$Power_0 = \alpha * C_0 * V_0^2 * F_0$$

where α is the activity factor, $Power_0$ is the power consumption and C_0 is the effective capacitance for a given design. The frequency is usually approximated as being linearly proportional with the operating voltage, namely

$$F_0 \cong K_f * V_0$$

where K_f is the proportion constant. This leads to the cubic dependency of power on the operating voltage

$$Power_{max} = \alpha * C_0 * V_0^3 * K_f$$

The performance at this operating condition is given by

$$Perf_0 \cong IPC_0 * F_0$$

where IPC_0 indicates the Instruction Per Cycle in Frequency F_0 [7].

It can be derived from the above formulae (see also [3]) that by increasing the voltage by 1%, for example, one can increase performance by 1% through increased frequency. This would result in a power increase of approximately 3%. Thus, an alternate microarchitectural feature that gains less than 1% in performance for a power increase of 3% or more should be rejected upfront. In general, a microarchitectural feature can be regarded as power-aware, if the % ratio between the power increase and the performance gain is less than 3.

Minimizing Energy Per Task

Energy consumption in general is a sum of two components: active energy and idle energy. Minimizing the idle energy consumption is relatively straightforward and does not involve conflicting design tradeoffs: the processor enters a deep-sleep power state, stops the clocks, and lowers the operating voltage to the minimum allowed to sustain the internal state. Optimizing active energy is more complex. A very slow execution consumes less power for a longer period of time, while heavy parallelism reduces the active time but increases the active power.

$$Energy_{active} = Power_{active} * Time_{active}$$

or

$$Energy_{active} \cong Power_{active} / Perf_{active}$$

This implies that in order to improve overall battery life, the % performance benefit must be greater than the additional power consumed.

[5] In the ideal case where the 10% power is totally eliminated, battery life would be extended by only 11%.

[6] The Intel Pentium M processor's design assumed power envelopes from 7W for passively cooled boxes up to 24W for Thin and Light platforms.

[7] For the sake of this discussion, we assume performance scales linearly with frequency. In reality, mainly due to off-chips accesses (memory and I/O), performance does not scale with frequency.

During the definition of the Intel Pentium M processor we tended to use the stricter criterion in each case:

- Performance improvement features were usually included if they *"minimized energy per task,"* that is, they save energy. Features that pass this criterion improve performance and extend battery life–the ultimate win/win situation. Faster execution also implies longer idle time, allowing active units to be shut off, thus saving even more energy.
- Power-saving features may reduce Instructions Per Cycle (IPC) resulting in a performance loss. Such power-saving features were usually included only when they "*maximized performance at given thermal constraints,"* that is, the performance loss was smaller than would have been achieved by just using voltage and frequency scaling. In practice, by applying this criterion, a tradeoff can be made between saved power and increased frequency, thus squeezing more performance at the peak allowed power.
- Performance-improvement features that met the "*maximize performance at given thermal constraints"* criterion, but failed the *"minimizing energy per task"* one, were carefully judged and in many cases included. Such features do increase performance but consume more energy. This loss is negligible, since, as mentioned above, the processor's average power as a portion of the overall system average power is relatively small. In fact, the faster execution results in a longer idle time, potentially allowing additional energy savings.
- Figure 1 illustrates the design tradeoffs from a performance feature viewpoint. The magenta area (in the lower right side) indicates a clear win–improving both performance at the power envelope and battery life. The orange area (resides in the upper right side) indicates a tradeoff where constrained performance is preferred over lower battery life. The green area (resides in the lower left side) indicates a tradeoff where improved battery life is preferred over constrained performance. The white area (in the upper left side) is a clear "drop."

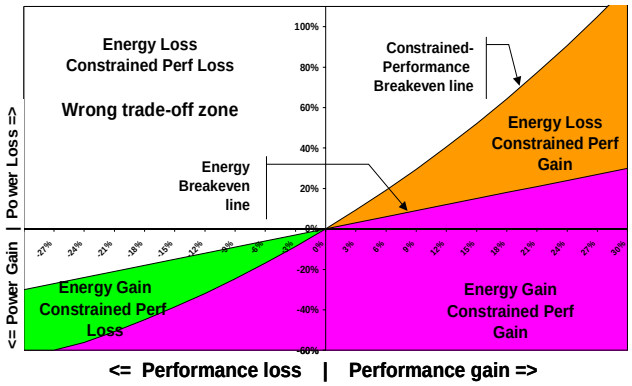

Figure 1: Performance/power tradeoff zones

Now that the tradeoffs are known, we examine the strategies we used to identify power-aware features. Power-awareness means attacking power and energy consumption at all levels:

- Reducing number of instructions per task.
- Reducing number of micro-ops per instruction.
- Reducing number of transistor switches per micro-op.
- Reducing the amount of energy per transistor switch.

Reducing the Number of Instructions Per Task

From an architectural point of view the number of instructions per task is fixed. However, from a microarchitectural point of view, this is true only for the number of retired instructions. With branch prediction, there are many speculated instructions running within the processor that are not retired. A better branch predictor decreases the number of the speculated instructions, thus practically reducing the number of overall processed instructions. Indeed, the extra logic involved in a better branch predictor does consume power, but the gain in the reduced number of instructions exceeds that extra cost.

Reducing the Number of Micro-ops Per Instruction

Out-of-order implementations of the IA32 Instruction Set Architecture (ISA) break macro-instructions into a sequence of one or more simple operations, called micro-operations, or *micro-ops* [4]. Handling and executing each micro-op consumes power. Eliminating micro-ops from the micro-op stream or combining several micro-ops together reduces the overall power. The Intel Pentium M processor micro-ops fusion and dedicated stack engine do exactly that. Here, as well, the gain in reduced micro-ops exceeds the cost of the extra logic involved.

Reducing the Number of Transistor Switches Per Micro-op

This is more straightforward and intuitive. High performance processors run at a high frequency and

provide a high degree of instruction-level parallelism–switching a lot of transistors on the way. There is a clear gain in doing the same operation with a smaller number of switches. In some cases, this is simple and involves only local optimizations such as accessing only the portion of a register or a cache line that is actually needed. In other cases, it calls for a global optimization to decide whether a unit will not be used for the next cycle, and thus can be shut off. Occasionally, such power saving may involve a performance loss.

Reducing the Amount of Energy Per Transistor Switch

Energy per switch depends on the transistor size and type, and on the operating voltage. Smaller transistors and lower operating voltages reduce energy per switch [5]. The effect of microarchitecture here is rather limited. Transistor size and type are tuned so that they meet the timing constraints without wasting unnecessary power. The Enhanced Intel SpeedStep technology reduces the operating voltage at low activity periods, thus reducing the energy per transistor switch. Microarchitecture can help reduce energy per switch by optimizing the amount of interconnect in the processor.

Each strategy affects performance, power, and energy in a different way. In most cases, features that fall under one of these strategies save energy per task. However, performance-improving features are likely to result in increased power consumption, e.g., better branch prediction reduces energy, but also reduces stalls, thus increasing power. This may look bad, but, in fact, it is not. Higher performance at lower energy can be traded for lower power by slowing down the processor either by using voltage/frequency scaling or microarchitectural throttling.

Static Power

The power consumed by a processor consists of active power (used to switch transistors) and static power (leakage of transistors under voltage). In this paper we focus mainly on active power reduction, but it is worth mentioning how the Intel Pentium M processor also reduces static power consumption.

The static power is roughly a function of the number of transistors, their type, the operating voltage, and the die temperature. The Pentium M processor reduces static power by several means:

- *Low-leakage devices*. The processor's 1MB power managed L2 cache, which contains roughly two-thirds of the transistors in the processor, is built with low-leaking transistors. Low-leaking transistors are somewhat slower, thus slightly increasing the cache access latency, but the significant power saved justifies the small performance loss.
- *Enhanced Intel SpeedStep technology*. This advanced technology significantly reduces the processor voltage (and temperature), hence leakage power, when processor activity is low.

POWER-AWARE FEATURES

The following sections describe several of the Intel Pentium M processor's power-aware features. These features cover all the above-mentioned strategies:

- Reducing the number of instructions per task: advanced branch prediction.
- Reducing the number of micro-ops per instruction: micro-ops fusion and dedicated stack engine.
- Reducing the number of transistor switches per micro-op: the Intel Pentium M processor bus and various lower-level optimizations.
- Reducing the amount of energy per transistor switch: Intel SpeedStep technology.

ADVANCED BRANCH PREDICTION

For high-frequency pipelined microprocessors, branch prediction continues to be one of the biggest ticket items for gaining performance. In the Intel Pentium M processor, the benefits are actually twofold: the decrease in speculative code gains performance and reduces the energy spent per instruction retired.

The advanced branch prediction in the Pentium M processor is based on the Intel Pentium® 4 processor's [6] branch predictor. On top of that, two additional predictors to capture special program flows, were added: a Loop Detector and an Indirect Branch Predictor.

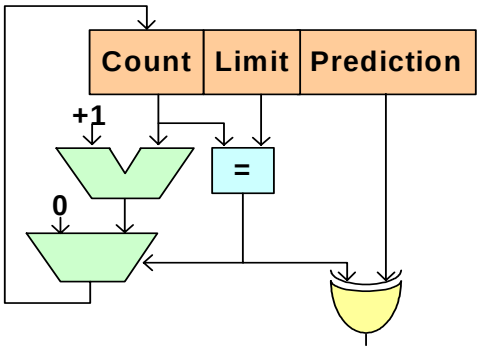

Figure 2: The Loop Detector logic

®Pentium 4 is a registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

The Loop Detector (Figure 2) analyzes branches to see if they have loop behavior. Loop behavior is defined as moving in one direction (taken or not-taken) a fixed number of times interspersed with a single movement in the opposite direction. When such a branch is detected, a set of counters are allocated in the predictor such that the behavior of the program can be predicted completely accurately for larger iteration counts than typically captured by global or local history predictors.

The Indirect Branch Predictor solves the problematic data-dependent indirect branches. Indirect branches are heavily used in object-oriented code (C++, Java), hence they became a growing source of branch mispredictions. While most indirect branches have a single target at run time, some, such as a case statement in a byte-code interpreter, may have many targets. These targets are chosen in a data-dependent manner.

The Indirect Branch Predictor (Figure 3) chooses targets based on a global control flow history, much the same way a global branch predictor chooses the direction of conditional branches using global control flow history. As can be seen in the figure, it is an adjunct to the normal target prediction device. Targets are always allocated in the Instruction Pointer tagged table along with the type of branch. When a misprediction occurs due to a mispredicted target on an indirect branch, the Indirect Branch Predictor allocates a new entry corresponding to the global history leading to this instance of the indirect branch. This construction allows monotonic indirect branches to predict correctly from the IP-based Target array, and data-dependent indirect branches to allocate as many targets as they may need for different global history patterns, which correlate with the different targets. Entries in the Indirect Branch Predictor are tagged with the hit and type information in the IP-based target array to prevent "false positives" from the Indirect Branch Predictor to lead to new sources of target mispredictions.

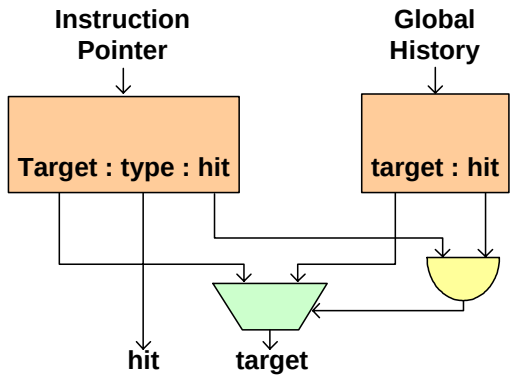

Figure 3: The Indirect Branch Predictor logic

The Intel Pentium M processor branch predictor misprediction rate is 20% lower than that of previous generation designs, resulting in as much as 7% in real performance. Approximately 30% of this benefit comes from the combination of the Loop Detector and Indirect Branch Predictor. The Loop Predictor captures a common program behavior and benefits many applications regardless of compilation techniques. The Indirect Target Predictor shows its gains more in specific applications where indirect branches are used to select data-dependent targets. In such applications, when the compilers use calculated branches rather than if-trees made from conditional branches, performance gains can be a few percentage points.

MICRO-OPS FUSION

Out-of-order implementations of the IA32 Instruction Set Architecture (ISA) break macro-instructions into a sequence of one or more simple operations, called micro-operations, or *micro-ops*. A conventional micro-op consists of a single operation operating on two sources. The Instruction Decoder breaks a macro-instruction into multiple micro-ops whenever the macro-instruction operates on more than two sources or when the nature of the operation requires a sequence of unrelated operations. There are quite a few cases of macro-instructions that break into several micro-ops, two of which are store operations and load-and-op (read-modify) operations.

Macro-instructions that store data in memory are decoded as two independent micro-ops. The first operation–*store-address*–calculates the address of the store, while the second operation–*store-data*–stores the data into the Store Data buffer[8]. The separation between the store-data and the store-address operations is important for memory disambiguation. Breaking the store operation into two micro-ops allows the store-address operation to dispatch earlier, even before the stored data are known, enabling resolution of address conflicts and opening the memory pipeline for other loads.

A typical load-and-op macro-instruction consists of two micro-ops: the first operation reads the operand from an address in memory, and the second operation calculates the result based on the data read from memory and the register operand. A load-and-op macro-instruction may have up to three register operands, so it must be implemented by two micro-ops. The atomic operations

[8] The actual write to memory is done when the store retires. Only then are the data in the respective Store Data buffers written into the specified address.

are inherently serial, and the second operation cannot start until the first operation completes.

Splitting the macro-instruction into multiple micro-ops also has its toll:

- The increased number of micro-ops creates pressure on resources with limited bandwidth (rename, retire) or limited capacity (Reorder-Buffer, Reservation-Station). This pressure eventually results in performance loss.
- Splitting a macro-instruction into more than one micro-op is a complex operation that requires a significantly more capable decoder. Due to its complexity, most implementations opt to have only one complex decoder; all other decoders are left to handle macro-instructions that break only into a single micro-op.
- Delivering more micro-ops through the system increases the energy required to complete a given instruction sequence.

The Pentium M processor features the micro-ops fusion mechanism to reduce this performance and energy cost while maintaining the benefit of the out-of-order execution. With micro-ops fusion, the Instruction Decoder *fuses* two micro-ops into one micro-op and keeps them united throughout most parts of the out-of-order core of the processor–at allocation, dispatch, and retirement. To maintain their non-fused behavior benefits, the micro-ops are executed as non-fused operations at the execution level. This provides an effectively wider instruction decoder, allocation, and retirement. Figure 4 describes the different domains in which the micro-op is fused and unfused.

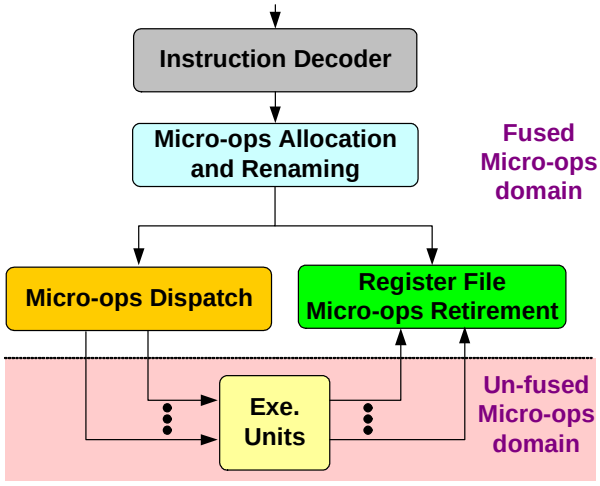

Figure 4: Micro-ops fusion domains

The macro-instruction is decoded into a single fused micro-op by the Instruction Decoder. The fused micro-op is allocated, renamed, and then issued into a single entry in the Reorder-Buffer and the Reservation-Station. To support fused micro-ops, each reservation-station entry can accommodate up to three source operands. When dispatching to the execution units, the Dispatcher controls the separate execution of each portion of the fused micro-op according to the readiness of its sources. In a sense, the Dispatcher treats each portion as if it occupied the whole entry for itself. The Execution of each operation is performed in the same way as a non-fused micro-op with only minor changes made to the execution units.

The fused store operation is depicted in Figure 5.

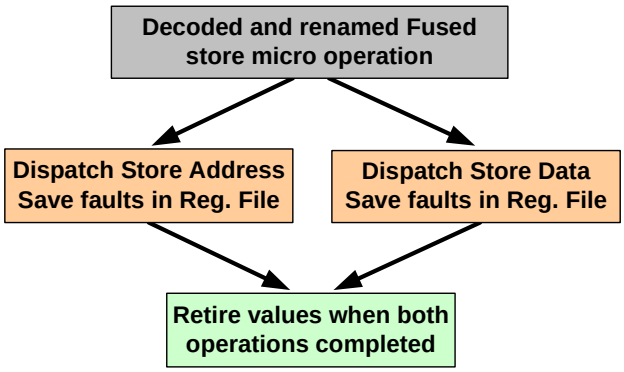

Figure 5: Fused store flow

The two micro-ops making up the fused store micro-op can be issued to their relevant execution units in parallel. The dispatch of the store-address operation to the address-generation unit is performed when its sources (the base and index registers) are ready. The dispatch of the store-data operation to the store data buffer unit can occur independently when its source operand is available. The retirement of the fused store can occur only after both operations complete.

The fused load-and-op operation is depicted in Figure 6.

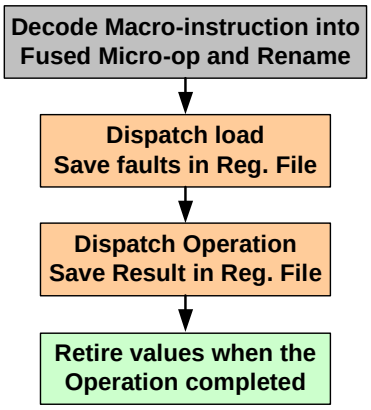

Figure 6: Fused load-and-op flow

The two micro-ops making up the fused load-and-op micro-op are issued serially to the relevant execution units. The dispatch of the load operation is performed when its sources (the base and index registers) are ready. The dispatch of the "op" portion of the load-and-op operation to the execution unit can occur only after the load completes and the other operand is ready. The retirement of the fused load-and-op micro-op can occur only after both operations complete.

We have found that the fused micro-ops mechanism reduces the number of micro-ops handled by the out-of-order logic by more than 10%. The reduced number of micro-ops increases performance by effectively widening the issue, rename, and retire pipeline. The biggest boost is obtained during a burst of memory operations, where micro-op fusion allows all decoders, rather than the one complex decoder, to process incoming instructions. This practically widens the processor decode, allocation, and retirement bandwidth by a factor of three.

The typical performance increase of the micro-op fusion is 5% for integer code and 9% for Floating Point (FP) code. The store fusion contributes most of the performance increase for integer code. The two types of fused micro-ops contribute about equally to the performance increase of FP code.

Delivering less micro-ops through the processor decreases the energy required to complete a given instruction sequence since the same task is accomplished by processing fewer micro-ops. The gain in power consumption is higher than the loss, due to the additional logic required to implement the micro-ops fusion mechanism.

DEDICATED STACK ENGINE

The IA32 Instruction Set Architecture (ISA) features instructions for hardware-assisted stack management that are typically used to implement a combined parameter and control-flow stack used in high-level programming languages. The ISA provides PUSH, POP, CALL, and RET, which have the obvious parameter stack and control-flow stack behaviors. The ISA dedicates the hardware Stack Pointer register (ESP) as the machine stack pointer, and this register is modified as a side effect of each of these instructions. Sequences of such instructions are quite common, for instance, PUSHing a set of operands and then using a CALL instruction is the standard mechanism for making a Procedure or Function Call.

In traditional out-of-order implementations of the IA32 ISA, these side-effect operations were performed by sending with each stack-related macro-instruction an additional micro-op to update the ESP register. This micro-op adds or subtracts an immediate value to the ESP register.

The Intel Pentium M processor chose to implement the ESP "side-effect" behavior in a more efficient way, using dedicated logic near the superscalar decoders. The idea is to represent the programmer's view of ESP (ESP_P) at any given point in time by some historic ESP living in the out-of-order execution core (ESP_O) added to a delta (ESP_D) that is maintained in the front end (see also [7]):

$$ESP_P := ESP_O + ESP_D$$

When, for example, a sequence of PUSHes and POPs is encountered in the instruction stream, the dedicated Stack Hardware executes the ESP side-effects in the decoders and updates the ESP_D register. Referring to Figure 7, we can see a superscalar implementation for N+1 decoders passing the accumulated delta value across the decoders and updating the delta register with the result, after the instructions are decoded. The hardware also patches the in-flight ESP_D value into the address syllable of each of the stack referencing micro-ops (patch HW not shown for clarity) so that the address generation unit (AGU) can calculate the proper memory location referenced by ESP_P. This provides the following benefits:

- Dependencies on ESP are removed since the ESP_O value used for scheduling in the out-of-order machine is not changed during the sequence of stack operations. This allows more parallelism opportunities to be realized in the out-of-order execution.
- ESP_D updates are done using a small specialized dedicated adder, thus freeing the general execution units to work on other micro-ops. This allows a higher degree of superscalarity for these instructions

without the cost associated with going to a higher degree of superscalarity for all integer operations. Additionally, since the ESP updating micro-ops have been eliminated, the ALUs are free to be utilized by more complex operations that would have been blocked by the ESP updates, increasing execution bandwidth.

- Updating the delta register in the front end eliminates the ESP updates micro-ops from the out-of-order machine. Thus, power saving is realized since the large adders are not used for small operations, and the eliminated micro-ops do not toggle bits throughout the machine.

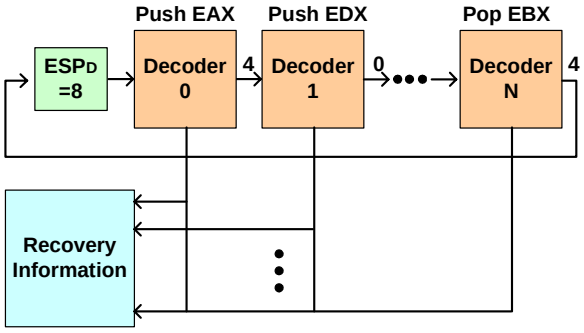

Figure 7: The dedicated stack engine logic

There are two complications with the Dedicated Stack Engine. Since it lives in the front of the pipeline, all its calculations are speculative. In order to recover a precise state at any point in the machine's life, the value of ESP_O and ESP_D must be able to be recovered for all instructions in the machine. ESP_O is maintained by the out-of-order core as any other general-purpose register. The Intel Pentium M processor adds an additional table (also shown in Figure 7) that saves the ESP_D value and a code relating to the effect on the ESP_D register for every instruction in the machine. This allows the value of ESP_P to be recovered for all instructions either pre- or post-execution. This allows for handling of either Faults or Traps as defined in IA32.

The second complication occurs when the architectural value of ESP is needed inside the out-of-order machine, for instance, *"XOR ESP,3"* or, more commonly, when ESP is used in an address syllable. In this case, the decode logic automatically inserts a micro-op that carries out the ESP_P calculation. The ESP_D register can be cleared since the architectural value is now coherent. A sync is not generated when the ESP_D register is zero, so continued usage of ESP as a general-purpose register will have no ill effects.

The Dedicated Stack Engine ESP typically eliminates 5% of the micro-ops from an IA32 program compared to a processor not including this feature, even when including the ESP synchronization micro-ops. Clearly this makes the out-of-order machine look virtually larger and frees execution bandwidth. However, the major performance gain is to increase the front-end bandwidth on these common instruction sequences to the full width of the superscalar decoders. These 5% of eliminated micro-ops result in a similar decrease in energy per instruction–in line with the Pentium M processor's power-awareness direction.

THE INTEL PENTIUM M PROCESSOR BUS

The Intel Pentium M processor bus was designed to provide a desktop-like performance while consuming significantly less power. Power saving is achieved by the combination of protocol and circuit methods that are unique to the Pentium M processor bus.

The processor bus supports 100 MHz bus clocks with a data rate of 400M transfers per second. It is a latched bus with an in-order queue of 8-pipelined transactions. Designed for mobile systems, the bus is optimized for a uni-processor environment. For example, it allows us to reduce the number of pins to save power:

- There are only 32 address bits that cover 4GB of physical address space.
- The bus does not support dual-processors, since the mobile systems' power budget cannot support dual processors anyway.

The Pentium M processor bus saves power aggressively when idle; it carefully controls its input buffer's sense-amplifiers that sample the activity on the bus. When the bus is idle, all sense amplifiers are disabled and do not consume any power. When the bus is active and address and data are driven on the bus, the input buffers are enabled in advance to ensure all information is captured with no delay.

The bus features many innovative mechanisms to reduce power while maintaining performance. Several of them are described below.

DPWR#: Data Bus Power Control. This is a special signal driven by the 855PM chipset whenever data are transferred to the processor. DPWR# is used to dynamically enable the processor's 64-bit data bus input sense amplifiers and their related controls (~80 signals) only when data are transferred to the bus.

BPRI Control: This is a method to achieve the DPWR# functionality for the address bus. BPRI# is asserted whenever the 855PM chipset attempts to drive the bus. It is used to dynamically enable the 32-bit address bus

input sense amplifiers and their related controls (~40 signals) only when a transaction is issued to the bus.

Low Vtt: The Intel Pentium M processor's I/O buffers work at a low voltage of 1.05V (Vtt). The low Vtt is an essential element to reduce the bus power. However, operating at low Vtt introduces a new set of problems because the I/O buffer is working at the low linear point, which affects the buffer's characteristics. The bus includes a special Resistor Compensation (RCOMP) method to adjust the buffer strength dynamically during run time. It accommodates the impacts of temperature, voltage drift, and bus topology. Thus, at any thermal and power state the Pentium M processor bus has full impedance termination. It has split power planes that allow setting the I/O operating voltage to a fixed value of 1.05V even though the core may be operating at a higher Enhanced Intel SpeedStep technology operating point.

PSI: Power Status Indicator. The Pentium M processor bus provides a signal to reduce the overall platform power (not just the processor power!). This signal is driven by the processor to control the current consumption of the Voltage Regulator (VR) when the processor operates at a low power state.

LOWER-LEVEL POWER OPTIMIZATIONS

This section describes several lower-level mechanisms that demonstrate the power awareness of the Intel Pentium M processor.

A simple, yet effective method that was pursued in order to reduce power was to identify idle logic and shut it off. This was done locally and globally.

Locally, the design was thoroughly reviewed for any inefficiency during idle states. The goal was to gate the clocks as much as possible. Ideally, the clocks should be shut off for each pipe stage separately. However, if such a naive approach is used, the added complexity may sometimes outweigh the gain. In these cases the logic is shut off for the entire unit only at the end of the operation, resulting in a smaller power saving.

Globally, idle time identification is done at a higher microarchitectural level, when the unit alone cannot identify the idle period. For example, the first stage of a unit is always kept awake in order to respond to incoming messages. So, the goal was to create a few central controllers that can microarchitecturally identify or predict idle periods and instruct the units to reduce power (either by shutting off their clocks or by disabling parts of their logic). Also, the prediction logic should allow operations to resume seamlessly with no performance penalty.

One example for such a power predictor is the "Allocate stall" predictor. Whenever the Reorder-Buffer is full, the Allocator stalls the pipeline. However, the Allocator cannot tell if the Reorder-Buffer will remain full on the next cycle. It therefore needs to re-evaluate the stall condition every cycle. It turns out that in many cases when the Reorder-Buffer is full, it stays so for very long periods. Therefore the power penalty in this case is high. A specialized logic was defined to collect information from the Reorder-Buffer and other units in order to predict the nature of the next cycle. This logic instructs the Allocator to hold on to the stall condition and shut off its clocks.

Another type of a microarchitectural power feature identifies the logic that is necessary for a specific operation and activates only that part of the processor. Here are two examples for an implementation of such a power feature.

In a conventional processor, all the units of a specific execution port electrically share the same source bus wires. However, power can be saved if instead of driving the sources to all the execution units (EUs), only the wires that belong to the target EU are driven. Therefore, the Pentium M processor execution units were divided into a few segments (stacks), and a special logic was created in order to control the flow of data to every stack according to its actual destination (see Figure 8).

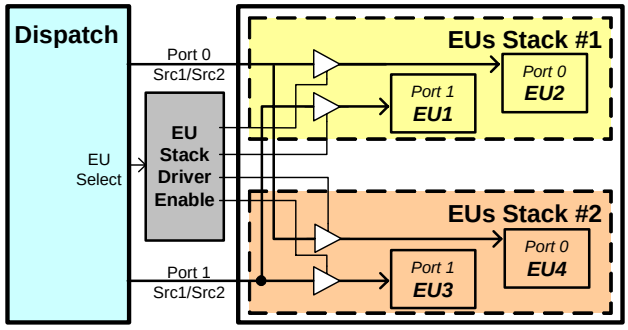

Figure 8: Execution units stacking

A generic processor operates on several different data types with different widths. In an IA32 processor there are integer operations, operating on 32 bits, multimedia operations, operating on 64 bits or 128 bits, and floating-point operations, operating on 80 bits. The most common instructions are integer instructions that access only a limited set of registers and use only 32-bit data values. Toggling a wider bus and reading from a bigger register file consumes more power than is actually

required. The Intel Pentium M processor saves power by identifying integer operations in advance and activating only the appropriate hardware (see Figure 9). The savings include the narrower buses to and from the EU during dispatch and writeback, and also other elements in the renaming logic that are not accessed while an integer operation is executing. This effectively transforms the processor into a 32-bit machine that utilizes only resources needed for integer operations while operating on integer data types.

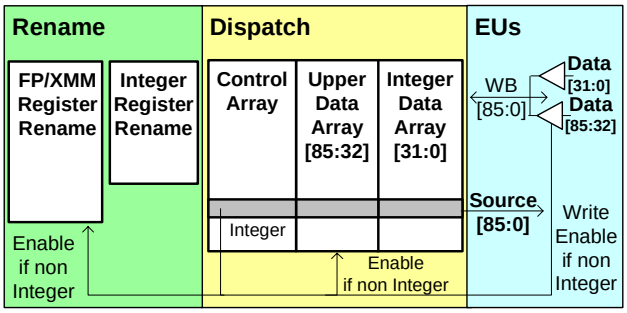

Figure 9: Early identification of EU width

ENHANCED INTEL SPEEDSTEP TECHNOLOGY

High-performance processors tend to have high power consumption during execution. This is a simple derivative of the active power equation

$$Power = \alpha * C * V^2 * F$$

where V is the core voltage, F is the operating frequency, and α is the activity factor. However, proven mobile usage models, indicate that typical usage is bursty in nature, requiring high performance only for short bursts of time. Average power reduction can be achieved by switching voltage and frequency to a lower operating point, when demand is low. The efficiency of the solution depends on the ability to execute this operation-point switch frequently and efficiently, to track demand.

Previous generations of Intel mobile processors implemented the Intel basic SpeedStep technology [8]. It switches both voltage and frequency between two distinct states: Lowest Frequency Mode (LFM) and Highest Frequency Mode (HFM) by using the platform C3-idle state. While achieving low-power operation during LFM, the basic SpeedStep architecture does not fully address demand-based switching needs. The long system unavailability time during transitions limits the switching frequency due to interaction with streaming devices such as Audio Codec '97 (AC '97) and the Universal Serial Bus (USB). Additionally, having only two fixed operating points limits operating point optimization according to the load. The Intel Pentium M processor introduces a multi-point Enhanced Intel SpeedStep technology optimized for demand-based switching.

The Enhanced Intel SpeedStep technology attempts to address the following challenges:

- *Minimizing system and processor unavailability.* Operating point switching requires voltage to be transitioned over a wide range (e.g., from 0.9V to 1.5V). Physical limitations of the power delivery system translate this demand to over 100µs delay. A full clock generator Phase-Locked-Loop relock requires approximately 30µs. The architecture needs to ensure system memory access unavailability will not exceed 10-15µs, to match isochronous device needs.
- *Self-managed voltage and frequency stepping.* The Enhanced Intel SpeedStep technology requires the migration of the mechanism from the chipset into the processor. This introduces two challenges: (a) how to sequence the operation when the processor clock is halted and (b) how to prevent loss of system events, such as interrupts and snoops, previously blocked by the chipset during the transition.

Figure 10 depicts the high-level block diagram of the Enhanced Intel SpeedStep technology instantiation in the Intel Pentium M processor.

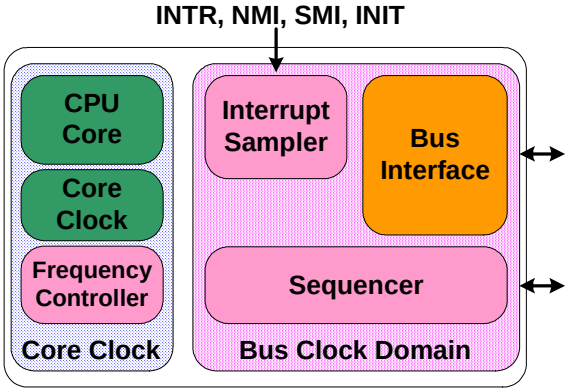

Figure 10: Enhanced Intel SpeedStep technology block diagram and clocking

The Enhanced Intel SpeedStep technology uses three novel principals to address the challenges stated above:

1. *Voltage-Frequency switching separation.* Unlike previous architectures, the Enhanced Intel SpeedStep technology separates the voltage and frequency transition stages (Figure11).

Voltage is stepped in short increments, preventing clock noise and allowing processor execution during the voltage transition stage. Thus, system memory and the processor are made available during the longest segment of the operating point transition, thereby minimizing unavailability time to only the frequency transition stage.

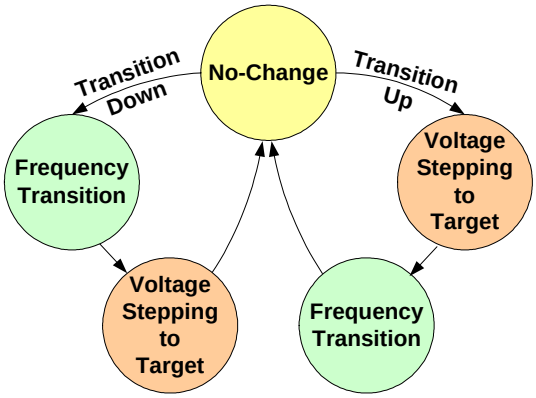

Figure 11: Enhanced Intel SpeedStep® technology transition sequencing

2. *Clock partitioning and recovery.* During the Enhanced Intel SpeedStep technology transition, only the core clock and Phase-Locked-Loop are stopped, while the bus-clock is kept running. The Enhanced Intel SpeedStep technology logic was partitioned such that only the command interface and core controls operate on the core clock, while the sequencer and interrupt interface operate on the bus clock. Thus the logic can be kept active constantly, even though the core clock has been halted.

   Additionally, the clock circuitry of the Pentium M processor was designed to utilize the active bus-clock to shorten core-clock relock time considerably. Thus core-clock restart time is set to only 10µs, minimizing the processor inactive time.

3. *Event blocking.* Interrupts, pin events, and snoop requests sent during the frequency transition stage must not be lost, even though the core clock is not available to serve them.

   The Enhanced Intel SpeedStep technology logic samples all pin events when the core clock is stopped. These are re-sent to the processor once the core clock is available, preventing loss of events.

   Bus events (such as snoops and interrupt messages) are blocked off using the native BNR# protocol, which captures the bus for the frequency transition period. Thus bus and pin events are not missed; they are serviced once the core is capable and running.

Consequently, the Enhanced Intel SpeedStep technology provides the Pentium M processor a flexible, multi-point operating mode, completely self managed, and with a very low CPU and memory unavailability time, which optimizes its power and performance according to demand.

PERFORMANCE

The Intel Pentium M processor architecture delivers breakthrough mobile performance and enables extended battery life in notebook PCs.

Comprehensive information about this processor performance can be found in [9]. In this paper, we choose to demonstrate the power-awareness of the Pentium M processor. We measured the processor performance and average processor power consumption in various operation modes on several benchmarks and compared its performance and efficiency with those of other mobile processors in similar configurations[9]. Efficiency reflects energy per task. Benchmark efficiency is measured by dividing the benchmark performance (1/execution-time) by the average processor power of that benchmark. As will be shown, the Pentium M processor exhibits higher performance and superior efficiency.

The set of benchmarks includes (see more in Table 1):

- Mobile Representative Office Productivity Workload
- Internet Experience workload
- SPEC CPU 2000 V1.2 [10]

We compared the Pentium M processor with the following: (see Table 2 for detailed system configurations):

- Intel® Pentium® M processor (1.6 GHz/600 MHz)
- Mobile Intel® Pentium® 4 Processor - M (2.4/1. 2 GHz)
- Mobile Intel® Pentium® III Processor - M (1.2 GHz/ 800 MHz)

All system run the Windows* XP* operating system. The operation modes used are as follows:

[9] The information included in this section was prepared specifically for this paper to provide insight into the success of the design criteria using known benchmarks as a workload. Some measurements were collected on reference boards that are not publicly available. Therefore, these results should be considered only as an estimate for relative performance and efficiency.

- Always On (Max Frequency)
- Portable/Laptop (Adaptive Frequency)
- Maximum Battery (Min Frequency)

All scores are normalized to the Intel Mobile Pentium 4 Processor - M (orange bar) score. The efficiency–performance over power–is obtained from the benchmark performance score divided by the average processor power for the duration of that benchmark.

Always On Mode

In the Always On mode the processor always runs at its highest frequency. This mode is mostly used when the system is connected to an AC power source. This mode demonstrates the inherent performance and power-awareness of the Intel Pentium M processor without utilizing the Enhanced Intel SpeedStep technology.

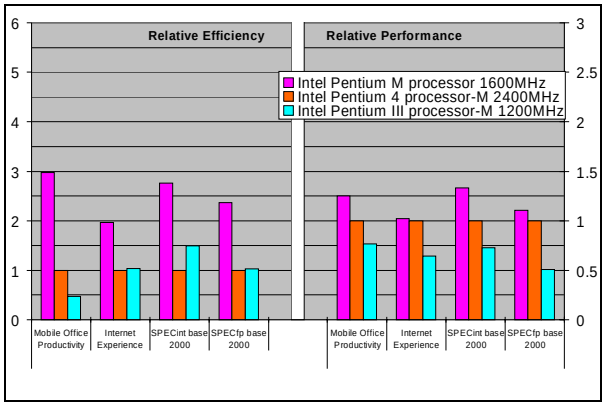

Figure 12: Always On mode performance and efficiency

Figure 12 presents comparative performance and efficiency results in the Always On mode.

In this mode, the Intel Pentium M processor performs equal or better (2%-25%) than the Mobile Intel Pentium 4 Processor - M on all benchmarks, and is significantly more efficient (2X-3X) than it. This shows that the Pentium M processor power-awareness philosophy works: it does more work and consumes significantly less power in the same thermally constrained environment. The Intel Mobile Pentium 4 Processor - M, designed for higher performance at higher power envelopes, cannot exploit its full performance potential in this thermally restricted environment and has to slow down.

SPECint_base2000 and SPECfp_base2000 are of particular interest here. The Pentium M processor exhibits nearly double the efficiency advantage over the Intel Mobile Pentium 4 Processor - M on most of the tests in these benchmarks. Several tests, *179.art* and *300.twolf*, exhibit an even greater efficiency gain (over 8X and 5X respectively) mainly due to the large 1MB power-managed L2 cache of the Intel Pentium M processor.

Portable/Laptop Mode

In Portable/Laptop mode, frequency and voltage changes depend on the application demand. This mode is the normal usage mode when the system is not connected to an AC power source. This mode demonstrates the effectiveness of combining the performance and power-awareness of the Pentium M processor with the energy-saving nature of the Enhanced Intel SpeedStep technology to provide end users with breakthrough mobile performance and extended battery life.

In this mode, all three processors operate between their highest and lowest frequency operating points, depending on the amount of work to be done. For processor-intensive workloads, each processor operates at its highest operating voltage and runs at its maximum frequency: the Intel Pentium M processor @ 1.6 GHz, the mobile Intel Pentium 4 Processor - M @ 2.4 GHz, and the Mobile Intel Pentium III Processor - M @ 1.2 GHz. When there is no activity (idle period), each processor runs at its lowest frequency and voltage to conserve energy: the Intel Pentium M processor @ 600 MHz, the Mobile Intel Pentium 4 Processor - M @ 1.2 GHz, and the Mobile Intel Pentium III Processor - M @ 800 MHz. Using the efficient switching algorithms of the Enhanced Intel SpeedStep technology, the Pentium M processor is transparently switched between the highest and lowest frequency and voltage states, giving the user the best of both worlds: maximum performance under demanding applications and lowest power during idle periods.

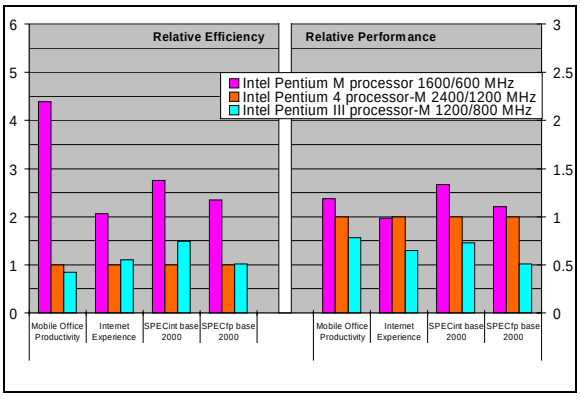

Figure 13: Portable/Laptop mode performance and efficiency

*Other brands and names are the property of their respective owners.

Figure 13 presents comparative performance and efficiency results in the Portable/Laptop mode. Results show that in this mode, the Pentium M processor performs equal or better (0%-30%) than the Mobile Intel Pentium 4 Processor - M (a similar advantage to the Always On mode). However, the relative efficiency over the Mobile Intel Pentium 4 Processor - M in benchmarks that exhibit periods of lower activity went up, e.g., from 3X to over 4X on the mobile representative Office Productivity workload. This improved efficiency results from the much lower power consumption of the Intel Pentium M processor at its low frequency mode @ 600 MHz compared with the power consumption of the Mobile Intel Pentium 4 Processor - M @1.2 GHz.

Maximum Battery Mode

In the Maximum Battery mode the processor runs at its lowest frequency. This mode is usually used when the user is away from an AC power source for a long time. This mode demonstrates the ability of the Intel Pentium M processor to minimize energy consumption when longer battery life is crucial.

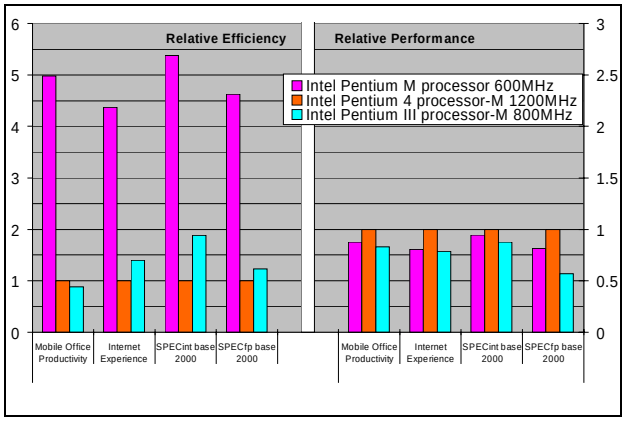

Figure 14: Maximum Battery mode performance and efficiency

Figure 14 presents comparative performance and efficiency results in the Maximum Battery mode. When all three platforms are locked at their lowest processor frequencies, the Intel Pentium M processor running at 600 MHz draws much lower power than the other two processors. This puts the Pentium M processor in a better position to get more work done for the power consumed under the workloads tested. The processor does compromise performance in this mode: it is about 20% slower than the Mobile Intel Pentium 4 Processor - M. However, it is extremely more efficient–about 5X more–allowing it to do significantly more work with the same energy.

Table 1: Benchmark description

Mobile Representative Office Productivity Workload: Targeted to evaluate notebook user experience under popular business-oriented applications in a Microsoft Windows* operating environment. Some usage models represented in this productivity workload include applications from Microsoft Office XP* (i.e., Word 2002, Excel 2002, PowerPoint 2002, Outlook 2002), McAfee* VirusScan*, Adobe* Photoshop*, WinZip* and others.
Internet Experience Workload: Measures PC client performance under a range of popular Internet technologies such as SSL*, XML*, VML*, Java*, etc. using applications such as Adobe Acrobat*, Apple Quicktime*, Cycore*, Cult3D*, Macromedia Flash*, Windows Media Player*, and RealNetworks RealVideo*.
SPEC CPU2000: The industry-standard benchmark that evaluates compute-intensive integer and floating-point application performance [10].

Table 2: System configurations

Platform	Dell Latitude C610	Intel Reference Platform	Intel Reference Platform
CPU	Mobile Intel® Pentium® III Processor-M	Intel® Pentium® M Processor	Mobile Intel® Pentium® 4 Processor-M
CPU Core Freq (MHz)	1200/800	1600/600	2400/1200
CPU Bus Freq	133	400	400
L2 Cache (KB)	512	1024	512
Chipset	Intel 830M	Intel 855PM	Intel 845
Mem Size (MB)	512		
Mem Type/Speed	PC133	DDR 266	
Mem CAS Latency	2-2-2	2-3-3	
Graphics Core	ATI Mobility Radeon M6	ATI Radeon 9000	
Graphics Mem	16MB	64MB	
Gfx Driver	6.13.10.3293	6.13.10.6200	
Screen Resolution	1024 x 768 x 32bpp 60Hz		
HDD Mfr/Model	IBM IC25N040ATCS05-0		
HDD Size, Buffer, RPM	40GB IDE 8MB 5400RPM		
OS, Build, File System	WinXP, SP1 5.1.2600, FAT 32		
LAN	Intel ICH3 Integrated Ethernet Ctrl.	Intel ICH4 Integrated Ethernet Ctrl.	Intel ICH3 Integrated Ethernet Ctrl.

* Other brands and names are the property of their respective owners.

CONCLUSION

The Intel Pentium M processor is Intel's first microprocessor designed specifically for the requirements of tomorrow's mobile PCs. It provides uncompromised performance while observing the thermal and energy requirements and limitations of the mobile platform. Performance-enhancement features were included only if proved to be power-efficient. The processor features many novel power-aware performance mechanisms such as advanced branch prediction, micro-operation fusion, a dedicated stack engine, and the optimized Pentium M bus. It also features the Enhanced Intel SpeedStep technology to reduce energy consumption.

These unique features enable the Pentium M processor to deliver breakthrough performance and enable extended battery life thereby providing users with a superior mobile experience.

ACKNOWLEDGMENTS

The authors thank all the researchers, architects, designers, and validators who took the Intel Pentium M processor from a vision to a real product. Special thanks go to Wilson Huang from Intel's Mobile Platforms Group who conducted the performance and power measurements used in this paper.

REFERENCES

[1] R. Ronen, A. Mendelson, K. Lai, S.L. Lu, F. Pollack, and J.P. Shen, "Coming Challenges in Microarchitecture and Architecture," in *Proceedings of the IEEE*, Vol. 89, No. 3, March 2001, pp. 325-340.

[2] *IA-32 Intel Architecture Software Developer's Manual Volume 1*: Basic Architecture at http://developer.intel.com/design/pentium4/manuals/245470.htm

[3] D. M. Brooks et al., "Power-aware microarchitecture: design and modeling challenges for next-generation microprocessors," *IEEE Micro*, Vol. 20, Issue: 6, Dec. 2000, pp. 26-44.

[4] D.B. Papworth, "Tuning the Pentium® Pro microarchitecture," *IEEE Micro,* Vol. 16-2, April 1996, p. 8.

[5] C. Mead and L. Conway, *Introduction to VLSI systems*, Addison-Wesley Publishing Company, Boston, Dec. 1980.

[6] G. Hinton, D. Sager, M. Upton, D. Boggs, D. Carmean, A. Kyker, and P. Roussel, "The Microarchitecture of the Pentium 4 Processor," *Intel Technology Journal,* Issue 1, 2001, article 2.

[7] M. Bekerman, A. Yoaz, F. Gabbay, S. Jourdan, M. Kalaev, and R. Ronen, "Early Load Address Resolution via Register Tracking," in *Proceedings of the 27th International Symposium on Computer Architecture,* June 2000, pp. 306-315.

[8] Mobile Pentium® III processors–Intel SpeedStep® Technology http://www.intel.com/support/processors/mobile/pentiumiii/ss.htm.

[9] Intel® Centrino™ Mobile Technology Performance Brief http://www.intel.com/performance/resources/mobiletechnology/

[10] SPEC CPU2000 V1.2 http://www.specbench.org/osg/cpu2000/.

AUTHORS' BIOGRAPHIES

Simcha Gochman is a principal engineer/architect with Intel's Mobile Platforms Group in Haifa, Israel, leading the microarchitecture definition activities of the Intel Pentium M processor and its future derivatives. Simcha has been with Intel for 18 years. Earlier in Intel he led the microarchitecture definition of the Intel Pentium Processor with MMX™ technology and was involved with the design of the 80860 processor and the 80387 numeric coprocessor. Simcha received his M.Sc. degree from the Technion, Israel Institute of Technology in 1984. His e-mail is simcha.gochman@intel.com.

Ronny Ronen is a principal engineer/researcher and director of Intel's Microprocessor Research Lab in Haifa, Israel, focusing on microarchitecture research. Ronny was heavily involved in the definition stages of the Intel Pentium M processor. He has been with Intel for 22 years. Earlier in Intel he led the compiler and performance simulation activities in the Intel Israel Software department. Ronny received his M.Sc. degree from the Technion, Israel Institute of Technology in 1979. His e-mail is ronny.ronen@intel.com.

Ittai Anati is a senior architect with Intel's Mobile Platforms Group in Haifa, Israel, focusing on the micro-architecture of the Intel Pentium M processor family. Ittai has been with Intel for 14 years. Earlier in Intel, Ittai was part of the i860XP, Pentium Overdrive, and Pentium with MMX technology design teams. He received his B.Sc. degree from the Technion, Israel Institute of Technology in 1990. His e-mail is ittai.anati@intel.com.

Ariel Berkovits is a senior architect with Intel's Mobile Platforms Group in Haifa, Israel, leading the Intel Pentium M processor performance analysis, projection and performance validation activities. Ariel has been with Intel for 11 years. Earlier in Intel he led the

---

™ MMX is a trademark of Intel Corporation or its subsidiaries in the United States and other countries

development of performance simulators and was involved in the definition of graphic chipsets. He holds a B.Sc. degree from the Hebrew University in Jerusalem. His e-mail is ariel.berkovits@intel.com.

Tsvika Kurts is a senior architect with Intel's Mobile Platforms Group in Haifa, Israel, focusing on the Intel Pentium M processor bus, platform, and debug hooks. Tsvika has been with Intel for 18 years. Earlier in Intel Tsvika was part of the Intel Pentium® Pro processor bus architecture and system validation team and was involved in the Intel Pentium 4 processor bus protocol development. He received his B.Sc. degree in 1984 and his M.Sc. degree in 1992, both from the Technion, Israel Institute of Technology. His e-mail is tsvika.kurts@intel.com.

Alon Naveh is a senior architect with Intel's Mobile Platforms Group in Haifa, Israel, focusing on processor and platform power management. He received his B.Sc. degree from the Technion, Israel Institute of Technology in 1983, and holds an MBA degree from San Jose State University. Alon was involved in the definition of the Intel Pentium M processor power management, PCI-E, and the Odem chipset. Prior to Intel, Alon worked in Motorola Semiconductor and in National Semiconductor. His e-mail is alon.naveh@intel.com.

Ali Saeed is a senior systems architect in Intel's Mobile Platforms Group. His primary research area includes performance and power profiling of processors and chipsets with an emphasis on performance, power, and battery life estimations. Prior to joining Intel, Ali was manager of performance analysis in Compaq's Commercial PC Group. He holds a Bachelors degree in Electrical Engineering from the University of Houston. His e-mail is ali.saeed@intel.com.

Zeev Sperber is a senior architect with Intel's Mobile Platforms Group in Haifa, Israel. He received his B.Sc. degree from the Technion, Israel Institute of Technology in 1984. Zeev has been with Intel for 19 years. He was involved in the definition of the Intel Pentium M processor out-of-order and execution sections. Earlier in Intel, Zeev was part of the Intel Pentium processor with the MMX technology team. His e-mail is zeev.sperber@intel.com.

Robert C. Valentine is a senior architect with Intel's Mobile Platforms Group in Haifa, Israel, focusing on the Intel Pentium M processor's fetch and decode cluster. Robert has been with Intel for 13 years and has worked both as a microarchitect in the Haifa's Microprocessor Platforms Group and as a member of Haifa's Research team. Prior to Intel, Robert worked in Prime Computer and Honeywell Information Systems. He holds an M.Sc. degree in Computer Engineering from Boston University. His e-mail is bob.valentine@intel.com.