

Designing Energy Aware Systems

N. Vijaykrishnan and Jie S. Hu
The Pennsylvania State University
University Park, PA 16802
email: vijay.jhu@cse.psu.edu

Abstract

The design of power-efficient systems has been a main concern for industrial designers and has been the focus of many academic and industrial labs. Power consumption has a significant impact on various aspects of the system such as power grid design, packaging and cooling, battery lifetime and system reliability. First, we will provide an overview of the metrics and tools used for power estimation. The second part of this proposal shows the need for a holistic power optimization approach that spans from circuit to software design issues. We illustrate this showing power optimizations applied to the memory system.

Index Terms: low power design, estimation tools, memory system, leakage power.

1 Introduction

The design of computing systems needs to consider various factors such as performance, cost, power dissipation and reliability. Among these power dissipation is considered as the biggest stumbling block in designing the next generation systems. Power problems are a significant issue ranging from small sensors to large compute servers. However, the underlying reasons for their importance are different.

In small embedded and mobile systems, the limited battery capacity is a main concern. The battery technology improvements have not matched to the increasing power requirements of the computing resources. Current lithium-ion batteries provide only 100W-hr per pound compared to around 10W-hr/lb capacities in the 1960's. In contrast, the power consumption numbers of the processors have increased from much less than a watt in 1970s to around 100Watts in current microprocessors. Consequently, battery technology has been a bottleneck making the battery pack a dominant part of the system weight and influencing the duration required between battery recharges.

Power dissipation has become an important issue in desktop systems and server environments for a variety of other reasons. The increasing power density due to the miniaturization of the circuits makes the task of packaging and cooling harder and costlier. Higher power densities also translate to higher on-chip temperatures and make it necessary to support costlier packaging. Power and cooling

requirements are also a major bottleneck for many data centers and is considered a significant part of the operating cost. The higher power densities also degrade system reliability. Furthermore, the increasing current draw poses difficulties in the power supply grid design.

2 Sources of Power Consumption

The three main sources of power consumption in a CMOS chip occur due to the switching activity of the signals, short-circuit current and leakage currents. Power is consumed whenever current is drawn to charge a node or wire from zero to one. This is referred to as the dynamic power consumption and is represented as CV^2f , where C is the capacitance of the node or wire being switched, V is the voltage swing associated with a change from a logical zero to a logical one and f is the operating frequency. Dynamic energy consumption has been the dominant source of power consumption and has been the focus of most power optimization efforts. The second source of power consumption is due to the short-circuit current that flows when both the pull-up and pull-down circuits are both on for a short duration when the inputs are changing. Short circuit current is not a major concern for well designed circuits. The third source of power consumption is due to leakage current that flows even when the transistors are turned off. Leakage power is consumed immaterial of whether there is switching activity or not and is becoming a major concern with the scaling down of threshold voltages and the reducing thickness of the gate oxide.

In order to reduce power consumption, tools for estimating the various sources of power consumption are essential. The estimation tools are useful in identifying the components that are problematic from a power consumption perspective and in evaluating the effectiveness of optimizations proposed to overcome these problems. The power estimation tools can be used at different stages in the system design (See Figure 1). Tools for accurate power-performance prediction are essential for designing power-aware architectures, compilers, run-time support, communication protocols, and applications. Currently, there are tools to measure the power at either a very fine-grain (circuit or gate) level or coarse-grain (procedural or program) level. With fine-grain estimation, it is difficult or impossible to measure power usage in (future) billion transistor designs or for large programs. However, this is the most accurate approach to

power estimation. On the other hand, coarse-grain measurements can only give gross estimates, but do so quite efficiently.

At the earliest stage of the design, power is estimated for a given system with little implementation detail at the architectural level. Architectural power simulators have been typically built on top of performance simulators that keep track of accesses to the different components in the architecture and obtain the power consumption by modeling the per access energy cost of a single access for each component. The models for the different components are built using the structure, bitwidth and design style. For example, the power consumed for a cache access can be modeled using information such as the size of the cache, the number of ports and the use of single-ended or double-ended sense amplifiers for reading the data. Examples of architectural level estimation tools include Simplepower [24], Softwatt, Watch [2] and Wattwatcher.

Once the RTL and the corresponding gate level implementation of the architecture are available more accurate power estimation is possible. Even at this level actual layout or circuit implementation is not available and gate and wiring capacitances are estimated using models. Gate capacitances are modeled using the sizing information while wire loads are obtained from number of pins incident on the net and based on placement information. The switching activity at the nodes is estimated through simulation. Once capacitance and switching information is available the dynamic power estimation can be performed.

Later in the design cycle, even more accurate estimates can be obtained at the circuit level as more information is available to make estimates of the capacitance more accurately. However, estimation at the higher levels is becoming more and more important as power problems identified later in the design cycle are hard to fix. Consequently, tools for estimating the power at the earliest stage of design are becoming popular. Since software is becoming an integral part of most systems, tools for estimating the power consumed by software and the influence of software optimizations on the power consumption behavior have also become important.

These tools also help to identify the specific components that pose the main concern from a power consumption perspective. In many embedded systems, the design of the memory system is a critical factor influencing the power consumption profile. The rest of this paper shows how memory power optimizations can be reduced at different levels of system design.

3 Memory Power Optimizations

A host of hardware optimizations have been proposed to reduce the energy consumption. Focusing on SRAM memories, common techniques used for optimization include partitioning the memory into smaller parts, dividing the bit lines, dividing the word lines and using reduced voltage swings [12]. Two common optimizations applied to cache memories are block buffering [17] and cache subbanking [23]. In the block buffering scheme, the previously accessed cache line is buffered for subsequent accesses [17]. If the

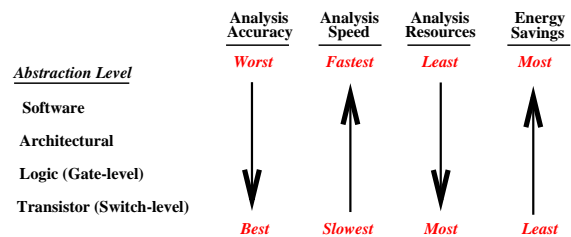


Figure 1. Comparison of Energy Optimizations at Different Levels.

data within the same cache line is accessed on the next data request, only the buffer needs to be accessed. This avoids the unnecessary and more energy consuming access to the entire cache data and tag array. Multiple block buffers can be thought of as a small sized Level 0 cache. In the cache subbanking optimization, which is also known as column multiplexing [23], the data array of the cache is divided into several subbanks and only the subbank where the desired data is located is accessed. This optimization reduces the per access energy consumption.

A common power optimization technique employed in DRAM memories include the support for multiple low power modes. Each mode is characterized by its *power consumption* and the time that it takes to transition back to the active mode (*resynchronization time*). Typically, lower the energy consumption, higher the resynchronization time [1]. These modes are characterized by varying degrees of the module components being active. The power mode transitions can be effected either by hardware or through software.

In the hardware approach, there is a Self-Monitoring and Prediction Hardware block which monitors ongoing memory transactions. It contains some prediction hardware to estimate the time until the next access to a memory bank and circuitry to ask the memory controller to initiate mode transitions. Limited amount of such self-monitored power-down is present in current memory controllers (e.g., Intel 82443BX [13]). The specific details of different prediction mechanisms that can be employed is given in [7]

In the software-directed approach, the memory controller is explicitly told to issue the control packets for a specific module's mode transitions. A set of configuration registers in the memory controller that are mapped into the address space of the CPU (similar to the registers in the memory controller in [13]) are used to set the power mode. Programming these registers using one or more CPU instructions (stores) would result in the desired power mode setting. The power modes are set based on analyzing the code and data access patterns using the compiler.

In addition, it is also possible to use code optimizations to improve the effectiveness of the power mode control. For example, all data accessed can be clustered together in a single module instead of being scattered across in different modules. This enables to put the unused modules into a lower power mode.

4 Reducing Leakage Energy

There have been several efforts spanning from the circuit level to the architectural level at reducing the cache leakage energy. Circuit mechanisms include adaptive substrate biasing, dynamic supply scaling and supply gating. Many of the circuit techniques have been exploited at the architectural level to control leakage at the cache bank and cache line granularities.

The approaches that target reducing cache leakage energy consumption can be broadly categorized into three groups: (i) those that base their leakage management decisions on some form of performance feedback (e.g., cache miss rate) [19], (ii) those that manage cache leakage in an application insensitive manner (e.g., periodically turning off cache lines) [8, 15, 16], and (iii) those that use feedback from the program behavior [15, 27, 25, 11].

The approach such as DRI I-Cache [19] in category (i) is inherently coarse-grain in managing leakage as it turns off large portions of the cache depending on a performance feedback that does not specifically capture cache line usage patterns.

Approaches in category (ii) turn off cache lines independent of the instruction access pattern. An example of such a scheme is the periodic cache line turn-off proposed in [8]. The success of this strategy depends on how well the selected period reflects the rate at which the instruction working set changes. Specifically, the optimum period may change not only across applications but also within the different phases of the application itself. A second example of a fixed scheme in category (ii) is the technique proposed in [16]. This technique adopts a bank based strategy, where when execution moves from one bank to another, the hardware turns off the former and turns on the latter. Another technique in category (ii) is the cache decay-based approach (its adaptive variant falls in category (iii)) proposed by Kaxiras et al [15]. In this technique, a small counter is attached to each cache line which tracks its access frequency. If a cache line is not accessed for a certain number of cycles, it is placed into the leakage saving mode. While this technique tries to capture the usage frequency of cache lines, it does not directly predict the cache line access pattern. Consequently, a cache line whose counter saturates is turned off even if it is going to be accessed in the next cycle. Since it is also a periodic approach, choosing a suitable decay interval is crucial if it is to be successful.

The approaches in category (iii) attempt to manage cache lines in an application-sensitive manner. The adaptive version of the cache-decay scheme [15] tailors the decay interval for the cache lines based on cache line access patterns. They start out with the smallest decay interval for each cache line to aggressively turn off cache lines and increase the decay interval when they learn that the cache lines were turned off prematurely. These schemes learn about premature turn-off by leaving the tags on at all times. The approach in [27] also uses tag information to adapt leakage management. In [25], an optimizing compiler is used to analyze the program to insert explicit cache line turn-off instructions. This scheme demands sophisticated program analysis and modification support, and needs modifications in the ISA to implement cache line turn-on/off instructions.

Further, Hu et al., in [11], proposed a hotspot based leakage management scheme to capture the dynamic phase execution information of the running program for directing leakage control and a just-in-time activation scheme to significantly reduce the performance overhead due to the leakage control.

As Java technologies are more widely adopted in battery powered devices such as cellphones, PDAs, and pagers, optimizing the power consumption in Java environment is becoming a critical issue. Java virtual machine (JVM) relies on the garbage collector (GC) for automatic memory management. In [4], Chen et al. proposed a GC-controlled leakage energy optimization technique that shuts off memory banks that do not hold live data. Their schemes reduce the leakage energy consumed by the heap memory significantly. However, conventional GC is invoked at a fixed frequency to detect and turn off the memory banks containing no live objects. High frequent GC will unnecessarily decrease the performance of the virtual machine. On the other side, GC at very low frequency will lose the opportunities for leakage optimization. The optimal GC frequency depends on the behavior of a particular application. In [5], the authors further developed an adaptive scheme that dynamically adjusts the GC frequency according to the memory allocation behavior of the applications. This adaptive scheme provides a leakage reduction approaching that delivered by the optimal GC frequency of a given application.

In attempting to reduce leakage energy, we might increase the susceptibility to soft errors when reducing supply voltages. The work [26] provides a detailed investigation on impacts of soft errors in caches applying drowsy leakage control scheme. The results indicate that the single event upset rate (SER) increases dramatically from 2.5E-05 FIT/bit to 5E05 FIT/bit when the supply voltage is reduced from the normal voltage 1.0V to a drowsy voltage 0.3V. Hence to maintain the system reliability, more sophisticated error protection schemes that themselves will consume more energy will be required. Hence, as these reliability problems aggravate, devising techniques that will balance the tradeoffs between energy optimization and reliability will become important.

5 Conclusions

Power consumption has become a major design constraint influencing the design of next generation systems. Combating the power problem requires a holistic effort spanning from circuits to software. Furthermore, there is a complex interaction and tradeoff between power, performance and reliability that need to be balanced carefully.

Acknowledgment

This work was supported in part by grants from NSF CAREER Award 0093085, and Gigascale Silicon Research Center. The author would like to acknowledge Profs. Mary Jane Irwin and Mahmut Kandemir and students in the Microsystems Design Lab who were involved in parts of these projects reported here.

References

- [1] Rambus Inc. <http://www.rambus.com/>.
- [2] D. Brooks, V. Tiwari, and M. Martonosi. Watch: A framework for architectural-level power analysis and optimizations. In *Proc the 27th International Symposium on Computer Architecture*, Vancouver, British Columbia, June 2000.
- [3] R. C. Baumann. Soft errors in advanced semiconductor devices-part I: the three radiation sources. *IEEE Transactions on Device and Materials Reliability*, Vol. 1, No. 1, pp. 17-22, March 2001.
- [4] G. Chen, R. Shetty, M. Kandemir, N. Vijaykrishnan, M. J. Irwin, and M. Wolczko. Tuning Garbage Collection in an Embedded Java Environment. In *Proc. The 8th International Symposium on High-Performance Computer Architecture (HPCA'02)*, Cambridge, MA, February 2-6, 2002.
- [5] G. Chen, M. Kandemir, N. Vijaykrishnan, M. J. Irwin and M. Wolczko. Adaptive Garbage Collection for Battery-Operated Environments. In *Proc. The 2nd USENIX Java Virtual Machine Research and Technology Symposium (JVM'02)*, August 1-2, 2002.
- [6] V. Degalahal, N. Vijaykrishnan and M. J. Irwin. Analyzing Soft Errors in Leakage Optimized SRAM Designs. To appear in *Proc. of International Conference on VLSI Design*, Jan 2003.
- [7] V. Delaluz, M. Kandemir, N. Vijaykrishnan, A. Sivasubramaniam and M. J. Irwin. DRAM Energy Management Using Software and Hardware Directed Power Mode Control, 7th International Conference on High Performance Computer Architecture, Jan 2001.
- [8] K. Flautner, N. Kim, S. Martin, D. Blaauw, T. Mudge. Drowsy Caches: Simple techniques for reducing leakage power. In *Proc. the 29th International Symposium on Computer Architecture*, Anchorage, AK, May 2002.
- [9] S. Hareland, J. Maiz, M. Alavi, K. Mistry, S. Walsta, D. Changhong. Impact of CMOS process scaling and SOI on the soft error rates of logic processes. In *Proc. Symposium on VLSI Technology Digest of Technical Papers*. pp. 73-74.
- [10] P. Hazucha, and C. Svensson, Impact of CMOS Technology Scaling on the Atmospheric Neutron Soft Error Rate. *IEEE Transactions on Nuclear Science*, Vol. 47, No. 6, December 2000.
- [11] J. S. Hu, A. Nadgir, N. Vijaykrishnan, M. J. Irwin, M. Kandemir. Exploiting Program Hotspots and Code Sequentiality for Instruction Cache Leakage Management. In *Proc. of the International Symposium on Low Power Electronics and Design (ISLPED'03)*, Seoul, Korea, August 25-27, 2003.
- [12] K. Itoh, K. Sasaki, and Y. Nakagome. Trends in low-power ram circuit technologies. *Proceedings of the IEEE*, pages 524-543, Vol. 83. No. 4, April 1995.
- [13] Intel 440BX AGPset: 82443BX Host Bridge/Controller Data Sheet, April 1998.
- [14] B. Kang, N. Vijaykrishnan, M. J. Irwin and D. Duarte. Substrate Noise Detector for Noise Tolerant Mixed-Signal IC 2003 IEEE International SOC Conference, Sept 2003.
- [15] S. Kaxiras, Z. Hu, M. Martonosi. Cache decay: exploiting generational behavior to reduce cache leakage power. In *Proc. the 28th International Symposium on Computer Architecture*, Sweden, June 2001.
- [16] N. Kim, K. Flautner, D. Blaauw, and T. Mudge. Drowsy instruction caches: Leakage power reduction using dynamic voltage scaling and cache sub-bank prediction. In *Proc. the 35th Annual International Symposium on Microarchitecture*, November 2002.
- [17] J. Kin et al. The filter cache: An energy efficient memory structure. In *Proc. International Symposium on Microarchitecture*, December 1997.
- [18] L. Li, N. Vijaykrishnan, M. Kandemir and M. J. Irwin. Adaptive Error Protection for Energy Efficiency. In *Proc. of International Conference on Computer Aided Design*, Nov 2003.
- [19] M. D. Powell, S. Yang, B. Falsafi, K. Roy, and T. N. Vijaykumar. Reducing Leakage in a High-Performance Deep-Submicron Instruction Cache. *IEEE Transactions on VLSI*, Vol. 9, No. 1, February 2001.
- [20] R. Ramnarayanan, V. Degalahal, N. Vijaykrishnan, M. J. Irwin and D. Duarte. Analysis of Soft-Error Rate for Flip-Flops and Scannable Latches. 2003 IEEE International SOC Conference, Sept 2003
- [21] N. Seifert, D. Moyer, N. Leland, and R. Hokinson. Historical trend in alpha-particle induced soft error rates of the Alpha microprocessor. In *Proc. the 39th Annual International Reliability Physics Symposium*, pp. 259-265, 2001.
- [22] P. Sivakumar, M. Kistler, S. W. Keckler, D. C. Burger, and L. Alvisi. Modeling the effect of technology trends on the soft error rate of combinational logic. *International Conference on Dependable Systems and Networks*, June, 2002.
- [23] C.-L. Su and A. M. Despain. Cache design trade-offs for power and performance optimization: A case study, In *Proc. International Symposium on Low Power Electronics and Design*, pp. 63-68, 1995.
- [24] W. Ye, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin. The design and use of SimplePower: a cycle-accurate energy estimation tool. In *Proc. the 37th Design Automation Conference*, Los Angeles, CA, June 5-9, 2000.
- [25] W. Zhang et. al.. Compiler-directed instruction cache leakage optimization. In *Proc. the 35th Annual International Symposium on Microarchitecture*, November 2002.
- [26] W. Zhang, J. S. Hu, V. Degalahal, M. Kandemir, N. Vijaykrishnan, M. J. Irwin. Reducing Instruction Cache Energy Consumption Using a Compiler-Based Strategy. Accepted to publish in *ACM Transactions on Architecture and Code Optimization*.
- [27] H. Zhou, M. C. Toburen, E. Rotenberg, and T. M. Conte. Adaptive mode control: a static power-efficient cache design. In *Proc. the 2001 International Conference on Parallel Architectures and Compilation Techniques*, September 2001.
- [28] J. Ziegler. Terrestrial cosmic ray intensities. *IBM Journal of Research and Development*, Vol 42, No 1, January 1998.