



Intel[®] Technology Journal

Intel[®] Centrino[®] Duo Mobile Technology

Power and Thermal Management in the Intel[®] Core[™] Duo Processor

Power and Thermal Management in the Intel[®] Core[™] Duo Processor

Alon Naveh, Mobility Group, Intel Corporation
Efraim Rotem, Mobility Group, Intel Corporation
Avi Mendelson, Mobility Group, Intel Corporation
Simcha Gochman, Mobility Group, Intel Corporation
Rajshree Chabukswar, Software Solutions Group, Intel Corporation
Karthik Krishnan, Software Solutions Group, Intel Corporation
Arun Kumar, Software Solutions Group, Intel Corporation

Index words: Intel Core Duo, ACPI, power, thermal

ABSTRACT

The Intel[®] Core[™] Duo processor is the first mobile processor that uses Chip Multi-Processing (CMP) on-die technology to maximize performance and minimize power consumption. This paper provides an overview of the power control unit and its impact on the power-saving mechanisms. We describe the distribution of P-states and look at other techniques used to improve the system power consumption including the impact of multi-threading on power consumption. Then, we provide recommendations on optimizing for CMP and building power-friendly applications.

INTRODUCTION

Power and thermal management are becoming more challenging than ever before in all segments of computer-based systems. While in the server domain, the cost of electricity drives the need for low-power systems, in the mobility market, we are more concerned about battery life and thermal limitations. The increasing demand for computational power in conjunction with the relatively slow improvement in cooling technology caused power and thermal control methods to become primary parts of the architecture and the design process of the Intel Core Duo processor-based system.

Intel Core Duo is the first general-purpose, multi-core on die Chip Multi-Processing (CMP) processor Intel has developed for the mobile market. The core was designed to achieve two main goals: (1) maximize the performance under the thermal limitation the platform allows; and (2) improve the battery life of the system relative to previous generations of processors.

Comparing the Intel Core Duo processors with previous-generation Pentium[®] M processors [1] reveals that it uses newer process technology, runs faster, and uses the same-size caches, but the overall die size is increased in order to accommodate two cores. The use of new process technology, together with special design and layout optimizations, allows each core of the Intel Core Duo technology to control its dynamic power consumption. In this paper, therefore, we focus on the CMP-related aspects of this technology. Controlling the static power (leakage) was found to be a real challenge due to the fact that static power depends on the total area of the processor and on process technology. Since the overall area was increased and the new process was found to produce more leakage power, static and dynamic power management became one of the main issues when designing the system.

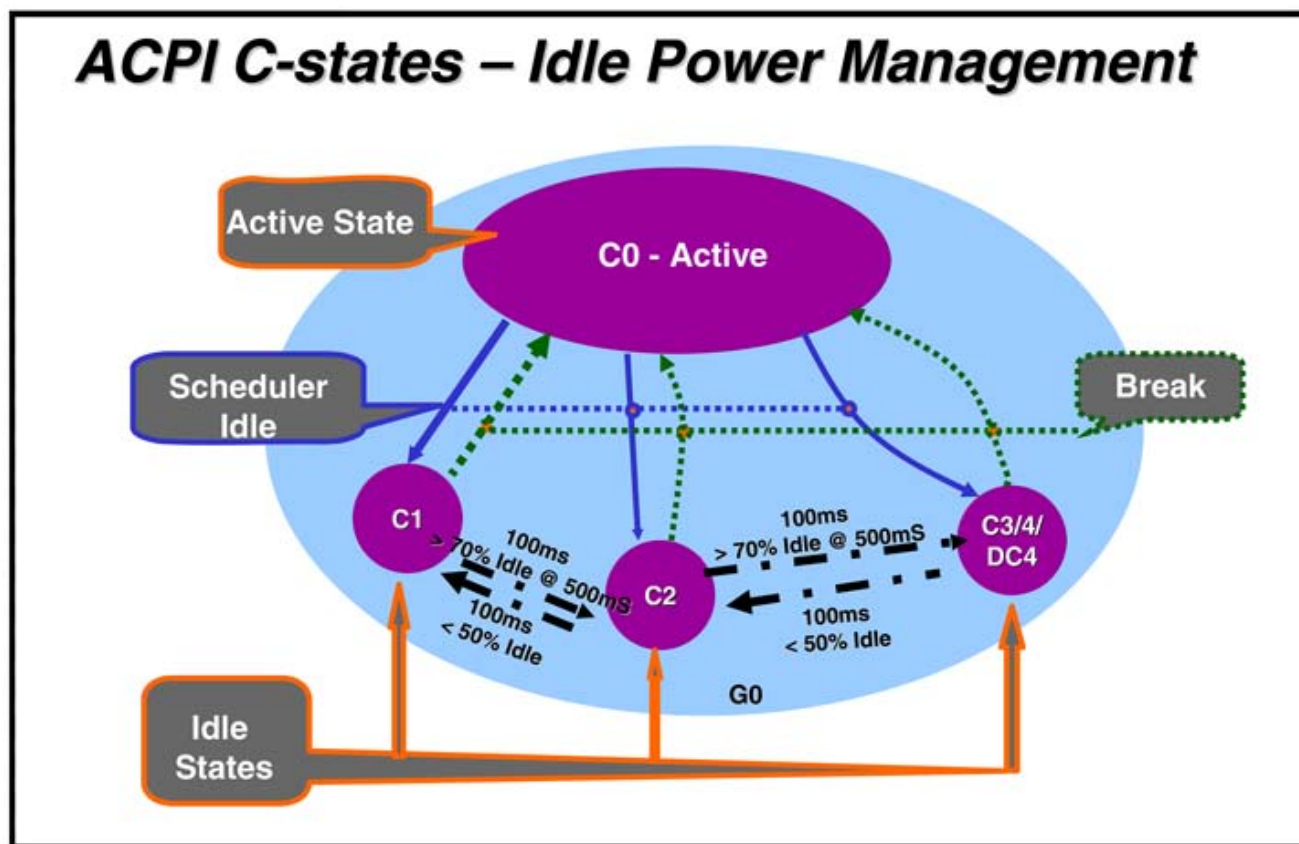


Figure 1: ACPI-based idle state management

Optimizing the system for maximum performance at the minimum power consumption is usually done as a combination of software (operating system) and hardware elements. Most modern operating systems (OS) use the ACPI [2] standard for optimizing the system in these areas. The ACPI processor sleep state control assumes that the core can be in different power-saving states (also termed sleep states) marked as C_0 to C_n . When the core is active, it always runs at C_0 , but when the core is idle, the OS tries to maintain a balance between the amount of power it can save and the overhead of entering and exiting to/from that sleep state. Thus C_1 represents the power state that has the least power saving but can be switched on and off almost immediately, while extended deep-sleep (DC_4) represents a power state where the static power consumption is negligible, but the time to enter into this state and respond to activity (back to C_0) is quite long. Figure 1 shows the general principle of operation of how a traditional ACPI software layer controls the sleep states of the processor. When the OS scheduler detects there are no more tasks to run, it transitions the processor into the “selected” idle state, by executing a BIOS defined instruction sequence. The processor will remain in that

idle state until a break event occurs and then return to the C_0 state. Break events would typically be interrupts and similar indicators that new tasks need to be executed. The OS evaluates the CPU activity factor over a registry-based time window and, periodically, modifies the selected target C-state for the next evaluation window.

The progressive increase in static power savings per state is achieved by employing more aggressive means as the C-state deepens. In C_1 idle state, only processor-centric measures are employed: instruction execution is halted and the core clocks are gated. In C_2 states and above, platform-level measures are also added to further increase the power savings. While in the C_2 state, the processor is obligated not to access the bus without chipset consent. Thus, the front side bus can be placed in a lower power state, and the chipset can also initiate power-saving measures. In C_3 , the processor also disables its internal Phase Locked Loops. In C_4 , the processor also lowers its internal voltage level to the point where only content retention is possible, but no operations can be done. Deep- C_4 , a new Intel Core Duo power state, is described later in this paper.

In order to control the dynamic power consumption, the ACPI standard tries to adjust the active power consumption to the “computational” needs of the software. The system controls the “active” state of the system (also termed P-states) by a combination of hardware and software interfaces: the hardware defines a set of “operational points” where each one defines a different frequency/voltage and therefore different levels of power consumption. The OS aims to keep the system at the lowest operational point yet still meet the performance requirements. In other words, if it anticipates that the software can efficiently use the maximum performance that the processor can provide, it puts it in the P₀ state. When the OS predicts (based on history) that the lower (and slower) operational point can still meet the performance requirements, it switches to that operational point (marked as P_n) in order to save power.

Power consumption produces heat that needs to be removed from the system. Naturally, each system has a total cooling capacity per its specific design, and each major component has a cooling limit or power consumption allowance. The system designers usually choose the maximum frequency the system can run at without overheating when running in a “normal” usage model; this is also called the system's Thermal Design Power (TDP). Thus, when an unexpected workload is used the thermal control logic aims to allow the system to provide maximum performance under the thermal constraints.

The remainder of this paper is organized as follows: we first describe the Intel Core Duo implementation of power and thermal control; next, we provide experimental numbers that examine the efficiency of the power and thermal mechanisms; and finally we extend the discussion to software optimizations that can help save power.

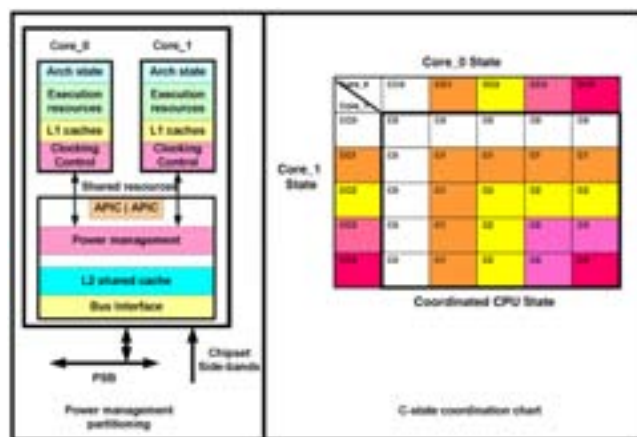


Figure 2: Internal power state in Intel Core Duo processor

POWER AND THERMAL MANAGEMENT

Power and thermal management of the Intel Core Duo processor raise a new challenge to the ACPI-based architecture since, from a software point of view, there is no difference between running under a CMP-based system and running under dual-core architecture, but from a hardware point of view, even though most of the logic is duplicated, major parts of the logic such as the power supply and the L2 cache are still shared. For example, in a Dual-Processor system, when the OS decides to reduce the frequency of a single core, the other core can still run at full speed. In the Intel Core Duo system, however, lowering the frequency to one core slows down the other core as well.

C-state Architecture

Since the OS views the Intel Core Duo processor as two independent execution units, and the platform views the whole processor as a single entity for all power-management related activities (C2 state and beyond), we chose to separate the core C-state control mechanism from that of the full CPU and platform C-state control.

This was achieved by making the power and thermal control unit part of the core logic and not part of the chipset as before. Migration of the power and thermal management flow into the processor allows us to use a hardware coordination mechanism in which each core can request any C-state it wishes, thus allowing for individual core savings to be maximized. The CPU C-state is determined and entered based on the lowest common denominator of both cores’ requests, portraying a single CPU entity to the chipset power management hardware and flows. Thus, software can manage each core

independently, while the actual power management adheres to the platform and CPU shared resource restrictions.

As can be seen from Figure 2, the Intel Core Duo processor is partitioned into three domains. The cores, their respective Level-1 caches, and the local thermal management logic each operates as a power management domain. The shared resources—including the Level-2 cache, bus interface, and interrupt controllers (APICs)—are yet another power management domain. All domains share a single power plane and a single-core PLL, thus operating at the same frequency and voltage levels. However, each of the domains has an independent clock distribution (spine). The core spines can be gated independently, allowing the most basic per core power savings (C1-Halt state). The shared-resource spine is gated only when both cores are idle and no shared operations (bus transactions, cache accesses) are taking place. If needed, the shared-resource clock can be kept active even when both cores' clocks are halted, serving L2 snoops and APIC message analysis.

The coordination mechanism serves as a transparent layer between the individually controlled cores and the shared resource on die and on the platform. It determines the required CPU C-state, based on both cores' individual requests, controls the state of the shared resources, such as the shared clock distribution, and also implements the C-state entry protocol with the chipset, emulating a legacy single-CPU mobile processor. When detecting that both core states are deeper than C1, the coordination mechanism issues the proper indication to the chipset, triggering the platform C2-4 entry sequence.

When the platform detects a break event (such as an interrupt request), it negates the proper sideband signals (such as the STPCLK, SLP, DPSLP and DPRSTP). The coordination logic then analyzes the platform signaling, and initiates the proper C-state exit sequences for the shared resources, and if needed, the cores.

Thus, the required goal of coordinating across the two cores is achieved in an efficient and transparent manner. Software operates as if it is managing two independent cores, while the platform and the shared resources are controlled as one by the coordination logic, reflecting a single platform Level C-state in a backwards-compatible manner.

As part of the per core power savings, the independent cores' L1 cache is also flushed during core C3 and C4 states. Due to the ratio between L1 and L2 cache size, it is assumed that nearly all of the L1 is included in the L2. Therefore, the flushing should not incur a high overhead of a write-back into the system memory, and it will not

incur a high warm-up penalty when restarting execution afterwards, since the data already reside nearby in the L2 cache. By flushing the caches, the cores can be kept asleep even when the L2 cache is accessed heavily by the other core or by a system device, thus improving power savings even further.

Dynamic Cache Sizing and Deep C4 State

Now that the processor has been able to enter C4, we face the challenge of lowering the C4-state leakage power even further. Since leakage power is directly proportional to the operating voltage, the most efficient means to save leakage static power would be to lower the C4 operating point. Unfortunately, lowering the voltage impacts data retention, and the first cells to be affected are the small transistor data arrays such as in the L2 cache. Therefore, the first step in achieving a lower voltage idle state is to implement a mechanism that can dynamically shut down the L2 cache in preparation for the Deep C4 state.

Cache Sizing

When defining the L2 cache dynamic sizing algorithm, the following considerations need to be addressed:

- L2 is a large array; therefore flushing it will incur some power and potentially C-state latencies, especially if done all at once or too frequently.
- Many applications will suffer performance degradation if running for a long period of time with little or no L2 cache. However, it has been proven that the short interrupt handling tasks, occurring at periodic timer ticks, do not have much use for the L2 cache and are not visibly impacted by running with the cache closed.

To accommodate the above restrictions, the dynamic sizing mechanism is implemented as an adaptive algorithm, with various built-in filtering properties and heuristics.

At the algorithms' base is an assumption that during long periods of very low utilization and idle residency (mapping to the C4 state), shutting down the cache will not result in a perceivable performance impact. This condition is detected by a state machine, described in Figure 3. In order to start shrinking the cache, the Finite State Machine (FSM) checks that the CPU frequency, controlled by the OS, is below a programmable threshold. It also checks that the CPU on the whole has not stayed too long in C0—which may indicate a streaming task being executed (e.g., DVD playback). This is done by a pre-programmed count-down timer, reloaded once the whole CPU enters C2-4 (package) states. Finally, the FSM checks the second core's idle state, requiring it to be in C4 as well, before allowing a shrink operation to begin.

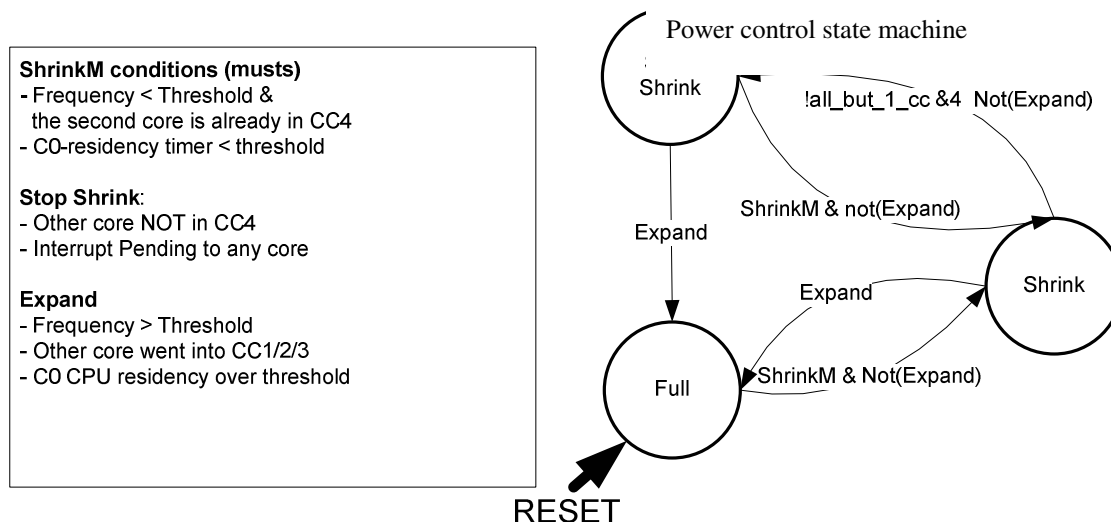


Figure 3: Shrink/Expand heuristics

The FSM freezes the shrink operation if a pending interrupt is detected, or if either core is in the active C0 state.

Cache expansion is requested once the activity indicators show that performance may be required. This is inferred in one of three ways: either the frequency is being increased over the programmable threshold, the CPU is staying in C0 for a period exceeding the pre-programmed timer, or one of the cores is entering a non C4 idle state (this is the OS's way of signaling that the core was not idle enough).

The actual cache flushing flow is performed by the microcode of the last core entering C4-state. In order to minimize the power impact and to filter too short C4 periods, the microcode flushes only part of the L2 (1/8 through 1/2 of the total cache size) during each consecutive C4 entry. The cache is flushed in chunks of lines (between 4 to 256 lines) checking for interrupts in between. Once a whole way is flushed, it is power-gated with sleep transistors, further reducing its leakage.

Microcode typically automatically expands the cache to a minimum of two ways upon every DeepC4 exit. Once the CPU enters C4 again, microcode will shrink the cache back down to 0. At the initial expansion it is assumed that the CPU has just exited from DeepC4. As such the L2 valid array may not be valid. Therefore, as part of the DeepC4 exit flow, the microcode also clears all of the L2 valid bits, ensuring the cache is indeed perceived as empty by the snoop logic. The same initialization flow can be applied also to other sensitive arrays should testing detect them as unstable.

DeepC4 (DC4) Entry

After the L2 cache has been shrunk to 0 and the CPU enters C4, the CPU voltage may be further reduced. Moreover, since no data are cached, the data cache does not need to be awakened for snoops. This feature is performed by the chipset, during the DC4 state. Once this is detected, the chipset starts diverting snoopable traffic directly to memory. During the DC4 state, the chipset scans the incoming traffic for interrupts and APIC messages, and once they are detected, queues them separately, while initiating a break sequence for the processor. Once the processor is fully awake, the interrupts are delivered to the processor and the memory traffic is diverted back to the CPU for snooping.

Handling P-states in a CMP Environment

ACPI P-state (Performance) control algorithm's goal is to optimize the runtime power consumption without significantly impacting performance. The algorithm dynamically adjusts the processor frequency such that it is just high enough to service the SW execution load. Operating point selection is done by the OS power management algorithms (OSPM) based on the CPU load observed over a window of time. Once the target point is set, the CPU is expected to modify its operating voltage and frequency to match the OSPM's request.

Figure 4 shows one example of the relationship between different working points (P-state points) in the Intel Core Duo processor and their relative power consumption. As can be seen, the benefit of going to a lower working point can be divided into an "exponential" part and a "linear" part. The exponential part represents a range of operating points where both frequency and voltage can be scaled to meet the new working point, while the linear part

represents a range of operating points where the system has already reached the minimum voltage allowed and so only the frequency can be scaled.

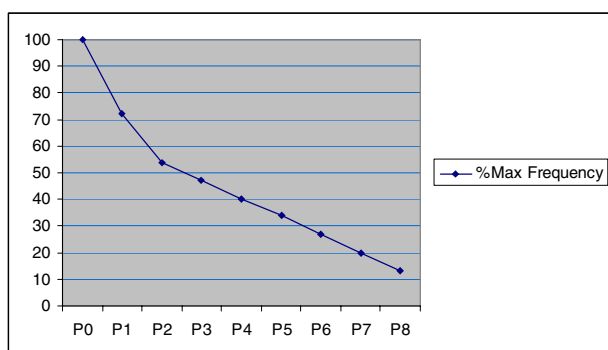


Figure 4: P-state CPU frequencies

When a multi-processor system was used, each core could be controlled independently by the OS and the individual processor state was set accordingly. The Intel Core Duo CMP implementation presents new challenges since some parts of the system, such as the PLLs and power planes, are shared and so both cores and the shared area need to operate at the same effective frequency/voltage point (i.e., the same P-state).

Initial SW versions that ran on the hyper-threading Intel NetBurst[®] microarchitecture-based CPUs utilized System Management Mode (SMM) code to coordinate the target P-state between the threads. This solution was targeted to optimize performance, which suited desktops and servers at which most hyper-threading CPUs were targeted. However, it incurred inherent overhead for every P-state request made by the OSPM, resulting in an average power impact prohibitive to the mobile focused Intel Core Duo architecture.

Consequently, a HW coordination approach was adopted for the P-state architecture as well. As with the C-state mechanism, each core's OS Power Management component can request a P-state separately via the standard IA32_PERF_CONTROL MSR. The HW coordination logic in the shared region tracks these P-state requests from both cores and determines the required CPU level target operating point based on the current execution state of the CPU:

- **No thermal control state:** In this case, the coordination logic will prefer performance over power, so as not to starve the SW threads. As a result, the higher operating point will be selected for the total CPU operating point, causing the whole CPU to execute the known advanced Intel SpeedStep[®] technology-based architecture transitions between operating points.

- **Thermal_Controlled_State:** In this case, the HW has detected an overheating condition and is trying to drive the operating point down for cooling. Here, the coordination HW will select the lower of the two cores' requests, allowing a quicker cooldown than if the maximum of both cores was used. Initial Intel Core Duo CPUs will use the same operation point for both cores; however, future implementations may choose to select a different operation point per core, thus taking advantage of this capability.

Due to the hardware coordination nature of the P-state architecture, OSPM may have a hard time tracking the exact frequency each core has run at, since the actual runtime frequency may be higher than the OSPM requested, and may vary without the core's OSPM knowledge. This may cause some confusion to the OSPM when trying to use the core's activity factor (non C0 state %) and the expected frequency to determine the actual code load. Intel Core Duo technology augments the previous frequency visibility mechanism by supplying two 64-bit counters, providing the OSPM information on the actual execution frequency of each core. ACNT (ActualCount) counts executed clocks (not Idle) at the currently coordinated CPU frequency. MCNT (MaxCount) counts the maximum number of non-idle clocks that the core could have run at, during the same period of time, had it run at the nominal frequency defined for the part. By dividing the ACNT by MCNT, the OSPM can determine the actual frequency each core has run at over a window of time, assisting the OSPM to assess the correct execution load on that core and then in turn determine the next operating point.

Thermal Control in a CMP Environment

The Intel Core Duo system was designed with power efficiency in mind and is aimed at different form factors, some of which are power and thermally constrained. Thermal management is a fundamental capability of all mobile platforms. Managing platform thermals enables us to achieve the maximum CPU and platform performance within thermal constraints. Thermal management features also improve ergonomics with cooler systems and lower fan acoustic noise.

In order to better control the thermal conditions of the system, Intel Core Duo technology presents two new concepts: the use of digital sensors for high accuracy die temperature measurements and dual-core multiple-level thermal control.

The general structure of the digital thermometer is described in Figure 5. It supports the legacy Intel[®] Centrino[®] mobile technology thermal sensor, PROCHOT and THERMTRIP, and the adding of a temperature reading capability to the fixed threshold sensor. A control

logic built around the thermal sensor performs a periodic scan and generates an output that represents the current temperature reading. The reading is loaded into an MSR, accessible to software. In order to improve temperature reading, multiple sense points monitor different hot spots on the die and report the maximum temperature of the die. An independent temperature reading from each core is available, with optional reporting of the maximum temperature of the entire die, e.g., the highest temperature of both cores.

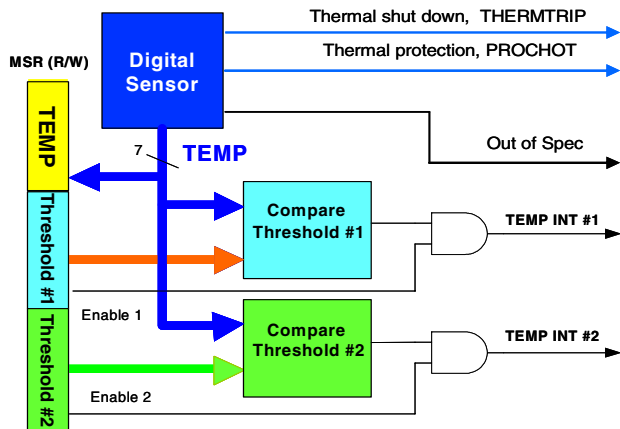


Figure 5: Digital thermometer block diagram

Interrupt generation capability is provided in addition to the temperature reading. Two programmable thresholds are loaded by S/W and a thermal event is generated upon threshold crossing. This thermal event generates an interrupt to a single core or to both cores simultaneously through the APIC settings.

The digital thermometer is intended to be used as an input to a software-based thermal control such as the ACPI. An interrupt threshold is defined to indicate the upper and lower temperature thresholds. An example of digital thermometer usage is illustrated in Figure 6.

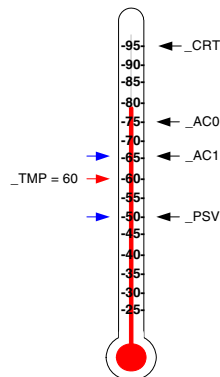


Figure 6: Digital thermometer and ACPI

In the above example, the die temperature is at 60°C and the thresholds are set to 50°C and 65°C. If the temperature rises above 65°C, a low-to-high interrupt is generated. The control software identifies the new temperature and initiates action, such as activating fans or initiating some passive cooling policy. The activation thresholds and policies are defined using BIOS and ACPI. New thresholds are loaded around the new temperature to further track new changes.

In previous Intel Pentium M processor-based systems, a single analog thermal diode was used to measure die temperature. A thermal diode cannot be located at the hottest spot of the die and therefore some offset was applied to keep the CPU within specifications. For these systems it was sufficient, since the die had a single hot spot. In the Intel Core Duo system, there is more than a single hot spot that moves as a function of the combined workload of both cores. Figure 7 shows the differences between the usage of the traditional analog sensor and the new digital sensors.

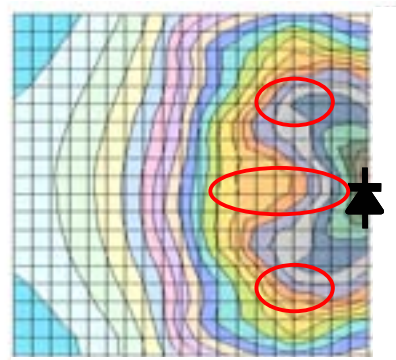


Figure 7: Analog vs. digital sensors in Intel Core Duo systems

As we can see, the use of multiple sensing points provides high accuracy and close proximity to the hot spot at any time. An analog thermal diode is still available on the Intel Core Duo processor. An example of the difference between the diode and the DTS is presented in Figure 8. The information presented in this graph was taken from simulations and not from a real system, but was correlated with data from real systems and found to be close enough.

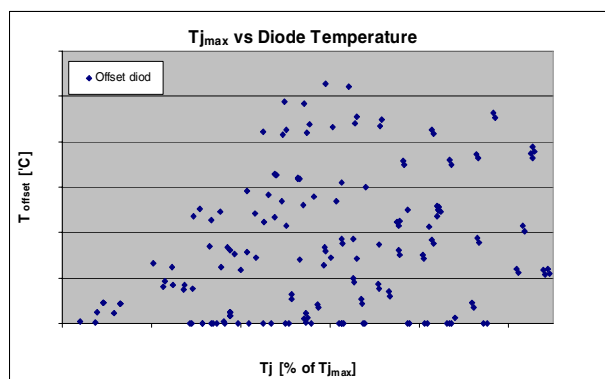


Figure 8: Diode to DTS temperature difference

It can be seen that significant temperature gradients exist on the die. The use of a digital thermometer provides improved temperature readings, enables higher CPU performance within thermal limitations, and improves reliability.

The Intel Core Duo system also implements hardware-based thermal control. Hardware-based thermal management is intended to handle abnormal thermal conditions and to protect the die from transient effects. Hardware-based thermal control ensures that the CPU will always operate within specified conditions. This improves reliability and allows higher performance with tighter control parameters.

Legacy thermal control features implement two externally visible signals:

- THERMTRIP: a fixed temperature sensor to detect catastrophic thermal conditions and to shut down the system if thermal runaway occurs.
- PROCHOT: a fixed temperature threshold that provides the DVS with a self-control mechanism that drops frequency and voltage to a new working point (a more detailed description of this mechanism can be found in [1]).

Intel Core Duo technology implements a new multiple-core thermal control algorithm. Both cores synchronize action requests and activity. A programmable policy can select DVS on both cores and linear control on each core either independently or as a locked operation. PROCHOT was also made available as an input, to allow thermal protection of platform components such as the voltage regulator (VR).

A finer grain overheat detection is performed by monitoring the thermal activity. If an extreme thermal condition occurs (fan malfunction, operation within a bag, etc.) the processor self thermal management may not be sufficient and the temperature may not drop below the threshold point. The internal control algorithm tracks the

control behavior and further reduces power as needed. Continuous extreme conditions will eventually initiate an “Out Of Spec” thermal interrupt and register the warning in an internal status bit. The “Out Of Spec warning” signal is intended to initiate a managed system shutdown before a THERMTRIP shutdown occurs.

Platform Power Optimization

Intel Core Duo technology implemented power and thermal management features as described above to control the CPU power and thermals. Other components on the platform also contribute to the total platform power and work closely with the CPU. The CPU VR losses can get as high as 5W while running high workloads. A 3-phase VR was built to deliver high current operating at low efficiency while working at low utilization. Turning off phases or switching to asynchronous mode can save a significant amount of power. Efficiency, as a function of the number of phases, is described in Figure 9.

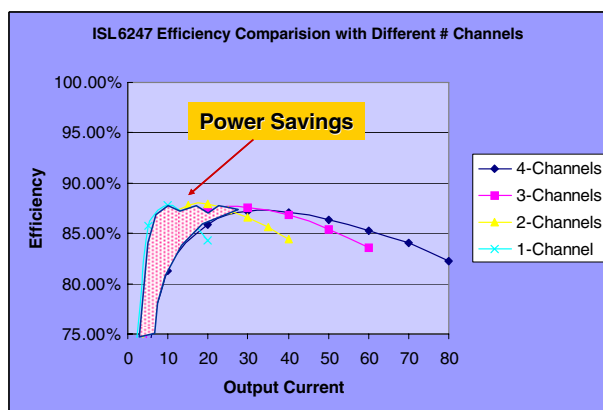


Figure 9: Efficiency as a function of number of phases

It can be noted that improved power efficiency mode cannot handle high currents and there is a need for feedback on the current requirements. The Intel Core Duo processor keeps track of the power consumption and gives feedback to the VR using the PSI-2 and VID signals, as described in Figure 10.

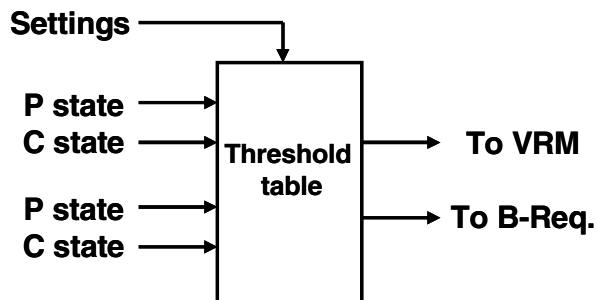


Figure 10: PSI-2 overview

The CPU tracks the activity requirements based on P- and C-states. When the OS initiates a request to go to a higher P-state, there is sufficient time to signal the VR to change to high current mode before the actual power consumption increases.

Another optimization at low workloads is done on the load line. The voltage delivery has serial resistance, and a voltage drop between the VR and the CPU is a function of the current consumption. At lower power consumption, the voltage drop is lower and the CPU voltage increases, driving higher power. The same mechanism on the CPU detects low power consumption and drives lower VID code to the VR. Voltage is increased ahead of time, before actual power consumption increases.

Communicating the CPU requirements to the rest of the platform enables additional power savings on the platform, increasing the battery life.

EXPERIMENT RESULTS OF POWER MANAGEMENT

In this section, we provide measurements that were taken from an Intel Core Duo processor-based system in order to examine the efficiency of its power and performance. We mainly focus on the relation between the use of MT applications and the utilization of the Intel Core Duo system.

Distribution of P-states

The OS along with the hardware optimizes the platform power by decreasing the processor speed when there is no demand and increasing the frequency as demand increases. Figure 11 shows a typical snapshot of the system power consumption during some period of time. As can be seen, the OS sets both processors to a single power state (P-state), but the total power consumption depends on the number of tasks run in parallel (if one task is run, we assume it is run in an ST environment) and the computational time of these tasks. We can clearly observe the phases of execution: the length of each phase reflects the total time it takes while the area below the curve reflects the total energy for this phase. We use this energy

calculation methodology in the following sections to compare ST and MT applications.

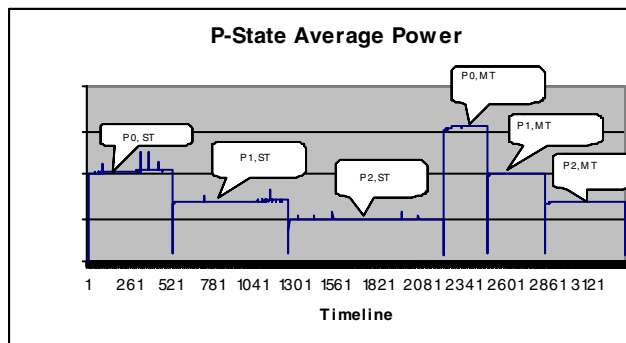


Figure 11: Energy calculation methodology

Power Consumption of Different Power States

In order to evaluate the benefit of using MT applications, we measure the power consumption of the system while in different working points when running an ST application vs. an MT application. Figure 12 shows the experimental data with both ST and MT code under different P states on an Intel Core Duo system. As can be seen, the power consumed by an MT code is less than twice the power consumed by an ST code in corresponding P states. This implies that the applications that are MT and demonstrate excellent scaling will not only reduce the total execution time, but also show greater savings in power consumed.

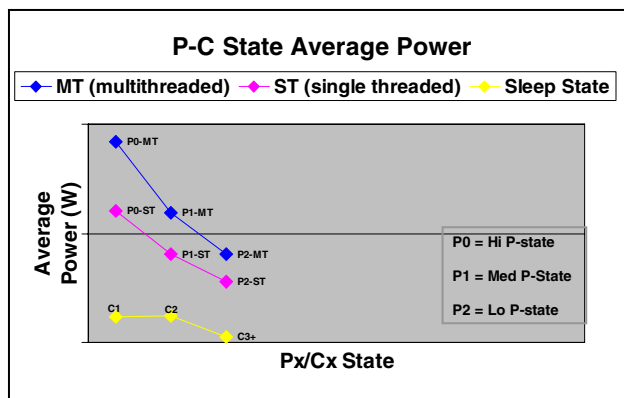


Figure 12: P/C-state average power

Impact of Clock Rate on Power Consumption

The OS uses a periodic clock interrupt to keep track of time, trigger timer objects, and schedule application threads. While the default interrupt rate is set by the OS, applications can increase the interrupt rate to any desired frequency (as low as 1ms).

While some multimedia applications with high-definition content may require a high interrupt rate for timely

scheduling of timer-based threads, some with enough processing headroom may not require it. Thus, in both cases, increasing the clock interrupt rate may have an important impact on the overall power (and therefore on the battery life).

Figure 13 shows the processor power impact due to a high interrupt rate on an Intel Core Duo system. Average power increases from 0.5 W to 1.05 W when the interrupt frequency is increased to 1ms.

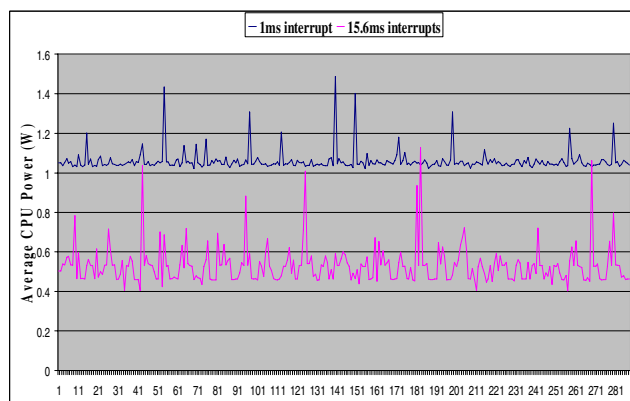


Figure 13: High interrupt rate impact

As can be seen, the “baseline” of the 1ms curve is higher than the normal interrupt rate. The reason for that is that every interrupt causes the CPU to go back from C3 to C0 (active state) and when the CPU utilization is high enough, it will also execute the clock service routine in high P-state.

Balance and Imbalance Threading Environment

This section includes a comparative study of the total energy consumed on the Intel Core Duo platform (CPU plus platform) for a given task with both MT and ST methodologies. We shall also analyze applications featuring various threading scenarios, such as “balanced threading” vs. “imbalanced threading.”

Applications with a Balanced Threading Model

Real-world applications under study here are CPU bound workloads that were run in both ST and MT mode. The applications are CPU bound, but in ST mode, only one core is fully utilized, and in MT mode both the cores are fully utilized and consume equal processing resources. Adaptive mode here refers to the power-saving scheme where the OS optimizes overall power consumption by dynamically changing CPU frequency on demand using Intel SpeedStep technology (the GV3 technology).

The graphs (Figures 14 and 15) below indicate CPU and platform energy for adaptive mode. For each application run (ST and MT), power data gathering is normalized to the longest run-time. For example, a cryptography

workload runs for ~50 seconds in ST mode and ~25 seconds in MT modes. The power data is measured for 50 seconds in both ST and MT mode. The intent here is to identify if total energy can be saved by finishing the task faster (as in the case of MT) and entering deep sleep states after task completion.

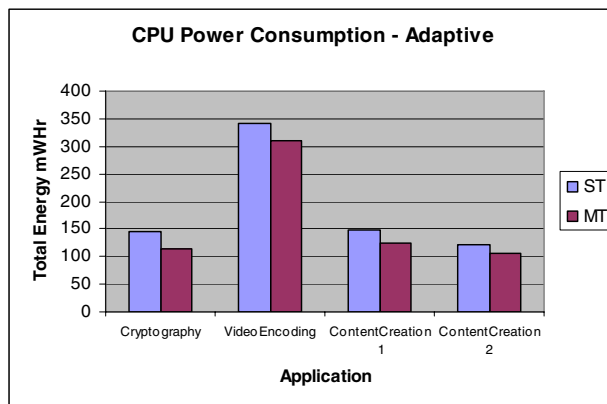


Figure 14: Balanced threading–CPU power

As indicated in Figure 14, the energy for a given task is reduced by completing the task sooner with efficient MT and letting the processor enter idle/sleep state. The MT applications above show linear performance scaling with the number of cores and demonstrate a ~9%-27% savings in CPU power.

The CPU power is a major component of the overall power, and the graph in Figure 15 indicates about an 8%-17% savings in the platform power due to MT.

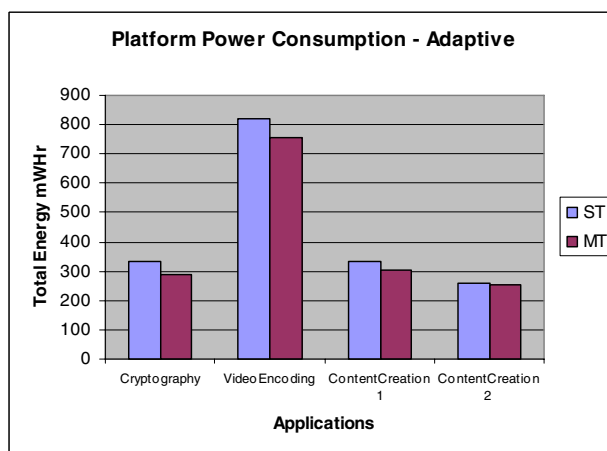


Figure 15: Balanced threading–platform power

Applications with an Imbalanced Threading Model

In this section, we examine power/performance implications on an application with imbalanced threading where the load on the threads is asymmetrical. The analysis here demonstrates how the imbalance in the

threads may cause certain OSs to incorrectly choose a sub-optimal P-state.

We used an in-house gaming prototype primarily composed of a physics computation (collision detection and resolution of graphics objects) and a rendering engine for the analysis. A balanced threading model was achieved by dividing the graphical objects into two cores with each thread taking care of collision detection and resolution of the objects it owned. For the imbalanced case, one thread was given the task of performing collision detection and resolution for the colliding objects while the other thread was given the task of calculating the updated positions. The imbalance was due to the fact that the first thread was more CPU intensive than the other.

The MaxPerf mode refers to the power scheme where the processor is always running at the highest clock speed. The following shows the total energy under different modes with a normalization similar to the one used before. The first data set in Figure 16 represents energy data with a MaxPerf power scheme. The second data set indicates energy consumption with an Adaptive scheme. We focus on these two data sets for now.

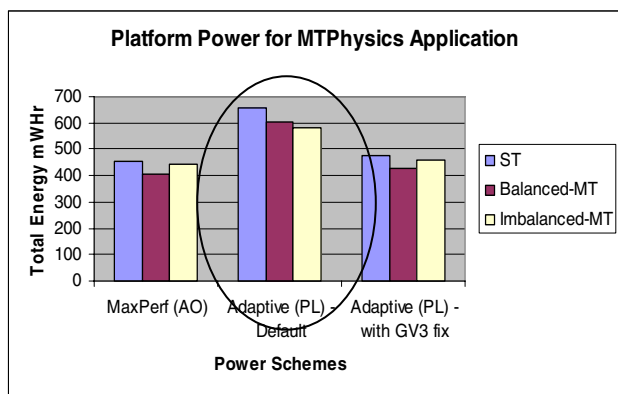


Figure 16: Imbalanced threading–platform power

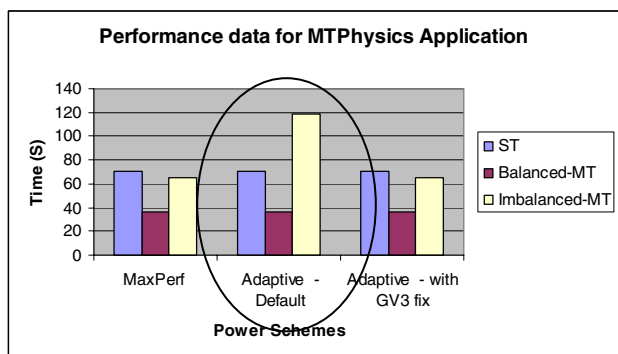


Figure 17: Imbalanced threading–performance data

As indicated in Figure 17 (circled data), the Imbalanced-MT implementation demonstrates 2x performance

degradation when running in the Adaptive power scheme as compared to MaxPerf.

Since we use a normalization technique for energy measurements, the platform energy consumption increased for the Imbalanced-MT because the run time is now doubled with the Adaptive-Default scheme (indicated with the circle in Figure 17).

What caused a 2x performance degradation with the Adaptive power scheme?

By looking at the application profile while running the Imbalanced-MT version it is observed that since the first thread is doing more of the work than the second thread, the first thread keeps migrating between the cores, making the effective CPU utilization on the cores ~50%. This is a natural artifact of the manner in which the Windows* scheduler works. However, on a system running in “Adaptive” (portable/laptop) power mode, this thread migration causes the Windows kernel power manager to incorrectly calculate the optimal target performance state for the processor, as the individual cores may appear less busy than the whole package. Due to this, the Windows OS tends to reduce processor frequency in order to save power in Adaptive mode and hence performance may be degraded in Adaptive mode. This in turn causes increased power consumption.

To address this issue of incorrectly calculating the optimal frequency, Microsoft provided a hotfix (KB896256) to change the kernel power manager to track CPU utilization across the entire package, rather than the individual cores, and hence calculate the optimum frequency for applications.

The third data set in Figure 16 (Adaptive–with GV3 Fix) indicates data with the kernel hotfix. In this case, Imbalanced-MT implementation in Adaptive scheme shows power/performance data similar to that of MaxPerf mode. With this fix, processors run at optimum frequency produce expected results.

This study shows that the imbalanced threading model/under-utilized CPU may cause degradation in performance causing increased power consumption. Hence it is recommended to use a balanced threading model while running MT applications or utilize the appropriate operating system fixes.

Platform Power Savings Measurements

The following measurement shows the benefit of platform power savings while running low workloads. The workloads tested in this experiment are mobile mark and DVD playback. The power losses of the VR and power delivery to the CPU are measured with the PSI enabled and disabled. The results are described in Figure 18.

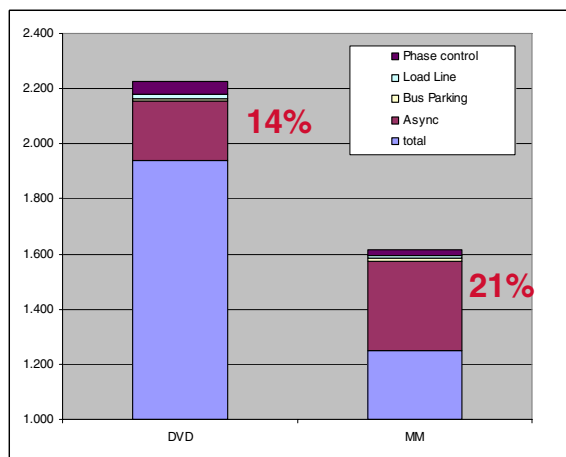


Figure 18: Power losses on VR with and without PSI-2

The above figure shows the overall power losses broken into its components. It can be seen in the above results that activating the PSI-2 reduces the VR losses by 14% while playing DVD and by 21% in Mobile Mark workload.

SOFTWARE RECOMMENDATIONS FOR POWER SAVING WITH MULTI-THREADING

In this section we provide a few highlights from the *Intel Core Duo Optimization Guide*[5]. We mainly focus on the power aspects of the optimizations:

- *Threading done right provides power savings.* As indicated by the data in the “Balanced Multi-threading Model” section, a properly multi-threaded application results in power savings. This includes MT implementations with minimum imbalance and synchronization points. While multi-threading an application, it is always recommended to use a threading model in which all the threads perform equal amounts of work independently. Minimizing synchronization points between threads leads to more time spent in the parallel section which translates to better power savings.
- *Thread imbalance may cause performance degradation and may not provide power benefits as compared to balanced threading.* As discussed in the “Imbalanced Threading Model” section, applications with an imbalanced threading model may show lesser performance improvement as compared to a balanced threading model; and hence consume more power than the balanced implementation. Thread imbalances may cause frequent thread migration across cores that may result in reporting of incorrect processor activity states. This may lead processors to go down to the lower frequency states (if the hotfix provided by

Microsoft is not enabled) in Adaptive scheme even if one of the threads is utilizing full processor resources. This issue may occur while running ST applications on a dual-core system in “Adaptive” mode as well.

- *Utilize GV3 hotfix (KB896256) from Microsoft for Windows.* In the case of applications having huge thread imbalances, for example, Thread A utilizing 100% CPU and Thread B utilizing 10% CPU, Thread A will keep migrating between cores. This makes effective CPU utilization 50%. On a system running in “Adaptive” (portable/laptop) power mode, this thread migration causes the Windows kernel power manager to incorrectly calculate the optimal target performance state for the processor. To address this issue, Microsoft provided a kernel hotfix that calculates optimal frequency by looking at the entire package.

Hence, if an MT application indicates increased power consumption due to reduced performance in Adaptive mode only, install the GV3 hotfix from Microsoft (as the issue might be the one described above). GV3 hotfix will track CPU utilization across the entire package rather than individual cores and will enable the OS to operate at optimum frequency.

- *Interrupt rate.* Applications need to be sensitive to the interrupt rate as the power penalty may increase further on future architectures. In-house testing with few multimedia applications that use a high interrupt rate have shown that the interrupt rate can actually be reduced to default frequency without impacting the user experience. It is recommended that applications tune the interrupt rate to the content being delivered and also to shutdown the interrupts after content delivery.
- *OS scheduling vs. hard affinitizing.* In general, for the Intel Core Duo processor-based systems, it is advisable to use OS scheduling instead of affinitizing threads or applications. OS scheduling may show better power savings due to better performance than affinitizing the threads. The OS scheduler will utilize any under utilized core; while hard affinitizing may potentially degrade performance, since applications may need to wait for the availability of a specific processor even though other processors in the system are idle.

CONCLUSION AND REMARKS

The Intel Core Duo processor-based technology introduces the enhanced power management-aware processor including CPU-based power management actions such as dynamic L2 resizing. Dynamically resizing/shutting down of the L2 cache is needed in

preparation for DeepC4 state in order to achieve lower voltage idle state for saving power. As described in this paper, efficient multi-threading provides optimal performance scaling as well as power savings. Multimedia applications reduce/shutdown interrupts after content delivery to save power.

REFERENCES

- [1] Intel® Centrino® Mobile Technology – Enabling Battery Life at http://www.intel.com/design/mobile/platform/platform_collateral.htm
- [2] ACPI Specification at <http://www.acpi.info/spec.htm>*
- [3] “Introduction to Intel® Core™ Duo Processor Architecture,” *Intel Technology Journal*, Volume 10, Issue 2, 2006.
- [4] “CMP implementation in Intel® Core™ Duo Processor,” *Intel Technology Journal*, Volume 10, Issue 2, 2006.
- [5] *IA-32 Intel® Architecture Optimization Reference Manual*, in <http://www.intel.com/design/Pentium4/manuals/248966.htm>

AUTHORS' BIOGRAPHIES

Alon Naveh is a senior architect with Intel's Mobile Platform 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 from San Jose State University. Alon co-lead the power management definition of the Intel Core Duo architecture and was involved in the definition of the Intel Pentium M processor-based 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 at intel.com.

Efi Rotem is a senior architect with Intel's Mobile Platform Group in Haifa, Israel, focusing on processor and platform power and thermal management. Efi joined Intel in 1995 and was involved in the definition and development of Pentium 4 and Pentium M processors. Earlier in Intel he led the Intel Pentium with MMX™ technology testing. He received a B.Sc. degree from the Technion, Israel Institute of Technology in 1986. His e-mail is efrain.rotem at intel.com.

Avi Mendelson is a principal engineer in Intel's Mobile Platform Group in Haifa, Israel, and adjunct professor in the CS and EE departments, Technion, Israel Institute of Technology. He received his B.Sc. and M.S.c degrees from the Technion, Israel Institute of Technology and his

Ph.D from the University of Massachusetts Amherst. Avi has been with Intel for 7 years. He started as senior researcher in Intel Labs, later he moved to the Microprocessor group where he serves as the CMP architect of Intel Core Duo processor. Avi's work and research interests are in computer architecture, low-power design, parallel systems, OS-related issues and virtualization. His e-mail address is avi.mendelson at intel.com.

Simcha Gochman is a senior principal engineer with Intel's Mobile Platform Group in Haifa, Israel. Simcha has been with Intel for 21 years. Lately he has led the microarchitecture development of the Pentium M processors and of the Core Duo processor. Prior to that he led the microarchitecture definition of the 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 at intel.com.

Rajshree Chabukswar is a software engineer working on client enabling in the Software Solutions Group that enables client platforms through software optimizations. Prior to working at Intel, she obtained a Masters degree in Computer Engineering from Syracuse University, NY. Her e-mail is rajshree.a.chabukswar at intel.com.

Karthik Krishnan is a Software Engineer with the Software Solutions Group at Intel. He holds a Masters degree in Mathematics from the Indian Institute of Technology. His current focus is on power and performance optimization of software applications on dual-core platforms. His e-mail is karthikeyan.krishnan at intel.com.

Arun Kumar is an engineering manager in Intel's Software Solutions Group in Dupont, Washington. He received his B.Tech degree from the Indian Institute of Technology, Kanpur and his Masters and Ph.D degrees from the University of Minnesota, Minneapolis, where his research was in the area of image processing and computer vision. Currently his team works on client platforms based on Pentium M, Pentium D and Core Duo processor architectures with special focus on CMP, application software performance, and platform power consumption. His e-mail is arun.kumar at intel.com.

Copyright © Intel Corporation 2006. All rights reserved. Intel, Core, Pentium, Intel NetBurst, Intel SpeedStep, Centrino, and MMX are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

* Other names and brands may be claimed as the property of others.

This publication was downloaded from

<http://developer.intel.com/>.

Legal notices at

<http://www.intel.com/sites/corporate/tradmarx.htm>.

For further information visit:

developer.intel.com/technology/itj/index.htm