Abstract

In today’s system-on-chip (SoC) platforms containing multiple heterogeneous processors, the system architect needs to know how to deal with power management facilities provided with different operating systems (OSes) running on them to balance the overall system power policies. Besides leveraging the right features, the design needs these OSes to work together at the system level. This white paper first provides an overview of the latest power management features of high-level and real-time OSes to support power savings hardware available with today’s SoC platforms. Next, we will walk through an example showing how interprocessor communication and state machine design can help reduce the overall system power on a real heterogeneous dual-core system.

The need to save power

Energy consumption is becoming more of a concern as this slice is getting an increasingly larger percentage of the overall operating costs. Imagine superstores with lines and lines of check-out lanes, each with a cash register, a credit-card reader, a scanner and a weight measuring station. It is really a waste if these equipments are not designed to be energy efficient with abilities to power down between customers or during non-operating hours. When multiplied by the number of stores, the number of cities and the operating life of the product, the total accumulated portion of the energy bill that could be saved is in the millions of dollars.

Many of today’s operating systems, like Linux®, come with power management support. The features have been available on the mainstream kernel since Linux made headways to lower power portable devices like smart phones, tablets and ebook readers. So even though your design is a plugged-in appliance, you can embrace the “go green initiative” from the ground up by taking advantage of the power management features that are already in place and incorporate them.

In this paper, we will first review power savings techniques available with today’s powered (i.e., plugged-in) system-on-chip (SoC)-based embedded systems and quickly move on to the discussion of how two operating systems, each with its own power methodologies, can cooperate at the system level to provide power management services.

Power management requires a system-level approach

There are two different components to the power equation from a silicon process standpoint: static (sometimes referred to as standby) and active. Static power is affected by leakage mainly and increases with temperature and supply voltage. Since leakage is a natural phenomenon that comes with shrinking process technology, the only way to really eliminate it is to shut that component down. Within the SoC, tactics employed so far include power islands, enabling part of the SoC to completely shut down. On the other hand, active power, which does increase with supply voltage, but not temperature, depends on chip activity. Strategies here include:
• Dynamic voltage and frequency scaling (DVFS), where the voltage and frequency can be dynamically adjusted to adapt to the performance required
• Clock domain to gate off unused peripheral
• Dynamic power switching (DPS), where software can switch between power modes based on system activity. The “software” is usually part of the operating system
• Adaptive voltage scaling (AVS), a closed-loop, hardware and software cooperative strategy to maintain performance while using the minimum voltage needed based on the silicon process and temperature

From the system standpoint, operations needed for power management include the ability to:
• Go to standby (user-application- or system-initiated system service)
• Hibernate to memory or storage (user-application- or system-initiated system service)
• Suspend and resume (user-application-initiated system service)
• Transition to different power profiles (user-application condition or state, system initiated and controlled)

Power can also be affected by how the application code is designed. For example, since input/output (I/O) buffers at the pin need to drive current, the traffic through major peripherals like memory controllers, especially double data rate (DDR), can be a good place to inspect. Unnecessarily moving data in and out of the SoC can waste energy.

Let’s take a look at the block diagram of a typical modern embedded system as shown in Figure 1. The processor is highly integrated and includes several types of processors and accelerators for application-specific needs as well as all the I/O peripherals to get the data in and out. The system board has external voltage regulators for the different power rails in addition to battery and clock management support integrated circuits (ICs). It also contains external I/O modules and hot swappable devices. To save energy, the application can take advantage of the internal memories by aligning code and data. In this way, algorithms in the pipeline can reuse buffers locally so that the I/O buffers at the pin level do not have to toggle needlessly. Other techniques include matching data types to architecture, correct alignment and use power of two for array sizes to simplify address calculation. These techniques can help reduce power consumption because the lower MIPS required can lower the temperature. Some call this “energy coding,” the third optimization vector besides speed and code size.

![Figure 1. Modern embedded system using complex SoC](image-url)
Power management is a concerted effort. Actions such as going into standby mode can involve a series of hardware and software steps. Therefore, to really “do the job right,” power management needs to be a system-level (i.e., where hardware meets software) design goal, especially if the processor is a complex SoC with multiple internal bus masters.

For example, for the “suspend” operation, the software has to take the hardware through the following actions:

- Notify drivers and pending tasks that the system is powering down
- Wait for the safe state to start the shutdown sequence
- Turn off I/Os and accelerators by gating power or clocks
- Save system state to memory (shown as mobile mDDR)
- Adjust voltage regulators to throttle down
- Set up battery management for suspend
- Transition clocking to a suspend state (usually involving just the real-time clock and mDDR running)

To get into the details of how power management is implemented, we now need to move our discussion on a real device and software.

Power management with Linux and DSP/BIOS kernel on TI’s OMAP-L138 C6-Integra™ DSP+ARM processor

On SoCs such as the OMAP-L138 C6-Integra DSP+ARM processor, there are two on-chip processors – an ARM 926EJS general-purpose processor (GPP) and a TMS320C6748 DSP. The ARM9™ usually runs embedded Linux for I/O and graphical user interface tasks. The DSP needs a more deterministic and lightweight operating system such as the DSP/BIOS™ software kernel foundation to perform signal processing. The chip is highly integrated with multiple I/O peripherals like Ethernet, USB, SATA, an LCD controller and more, as you can see in Figure 2.

Figure 2. Block diagram of the OMAP-L138 C6-Integra DSP+ARM processor
Each processor has equal access to the on-chip peripherals, enabling I/Os to be distributed between the two depending on system response time requirements.

For support of embedded systems requirements, the Linux kernel version 2.6 implements power management using a network of frameworks and drivers. To help understand this, we just need to remind ourselves that there are key functions such as suspend, resume, idle, DVFS and the mechanism to achieve them involving controlling the central processing unit (CPU) (CPUIdle for CPU sleep states), the clocks (clock framework and tickless option), the voltage regulators (regulator framework) and the helper drivers (mainly I2C and SPI). Specific power targets are defined by operating performance points (OPP), and governors manage how to transition between these OPPs. Recently, the concept of Power Management Quality of Service (PM QoS) was introduced to tie computing resources and capabilities in the hardware with latencies and throughput needs to define the minimum OPP required across platforms.

Going into details of the application programming interface (APIs) and data structure is beyond the scope of our paper, but to help clarify work needed, Figure 3 shows how power management is currently implemented for the OMAP-L138 C6-Integra DSP+ARM processor embedded Linux port to the evaluation module (EVM). At the application level, power management policies like OPPs can be accessed via the sysfs interface. In the kernel space, the frameworks with their governors and drivers have Linux generic portions, as well as platform (SoC)- and board-specific driver portions that can be customized.

On the DSP/BiOS operating system, power management services are consolidated under the PWRM (power manager) framework. Similar to the Linux side, PWRM abstracts the low-level work needed to gate clocks and clock domains on/off, control DSP sleep modes and coordinate with the internal resources to

<table>
<thead>
<tr>
<th>User Space</th>
<th>Kernel Space (generic)</th>
<th>Kernel Space (platform specific)</th>
<th>Kernel Space (board specific)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>System PM Policies - /proc/pm</td>
<td>/sys/.../cpufreq</td>
<td>IDLE driver(s)</td>
</tr>
<tr>
<td></td>
<td>Governors sys if</td>
<td></td>
<td>CPU Operating Points</td>
</tr>
<tr>
<td></td>
<td>/proc/pm</td>
<td>Regulator API</td>
<td>CLK / RTC drivers</td>
</tr>
<tr>
<td></td>
<td>Linux APM</td>
<td>Susp/ resume</td>
<td>CPU freq driver</td>
</tr>
<tr>
<td></td>
<td>PM core</td>
<td>PM QoS</td>
<td>Board Operating Points</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PMIC driver</td>
</tr>
</tbody>
</table>

Figure 3: Power management implementation for embedded Linux
govern the safe switching between OPPs. PWRM sits on top of the Power Scaling Library (PSCL) and a Power Management Interface (PMI) as in Figure 4.

To coordinate between the Linux and DSP/BIOS environments, a hardware mechanism is needed to communicate.

The inter-processor communication (IPC) hardware mechanism for the OMAP-L138 C6-Integra DSP+ARM processor is very straightforward as in Figure 5 on the following page. At the SoC level, five CHIPINT bits are allocated in the CHIPSIG memory-mapped register (MMR) located in the SYSCFG system configuration module for the signaling between the DSP and ARM. Up to four signals can be mapped for the DSP to notify the ARM and two for the ARM to notify the DSP with an additional signal just for the DSP non-maskable interrupt (NMI) event. Note that two bits are common to both the DSP and ARM so that they can be both interrupted at the same time. This is a useful future for debugging purposes. Writing a “one” to the bit will generate the signal or event. These events are fed to the respective interrupt controller (INTC) to get mapped to the core interrupt inputs.

To pass data, any of the internal or external memory areas can be used as shared memory area(s). Mutual exclusivity can be controlled using the mutex or semaphore mechanisms provided with the operating system. The SoC provides a system-level memory protection unit (MPU) that can protect a memory region from being
overwritten by internal bus masters like the ARM or DSP cores or the DMAs. This feature can be useful during development to debug the IPC software mechanism or detect ill-behaved programs or memory leaks.

For Linux and DSP/BIOS kernel, the IPC is abstracted by a software component called DSPLink. It consists of several modules to provide DSP control and code loading, buffer passing and control, and message passing and control. On the Linux side, it is a kernel mode driver. On the DSP/BIOS kernel side, it is a regular driver that can be called at the task level.

For applications that use the DSP as an accelerator, DSPLink PROC functions can be used to shutdown the DSP if the application no longer needs its service. This is adequate for most embedded systems where the ARM is running Linux as the master processor, and controls when it’s time to disable I/Os and accelerators to go standby. Disabling the DSP will enable more than 91 percent power savings just on the processor alone, as shown in Figure 6 on the following page.

For other states of DSP idle, power savings can be realized by expanding the Linux kernel space platform drivers or creating a user space proxy that uses DSPLink as the message transport to communicate with the DSP/BIOS side application to send request to PWRM. For example, to get to an OPP that only idles the DSP (90 percent power savings), add a service to the Linux suspend framework driver that will send a message to the DSP to initiate a PWRM_sleepDSP operation. Note that since DSPLink for OMAP-L138 C6-Integra DSP+ARM processors is not yet power aware (it is for OMAP™ devices), care needs to be taken to include mechanism to wake up the DSP.
With power management services becoming available in mainline Linux kernel, it is possible to achieve substantial power savings by just “turning them on.” We hope that this white paper will encourage you to look into the hardware features of your platform to see if it is capable of reducing operating power to embrace “Go Green.”

For those interested in experimenting with OMAP-L138 C6-Integra DSP+ARM processor power management features described in this paper, information about the EVM and software development kit (SDK) is available at http://focus.ti.com/docs/prod/folders/print/omap-l138.html
IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause serious personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal and regulatory requirements concerning their products and any use of TI products in such safety-critical applications. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for such statements.

TI products are not authorized for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal and regulatory requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any application-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in automotive applications or environments unless the TI products are specifically designated by TI as automotive. Buyers acknowledge and agree that the Buyer's use of TI products which TI has not designated as automotive is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

<table>
<thead>
<tr>
<th>Products</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audio</td>
<td>Communications and Telecom</td>
</tr>
<tr>
<td>Amplifiers</td>
<td>Computers and Peripherals</td>
</tr>
<tr>
<td>Data Converters</td>
<td>Consumer Electronics</td>
</tr>
<tr>
<td>DLP® Products</td>
<td>Energy and Lighting</td>
</tr>
<tr>
<td>DSP</td>
<td>Industrial</td>
</tr>
<tr>
<td>Clocks and Timers</td>
<td>Medical</td>
</tr>
<tr>
<td>Interface</td>
<td>Security</td>
</tr>
<tr>
<td>Logic</td>
<td>Space, Avionics and Defense</td>
</tr>
<tr>
<td>Power Mgmt</td>
<td>Transportation and Automotive</td>
</tr>
<tr>
<td>Microcontrollers</td>
<td>Video and Imaging</td>
</tr>
<tr>
<td>RFID</td>
<td>Wireless</td>
</tr>
<tr>
<td>RF/IF and ZigBee® Solutions</td>
<td></td>
</tr>
</tbody>
</table>

TI E2E Community Home Page: e2e.ti.com

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2011, Texas Instruments Incorporated