The design and application of the PowerPC 405LP energy-efficient system-on-a-chip

K. J. Nowka
G. D. Carpenter
B. C. Brock

The PowerPC® 405LP system-on-a-chip (SoC) processor, which was developed for high-content, battery-powered application space, provides dynamic voltage-scaling and on-the-fly frequency-scaling capabilities that allow the system and applications to adapt to changes in their performance demands and power constraints during operation. The 405LP operates over a voltage supply range of 1.95 to 0.9 V with a range of power efficiencies of 1.0 to 3.9 MIPS/mW when executing the Dhrystone benchmark. Operating system and application software support allow the applications to take full advantage of the energy-efficiency capabilities of the SoC. This paper describes the organization of the SoC design, details the capabilities provided in the design to match the performance and power consumption with the need of the application, describes how these capabilities are employed, and presents measured results for the PowerPC 405LP processor.

Introduction

The high-content battery-powered segment of the marketplace continues to demand greater performance while strictly limiting the power consumption of device electronics. This market segment includes information appliances such as Web pads, advanced personal digital assistants (PDAs), cell phones, and small-form-factor PCs. Peak performance demands can exceed 500 MIPS. Such applications are also characterized by significant fractions of idle time. During active computation, their required performance tends to vary widely and rapidly as a function of the workload. The constraints of battery lifetime and low-cost packaging place stringent limits on standby and active power consumption. During peak activity, the power consumption of the processor core is best kept at or below about 500 mW. Because these applications may have long periods of inactivity, the standby power of the inactive processor and the energy consumption of the sleep monitor must be minimized. To address battery-powered applications, we have developed a voltage-scalable system-on-a-chip (SoC) platform [1] in the IBM 0.18-μm, 1.8-V bulk CMOS foundry process [2, 3]. The processor contains a 32-bit PowerPC* core with instruction and data caches. The SoC uses IBM CoreConnect* technology [4] to integrate a rich set of memory and I/O interfaces. In addition, on-chip hardware accelerators have been developed to improve the performance of important tasks and decrease their power consumption. A block diagram of the SoC is shown in **Figure 1**.

Dynamic performance and energy control

The performance of a processor is determined by the frequency of operation and the number of operations that can be completed on average per processor clock cycle. The maximum frequency of the processor can be accurately modeled by the Sakurai α-power delay model [5], which uses a fitting parameter α to represent the degree of velocity saturation ($\alpha = 2$ implies no velocity saturation, and $\alpha = 1$ implies full velocity saturation). The

0018-8646/03/$5.00 © 2003 IBM

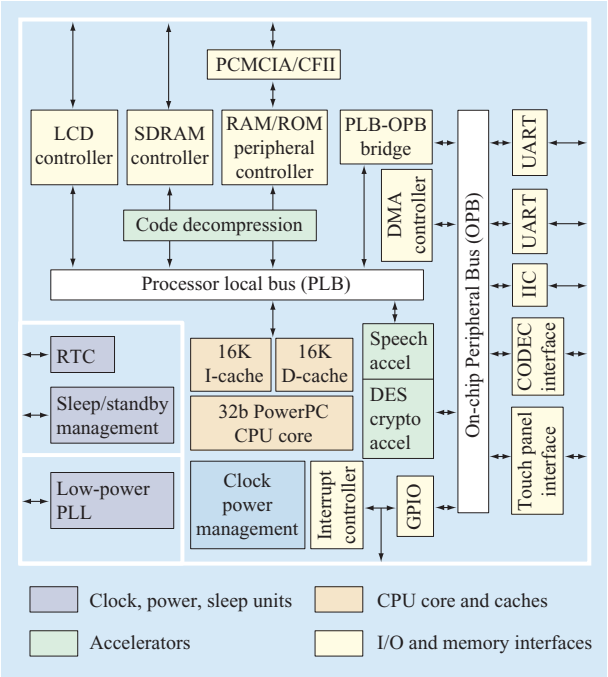

Figure 1

PowerPC system-on-a-chip structure.

delay, T_d, of switching a load capacitance through a transistor is modeled as

$$T_d = kV_{DD}/(V_{DD} - V_t)^{\alpha},$$

where V_{DD} is the supply voltage and V_t is the threshold voltage of the transistor. By using this model, the frequency of the system is determined by the summation of delays along a critical path through the logic:

$$f = 1/(\Sigma_i\, Td_i) = 1/\Sigma_i\, [k_i\, V_{DD}/(V_{DD} - V_t)^{\alpha}]. \quad (1)$$

The dynamic power consumed in a CMOS transistor resulting from the switching of the load capacitance, C_L, through a voltage, V_{DD}, at a frequency, f, is

$$P = \tfrac{1}{2} C_L V_{DD}^2 f.$$

The dynamic power consumed in a chip is the sum of the power of all switching nodes. It can be modeled as the power of switching the average switching capacitance of the system, C_{sw}, through a voltage of V_{DD}:

$$P = \tfrac{1}{2} C_{sw} V_{DD}^2 f. \quad (2)$$

Examination of Equation (2) shows that if the slight voltage dependence of the switching capacitance is ignored, the dynamic power consumption of a system is quadratically more sensitive to power-supply voltage than is the frequency. Voltage-scaled systems [6–8] take advantage of this greater sensitivity to improve the power efficiency of the operation of the system by reducing the power-supply voltage and thereby reducing both the frequency of operation and the power consumption when demands on the system are low. Dynamic voltage-scaled systems adjust the supply voltage dynamically to meet performance demands while minimizing power consumption [9–13]. The dynamic energy consumption is reduced quadratically with the decreasing supply, while the commensurate maximum frequency decreases approximately linearly near the nominal supply and superlinearly farther from the nominal supply. For battery-powered applications in which both energy efficiency and performance are crucial, voltage scaling allows a wide range of options in the tradeoff between performance and power consumption.

PowerPC 405LP system organization

The PowerPC 405LP SoC was developed to take advantage of the power-efficiency potential of dynamic voltage scaling (DVS). This SoC design consists of a high-performance embedded 32-bit PowerPC processor core. The processor core for this design was based upon an existing, fixed-voltage PowerPC 405 core [14]. The core includes a five-stage pipelined CPU with a hardware multiply–accumulate unit, hardware division, static branch prediction support, and a 64-entry, fully associative translation lookaside buffer. Single-cycle-access, two-way set-associative 16-KB SRAM instruction and data caches are connected to the processor core.

The processor core connects to external SDRAM, and to external memory, storage, and network through the PCMCIA/Compact Flash interface by way of the 64-bit processor local bus (PLB). An integrated liquid crystal display (LCD) controller is also attached to the PLB. Lower-bandwidth on-chip peripheral bus (OPB) I/O interfaces include dual universal asynchronous receivers–transmitters (UARTs), an I2C interface, general-purpose I/O lines, an audio coder–decoder (CODEC), and a touch panel controller interface.

A custom dedicated speech accelerator, an instruction decompression engine [15], and a data encryption standard (DES) accelerator core are included on the SoC to accelerate key tasks (**Figure 2**). The SoC contains a low-voltage phase-locked-loop (PLL) core and a real-time-clock (RTC) core for on-chip clock generation, as well as a clock power-management core and a sleep-management core. Figure 2 shows a die photograph of the SoC after processing of the third level of metal. The die is 6.02 mm on a side and is ringed by peripheral I/O pads. This device was fabricated by using the IBM CMOS 7sf, 0.18-μm bulk CMOS process [2, 3] with five levels of copper interconnect. This technology has a nominal supply voltage of 1.8 V, threshold voltages of 0.43 V/−0.38 V for

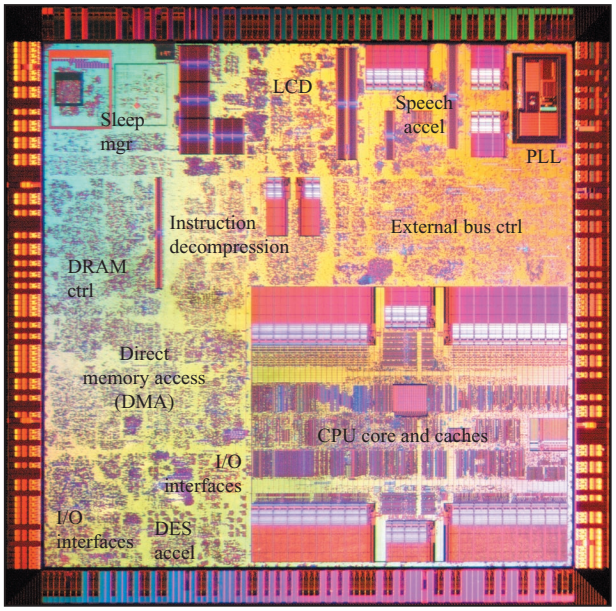

Figure 2

PowerPC SoC die photograph.

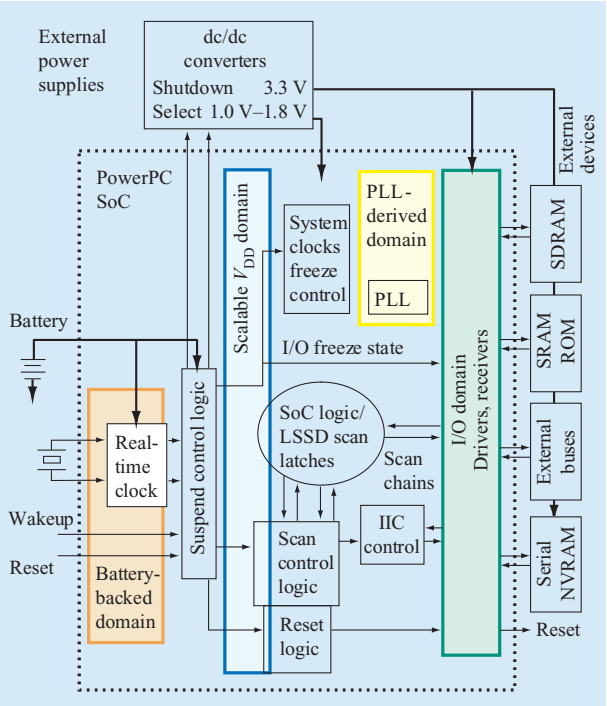

Figure 3

SoC supply domains and signal interfaces.

n-MOS/p-MOS devices, and a gate-oxide thickness of 3.5 nm.

The PowerPC processor core satisfies the performance demands of the information appliance applications. Additional capabilities were developed for this processor to reduce both the active and the standby power consumption of the device. The active power consumption is reduced when resource demands are lowered through the use of dynamic voltage scaling, dynamic frequency scaling, and unit- and register-level functional clock gating [16]. In addition, dedicated hardware accelerators perform key tasks more efficiently. Finally, the SoC integration allows this design to avoid costly off-chip accesses.

The measured power consumption for the PowerPC SoC while executing the synthetic benchmark Dhrystone Version 2.1 [17, 18] is 570 mW. These measurements are for nominal hardware with a 1.8-V supply voltage and a processor clock frequency of 333 MHz. The performance under these conditions corresponds to 500 MIPS. When the supply is lowered to 0.85 V, the nominal hardware SoC consumes just 33 mW at 66 MHz. This represents a frequency range of 5:1 with a power range of 17:1.

For periods of device inactivity, several levels of standby power reduction can be achieved [1]. Simply lowering the voltage and frequency to their minimum levels saves standby power while allowing a very rapid transition to full performance. By disabling the clocks, a low-leakage sleep state can be entered in which further power can be saved in standby mode. For periods of extended inactivity, the 405LP supports two hibernation modes: One mode requires the software to write any necessary state to some persistent storage, while the second mode requires the state of the machine to be moved to nonvolatile storage by way of the scan-chain prior to hibernation.

System capabilities for dynamic voltage and frequency scaling

Providing a broad range of active operating points requires additional capabilities in the power supply and its distribution, and clock generation and its distribution.

Power-supply capabilities

To support data voice switching (DVS) in this SoC, the power distribution has been divided into four distinct domains (**Figure 3**): two persistent[1] voltage domains, one dynamically voltage-scaled domain, and one internally derived domain. The I/O drivers and receivers are powered by a persistent 3.3-V supply. The real-time clock and the logic associated with controlling the voltage of the cores are powered by a persistent, battery-backed 1.8-V supply. The supply voltage for the logic in the processor core, the caches, the peripheral SoC cores, and the accelerators is dynamically varied between 1 V and 1.8 V.

[1] The supply is at a fixed voltage and is not turned off.

Finally, the phase-locked loop (PLL) [19] is powered by an on-chip linear regulator, which derives a constant 1-V supply from the dynamically varying logic supply. The power domains and the signal interfaces between the domains are shown in Figure 3.

An on-chip supervisor controls the external dc-to-dc converter, which provides the logic supply. Under software control, the supervisor requests changes to the logic supply voltage during active operation and requests shutdown of the logic supply during hibernation.

Because of potential voltage differences encountered in traversing supply domains, voltage-translating level-shifters are employed at the domain interfaces. In addition, because the logic supply is not persistent during hibernation, signals passing from the logic domain to the persistent I/O supply domain and the persistent battery-backed domain are latched by level-shifters at these interfaces.

Clock generation subsystem capabilities

The clock generation subsystem is key to supporting a voltage-scalable design. The system clocks can be derived from several sources: a low-frequency external source, which is used for the real-time clock, an external auxiliary clock, the PLL reference clock, or the clock from the on-chip PLL.

The PLL uses an interleaved five-stage voltage-controlled ring oscillator [20]. The voltage-controlled oscillator (VCO) tuning range is 513 to 1017 MHz across full process corners. Duty-cycle correction is accomplished through a divide-by-two of frequency performed in the output path-level shifter. Adjustment to the output clock frequency can be performed on-the-fly by modifying the output clock divider through a write operation to a control register. This allows adjustment of the frequency across a range of ½ to 1/128th of the VCO frequency. A glitchless output multiplexor avoids spurious clock pulses during this operation and allows for clock freezing for sleep and hibernation modes.

These on-the-fly frequency-modification techniques can be used to provide dynamic frequency scaling for additional active power reduction. When the performance demands of the application decrease, the system software can lower the operating frequency. Under software control, at a given supply voltage, the frequency of the core can be varied from the maximum frequency down to 1/64th of the maximum. This allows the frequency to be dynamically set as low as 4.2 MHz.

The dynamic voltage- and frequency-scaling capabilities provided by the power and clocking subsystems require a software infrastructure to manage the performance and power consumption of the SoC. By specifying the possible device operating points, managing the state of the device, initiating changes in the power operating state, and by implementing power policies through a power management system, the operating system and application software can take advantage of the improved energy efficiency.

Operating system power management architecture

Background

We are currently examining a general and flexible dynamic power management architecture for embedded systems. Although the concepts should be applicable to a broad class of operating systems and scalable processors, our initial focus and implementation will be for embedded Linux** for the PowerPC 405LP. There are several existing research and production implementations of processor voltage and frequency scaling for other processors [6–12]. Our research extends and augments the capabilities of these systems in several important ways.

We recognize that the overriding power management goal in portable systems is to reduce system-wide energy consumption. The current generation of embedded processors are so power-efficient that the SoC processor may no longer be the major energy consumer in systems that include high-performance memories and large color displays. Therefore, a dynamic power management system that is concerned only with voltage- and frequency-scaling the processor core may be of limited use. Our dynamic power management architecture supports the ability of the PowerPC 405LP to rapidly scale internal and external bus frequencies, either in concert with or independent of the CPU frequency. Scaling bus frequencies at key points can produce significant reductions in system-wide energy consumption.

Another observation is that the breakdown of system-wide energy consumption, as well as the most effective way to manage this consumption, is highly application[2] dependent. Therefore, a dynamic power management architecture must be flexible enough to support multiple platforms with differing requirements. We believe that requirements for simplicity and flexibility are best served by leaving the workings of the dynamic power management completely transparent to most tasks and even to the core of the operating system itself. Our current research prototype requires no changes to programs or to the well-understood process management implementation of the operating system in order to achieve significant results. Furthermore, the tasks that control the system-wide power management policies can run entirely in user space and communicate their requirements to the operating system through a small set

[2] Throughout this section we use the terms *system* and *application* to indicate a complete embedded system (e.g., a cell phone or PDA) and the terms *program* and *task* to refer to software.

of system calls. The architecture also supports the ability of tasks to set their own power–performance characteristics for those special cases in which this is required.

Architecture

Our dynamic power management architecture for embedded Linux is based on a hierarchy of abstract objects. We generally expect a complete power management framework to be defined in advance for each application, by a system designer familiar with the characteristics of the embedded processor and any special features and requirements of the application. A parameterized framework is used by the operating system to control low-level details of the power management strategy, while a higher-level management task controls the overall dynamic power consumption of the application.

The dynamic power management hierarchy is illustrated in **Figure 4(a)**. In the following, the dynamic power management architecture is described from the bottom up.

1. Operating points

The lowest-level object in the power management architecture is the operating point. An operating point encapsulates a set of mutually constrained physical and abstract parameters that bear on a dynamic power management policy. At any given point in time, the system is executing at a particular operating point, and a dynamic power management system could properly be defined as the set of rules and procedures that move the system from operating point to operating point as the system evolves. By their nature, operating points are processor- and application-dependent. Operating points for the PowerPC 405LP currently specify a core voltage level, CPU and bus frequencies, memory timing parameters, and other clocking-related data. The system designer is responsible for defining as many operating points as are necessary for the power management needs of the application. We explain below how the system selects an operating point and shifts from one operating point to another.

2. Device management and operating point congruence classes

The states of on-board and external peripheral devices have a tremendous influence on system-wide energy consumption and on the choice of operating point. For example, the PowerPC 405LP has an on-board LCD controller which uses an SDRAM framebuffer.[3] If the LCD controller is enabled, any valid operating point for the system must specify a bus frequency high enough to satisfy the refresh rate of the display, which is ultimately controlled by a variable pixel clock frequency that is also specified by the operating point. When the LCD is disabled (for example, when a PDA is used simply as an MP3 player), significant system-wide energy reductions may be achieved by reducing these frequencies.

[3] The framebuffer is a dedicated memory that is used by the display subsystem.

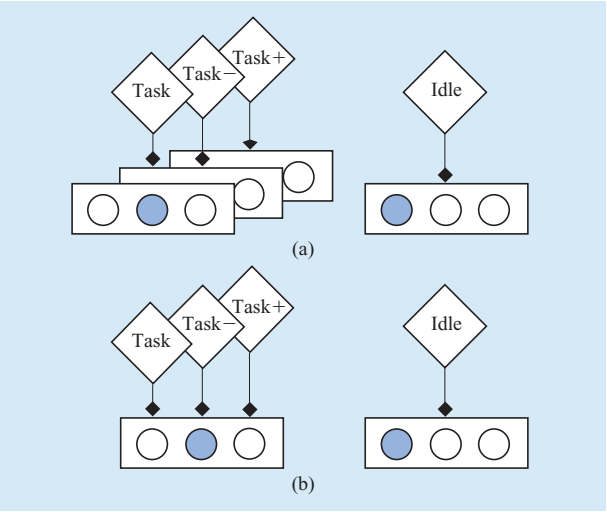

Figure 4

(a) Fully enumerated power-management policy mapping each operating state to a congruence class of operating points, only one of which is selected at any given time. (b) Simplified power-management policy mapping all task states to a common congruence class of operating points.

In general, our power management architecture attempts to relieve the high-level power management task from the responsibility of managing device states, and from having to respond to changes in device states. This feature is implemented in two ways. First, we expect low-level device drivers to aggressively manage the power consumption of the devices they control. For example, if a PowerPC 405LP system is not currently producing or consuming audio data, the device driver for the audio CODEC interface is expected to power-down the external CODEC; it also commands the on-board clock and power manager to clock-gate the CODEC interface peripheral. This may change the bandwidth (frequency) requirements for the on-board peripheral bus, and hence trigger a change in the operating point. This all happens without the knowledge or intervention of the power management task.

When devices change state (and hence their requirements for system resources), these state changes are communicated to the operating system power management system for a potential change in the operating point. The next-higher-level object in the power management hierarchy, the congruence class of operating points, implements this feature. This object groups together operating points that the system designer has declared to be equivalent, in terms of a power management strategy, adjusting for any constraints imposed by devices. Simple rules are defined to

automatically select one of several possibly valid operating points from the congruence class whenever device states change. This mechanism frees the power management task to focus on high-level management while ensuring that the system always operates at the best operating point (as defined by the system designer) consistent with the current policy and device states.

3. Operating states

The next-higher-level object in the dynamic power management architecture is the operating state. This concept refers to the dynamic state of the system as it relates to power management and the choice of an appropriate operating point. In our hierarchy, an operating state object simply maps an abstract system state to a congruence class of operating points.

The introduction of the concept of the operating state was first motivated by the observation that significant system-wide energy savings can be achieved by reducing CPU and bus frequencies as well as core voltage while the system is idle. Therefore, a mechanism is required to specify a different operating point during the times when programs are executing and the times when the system is idle. This naturally leads to a distinction between a task state and an idle state, each with a potentially different operating point. The fact that the PowerPC 405LP can scale frequencies with a latency measured in microseconds means that this feature can be exploited for even relatively short idle periods while the system is blocked on I/O operations or timeouts. The transition from a task to an idle operating point and back is smoothly and efficiently managed by the operating system.

The concept of an operating state also provides for task-specific operating points for power-aware tasks. The dynamic power management architecture for the PowerPC 405LP includes several task operating states, each associated with a particular power–performance level. The default task state is expected to be used by the large majority of tasks as most tasks now use the default scheduling policy of the operating system. Tasks with special requirements may be specified to run in different task states.

Figure 4(a) shows an example in which each of the task states specifies a special set of operating points for tasks in that state via their congruence class mapping. For example, the task− (task-minus) state may specify a set of operating points that are more power-efficient but offer lower performance than the default state, while the task+ state may specify high-performance but less efficient operating points.

Alternatively, **Figure 4(b)** illustrates a policy in which all of the task states reference the same congruence class. Here, the activity of tasks in non-default states may simply be used by the power management task to control the global dynamic power management policy. For example, if the power management task observes that most of the system activity is in the task− operating state, it may decide to change the overall power management policy to a more power-efficient one.

4. Policies and power managers

The highest-level abstraction of our power management architecture is the policy, which maps operating states to congruence classes of operating points. A power management system design specifies at least one policy, and may specify as many different policies as necessary for different situations. The policy in effect at any given point in time completely controls the operating point of the system in any operating state.

As an example, we consider the implementation of a simple activity-based power manager for a dynamic voltage-scalable system like the 405LP. Systems like this use CPU utilization to drive the dynamic power management policy. As system activity increases, the power manager increases the system frequency (including the core voltage) in an attempt to provide adequate performance for the workload while minimizing power consumption. These simple types of power managers have been proven to be very effective for managing the diverse workloads of portable information appliances.

We have implemented such a power manager in the framework described here. In this design, power policies are associated with CPU core voltages. The power manager uses the mechanism of setting a policy to move the system from voltage (and frequency) level to level. Note that the abstraction relieves the power manager of all of the low-level details: The policy describes consistent operating points for the idle state as well as the task states, regardless of the state of peripheral devices, and if special operating points are required for non-default task states, these are transparently encoded by the congruence class mappings for those states. In fact, the power manager task is not even aware of the particular voltages and frequencies associated with the policies. The power manager simply interprets a set of abstract rules, specified by the system designer, that describe the events that should move the system from power policy to power policy.

The power manager operates by periodically querying the power management system as to the amount of time the application has been spending in the various operating states. When system activity increases past a certain threshold, indicated by the ratio of time spent in the task state vs. the idle state, the rule set causes the power manager to move to a higher performance (higher voltage and frequency) policy. Decreases in system activity trigger rules that move to a lower performance policy. The power manager can query the system at relatively high

frequencies with minimal overhead. The result is a system that provides both interactive responsiveness and very power-efficient operation during idle periods.

System applications of energy-efficient capabilities

Reducing power consumption through dynamic scaling can be useful at the application level as well as the operating system (OS) level. One of the more important and demanding applications for small, highly portable devices is displaying video content. One popular standard video format is MPEG-4 for playing quarter video graphics array (QVGA). This application is also good for demonstrating how some applications can be made aware of power management and thus more efficient. Delivering QVGA at 30 frames per second satisfies the high peak performance demands and real-time deadlines of users of handheld devices.

Two versions of the MPEG-4 player were written and instrumented on the 405LP evaluation board. The first was a standard implementation delivering QVGA at 30 frames per second, while the second was a power-management-aware version. **Figure 5** shows oscilloscope traces from the two MPEG-4 players running the same video clip. On the left is the standard implementation, and on the right the power-management-aware version. The scalable logic supply voltage is shown in yellow in the bottom third of both traces. The top two thirds contains the logic power consumption of the 405LP in green and the total power consumption of the 405LP in red. As the video is played, the nature of the power consumption is highly dynamic. Each frame consists of a high-power region as the CPU is busy decoding the frame and converting YUV to RGB color, followed by a period of low activity until the next frame is needed at the next frame interval (33 ms later). In the frame on the right, it can be seen that the duration of the high processor demand varies from frame to frame. In some cases, the frame calculation finishes just in time; in others, the demand is low for more than 50% of the frame interval. Because this duty cycle is determined by the content of each frame, it is not easily predictable. The application lends itself well to a power management scheme that can respond quickly and dynamically to real-time changes in workload. The workload changes are rapid enough to require the applications rather than the operating system to initiate power management.

To meet the peak rendering demand of this application, the system is run at 266 MHz on the CPU and 133 MHz on the memory bus. The trace on the left shows that during the decode and color conversion phase, approximately 550 mW is consumed by the 405LP logic, and 700 mW in total. When the frame is completed, the logic power drops to approximately 200 mW as the demand on the CPU is reduced and the processor idles, but the operating point is unchanged.

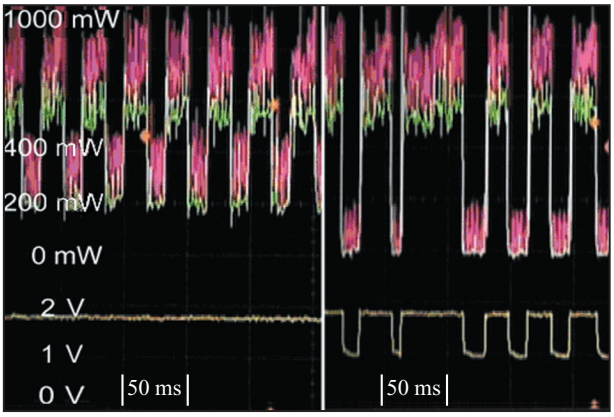

Figure 5

MPEG-4 decode measurements.

By rewriting the application to make it aware of the fast dynamic scalability of the 405LP, significant power can be saved during the low-demand state while waiting for the next frame interval. In the right half of the oscilloscope trace presented in Figure 5, it can be seen that when the application has completed the frame calculation and there is slack, it requests a change in the operating point of the 405LP to reach a much lower performance (CPU = 66 MHz and memory = 66 MHz) and a more efficient (V_{DD} logic = 1.0 V) operating point. This lower operating point is more than sufficient to sustain the demand between frames. The result is that between frames the idle power consumption drops to a few milliwatts, and the average power consumption drops 25–30% as the video is played.

The transitions between operating frequencies are occurring in less than 1 μs. The yellow trace at the bottom shows the logic supply shifting between 1.8 and 1.0 V in about 1 ms, limited here by the slew rate of the voltage converter on the reference board. During these transitions, the CPU is not stopped and can sustain other real-time demands of the system, such as an MPEG-3 audio stream.

Summary

The PowerPC 405LP is a 5.8-million-transistor, 36-mm^2 design optimized for energy efficiency. Measurements on nominal hardware show that the SoC consumes 570 mW when executing Dhrystone 2.1 at the nominal supply voltage at a frequency of 333 MHz. This corresponds to 500 Dhrystone MIPS. When the supply is lowered to 0.85 V, the nominal hardware SoC consumes just 33 mW at 66 MHz. This represents a frequency range of 5:1 with a power range of 17:1. The 405LP implements many

features to improve power efficiency when the SoC is active: This device makes use of SoC technology to integrate the full set of devices and interfaces demanded by the battery-powered mobile market, thereby eliminating power-inefficient off-chip interfaces. Under software control, both the voltage and the frequency of the processor can be modified, thereby allowing the performance demands of the application to be met while minimizing the dynamic power consumption. Unused storage and functions are not clocked, eliminating unnecessary energy consumption. Functional accelerators, which can perform key tasks more efficiently than the processor core, are included in the SoC.

In addition, the 405LP implements standby power-reduction features to ensure that power is not wasted when the SoC is inactive. This processor, under software control, can enter both a low-leakage sleep state and a state-preserving deep-sleep state to minimize standby power consumption. By applying these techniques, the active performance of the 405LP is not sacrificed, while standby power can be reduced as low as 54 μW.

The ability of the 405LP to dynamically adapt to changing requirements for performance and power is supported by the operating systems. The OS architecture for power management and the policies and power managers required in order to use dynamic scalability have been described. The MPEG-4 decode application is an example application in which the capabilities of the PowerPC 405LP have shown their value.

Acknowledgment

The PowerPC 405LP was developed jointly by the IBM Austin Research Laboratory, the IBM Microelectronics Division in Raleigh, NC, Burlington, VT, Austin, TX, and Yasu, Japan, as well as the IBM Tokyo Research Laboratory and the IBM Thomas J. Watson Research Center.

*Trademark or registered trademark of International Business Machines Corporation.

**Trademark or registered trademark of Linus Torvalds.

References

1. K. Nowka, G. Carpenter, E. MacDonald, H. Ngo, B. Brock, K. Ishii, T. Nguyen, and J. Burns, "A 0.9V to 1.95V Dynamic Voltage Scalable and Frequency Scalable 32-Bit PowerPC Processor," *IEEE International Solid-State Circuits Conference*, *Digest of Technical Papers*, 2002, pp. 340–341.
2. L. Su, S. Subbanna, E. Crabbe, P. Agnello, E. Nowak, R. Schulz, S. Rauch, H. Ng, T. Newman, A. Ray, M. Hargrove, A. Acovic, J. Snare, S. Crowder, B. Chen, J. Sun, and B. Davari, "A High-Performance 0.08 μm CMOS," *IEEE Symposium on VLSI Technology*, *Digest of Technical Papers*, 1996, pp. 12–13.
3. IBM Corporation, *0.18 Micron Technology*, Product Brief, Hopewell Junction, NY, February 21, 2003; see *http://www-3.ibm.com/chips/*.
4. IBM Corporation, *CoreConnect Bus Architecture*, Product Brief, Hopewell Junction, NY, September 1, 1999; see *http://www.chips.ibm.com/techlib/techlib.nsf/productfamilies/CoreConnect_Bus_Architecture/*.
5. T. Sakurai and A. Newton, "Alpha-Power Law MOSFET Model and Its Applications to CMOS Inverter Delay and Other Formulas," *IEEE J. Solid State Circuits* **25,** No. 2, 584–594 (April 1990).
6. E. Vittoz, "Low-Power Design: Ways to Approach the Limits," *IEEE International Solid-State Circuits Conference*, *Digest of Technical Papers*, 1994, pp. 14–18.
7. M. Horowitz, T. Indermaur, and R. Gonzalez, "Low-Power Digital Design," *IEEE Symposium on Low Power Electronics*, *Digest of Technical Papers*, 1994, pp. 8–11.
8. R. Gonzalez, B. Gordon, and M. Horowitz, "Supply and Threshold Voltage Scaling for Low Power CMOS," *IEEE J. Solid-State Circuits* **32,** No. 8, 1210–1216 (August 2000).
9. T. Burd and R. Brodersen, "Energy Efficient CMOS Microprocessor Design," *Proceedings of the Twenty-Eighth Hawaii International Conference on System Sciences*, 1995, pp. 288–297.
10. K. Suzuki, S. Mita, T. Fujita, F. Yamane, F. Sano, A. Chiba, Y. Watanabe, K. Matsuda, T. Maeda, and T. Kuroda, "A 300 MIPS/W RISC Core Processor with Variable Supply-Voltage Scheme in Variable Threshold-Voltage CMOS," *Proceedings of the IEEE Conference on Custom Integrated Circuits*, 1997, pp. 587–590.
11. T. Kuroda, K. Suzuki, S. Mita, T. Fujita, F. Yamane, F. Sano, A. Chiba, Y. Watanabe, K. Matsuda, T. Maeda, T. Sakurai, and T. Furuyama, "Variable Supply-Voltage Scheme for Low-Power High-Speed CMOS Digital Design," *IEEE J. Solid-State Circuits* **33,** No. 3, 454–462 (March 1998).
12. T. Burd, T. Pering, A. Stratakos, and R. Brodersen, "A Dynamic Voltage Scaled Microprocessor System," *IEEE International Solid-State Circuits Conference, Digest of Technical Papers*, 2000, pp. 294–295.
13. K. Nowka, G. Carpenter, E. MacDonald, H. Ngo, B. Brock, K. Ishii, T. Nguyen, and J. Burns, "A 32-Bit PowerPC System-on-a-Chip with Support for Dynamic Voltage Scaling and Dynamic Frequency Scaling," *IEEE J. Solid-State Circuits* **37,** 1441–1447 (November 2002).
14. IBM Corporation, *PowerPC Embedded Cores*, Product Brief, Hopewell Junction, NY, June 1, 2000; see *http://www.chips.ibm.com/techlib/techlib.nsf/products/PowerPC_405_Embedded_Cores/*.
15. T. M. Kemp, R. K. Montoye, J. D. Harper, J. D. Palmer, and D. J. Auerbach, "A Decompression Core for PowerPC," *IBM J. Res. & Dev.* **42,** No. 6, 807–812 (November 1998).
16. A. Correale, "Overview of the Power Minimization Techniques Employed in the IBM PowerPC 4xx Embedded Controllers," *IEEE Symposium on Low Power Electronics*, *Digest of Technical Papers*, 1995, pp. 75–80.
17. R. P. Weicker, "Dhrystone: A Synthetic Systems Programming Benchmark," *Commun. ACM* **27,** No. 10, 1013–1030 (1984).
18. R. P. Weicker, "Dhrystone Benchmark: Rationale for Version 2 and Measurement Rules," *SIGPLAN Notices* **23,** No. 8, 49–62 (1988).
19. D. Boerstler, G. Carpenter, H. Ngo, and K. Nowka, "Dynamically Scalable Low Voltage Clock Generation System," U.S. Patent 6,515,530, February 4, 2003.
20. D. Boerstler, "Multiphase Voltage Controlled Oscillator with Variable Gain and Range," U.S. Patent 6,353,369, March 5, 2002.

Received November 18, 2002; accepted for publication May 2, 2003

Kevin J. Nowka *IBM Research Division, Austin Research Laboratory, 11501 Burnet Road, Austin, Texas 78758 (nowka@us.ibm.com).* Dr. Nowka received a B.S. degree in computer engineering from Iowa State University in 1986 and M.S. and Ph.D. degrees in electrical engineering from Stanford University in 1988 and 1995, respectively. In 1996 he joined the IBM Austin Research Laboratory, where he has conducted research on CMOS VLSI circuits and arithmetic functions for application to the design of high-frequency and low-power CMOS processors. Prior to his graduate work, he was a Member of Technical Staff at AT&T Bell Laboratories. He holds twenty patents related to processor design.

Gary D. Carpenter *IBM Research Division, Austin Research Laboratory, 11501 Burnet Road, Austin, Texas 78758 (carpentg@us.ibm.com).* Mr. Carpenter received a B.S. degree in electrical engineering from the University of Kentucky in 1983. He is a Research Staff Member at the IBM Austin Research Laboratory, and is chief architect for ultralow-power embedded processors. He has worked in the areas of hardware system architecture, system design, and analog circuit and logic design. Currently his focus is research on energy-efficient circuits, processors, and systems.

Bishop C. Brock *IBM Research Division, Austin Research Laboratory, 11501 Burnet Road, Austin, Texas 78758 (bcbrock@us.ibm.com).* Mr. Brock received an M.S. degree in computer science from the University of Texas at Austin in 1987. He has been a Research Staff Member at the IBM Austin Research Laboratory since 1997. While at IBM, Mr. Brock has been active in the areas of scalable and low-power system design, performance monitoring, and hardware and software power-management techniques for embedded systems. Prior to joining IBM, he worked in the areas of automated reasoning and formal verification of hardware.