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.

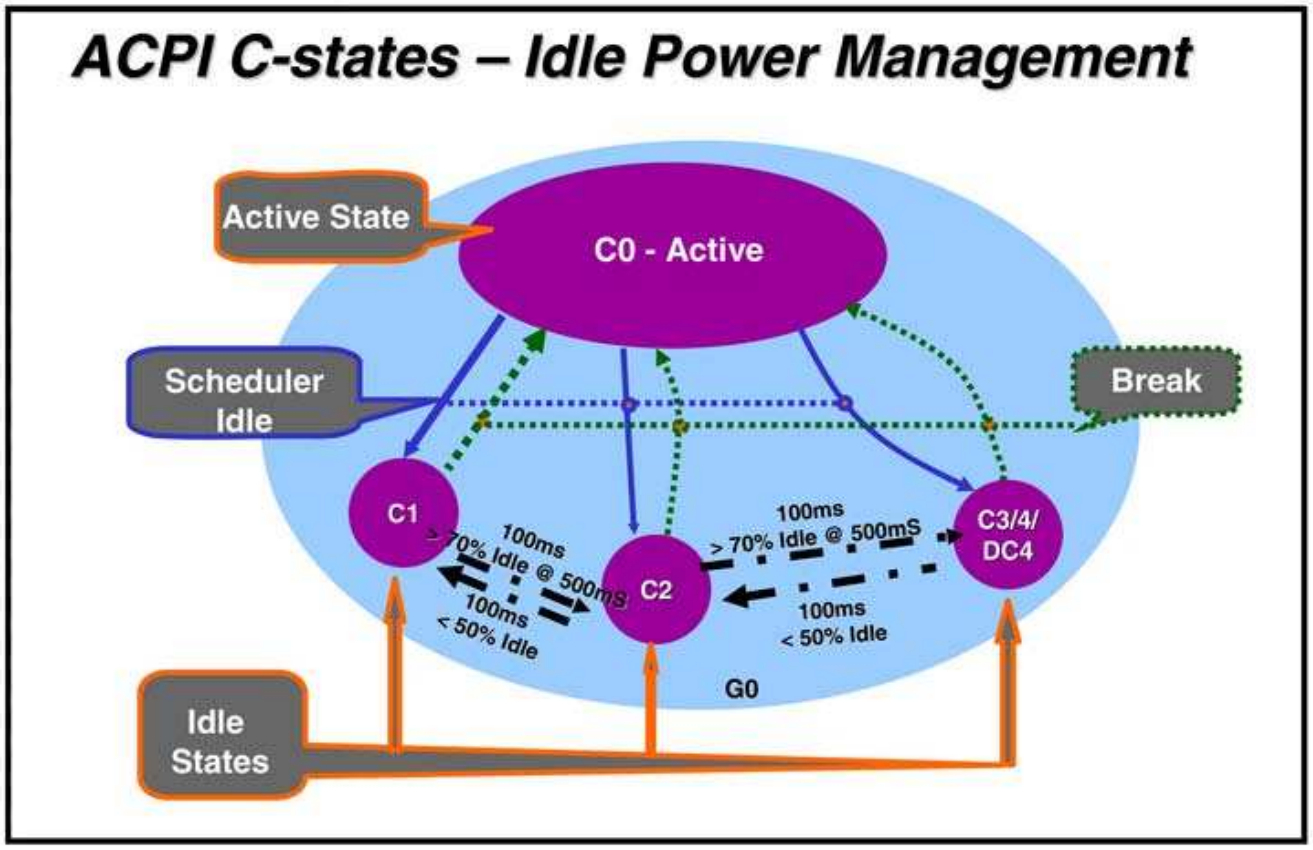

Figure 1: ACPI-based idle state management

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-C4, 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 P_0 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 Pn) 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.

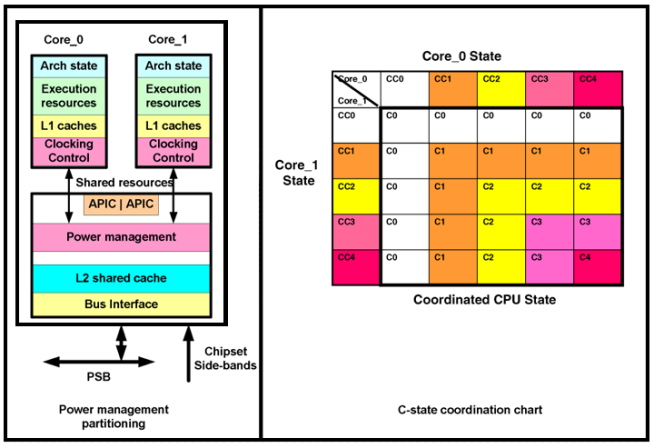

Figure 2: Internal power state in Intel Core Duo processor

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.

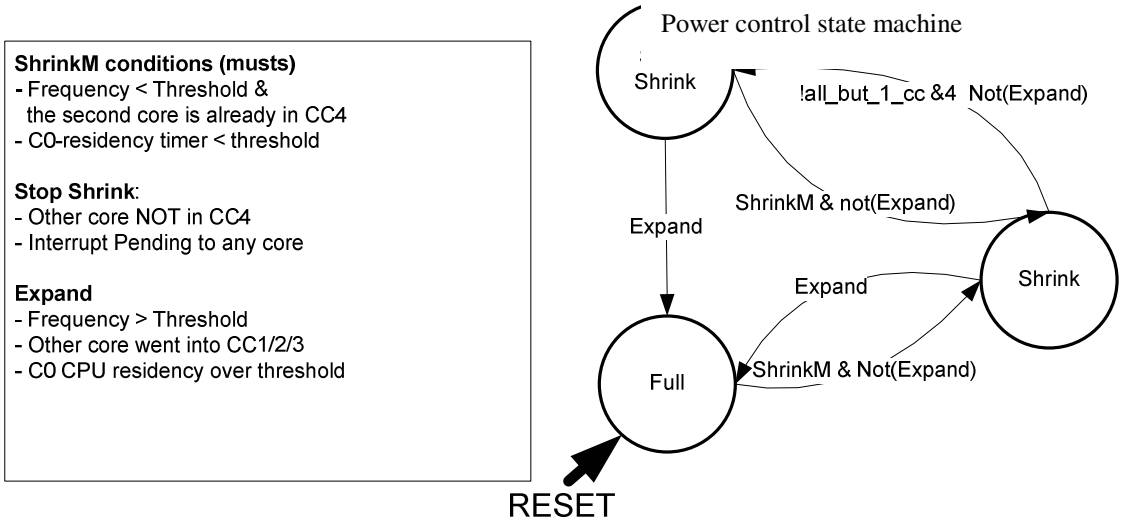

Figure 3: Shrink/Expand heuristics

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

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.