Hindawi Publishing Corporation
EURASIP Journal on Embedded Systems
Volume 2009, Article ID 725438, 7 pages
doi:10.1155/2009/725438

Research Article

Data Cache-Energy and Throughput Models: Design Exploration for Embedded Processors

Muhammad Yasir Qadri and Klaus D. McDonald-Maier

School of Computer Science and Electronic Engineering, University of Essex, Colchester CO4 3SQ, UK

Correspondence should be addressed to Muhammad Yasir Qadri, yasirqadri@acm.org

Received 25 March 2009; Revised 19 June 2009; Accepted 15 October 2009

Recommended by Bertrand Granado

Most modern 16-bit and 32-bit embedded processors contain cache memories to further increase instruction throughput of the device. Embedded processors that contain cache memories open an opportunity for the low-power research community to model the impact of cache energy consumption and throughput gains. For optimal cache memory configuration mathematical models have been proposed in the past. Most of these models are complex enough to be adapted for modern applications like run-time cache reconfiguration. This paper improves and validates previously proposed energy and throughput models for a data cache, which could be used for overhead analysis for various cache types with relatively small amount of inputs. These models analyze the energy and throughput of a data cache on an application basis, thus providing the hardware and software designer with the feedback vital to tune the cache or application for a given energy budget. The models are suitable for use at design time in the cache optimization process for embedded processors considering time and energy overhead or could be employed at runtime for reconfigurable architectures.

1. Introduction

The popularity of embedded processors could be judged by the fact that more than 10 billion embedded processors were shipped in 2008, and this is expected to reach 10.76 billion units in 2009 [1]. In the embedded market the number of 32-bit processors shipped has surpassed significantly that of 8-bit processors [2]. Modern 16-bit and 32-bit embedded processors increasingly contain cache memories to further instruction throughput and performance of the device. The recent drive towards low-power processing has challenged the designers and researchers to optimize every component of the processor. However optimization for energy usually comes with some sacrifice on throughput, and which may result in overall minor gain.

Figure 1 shows the operation of a typical battery powered embedded system. Normally, in such devices, the processor is placed in active mode only when required; otherwise it remains in a sleep mode. An overall power saving (increased throughput to energy ratio) could be achieved by increasing the throughput (i.e., lowering the duty cycle), decreasing the peak energy consumption, or by lowering the sleep mode energy consumption. This phenomenon clearly shows the interdependence of energy and throughput for overall power saving. Keeping this in mind, a simplified approach is proposed that is based on energy and throughput models to analyze the impact of a cache structure in an embedded processor per application basis which exemplifies the use of the models for design space exploration and software optimization.

The remainder of this paper is divided into five sections. In the following two sections related work is discussed and the energy and throughput models are introduced. In the fourth section experimental environment and results are discussed, the fifth section describes an example application for the mathematical models, and the final section forms the conclusion.

2. Related Work

The cache energy consumption and throughput models have been the focus of research for some time. Shiue and

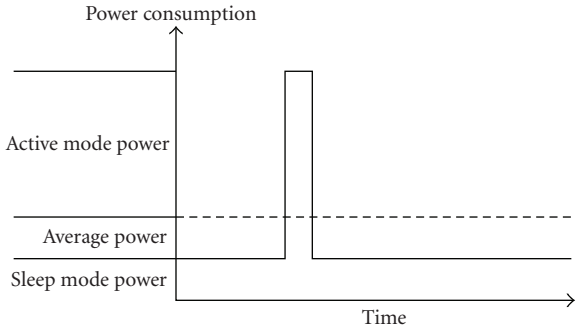

Figure 1: Power consumption of a typical battery powered processor (adapted from [3]).

Chakrabarti [4] present an algorithm to find optimum cache configuration based on cache size, the number of processor cycles, and the energy consumption. Their work is an extension of the work of Panda et al. [5, 6] on data cache sizing and memory exploration. The energy model by Shiue and Chakrabarti , though highly accurate, requires a wide range of inputs like number of bit switches on address bus per instruction, number of bit switches on data bus per instruction, number of memory cells in a word line and in a bit line, and so forth. which may not be known to the model user in advance. Another example of a detailed cache energy model was presented by Kamble and Ghose [7]. These analytical models for conventional caches were found to be accurate to within 2% error. However, they over-predict the power dissipations of low-power caches by as much as 30%. The low-power cache designs used by Kamble and Ghose incorporated block buffering, data RAM subbanking, and bus invert coding for evaluating the models. The relative error in the models increased greatly when the sub-banking and block buffering were simultaneously applied. The major difference between the approach used by Kamble and Ghose [7] and the one discussed in this paper is that the former one incorporated bit level models to evaluate the energy consumption, which are in some cases inaccurate as the error in output address power was found (by the Kamble and Ghose) in the order of 200%, due to the fact that data and instruction access addresses exhibit strong locality. The approach presented here uses a standard cache modelling tool, CACTI [8], for measuring bit level power consumption in cache structures and provides a holistic approach for energy and throughput for an application basis. In fact the accuracy of these models is independent of any particular cache configuration as standard cache energy and timing tools are used to provide cache specific data. This approach is discussed in detail in Section 4.

Simunic et al. [9] presented mathematical models for energy estimation in embedded systems. The per cycle energy model presented in their work comprises energy components of processor, memory, interconnects and pins, DC-to-DC converters, and level two (L2) cache. The model was validated using an ARM simulator [10] and the SmartBadge [11] prototype based on ARM-1100 processor. This was found to be within 5% of the hardware measurements for the same operating frequency. The models presented in their work holistically analyze the embedded system power and do not estimate energy consumption for individual components of a processor that is, level one (L1) cache, on-chip memory, pipeline, and so forth. In work by Li and Henkel [12] a full system detailed energy model comprising cache, main memory, and software energy components was presented. Their work includes description of a framework to assess and optimize energy dissipation of embedded systems. Tiwari et al. [13] presented an instruction level energy model estimating energy consumed in individual pipeline stages. The same methodology was applied in [14] by the authors to observe the effects of cache enabling and disabling.

Wada et al. [15] presented comprehensive circuit level access time model for on-chip cache memory. On comparing with SPICE results the model gives 20% error for an 8 nanoseconds access time cache memory. Taha and Wills [16] presented an instruction throughput model for Superscalar processors. The main parameters of the model are superscalar width of the processor, pipeline depth, instruction fetch method, branch predictor, cache size and latency, and so forth. The model results in errors up to 5.5% as compared to the SimpleScalar out-of-order simulator [17]. CACTI (cache access and cycle time model) [8] is an open-source modelling tool based on such detailed models to provide thorough, near accurate memory access time and energy estimates. However it is not a trace driven simulator, and so energy consumption resulting in number of hits or misses is not accounted for a particular application.

Apart from the mathematical models, substantial work has been done for cache miss rate prediction and minimization. Ding and Zhong in [18] have presented a framework for data locality prediction, which can be used to profile a code to reduce miss rate. The framework is based on approximate analysis of reuse distance, pattern recognition, and distance-based sampling. Their results show an average of 94% accuracy when tested on a number of integer and floating point programs from SPEC and other benchmark suites. Extending their work Zhong et al. in [19] introduce an interactive visualization tool that uses a three-dimensional plot to show miss rate changes across program data sizes and cache sizes. Another very useful tool named RDVIS as a further extension of the work previously stated was presented by Beyls et al. in [20, 21]. Based on cluster analysis of basic block vectors, the tool gives hints on particular code segments for further optimization. This in effect provides valuable feedback to the programmer to improve temporal locality of the data to increase hit rate for a cache configuration.

The following section presents the proposed cache energy and throughput models, which can be used to identify an early cache overhead estimate based on a limited set of input data. These models are an extension of the models previously proposed by Qadri and Maier in [22, 23].

3. The D-Cache Energy and Throughput Models

The cache energy and throughput models given below strive to provide a complete application-based analysis. As a result they could facilitate the tuning of a cache and an application

Table 1: Simulation platform parameters.

Parameter	Value
Processor	PowerPC440GP
Execution mode	Turbo
Clock frequency (Hz)	1.00E+08
Time	1.00E−08
CPI	1
Technology	0.18 um
Vdc (V)	1.8
Logic Supply (V)	3.3
DDR SDRAM (V)	2.5
VDD (1.8 V) active operating current IDD (A)	9.15E−01
OVDD (3.3 V) active operating current IODD (A)	1.25E−01
Energy per Cycle (J)	1.65E−08
Idle mode Energy (J)	4.12E−09

Table 2: Cache simulator data.

CACTI Data	
Cache Size	32 Kbytes
Block Size	256 bytes
R/W Ports	0
Read ports	1
Write ports	1
Access Time (s)	1.44E−09
Cycle Time (s)	7.38E−10
Read Energy (J)	2.24E−10
Write Energy (J)	3.89E−11
Leakage Read Power (W)	2.96E−04
Leakage Write Power (W)	2.82E−04

according to a given power budget. The models presented in this paper are an improved extension of energy and throughput models for a data cache, previously presented by the authors in [22, 23]. The major improvements in the model are as follows: (1) The leakage energy (E_{leak}) is now indicated for the entire processor rather than simply the cache on its own. The energy model covers the per cycle energy consumption of the processor. The leakage energy statistics of the processor in the data sheet covers the cache and all peripherals of the chip. (2) The miss rate in E_{read} and E_{write} has been changed to read_{mr} (read miss rate) and write_{mr} (write miss rate) as compared to total miss rate (r_{miss}) that was employed previously. This was done as the read energy and write energy components correspond to the respective miss rate contribution of the cache. (3) In the throughput model stated in [23] a term t_{mem} (time saved from memory operations) was subtracted from the total throughput of the system, which was later found to be inaccurate. The overall time taken to execute an instruction denoted as T_{total} is the measure of the total time taken by the processor for running an application using cache. The time saved from memory only operations is already accounted in T_{total}. However a new term t_{ins} was introduced to incorporate the time taken for the execution of cache access instructions.

3.1. Energy Model. If E_{read} and E_{write} are the energy consumed by cache read and write accesses, E_{leak} the leakage energy of the processor, $E_{c\to m}$ the energy consumed by cache to memory accesses, E_{mp} the energy miss penalty, and E_{misc} is the Energy consumed by the instructions which do not require data memory access, then the total energy consumption of the code E_{total} in Joules (J) could be defined as

$$E_{\text{total}} = E_{\text{read}} + E_{\text{write}} + E_{c\to m} + E_{\text{mp}} + E_{\text{leak}} + E_{\text{misc}}. \quad (1)$$

Further defining the individual components,

$$\begin{aligned} E_{\text{read}} &= n_{\text{read}} \cdot E_{\text{dyn.read}} \cdot \left[1 + \frac{\text{read}_{\text{mr}}}{100}\right], \\ E_{\text{write}} &= n_{\text{write}} \cdot E_{\text{dyn.write}} \cdot \left[1 + \frac{\text{write}_{\text{mr}}}{100}\right], \\ E_{c\to m} &= E_m \cdot (n_{\text{read}} + n_{\text{write}}) \cdot \left[1 + \frac{\text{total}_{\text{mr}}}{100}\right], \\ E_{\text{mp}} &= E_{\text{idle}} \cdot (n_{\text{read}} + n_{\text{write}}) \cdot \left[P_{\text{miss}} \cdot \frac{\text{total}_{\text{mr}}}{100}\right], \end{aligned} \quad (2)$$

where n_{read} is the number of read accesses, n_{write} the number of write accesses, $E_{\text{dyn.read}}$ the total dynamic read energy for all banks, $E_{\text{dyn.write}}$ the total dynamic write energy for all banks, E_m the energy consumed per memory access, E_{idle} the per cycle idle mode energy consumption of the processor, read_{mr}, write_{mr}, and total_{mr} are the read, write, and total miss ratio (in percentage), and P_{miss} is the miss penalty (in number of stall cycles).

The idle mode leakage energy of the processor E_{leak} could be calculated as

$$E_{\text{leak}} = P_{\text{leak}} \cdot t_{\text{idle}}, \quad (3)$$

where t_{idle} (s) is the total time in seconds for which processor was idle.

3.2. Throughput Model. Due to the concurrent nature of cache to memory access time and cache access time, their overlapping can be assumed. If t_{cache} is the time taken for cache operations, t_{ins} the time taken in execution of cache access instructions (s), t_{mp} the time miss penalty, and t_{misc} is the time taken while executing other instructions which do not require data memory access, then the total time taken by an application with a data cache could be estimated as

$$T_{\text{total}} = t_{\text{cache}} + t_{\text{ins}} + t_{\text{mp}} + t_{\text{misc}}. \quad (4)$$

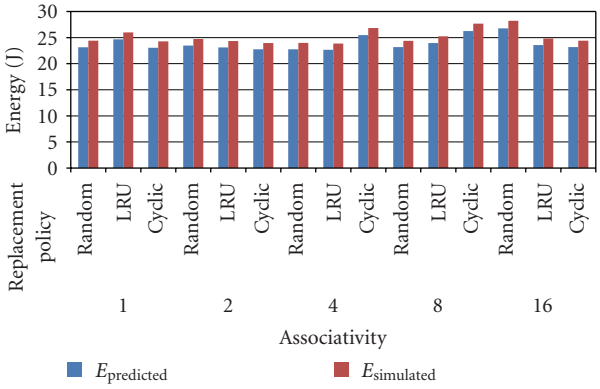

Figure 2: Energy consumption for write-through cache.

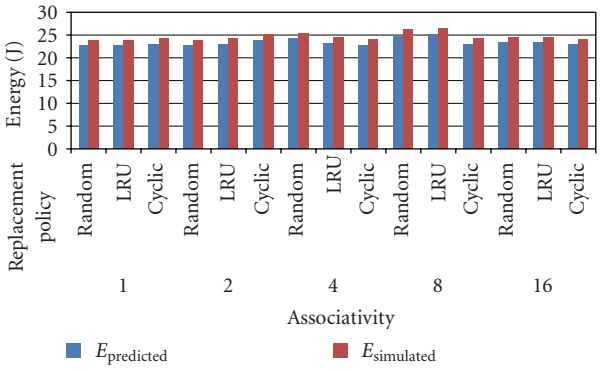

Figure 3: Energy consumption for write-back cache.

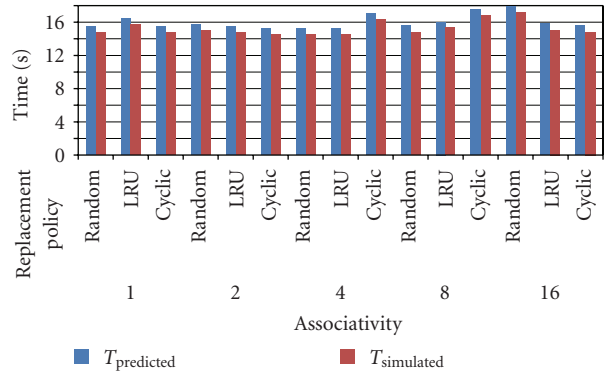

Figure 4: Throughput for write-through cache.

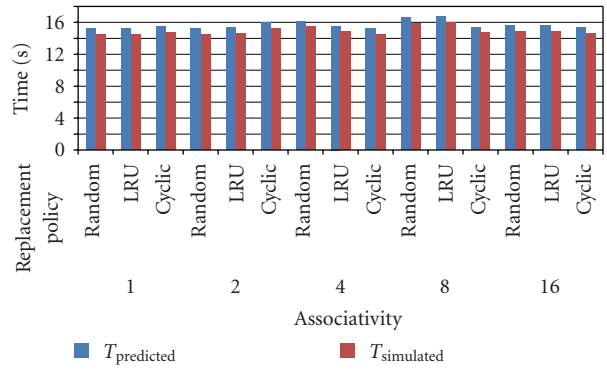

Figure 5: Throughput for write-back cache.

Furthermore,

$$t_{\text{cache}} = t_c \cdot (n_{\text{read}} + n_{\text{write}}) \cdot \left[1 + \frac{\text{total}_{\text{mr}}}{100}\right],$$
$$t_{\text{ins}} = \left(t_{\text{cycle}} - t_c\right) \cdot (n_{\text{read}} + n_{\text{write}}), \quad (5)$$
$$t_{\text{mp}} = t_{\text{cycle}} \cdot (n_{\text{read}} + n_{\text{write}}) \cdot \left[P_{\text{miss}} \cdot \frac{\text{total}_{\text{mr}}}{100}\right],$$

where t_c is the time taken per cache access and t_{cycle} is the processor cycle time in seconds (s).

4. The Experimental Environment and Results

To analyze and validate the aforementioned models, SIMICS [25], a full system simulator was used. An IBM/AMCC PPC440GP [26] evaluation board model was used as the target platform and Montavista Linux 2.1 kernel was used as target application to evaluate the models. A generic 32-bit data cache was included in the processor model, and results were analyzed by varying associativity, write policy, and replacement policy. The cache read and write miss penalty was fixed at 5 cycles. The processor input parameters are defined in Table 1.

As SIMICS could only provide timing information of the model, processor power consumption data like idle mode energy (E_{idle}) and leakage power (P_{leak}) was taken from PPC440GP datasheet [26], and cache energy and timing parameters such as dynamic read and write energy per cache access ($E_{\text{dyn.read}}$, $E_{\text{dyn.write}}$) and cache access time (t_c) were taken from CACTI [8] cache simulator (see Table 2). For other parameters such as number of memory reads/writes and read/write/total miss rate (n_{read}, n_{write}, read_{mr}, write_{mr}, total_{mr}), SIMICS cache profilers statistics were used. The cache to memory access energy (E_m) was assumed to be half that of per cycle energy consumption of the processor. The

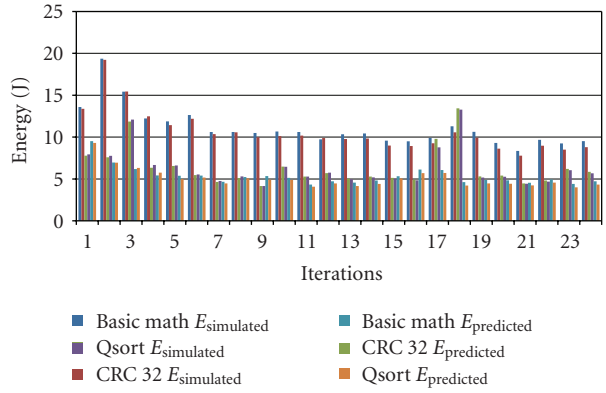

Figure 6: Simulated and Predicted Energy Consumption, varying Cache Size and Block Size (see Table 3).

Table 3: Iteration definition for varying Block Size and Cache Size.

Block Size/Cache Size	1 KBytes	2 KBytes	4 KBytes	8 KBytes	16 Kbytes	32 KBytes
64 Bytes	1	2	3	4	5	6
128 Bytes	7	8	9	10	11	12
256 Bytes	13	14	15	16	17	18
512 Bytes	19	20	21	22	23	24

Table 4: Cache Simulator Data for various Iterations.

CACTI Data								
Iteration	Associativity	Block Size (bytes)	Number of Lines	Cache Size (bytes)	Access Time (ns)	Cycle Time (ns)	Read Energy (nJ)	Write Energy (nJ)
1	0	64	16	1024	2.15	0.7782	0.160524	0.0918
2	0	128	8	1024	2.47	1.182	0.126	0.0695
3	0	256	4	1024	3.639	2.394	0.135	0.063
4	0	512	2	1024	8.185	6.955	0.171	0.068
5	0	64	32	2048	2.368	0.818	0.265	0.142
6	0	128	16	2048	2.58	1.206	0.186	0.095
7	0	256	8	2048	3.706	2.42	0.183	0.0755
8	0	512	4	2048	8.23	6.975	0.213	0.075
9	0	64	64	4096	2.2055	0.778	0.593	0.404
10	0	128	32	4096	2.802	1.25	0.307	0.145
11	0	256	16	4096	3.84	2.46	0.28	0.1
12	0	512	8	4096	8.316	7.016	0.298	0.087
13	0	64	128	8192	2.422	0.8175	0.96	0.5988
14	0	128	64	8192	2.633	1.206	0.619	0.407
15	0	256	32	8192	4.085	2.529	0.474	0.151
16	0	512	16	8192	8.48	7.09176	0.468	0.1125
17	0	64	256	16384	2.85	0.88	1.7	0.988
18	0	128	128	16384	2.8559	1.251	1.0049	0.602
19	0	256	64	16384	3.888	2.4557	0.834	0.413
20	0	512	32	16384	8.533	7.092	0.77	0.254
21	0	64	512	32768	3.783	0.985	3.177	1.7661
22	0	128	256	32768	3.3	1.33	1.776	0.991
23	0	256	128	32768	4.14	2.53	1.413	0.608
24	0	512	64	32768	8.534	7.092	1.263	0.4247

simulated energy consumption was obtained by multiplying per cycle energy consumption as per datasheet specification, by the number of cycles executed in the target application.

The results for energy and timing models are presented in Figures 2, 3, 4, and 5. From the graphs, it could be inferred that the average error of the energy model for the given parameters is approximately 5% and that of timing model is approximately 4.8%. This is also reinforced by the specific results for the benchmark applications; that is, BasicMath, QuickSort, and CRC 32 from the MiBench benchmark suite [27], while varying cache size and block size using a direct-mapped cache, are shown in Figures 6 and 7. The definition of each iteration for various cache and block size is given in Table 3, and the cache simulator data are given in Table 4.

5. Design Space Exploration

The validation of the models opens an opportunity to employ these in a variety of applications. One such application could be a design exploration to find optimal cache

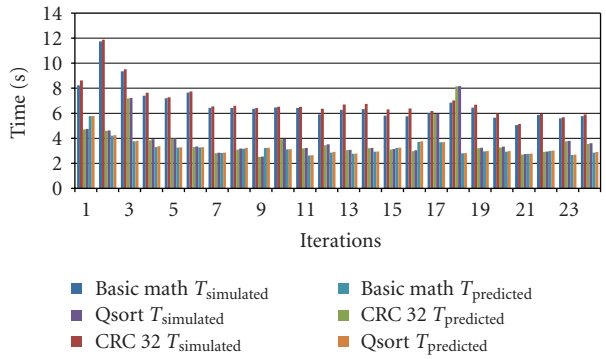

Figure 7: Simulated and Predicted Throughput, varying Cache Size and Block Size (see Table 3).

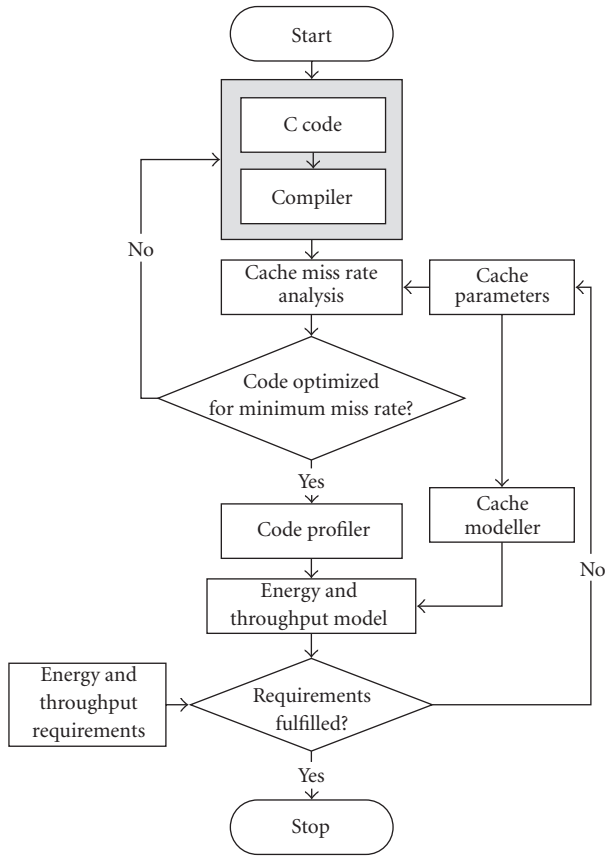

Figure 8: Proposed design cycle for optimization of cache and application code.

configuration for a set amount of energy budget or timing requirement. A typical approach for design exploration in order to identify the optimal cache configuration and code profile is shown in Figure 8. At first the miss rate prediction is carried out on the compiled code and preliminary cache parameters. Then several iterations may be performed to fine tune the software to reduce miss rates. Subsequently, the tuned software goes through the profiling step. The information from the cache modeller and the code profiler is then fed to the energy and throughput models. If the given energy budget along with the throughput requirements is not satisfied, then the cache parameters are to be changed and the same procedure is repeated. This strategy can be adopted at design time to optimize the cache configuration and decrease the miss rate of a particular application code.

6. Conclusion

In this paper straightforward mathematical models were presented with a typical accuracy of 5% when compared to SIMICS timing results and per cycle energy consumption of the PPC440GP processor. Therefore, the model-based approach presented here is a valid tool to predict the processors performance with sufficient accuracy, which would clearly facilitate executing these models in a system in order to adapt its own configuration during the actual operation of the processor. Furthermore, an example application for design exploration was discussed that could facilitate the identification of an optimal cache configuration and code profile for a target application. In future work the presented models are to be analyzed for multicore processors and to be further extended to incorporate multilevel cache systems.

Acknowledgment

The authors like to thank the anonymous reviewers for their very insightful feedback on earlier versions of this manuscript.

References

[1] "Embedded processors top 10 billion units in 2008," VDC Research, 2009.

[2] "MIPS charges into 32bit MCU fray," EETimes Asia, 2007.

[3] A. M. Holberg and A. Saetre, "Innovative techniques for extremely low power consumption with 8-bit microcontrollers," White Paper 7903A-AVR-2006/02, Atmel Corporation, San Jose, Calif, USA, 2006.

[4] W.-T. Shiue and C. Chakrabarti, "Memory exploration for low power, embedded systems," in *Proceedings of the 36th Annual ACM/IEEE Conference on Design Automation*, pp. 140–145, New Orleans, La, USA, 1999.

[5] P. R. Panda, N. D. Dutt, and A. Nicolau, "Architectural exploration and optimization of local memory in embedded systems," in *Proceedings of the 10th International Symposium on System Synthesis*, pp. 90–97, Antwerp, Belgium, 1997.

[6] P. R. Panda, N. D. Dutt, and A. Nicolau, "Data cache sizing for embedded processor applications," in *Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 925–926, Le Palais des Congrés de Paris, France, 1998.

[7] M. B. Kamble and K. Ghose, "Analytical energy dissipation models for low power caches," in *Proceedings of the International Symposium on Low Power Electronics and Design*, pp. 143–148, Monterey, Calif, USA, August 1997.

[8] D. Tarjan, S. Thoziyoor, and P. N. Jouppi, "CACTI 4.0," Tech. Rep., HP Laboratories, Palo Alto, Calif, USA, 2006.

[9] T. Simunic, L. Benini, and G. De Micheli, "Cycle-accurate simulation of energy consumption in embedded systems," in *Proceedings of the 36th Annual ACM/IEEE Design Automation Conference*, pp. 867–872, New Orleans, La, USA, 1999.

[10] *ARM Software Development Toolkit Version 2.11*, Advanced RISC Machines ltd (ARM), 1996.

[11] G. Q. Maguire, M. T. Smith, and H. W. P. Beadle, "SmartBadges: a wearable computer and communication system," in *Proceedings of the 6th International Workshop on Hardware/Software Codesign (CODES/CASHE '98)*, Seattle, Wash, USA, March 1998.

[12] Y. Li and J. Henkel, "A framework for estimation and minimizing energy dissipation of embedded HW/SW systems," in *Proceedings of the 35th Annual Conference on Design Automation*, pp. 188–193, San Francisco, Calif, USA, 1998.

[13] V. Tiwari, S. Malik, and A. Wolfe, "Power analysis of embedded software: a first step towards software power minimization," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 2, no. 4, pp. 437–445, 1994.

[14] V. Tiwari and M. T.-C. Lee, "Power analysis of a 32-bit embedded microcontroller," in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC '95)*, pp. 141–148, Chiba, Japan, August-September 1995.

[15] T. Wada, S. Rajan, and S. A. Przybylski, "An analytical access time model for on-chip cache memories," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 8, pp. 1147–1156, 1992.

[16] T. M. Taha and D. S. Wills, "An instruction throughput model of superscalar processors," *IEEE Transactions on Computers*, vol. 57, no. 3, pp. 389–403, 2008.

[17] T. Austin, E. Larson, and D. Ernest, "SimpleScalar: an infrastructure for computer system modeling," *Computer*, vol. 35, no. 2, pp. 59–67, 2002.

[18] C. Ding and Y. Zhong, "Predicting whole-program locality through reuse distance analysis," *ACM SIGPLAN Notices*, vol. 38, no. 5, pp. 245–257, 2003.

[19] Y. Zhong, S. G. Dropsho, X. Shen, A. Studer, and C. Ding, "Miss rate prediction across program inputs and cache configurations," *IEEE Transactions on Computers*, vol. 56, no. 3, pp. 328–343, 2007.

[20] K. Beyls and E. H. D'Hollander, "Platform-independent cache optimization by pinpointing low-locality reuse," in *Proceedings of the 4th International Conference on Computational Science (ICCS '04)*, vol. 3038 of *Lecture Notes in Computer Science*, pp. 448–455, Springer, May 2004.

[21] K. Beyls, E. H. D'Hollander, and F. Vandeputte, "RDVIS: a tool that visualizes the causes of low locality and hints program optimizations," in *Proceedings of the 5th International Conference on Computational Science (ICCS '05)*, vol. 3515 of *Lecture Notes in Computer Science*, pp. 166–173, Springer, Atlanta, Ga, USA, May 2005.

[22] M. Y. Qadri and K. D. M. Maier, "Towards increased power efficiency in low end embedded processors: can cache help?" in *Proceedings of the 4th UK Embedded Forum*, Southampton, UK, 2008.

[23] M. Y. Qadri and K. D. M. Maier, "Data cache-energy and throughput models: a design exploration for overhead analysis," in *Proceedings of the Conference on Design and Architectures for Signal and Image Processing (DASIP '08)*, Brussels, Belgium, 2008.

[24] M. Y. Qadri, H. S. Gujarathi, and K. D. M. Maier, "Low power processor architectures and contemporary techniques for power optimization—a review," *Journal of Computers*, vol. 4, no. 10, pp. 927–942, 2009.

[25] P. S. Magnusson, M. Christensson, J. Eskilson, et al., "Simics: a full system simulation platform," *Computer*, vol. 35, no. 2, pp. 50–58, 2002.

[26] "PowerPC440GP datasheet," AMCC 2009.

[27] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "MiBench: a free, commercially representative embedded benchmark suite," in *Proceedings of the IEEE International Workshop on Workload Characterization (WWC '01)*, pp. 3–14, IEEE Computer Society, Austin, Tex, USA, December 2001.