Software Design for Low-Power Embedded Systems

Youngsoo Shin‡and Tohru Ishihara§
‡Center for Collaborative Research, University of Tokyo, Tokyo 153-8505, Japan
§VLSI Design and Education Center, University of Tokyo, Tokyo 133-8656, Japan

Abstract

Power consumption has been a critical design constraint in the design of digital embedded systems, and there has been significant progress in the design of power efficient hardware, especially in the domain of digital integrated circuits, from technology up to system level. However, such hardware is not usually used in power-efficient way, mainly because of software designed not power in mind. This tutorial addresses several techniques to build software contents for embedded systems in a way that power consumption is reduced. We consider several design opportunities from application, compiler, and operating system.

1 Introduction

Recently, power consumption[1] has been a critical design constraint in the design of digital embedded systems due to widely used portable systems, which frequently require low-power consumption with high speed and complex functionality. The design of such systems often involves reprogrammable processors such as microprocessors, microcontrollers, and DSPs in the form of off-the-shelf components or cores. Furthermore, an increasing amount of system functionality tends to be realized through software, which is leveraged by the high performance of modern processors. Thus, design of software that leads to reduced power consumption of underlying hardware has increasing importance.

In this tutorial, we survey various techniques to obtain low-power in viewpoint of software. Especially we focus on two components: application and operating system (OS), which constitute most embedded systems.

2 Low-Power Hardware

In viewpoint of *hardware*, there has been significant progress to design power-efficient hardware in different design abstraction levels: device and technology level, layout level, transistor level, logic level, register transfer level, and so on. This is especially true in digital CMOS (Complementary Metal Oxide Silicon)-based integrated circuits. It is a well-known fact that power consumption in CMOS circuits can be decomposed into two parts: static and dynamic. The dynamic power consumption, which is a dominant factor, is described by

$$P = \alpha \cdot f \cdot C_L \cdot V_{dd}^2, \tag{1}$$

where α is the expected number of transitions per cycle, called switching activity, f is the clock frequency, C_L is the average load capacitance, and V_{dd} is the supply voltage. Thus, the energy dissipated during time period of T is given by

$$E = P \cdot T. \tag{2}$$

Equations (1) and (2) together suggest several facts worth of notice when we are to design low-power software[2] First, the software optimized for speed is energy-efficient. This indicates that techniques to develop speed-optimized software also can be used to design low-power software. Second, if the speed-optimized software is to be executed on a hardware with fixed f and V_{dd}, the only way to obtain low-power is to design software in a way that αC_L is reduced, which is the main topic of the next section.

If we turn our attention to processors themselves, there is another possibility. To reduce the power consumption of processors, two kinds of features are widely used: one is to bring a processor into a power-down mode when the processor is in an idle state, where only certain parts of the processor such as the clock generation and timer circuits are kept running; the other is to dynamically change the speed of a processor by varying the clock frequency along with the supply voltage when the required performance on the processor is lower than the maximum. Given a processor with these features, we are confronted with software design problems to *design application programs* and to *manage the mix of applications* in such a way that the power consumption is minimized. This is the topic of section 4.

3 Application Design for Low-Power

In order to design low-power applications, we first need methods to simulate, estimate, or analyze power consumption of processors when a certain application executes on them. This can be done by augmenting the information of power consumption for each instruction to the conventional performance estimation tool [1], [2]. To obtain system-wise power consumption, which includes, for example, power consumed by memory subsystem and DC-DC converter as well as that consumed by processors, power model of each component can be combined with that of processors [3].

There are not much *specific* methods to design energy-efficient applications, because applications optimized for speed are also energy-efficient. However, designing applications considering power management in OS-level can lead to promising result [4], [5], [6]. Once applications are designed, it is the responsibility of compiler to generate executable code. Most of power-efficient compilation techniques have been proposed during the back-end stage, because underlying hardware architecture is not exploited during the front-end stage [7]. These techniques include instruction scheduling [8], [9], register allocation [10], memory mapping [11], [12], and encoding [13], [14], [15]. If there is a possibility of architectural modification, power consumed by cache [16] or external memory [17], [18], [19], [20], [21] can be reduced. For embedded DSP application, there is more possibility of power reduction such as memory bank as-

[1] *Power consumption* and *energy dissipation* are used interchangeably. More precisely, power consumption in this paper means *average power consumption*, which is equal to energy dissipation divided by the time period of interest.

[2] *Low-power software* is misleading because software itself does not consume power. Nevertheless, we use the expression for brevity.

signment and instruction packing [22], which stem from the fact that DSPs frequently have irregular architectures.

4 Power Management

Depending on the way OS primitives are used by applications and on OS scheduling strategy, OS itself consumes sizable power. Thus, to design power-efficient embedded systems, we need to analyze power consumption of OS and applications [6].

In most embedded systems, a processor often waits for some event from its environment wasting its power. To reduce the waste, modern processors are often equipped with various levels of power modes. In the conventional approach employed in most portable appliances, called time-out based shutdown, a processor enters power-down mode after it stays in an idle state for a predefined time interval. Since the processor still wastes its energy while in the idle state, this approach fails to obtain a large reduction in energy when the idle interval occurs frequently but its length is short. In predictive method [23], [24], the length of next idle period is predicted based on a history of component usage. The predicted value becomes the metric to determine whether it is beneficial to enter power-down modes or not. This method focuses on event-driven applications such as user-interfaces where the latency caused by the mismatch between the predicted value and the actual value can be tolerated. If we turn our attention to power management of entire system, which consists of interacting components, the problem of power management can be formulated as one of optimization problems [25].

A digital system designed with a fixed supply voltage works at a fixed speed and then can be made idle if the computational demand is less than the maximum. If the supply voltage is lowered dynamically to the lowest value satisfying the required speed constraint of the system, less power would be consumed. A scheduling method to reduce power consumption of a variable speed processor (VSP) was first proposed in [26]. Static scheduling methods for real-time systems were proposed in [27], [28], [29]. Combined static and dynamic scheduling method for a processor with two levels of speeds is proposed in [30]. Specific methods for widely used priority-based real-time scheduling was proposed in [31], and it is combined with application design to result in better result [32].

References

[1] V. Tiwari, S. Malik, and A. Wolfe, "Power analysis of embedded software: a first step towards software power minimization," *IEEE Trans. on VLSI Systems*, vol. 2, no. 4, pp. 437–445, Dec. 1994.

[2] C. H. Gebotys and R. J. Gebotys, "An empirical comparison of algorithmic, instruction, and architectural power prediction models for high performance embedded DSP processors," in *Proc. Int'l Symposium on Low Power Electronics and Design*, Aug. 1998, pp. 121–123.

[3] T. Simunic, L. Benini, and G. De Micheli, "Cycle-accurate simulation of energy consumption in embedded systems," in *Proc. Design Automat. Conf.*, June 1999, pp. 867–872.

[4] Intel Corp., "Recommendations for writing power friendly software," http://www.intel.com/ial/ipm/winapp.htm.

[5] S. Lee and T. Sakurai, "Run-time power control scheme using software feedback loop for low-power real-time applications," in *Proc. Asia South Pacific Design Automat. Conf.*, Jan. 2000, pp. 381–386.

[6] R. Dick, G. Lakshminarayana, A. Raghunathan, and N. Jha, "Power analysis of embedded operating systems," in *Proc. Design Automat. Conf.*, June 2000, pp. 312–315.

[7] A. V. Aho, R. Sethi, and J. D. Ullman, *Compilers: Principles, Techniques, and Tools*, Addison-Wesley, Reading, Mass., 1988.

[8] C. L. Su, C. Y. Tsui, and A. M. Despain, "Low power architecture design and compilation technique for high-performance processors," in *Proc. IEEE COMPCON*, Feb. 1994, pp. 209–214.

[9] H. Tomiyama, T. Ishihara, A. Inoue, and H. Yasuura, "Instruction scheduling for power reduction in processor-based system design," in *Proc. Design, Automat. and Test in Europe Conference and Exhibition*, Feb. 1998, pp. 855–860.

[10] H. Mehta, R. M. Owens, M. J. Irwin, R. Chen, and D. Ghosh, "Techniques for low energy software," in *Proc. Int'l Symposium on Low Power Electronics and Design*, Aug. 1997, pp. 72–75.

[11] P. Panda and N. Dutt, "Reducing address bus transitions for low power memory mapping," in *Proc. European Design & Test Conf.*, Mar. 1996, pp. 63–67.

[12] S. Wuytack, J. Diguet, F. Catthoor, and H. De Man, "Formalized methodology for data reuse exploration for low-power hierarchical memory mappings," *IEEE Trans. on VLSI Systems*, vol. 6, no. 4, pp. 529–537, Dec. 1998.

[13] S. Ramprasad, N. R. Shanbhag, and I. N. Hajj, "A coding framework for low-power address and data busses," *IEEE Trans. on VLSI Systems*, vol. 7, no. 2, pp. 212–221, June 1999.

[14] Y. Shin and K. Choi, "Narrow bus encoding for low power systems," in *Proc. Asia South Pacific Design Automat. Conf.*, Jan. 2000, pp. 217–220.

[15] Y. Shin and T. Sakurai, "Coupling-driven bus design for low-power application-specific systems," in *Proc. Design Automat. Conf.*, June 2001.

[16] N. Bellas, I. Hajj, C. Polychronopoulos, and G. Stamoulis, "Architectural and compiler support for energy reduction in the memory hierarchy of high performance microprocessors," in *Proc. Int'l Symposium on Low Power Electronics and Design*, Aug. 1998, pp. 70–75.

[17] Y. Yoshida, B. Song, H. Okuhata, T. Onoye, and I. Shirakawa, "An object code compression approach to embedded processors," in *Proc. Int'l Symposium on Low Power Electronics and Design*, Aug. 1997, pp. 265–268.

[18] L. Benini, A. Macii, E. Macii, and M. Poncino, "Selective instruction compression for memory energy reduction in embedded systems," in *Proc. Int'l Symposium on Low Power Electronics and Design*, Aug. 1999, pp. 206–211.

[19] H. Lekatsas, J. Henkel, and W. Wolf, "Code compression for low power embedded system design," in *Proc. Design Automat. Conf.*, June 2000, pp. 294–299.

[20] T. Ishihara and H. Yasuura, "A power reduction technique with object code merging for application specific embedded processors," in *Proc. Design, Automat. and Test in Europe Conference and Exhibition*, Mar. 2000, pp. 617–623.

[21] L. Benini, A. Macii, E. Macii, and M. Poncino, "Synthesis of application-specific memories for power optimization in embedded systems," in *Proc. Design Automat. Conf.*, June 2000, pp. 300–303.

[22] M. T.-C. Lee, V. Tiwari, S. Malik, and M. Fujita, "Power analysis and minimization techniques for embedded DSP software," *IEEE Trans. on VLSI Systems*, vol. 5, no. 1, pp. 123–135, Mar. 1997.

[23] M. B. Srivastava, A. P. Chandrakasan, and R. W. Brodersen, "Predictive system shutdown and other architectural techniques for energy efficient programmable computation," *IEEE Trans. on VLSI Systems*, vol. 4, no. 1, pp. 42–55, Mar. 1996.

[24] C. Hwang and A. Wu, "A predictive system shutdown method for energy saving of event-driven computation," in *Proc. Int'l Conf. on Computer Aided Design*, Nov. 1997, pp. 28–32.

[25] G. A. Paleologo, L. Benini, A. Bogliolo, and G. De Micheli, "Policy optimization for dynamic power management," in *Proc. Design Automat. Conf.*, June 1998, pp. 182–187.

[26] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for reduced CPU energy," in *Proc. USENIX Symposium on Operating Systems Design and Implementation*, 1994, pp. 13–23.

[27] F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced CPU energy," in *Proc. IEEE Annual Foundations of Computer Science*, 1995, pp. 374–382.

[28] T. Ishihara and H. Yasuura, "Voltage scheduling problem for dynamically variable voltage processors," in *Proc. Int'l Symposium on Low Power Electronics and Design*, Aug. 1998, pp. 197–202.

[29] T. Okuma, H. Yasuura, and T. Ishihara, "Software energy reduction techniques for variable-voltage processors," *IEEE Design & Test of Computers*, vol. 18, no. 2, pp. 31–41, Mar. 2001.

[30] Y. Lee and C. Krishna, "Voltage-clock scaling for low energy consumption in

real-time embedded systems," in *Proc. Int'l Workshop on Real-Time Computing Systems and Applications*, 1999.

[31] Y. Shin and K. Choi, "Power conscious fixed priority scheduling for hard real-time systems," in *Proc. Design Automat. Conf.*, June 1999, pp. 134–139.

[32] Y. Shin, H. Kawaguchi, and T. Sakurai, "Cooperative voltage scaling (CVS) between OS and applications for low-power real-time systems," in *Proc. IEEE Custom Integrated Circuits Conf.*, May 2001, pp. 553–556.