**Intel and Core i7 (Nehalem) Dynamic Power Management**

Power management is a feature of electrical devices, including computers, which attempts to “turn off” or place certain or all portions of the device in some lower-power state when inactivity is present. Managing the energy consumed by a microprocessor prolongs battery life, reduces cooling requirements, reduces electrical noise, and reduces operating costs. Since mobile computing is becoming more ubiquitous, power management is becoming more important in mobile electrical devices such as laptops, cell phones, and personal digital assistants.

The standard for power management for microprocessor and computer systems is the Advanced Configuration and Power Interface (ACPI). ACPI defines platform-independent interfaces for hardware discovery, configuration, power management and monitoring. The specification is central to Operating System-directed Configuration and Power Management (OSPM), which describe a system implementing ACPI and thus, removes device management away from the system BIOS and places it at the heart of the operating system.

Intel has incorporated some form of power management technologies in their microprocessors since their Pentium III product line. However, with the introduction of the Core i7 (codenamed Nehalem), Intel has innovated greatly in the area of power management. Improvements have been made at all levels of the design: transistor level, circuit level, microarchitecture level, and architecture level.

This section on power management will first detail the various microprocessor core, package, and system level power management microprocessor states which Intel supports as per the ACPI standard. Subsequently, Core i7-specific features that allows for the processor cores, processor package, and computer system to implement and leverage the various power management states are discussed.

**Intel Power Management States**

Intel supports four power management states for their microprocessor, CPU package, and overall system. Table 1 provides the various power management state names along with a brief description.

<table>
<thead>
<tr>
<th>State Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-State</td>
<td>Microprocessor Performance State</td>
</tr>
<tr>
<td>T-State</td>
<td>Microprocessor Throttle States</td>
</tr>
<tr>
<td>C-State</td>
<td>Microprocessor and Package Idle States</td>
</tr>
<tr>
<td>S-State</td>
<td>System Sleep States</td>
</tr>
</tbody>
</table>

Table 1: Intel’s Microprocessor, CPU Package, and System Sleep States

T-States are microprocessor core frequency throttle states. Beyond this fact, T-States are not well documented and thus will not be discussed below. P-States, C-States, and S-States are discussed below.

**Microprocessor Performance (P) States**

Microprocessor performance states (P-States) are a pre-defined set of frequency and voltage combinations at which the microprocessor can operate when the CPU is active. The microprocessor
utilizes *dynamic frequency scaling* (DFS) and *dynamic voltage scaling* (DVS) to implement the various P-States supported by a microprocessor. DFS and DVS are techniques that dynamically changes the operating frequency and operating voltage of the microprocessor core based on current operating conditions.

Both scaling techniques work in tandem. Reducing the core voltage decreases leakage current of the CPU transistors, which reduces power. However, reducing the voltage causes transistors to switch slower. This in turn reduces the maximum operating frequency of the circuit. To maintain functional correctness and avoid meta-stability of state elements, the operating frequency must then be reduced to the new (lower) maximum operating frequency. This in turn makes an application program run longer. However, the net result is a significant reduction in the energy usage per second of the microprocessor, since the functional units are now idling much less per second than before.

The current P-State of the microprocessor is determined by the operating system. The time required to change from one P-State to another is relatively short. The operating system takes this time into account when it dynamically changes P-States. The OS must manage the tradeoff between power consumption by the microprocessor and the performance of the microprocessor. 


Performance state P0 is the highest performing P-State. Performance state P1 is the next lowest performing P-State. Performance State Pn is the lowest performing microprocessor P-State. Specific P-State information for the Intel Core i7 is not published. To provide an example, Table 2 below shows the various P-States and power consumption for the Intel Pentium M 1.6 GHz processor. [Enhanced Intel SpeedStep Technology for the Intel Pentium M Processor White Paper ftp://download.intel.com/design/network/papers/30117401.pdf].

<table>
<thead>
<tr>
<th>P-State</th>
<th>Frequency</th>
<th>Voltage</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>1.6 GHz</td>
<td>1.484 V</td>
<td>25 Watts</td>
</tr>
<tr>
<td>P1</td>
<td>1.4 GHz</td>
<td>1.420 V</td>
<td>~17 Watts</td>
</tr>
<tr>
<td>P2</td>
<td>1.2 GHz</td>
<td>1.276 V</td>
<td>~13 Watts</td>
</tr>
<tr>
<td>P3</td>
<td>1.0 GHz</td>
<td>1.164 V</td>
<td>~10 Watts</td>
</tr>
<tr>
<td>P4</td>
<td>800 MHz</td>
<td>1.036 V</td>
<td>~8 Watts</td>
</tr>
<tr>
<td>P5</td>
<td>600 MHz</td>
<td>0.956 V</td>
<td>6 Watts</td>
</tr>
</tbody>
</table>

Table 2: P-State Operating Points and Power Consumption for the Intel Pentium M 1.6 GHz CPU

**Microprocessor Idle (C) States**

A C-State is defined as an idle state. When nothing useful is being performed, various parts of the microprocessor can and should be powered down to save energy. There are three classifications of C-States: thread *(logical)* C-States, microprocessor *core* C-States, and microprocessor *package* C-States. All three categories of C-States are similar since they all represent some form of an idle state of a processor thread, processor core, or processor package. However, they are different in substantial ways.

**Thread (Logical) C-States**

A thread *(logical)* C-State represents the operating system’s view of the microprocessor’s current C-States, at the thread level. When an application asks for a processor’s core C-State, the application receives the C-State of a “logical core.” A logical core is what an application’s individual thread
perceives to be a core, since the thread perceives to have full ownership of a particular core. This is not
the case since the Core i7 supports simultaneous multi-threading (Intel Hyperthreading Technology),
which allows for two threads per core.

As an example, logical Core 0 (thread 0 executing on Core 0) can be in a specific idle state while logical
Core 1 (thread 1 on Core 0) can be in another idle state. The operating system can request any C-State
for a given thread.

**Core C-States**
A core C-State is a hardware-specific C-State. Any core of the four native cores residing on the Core i7
package can be in a specific C-State. Therefore, all four cores are not required to be in the same C-State.
Core C-States are mutually exclusive per-core idle states. This is intuitive since one core may be idle
while the other may be hard at work executing a Fast-Fourier Transform (FFT) or matrix inverse
algorithm.

The core C-State relates to a thread C-State. A core C-State is the highest power-consuming idle state to
satisfy the operating conditions of the most power-consuming thread.

**Package C-States**
A package C-state is an idle state in which all four cores of the Core i7 reside. The package C-State of the
Core i7 is related to the four individual core C-States. The microprocessor can only enter a low-power
package C-State when all four cores are ready to enter that same core C-State. Therefore, when all
cores are ready to enter the same lower power core C-State, then the package can safely transition into
the equivalent lower power package C-State. The conditions for entering a specific package C-State are
also intuitive. Since the native cores of the Core i7 often share resources like the L3 cache, the power
management system should not prevent a non-idle core to lose its state just because there are three
other cores idling.

To summarize the relationship between logical, core, and package C-States, the following equations can
be useful:

\[
\begin{align*}
\text{Package CState} &= \text{Highest Power Consuming CState of current Core CStates} \\
\text{Core CState} &= \text{Highest Power Consuming CState of all Current Thread CStates} \\
\text{Thread CState} &= \text{Anything the OS Wants to Request}
\end{align*}
\]

**Core C-State Specifics**
Similar to P-States, higher numbered C-States indicate more power savings. In the case of C-States,
higher numbered C-States indicate more inactive parts of the microprocessor and in turn, less power
consumed. However, the higher the C-State number, the wakeup time of the microprocessor modules is
higher. There are various C-States supported by Intel processors, including the Intel Core i7. Figure 1
summarizes the various C-States supported by Core i7 and various hardware that is powered down.
The following describes the various actions that are performed in the various core C-States supported by the Core i7 microprocessor as shown in Figure 1 above.

**Idle State C0**
When the microprocessor is in the active state (some P-State), the microprocessor is also defined to be in state C0. In state C0, instructions are actively being executed by a particular core or all cores, depending on the current workload.

**Idle State C1**
Idle state C1 is the first idle state and is defined as the halt state. When the processor is in state C1, no instructions are being executed. When a core is in C1, the Intel Core i7 controller turns off the clocks of all clock domains pertaining to the core pipeline. Even though the clocks are off, the a core in state C1 is still considered an active core.

The Core i7 controller turns off the clocks of all clock domains by utilizing a technique known as *clock gating*. Clock gating is a hardware-controlled power management technique which disables, or "gates", the clock to all state elements (flip flops and latches) of a synchronous digital system. Clock gating is accomplished by logically ANDing the clock signal of a particular clock domain with a conditional control signal, as demonstrated by Figure 2. Clock gating disables all frequency switching in the hardware and thus eliminates most of the dynamic (switching) power dissipated by the state elements. The only power still consumed is caused by leakage current.
Clock gating is an extremely difficult technique to implement, partially due to clock skew. Clock skew happens when a clock signal routed to various state elements arrives at different time instances. The varying arrival times are partially attributed to the extra delay introduced by the AND gates utilized by the clock gating technique. Clock skew may introduce race conditions in the hardware which may cause functional failures. Despite the implementation complexities, however, the design and verification effort required to correctly implement clock gating is well worth the power savings it provides.

**Idle State C3**
While in idle state C3, the core phase-locked-loops (PLLs) are turned off, and all the core caches are flushed. A core in C3 is considered an inactive core. The wakeup time for idle state C3 is significantly longer than in state C1 since the Core i7’s core caches must be restored. Additionally, since the PLLs are linear-feedback based control systems, which need to be turned back on, time must be allocated for the PLLs to lock (stabilize) to the correct frequency.

**Idle State C6**
While in idle state C6, the core PLLs are turned off, the core caches are flushed and the core state is saved to the Last Level Cache (LLC). The power gate transistors (discussed later) activated to reduce power consumption to a particular core to approximately zero Watts. A core in idle state C6 is considered an inactive core. The wakeup time a core in idle state C6 is the longest. The core state must be restored from the LLC, the core PLLs must be re-locked, the power gates must be deactivated, and core clocks must be turned back on.

An interesting situation may occur in C6. Since C6 is the deepest C-State, the energy cost to transition to and from this state is the highest. Frequent transition in and out of deep C-States can result in a net energy loss. To prevent this, Nehalem includes an auto-demote capability that uses intelligent heuristics to determine when idle period savings justify the energy cost of transitioning into a deep C-State and then transition back to C0. If there is not enough justification to transition to C6, the PCU demotes the OS C-State request to C3.

**Intel System (S) Power States**
Intel microprocessors, including the Core i7, can be in various system power states. There are a total of six different power states ranging from S0 (the active state) to S5 (the system is power off). The intermediate states are referred to as sleep states. In the various sleep states, the entire system appears to be turned off because of low power consumption. However, in the sleep states, enough of the architectural state is retained in order to return to the active state without requiring a system reboot.

A state diagram of the sleep state transitions is shown in Figure 3 below. As Figure 3 shows, the computer is “waking up” when it is transitioning from any lower power S-State to S0. The computer system is “going to sleep” whenever it is transitioning from S0 to any low power S-State. Figure 3 also shows that the system cannot enter one sleep state directly from another. All transitions must traverse through S0.
The following is a brief discussion of the actions performed by the system while in the various S-States supported by Intel Processors, including the Core i7.

**System Power State S0**
System power state S0 is the active state of the system. In S0, the system is completely operational and is keeping track of precise architectural state. **While a system is in S0, the processor can be in any package C-State, any core C-State, and any P-State.** System state S0 is also referred to as the “ON” state.

**System Power State S1**
While in S1, the system consumes less power than S0 state. All microprocessor and hardware state is maintained. The time it takes to “wake-up” from sleep state S1 back to active state S0 is minimal. **System interrupts such as mouse movements or keystrokes will usually wake the system.**

**System Power State S2**
While in S2, the system consumes less power than in state S1. In this state, the microprocessor loses power which causes all microprocessor state and cache contents are lost. The time it takes to “wake-up” from sleep state S2 to active state S0 is greater than that for sleep state S1. System interrupts such as mouse movements or keystrokes will usually wake the system.

**System Power State S3**
More commonly referred to as *Sleep* within the Windows OS, the system consumes less power than state S2. In S3, the microprocessor loses power. **In addition to the microprocessor and hardware context being lost, the chipset context is also lost.** Main memory contents (DRAM), however, are retained. The time it takes to “wake-up” from sleep state S3 to active state S0 is greater than that for sleep state S2. System interrupts such as mouse movements or keystrokes will usually wake the system.
System Power State S4
More commonly referred to as Hibernate within the Windows OS, the system consumes the least amount of power compared to all other sleep states (S1, S2, S3). While in S4, the system is considered to be “almost” completely off. Microprocessor and other hardware state is written to the hard disk. No microprocessor or other hardware state is retained. The time it takes to “wake-up” from sleep state S4 to active state S0 is greater than that for sleep state S3. System interrupts such as mouse movements or keystrokes will usually wake the system, at which architectural and other state information is retrieved from the hard disk before system can resume in the active state S0.

System Power State S5
More commonly referred to as Shut Down in within the Windows OS environment, this is the system OFF state. In S5, the system does not retain any state.

Core i7 (Nehalem) Package Level Power Management
Intel has made some radical changes to the microprocessor package level power management architecture. The two most important enhancements to the microprocessor package power management features are the dedicated power control unit (PCU) and Intel Turbo Boost technology. This section discusses both of these features in greater detail, beginning with the PCU.

The Power Control Unit (PCU)
Intel scientists and engineers have changed the way power management is implemented with the introduction of the Core i7 (Nehalem). Before the Core i7, the majority of power management actions were controlled with dedicated hardware. However, with the Core i7, Intel architects have migrated the power management control from hardware to software.

As Figure 4 shows, the Core i7 includes a “fifth” on-die microprocessor. This microprocessor is actually a simple (compared to the four Nehalem cores) microcontroller Intel calls power control unit (PCU). Budgeting nearly 1 million transistors (equivalent to the entire transistor budget of Intel’s 486 CPU) for the implementation of the PCU, the PCU resides on the microprocessor die referred to as the Uncore.
The PCU runs embedded firmware. The firmware globally takes inputs on temperature, current, power, and OS requests. The PCU firmware dynamically makes decisions about which power management states (P/T/C/S) to enter in order to effectively balance power/performance tradeoffs of the Core i7’s four native cores, the Core i7’s entire microprocessor package, and the overall computer system. The migration to firmware-based power management allows the sophisticated algorithms running on the CPU to be upgraded. This firmware-based power management solution is also fairly scalable.

Intel Turbo Boost Technology

In addition to deciding which power management states to place the various Nehalem cores, the Nehalem package, and the computer system, the PCU hardware and associated firmware implements a feature which Intel calls Turbo Boost Technology. Intel Turbo Boost technology automatically allows processor cores to run faster than the base operating frequency (and base operating voltage) if the processor is operating below rated power, temperature, and current specification limits. Turbo Boost technology can be utilized with any number of enabled or active physical or logical cores, thus resulting in increased performance of both multi-threaded (logical cores) and single-threaded (physical cores) workloads. The amount of time the microprocessor package spends in Turbo Boost state will depend on workload and operating environment.

As Figure 5 tries to illustrate, Intel® Turbo Boost technology provides the Core i7 the capability of maximizing core frequency while ensuring that the processor does not go above the upper-limits of temperature, power, and current. When lower power-consuming applications execute on a subset of active cores (cores in core C0 or core C1) with inactive cores (cores in core C3 or core C6) present in the package, the applications may take advantage of the additional power headroom provided by the inactive cores in the form of increased core frequency. The increase in frequency may be as little as one frequency step (133.33 MHz) and as much as two frequency steps (266.66 MHz). The number of frequency steps depends not only on the number of active and inactive cores present in the package, but also on the packages operating temperature, current power consumption, and electrical current values.

The aforementioned constraints are monitored and managed as a simple closed-loop control system. If measured temperature, power and current are all below factory-configured limits while the OS requests
performance state P0, the PCU firmware automatically steps up core frequency one step (+133.33 MHz) at a time until it reaches the upper limit dictated by the number of active cores. When temperature, power or current exceed factory configured limits and the active cores are operating above the base operating frequency, the PCU automatically steps down core frequency one step at a time in order to reduce temperature, power and current.

Core i7 (Nehalem) Microarchitecture and Circuit Level Power Management
Moving from the package to the microarchitecture and circuit technology, Intel has made additional improvements in order to save power. One major enhancement to the microarchitecture to save power includes the re-placement of the Loop Stream Detector (LSD). One major circuit-level enhancement to save power is the transition from domino-logic circuits to static CMOS circuits. This section discusses both of these features in greater detail, starting with the LSD.

Microarchitectural Power Management: Loop Stream Detector (LSD)
Intel micro-architects relocated the Core 2 Duo’s loop stream detector (LSD) on Nehalem. The Core 2 Duo’s LSD resided between the fetch and decode stages and buffered small loops of eighteen non-decoded x86 instructions. These instructions are executed repeatedly when needed without having to constantly re-fetch instructions from the L1 instruction cache. Whenever the LSD was in use, the hardware in the fetch stage could be shut off by clock gating, thus saving power.

Nehalem enhances Core 2 Duo’s LSD by placing it after the decode stage and increasing the amount of storage. Performing these enhancements allows Nehalem to potentially store larger loops and take advantage of additional power savings. The increase in storage capacity of the LSD allows Nehalem to store twenty-eight fetched and decoded micro-ops. Since the LSD now stores decoded instructions instead of raw x86 instructions, both the fetch and decode pipeline stages along with the associated branch prediction hardware can be shut off by clock gating. This provides even more dynamic power savings over the Core 2 Duo.
Circuit Level Power Management: Transition to static CMOS Circuits

The use of domino logic based circuits has dominated the design of microprocessors over the past twenty years. Domino logic circuits exhibit smaller parasitic capacitances which allow for higher switching frequencies. The higher switching frequencies, however, make domino logic circuits power hungry. To save power, Intel circuit designers decided to switch from domino logic to static CMOS based logic circuits when implementing Nehalem. CMOS based logic circuits consume substantially lower power than domino logic circuits. However, power savings come at a cost. CMOS technology is slower with respect to switching frequency since it exhibits considerably larger parasitic capacitances.

Core i7 (Nehalem) Transistor Level Power Management

Intel has made some radical changes in their transistor manufacturing technologies that ultimately have yielded power management benefits. The major enhancement includes the Hi-K Metal and Hafnium based transistor manufacturing. This new process technology has in turn yielded the use what Intel calls power gate transistors to help almost completely shut off the voltage delivered to a specific core of the Core i7. This section briefly discusses Hi-K Metal and Hafnium based transistor technology and transitions to a discussion on how the power gate transistors help in power savings.

Hi-K and Metal-Gate Transistor Technology

The fundamental goal of transistor design is to make the transistor smaller, faster, cheaper, and power efficient. Smaller transistors are faster and cheaper to make. Problems arise, however, when trying to manufacture smaller transistors. The gate dielectric, which is generally manufactured using multiple layers of silicon dioxide (SiO$_2$) and used to insulate the transistor and prevent current flow, becomes “thin.” The lack of many layers results in small amounts of current flow even when the transistor is in the OFF state. This current is called leakage current. Due to leakage current, power consumption remains significant even when utilizing clock gating to reduce power.

Over the last ten years, Intel, as well as many others, have been on a quest to find new materials to manufacture the gate dielectric in order to reduce leakage current. That new material is based on Hafnium (Hf), atomic number 72. Hafnium is thicker and provides a high dielectric constant (k) value to insulate current when the transistor is in the OFF state and to allow current to flow as freely and quickly as possible when in the ON state. However, due to Hafnium’s interaction with the poly-silicon gate electrode, the voltage level (called the threshold voltage) which allows the transistor to transition between binary states increases, thus limiting transistor switching speed. By using a metal gate instead of a poly-silicon gate, however, overcomes this problem.

With the combination of the Hafnium high-k material used for the gate dielectric and the metal gate electrode, Intel’s new process technology increases transistor switching speeds and enables higher core and bus clock frequencies. This also helps extend Moore’s Law well into the next decade. Intel claims record improvements gained from utilizing this new High-K/Metal Gate technology based on Hafnium. These include:

- Approximately twice the transistor density (for smaller chip sizes or increased transistor counts)
- Approximately 30% reduction in transistor switching power
• Greater than 20% improvement in transistor-switching speed or a greater than 5 times reduction in source-to-drain leakage power
• Greater than 10 times reduction in transistor gate oxide leakage for lower power requirements and increased battery life

Power Gate Transistors
In current multi-core microprocessor designs, every core on the microprocessor die is provided with voltage from a shared power plane. Increased motherboard costs and design complexity places this constraint on the multi-core microprocessor. Therefore, each of the four native cores on Nehalem run off the same core voltage. Consequently, this constraint indicates that leakage current, which causes power dissipation, is forced to be present on idle cores only because there exists one more active cores in the CPU.

![Figure 7: Nehalem's Power Gate Transistors](image_url)

To overcome this constraint, Intel design and manufacturing teams worked very closely and managed to manufacture a very particular material that could be used to manufacture a specific type of transistor (based on Hi-K Hafnium and Metal Gates) that can act as a power gate, or voltage barrier, between the voltage provided by the shared power plane to a core and the actual core itself. The usage of power gate is shown in Figure 7. Despite the existence of a single power plane (shared core voltage), decoupling the voltage power plane and an individual core with a power gate transistor allows individual Nehalem cores to be almost completely shut off when they are in certain idle (C) or sleep (S) states. The power gate transistors allow one or more cores to be operating in an active state at a nominal voltage, while allowing the remaining idle cores to have power completely shut off to them.