An Energy Characterization Framework for Software-Based Embedded Systems

Lee, Donghoon
Faculty of ISEE, Kyushu Univ.

Ishihara, Tohru
System LSI Research Center, Kyushu Univ.

Muroyama, Masanori
System LSI Research Center, Kyushu Univ.

Yasuura, Hiroto
System LSI Research Center, Kyushu Univ.

他

出版情報：ESTIMedia2006. 1, pp.59-64, 2006-10-26. ESTIMedia
バージョン：
権利関係：

An Energy Characterization Framework for Software-Based Embedded Systems

Donghoon Lee, Faculty of ISEE, Kyushu Univ., *donghoon@c.csce.kyushu-u.ac.jp*
Tohru Ishihara, Masanori Muroyama, Hiroto Yasuura, System LSI Research Center, Kyushu Univ., *{ishihara, muroyama, yasuura}@slrc.kyushu-u.ac.jp*
Farzan Fallah, Fujitsu Labs. of America, *farzan@us.fujitsu.com*

***Abstract*—** *This paper proposes an energy characterization framework which helps designers in developing a fast and accurate energy model for a target processor-based system. We use a linear model for energy estimation and we find the coefficients of the model using Linear Programming (LP). We use our approach for estimating the energy consumption of two commercial microprocessors with their on-chip caches and an off-chip SDRAM. Experimental results demonstrate that the error of our technique is on an average 3% and worst case 16% compared to the gate-level estimation results. Once the model has been developed, the energy consumption of an application program can be estimated with the speed of 300,000 instructions per second.*

I. INTRODUCTION

The increasing demand for portable devices such as cellular phones, PDAs and MP3 players makes low power consumption one of the most important design criteria. In many embedded applications, systems or sub-systems are implemented in software running on a dedicated processor. Therefore, the power consumption depends on the software being executed. As more and more functions are implemented in software, it is becoming more important to optimize software for reducing the energy consumption. The optimization can be done by modifying the algorithm implemented in the software or by using compilation techniques for optimizing the object code.

Although it is possible to estimate the energy consumption based on the number of instructions executed, the supply voltage, and the average energy consumption of the processor per instruction, the accuracy may not be very good. Better estimation can be achieved if parameters like the number of cache misses and the number of branch misses are used in the energy model.

Although in many cases it is sufficient to estimate the total or average energy consumption, in some cases it is necessary to know the amount of energy each module in the design consumes for each program state. A circuit level or a gate-level simulator may be used for this purpose. However, it is too time-consuming. Although RT-level simulation is faster, it is not practical to use it to estimate the energy consumption of a large application program.

In this paper, we propose a characterization technique which assists designers in finding a good energy model for a processor. We use a linear model to estimate the energy consumption of a processor. Once the energy model is developed, the energy consumption of software running on the processor is estimated based on the instruction trace obtained using a cycle-*inaccurate* instruction-set simulator (ISS). Using our method, we can estimate the average energy consumption quickly and accurately even for short instruction traces. Note that reducing the worst case estimation error for short instruction traces is not trivial.

The rest of the paper is organized as follows. In Section 2, we present related works. Our characterization framework for generating training benches and finding the coefficients of the linear equation are presented in Section 3. Section 4 presents experiments and results. The paper concludes in Section 5.

II. RELATED WORK

The most accurate and fastest approach to find the energy consumption of software running on a processor is to measure the power consumption of the actual chip. Tools like PowerScope [1] and Itsy [2] use computer-controlled multi-meters or A/D converters to measure energy consumption. The major drawback of using PowerScope is that it cannot measure the energy consumption of individual subsystems (e.g., a memory system) separately. Although Itsy overcomes this issue, it cannot measure the energy consumed in a short period of time because the energy consumption is averaged out over the entire execution time.

In recent years, many instruction-level energy modeling and analysis techniques have been proposed. The idea of instruction-level energy modeling by measuring the power consumption of each instruction while executed in a loop was introduced in [3-4]. The accuracy of these methods were improved by accounting for data dependencies, the effects of instruction and data addresses, register file addresses, and operand values [5]. The main drawback of these techniques is that they rely on exhaustive simulation to find the energy consumption of each instruction and the inter-instruction effects on the power consumption. The efficiency of the characterization can be improved by performing measurements on a limited subset of instructions and instruction sequences [6-7].

In [8-9], the same average energy is assumed for all instructions and the average power consumption while running an application program is calculated using the operating voltage of the processor and the clock frequency. In [10], the authors measure average energy consumed in each pipeline stage of a VLIW processor using a cycle-accurate simulator (e.g., Trimaran [11]) to improve the accuracy. The techniques estimate the average energy consumption over the entire program execution, while many

software-level energy optimization techniques need cycle-accurate energy estimation. The technique described in [12] estimates the power consumption of the target processor cycle-by-cycle. However, this requires calculating the power consumption of every gate for each instruction which is very time consuming.

Most of existing energy estimation techniques including the techniques presented in [3-10][12-20] assume a linear approximation model for estimating the energy consumption of software running on a processor. However, none describes how to decide which parameters to use and how to calculate the coefficients to accurately estimate the energy consumption of a processor. In [20], Tan et al. modeled the software energy consumption using a linear equation and discussed the parameters required to accurately estimate the energy consumption. However, they did not provide any method to find parameters, corresponding coefficients and test benches required for the accurate energy modeling.

We propose an energy characterization framework which assists designers to find an accurate linear model for estimating the energy consumption of a processor-based embedded system. In our framework, designers can find a training bench suitable for the energy characterization of the target embedded system and can optimize the corresponding coefficients using Linear Programming. We use parameters which can be extracted through GNU C debugger which is provided for almost of all types of commercial embedded processors. Therefore, our approach does not need a cycle-accurate ISS which is not provided for many types of embedded processors and is usually more expensive than a cycle-*inaccurate* one like a GNU debugger.

III. CHARACTERIZATION FRAMEWORK

A. General Approach

The energy consumption of a processor can be estimated using the following linear formula,

$$E_{estimate} = \sum_{i=0}^{N} c_i \cdot P_i \qquad (1)$$

where P_i's, c_i's and N are the parameters of the model, the corresponding coefficients and the number of parameters, respectively. The first step for the modeling is to find P_i's required for estimating the energy consumption of the target processor system. The P_i's should be parameters whose values can be easily obtained using a fast simulator like an ISS. For example, P_i's can be the number of load and store instructions executed, the number of cache misses, etc. Once the required set of parameters is obtained, the next step is to find a training bench for the energy characterization. This is the most important step. We discuss this in Section III.C. The final step is to find the coefficients, c_i's corresponding to the P_i's. In Section III.D, we describe an optimization problem which gives the optimal values of coefficients to minimize the energy estimation error for a given set of parameters. The energy consumption $E_{estimate}$ is then calculated using Equation (1).

B. Energy Charactarization Flow

Figure 1 shows an overview of our energy characterization flow. To obtain the reference energy values, we simulate the processor system at gate-level for a fixed number of instructions. We refer to this fixed number of instructions of test sequence as the instruction frame.

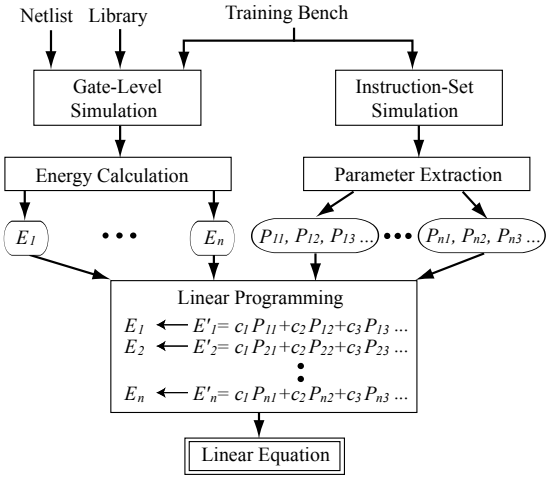

Figure 1. Overview of Energy Characterization

The width is the same for all instruction frames as shown in Figure 2. Since we perform gate-level simulation and calculate the energy consumption values for all instruction frames, this step is time-consuming. However, it needs to be done only once for the characterization.

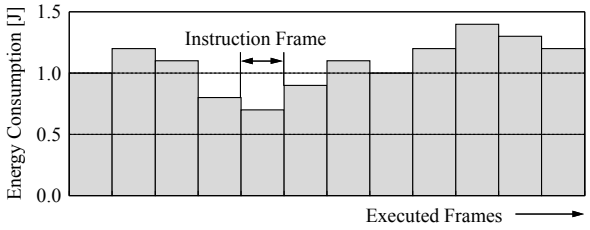

Figure 2. An Example of Instruction Frame

We, next, obtain an instruction trace for each application program using an instruction-set simulator. The traces are divided into small segments corresponding to instruction frames. P_i's should be parameters that can be easily extracted from instruction traces. For example, P_i's can be the number of multiply instructions executed within an instruction frame, the number of cache misses within an instruction frame, etc. For a set of P_i's, we find coefficients which minimize $\Sigma|E_{estimate}(i) - E_{gate\text{-}level}(i)|$, where $E_{gate\text{-}level}(i)$ and $E_{estimate}(i)$ are the energy consumption values obtained by gate-level simulation and Equation (1) for the *i-th* instruction frame, respectively.

C. Motivational Example and Our Approach

As we mentioned in Section III.A, a selection of training bench is the most important task. Figure 3 shows a motivational example. Both left and right of the Figure 3 show energy estimation results for JPEG encoder and MPEG2 encoder run on a target processor system.

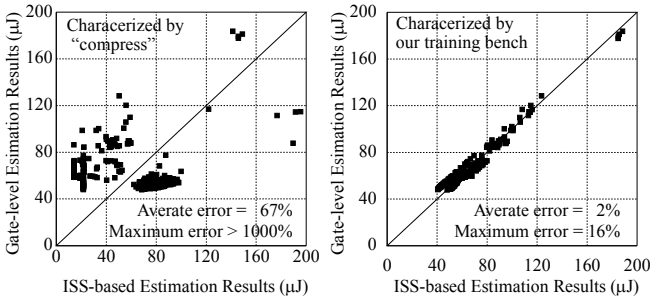

Figure 3. An Impact of Training Bench

For the results of the left figure, we used "compress", a file compression program, as a training bench and executed 500,000 instructions of gate-level simulation for the energy characterization. Through the characterization, we obtained a linear equation for estimating the energy consumption of a target processor system. Then we compared the estimated energy consumption values with the gate-level energy consumption results. As one can see, the energy estimation error is huge. The error is on an average 67% and more than 1000% for the worst case. There are two major reasons of the huge estimation error as follows.

1. Standard deviations of some parameter values are too small. For example, the numbers of cache misses are constant for all instruction frames.
2. Some parameters are strongly correlated to each other. For example, the numbers of cache misses and the numbers of branch misses have similar trends in an entire training bench.

If we carefully generate the training bench, the accuracy of the energy estimation can be improved drastically. The right of the Figure 3 shows estimation results of our approach. Only the difference between left and right results in Figure 3 is the training bench. We generate the training bench considering the standard deviations of every parameter values and correlation factors between any two parameters as criteria for generating the training bench. As a result, the estimation error of our approach is on an average 2% and 16% even for the worst case.

Figure 4 shows the flow of our approach for generating the training bench. We start from a template of training bench which consists of subroutines which execute power-hungry instructions like a multiply instruction repeatedly and produce many cache misses, many read-after-write hazards, and other pipeline stalls. We calculate parameter values using ISS. This process takes a few seconds. Then we evaluate the standard deviations of every parameter values and correlation factors of any two parameters. If standard deviations of some parameter values are lower than a specific value or the correlation factors of some parameters are higher than a specific value, we modify the initial training bench so that the standard deviations and the correlation factors are improved. This process is repeated until those two criteria are satisfied.

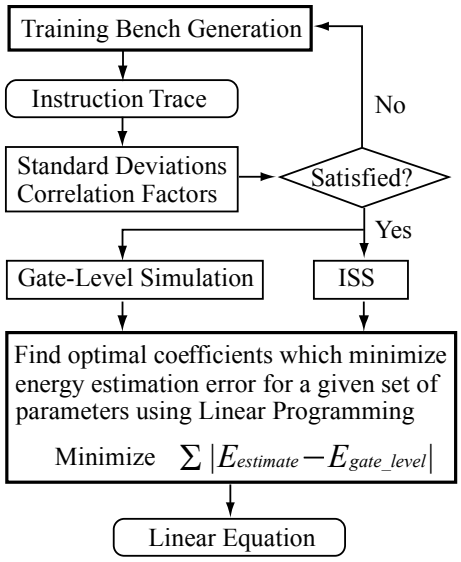

Figure 4. Training Bench Generation

D. Linear Equation Generation

We formulate an optimization problem which gives the optimal values of coefficients for minimizing the energy estimation error for a given set of parameters. The variables used in the problem formulation are defined as follows:

i, j: Indices of Instruction frame and parameter, respectively.
N, M: The number of parameters and instruction frames, respectively
E_i: The energy value for the i^{th} frame estimated using gate-level simulation
E'_i: The energy value for the i^{th} frame obtained using Equation (1)
Y_i: The absolute error value $|E'_i - E_i|$
P_{ij}: The parameter extracted from the instruction trace corresponding to the i^{th} frame
c_j: Coefficient whose value has to be determined

The problem can be formally defined as follows:
"For given sets of E_i's and P_{ij}'s, find a set of coefficients c_j's which minimize the total error, Y_{total}".

$$Minimize \quad Y_{total} = \sum_{i=0}^{M} Y_i$$

$$Subject\ to \quad -Y_i \le \left(\sum_{j=0}^{m} c_j \cdot P_{ij} \right) - E_i \le Y_i \quad (i = 0,...,M-1)$$

$$c_j,\ P_{ij},\ E_i,\ Y_i \ge 0 \quad (i = 0,...,M-1,\ j = 0,...,N-1)$$

IV. EXPERIMENTS AND RESULTS

A. Target System

We target a system which consists of a CPU core, on-chip cache memories, and SDRAM as an off-chip main memory as shown in Figure 5.

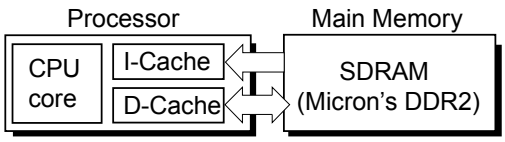

Figure 5. A Target System Model

For the off-chip main memory, we assumed a Micron's SDRAM. We used an M32R-II processor and an SH3-DSP processor as CPU cores as follows.

- **M32R-II processor**
  A 32-bit RISC microprocessor with 5-stage pipeline developped by Renesas Technology Corporation. It has 8KB 2-way set associative caches, a 32KB SRAM, and a 16-entry TLB on the chip.
- **SH3-DSP processor**
  A 32-bit RISC microprocessor developped by Renesas Technology Corporation. It has a digital signal processor core, a 32KB 4-way set associative cache, a 128KB SRAM, and a 18KB SRAM on the chip.

We synthesized the above two processors using 0.18μm CMOS standard cell library and SRAM module library.

B. Benchmark Programs

TABLE I. DESCRIPTION OF BENCHMARK PROGRAMS

	Program Description
JPEG	JPEG encoder version 6b
MPEG2	MPEG2 encoder version 1.2
compress	File compression program
FFT	Fast Fourier Transform
DCT	Discrete Cosine Transform

In our experiment, we used five benchmark programs shown in Table I. We compiled each benchmark program with two different optimization options. Each benchmark program was simulated 1,000,000 instructions for evaluating our approach. Each instruction frame was 5,000 instructions long and there were total of 200 instruction frames.

C. Detailed Characterization Flow

Figure 6 shows the details of the proposed framework. First, we generate the Switching Activity Interchange Format (SAIF) file through gate-level simulation using *NC-Verilog*^{TM} from Cadence design systems. The SAIF file has the information about the values of the signals that change during simulation. Then, the energy consumption, E_i, is calculated for the i^{th} instruction frame using *DesignPower*^{TM}, a gate-level power calculation tool from SYNOPSYS. The average energy consumption per access for the instruction cache and the data cache are calculated using library data sheets. We used the Micron System Power Calculator [21] for calculating the energy consumption of SDRAM. Similarly, we generate an instruction trace using GNU debuggers for M32R-II and SH3-DSP processors. Note that the many of ISSs used in the GNU debugger is cycle-*inaccurate*. We divide the instruction trace into small sub-traces each of which corresponding to an instruction frame and calculate the value of each parameter for each instruction frame. Finally, the optimal set of coefficients is found using *CPLEX*^{TM}, a Linear Programming solver form ILOG. The set of coefficients found minimizes the sum of estimation errors (i.e.,$\Sigma|E_i-E_i'|$). After finding the optimal values of coefficients, we can use the linear equation to estimate the energy consumption for any instruction trace.

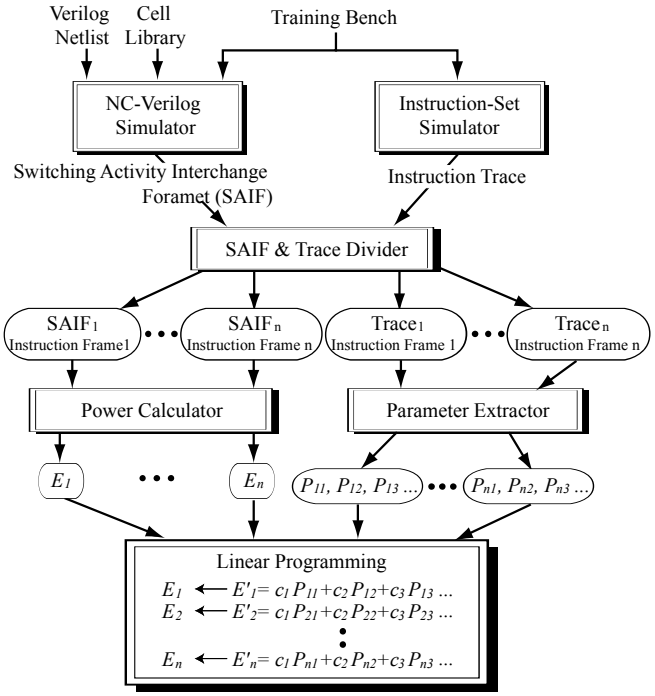

Figure 6. Detailed Characterization Flow

D. Characterization Results

TABLE II. CPU-TIME FOR CHARACTERIZATION (MINUTES)

Target Processor	M32R-II	SH3-DSP
Gate-Level Simulation	127	328
Power Calculation	32	41
Instruction-Set Simulation	< 1	< 1
LP Solver	< 1	< 1
Total CPU Time	160	370

Table II shows the characterization results. The characterizations for M32R-II and SH3-DSP took 160 minutes and 370 minutes, respectively. Although this step is time-consuming, it needs to be done only once for a target processor system. We start with a set of predetermined parameters which include 82 parameters and select some of them for a given microprocessor. We generate the training bench so that the standard deviations of every predetermined parameter values are greater than 100 and every correlation factors between any two parameters are less than 0.5. The generated training benches are simulated 475,000 instructions and 140,000 instructions for M32R-II and SH3-DSP processors, respectively. If the value of the parameter multiplied by its corresponding coefficient is very small compared to the other values, the parameter will not be used due to its weak impact on the energy estimation. In addition to this, several parameters are merged into a single parameter if corresponding coefficient values are very close to each other. As a result, we chose 30 and 19 parameters for M32R-II and SH3-DSP processors, respectively. The parameters include the following:

- The number of the following classes of instructions executed: 1) multiply, 2) divide, 3) multiply-add, 4) the other arithmetic operations, 5) logic, 6) shift, 7) register transfer, 8) load, and 9) store operations.
- The number of taken and untaken branches executed.
- The number of data and instruction cache misses.
- The number of times the instruction and data caches simultaneously miss.
- The number of times the read-after-write hazard occurs.
- The numbers of other events which cause a pipeline stall occur.

TABLE III. RESULTS FOR M32R-II PROCESSOR

	Average Error	Maximum Error	Standard Deviation of Error Percentage
JPEG	2.70 %	10.32%	2.76
JPEG_O	6.09 %	16.46%	6.17
MPEG2	1.54 %	3.97%	0.94
MPEG2_O	1.78 %	5.15%	0.96
compress	5.00%	6.41%	1.19
compress_O	4.35%	7.18%	0.93
FFT	1.55%	6.87%	0.92
FFT_O	1.45%	5.59%	0.89
DCT	1.42%	8.58%	0.72
DCT_O	1.47%	8.07%	0.69
Total	2.74%	16.46%	2.82

TABLE IV. RESULTS FOR SH3-DSP PROCESSOR

	Average Error	Maximum Error	Standard Deviation of Error Percentage
JPEG	3.17 %	11.89%	3.11
JPEG_O	6.33 %	10.02%	2.79
MPEG2	1.32 %	3.41%	0.98
MPEG2_O	1.31 %	5.63%	0.97
compress	5.73%	10.84%	1.37
compress_O	1.73%	15.15%	1.27
FFT	1.27%	3.26%	0.76
FFT_O	1.15%	4.75%	0.88
DCT	1.12%	2.20%	0.46
DCT_O	1.51%	3.04%	0.52
Total	2.47%	15.15%	2.45

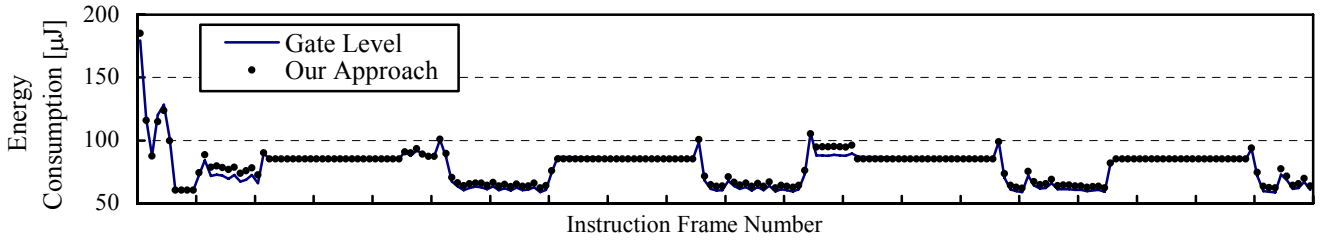

Figure 7. Energy Estimation Results for JPEG Encoder Executed on a M32R-II Processor

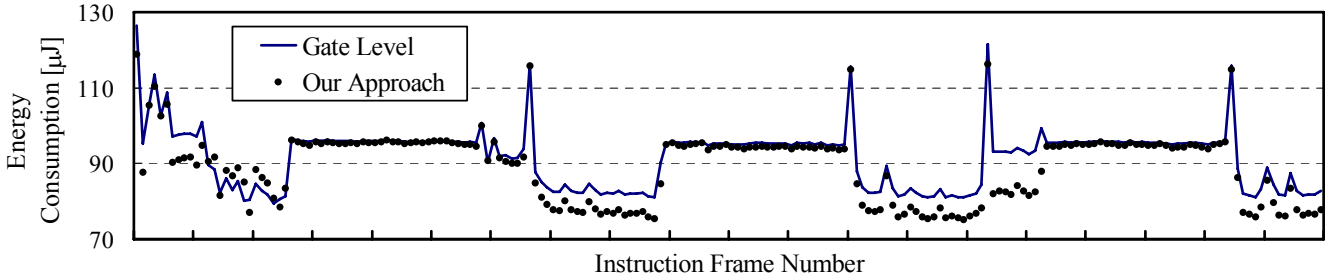

Figure 8. Energy Estimation Results for JPEG Encoder Executed on a SH3-DSP Processor

E. *Energy Estimation Results*

Average, maximum, and standard deviation of energy estimation errors for M32R-II and SH3-DSP processors are shown in Table III and IV, respectively. A suffix of each benchmark program "_O" represents that the program is compiled with a "-O3" option. The energy estimation error of our approach is on an average 2.7% and worst case 16.5% for M32R-II processor. For SH3-DSP processor, the error is on an average 2.5% and worst case 15.2%. The accuracy of energy estimation is overall very good. The notable point is that the standard deviation of error percentage is very small. This shows that our estimation results have a similar trend to the gate-level results even though absolute errors are not very small in some cases.

Figure 7 and 8 show the detailed results for JPEG encoder which runs on M32R-II and SH3-DSP processors, respectively. Horizontal and vertical axes represent instruction frame number and energy consumption per instruction frame, respectively. The energy consumption includes the energy for a CPU core, on-chip caches, and off-chip SDRAM. As one can see, the estimation errors for every instruction frames are very small.

V. SUMMARY AND CONCLUSIONS

An energy characterization framework for processor-based embedded system is proposed. This paper showed a guideline of the training bench generation for the accurate energy modeling. Experimental results using two commercial microprocessors with their on-chip instruction and data caches, and an off-chip SDRAM demonstrated that the error of our technique is on an average 3% and worst case 16% compared to the gate-level estimation results. Our energy estimation method works well even with a cycle-*inaccurate* simulator like a GNU debugger which is a de facto standard of software debugger. Once the model has been obtained, the energy consumption can be calculated with the speed of 300,000 instructions per second. Our future work will be devoted to extending the current framework to consider multi-core processor systems.

ACKNOWLEDGEMENT

This work is supported by VLSI Design and Education Center (VDEC), The University of Tokyo with the collaboration of Renesas Technology, Hitach, Ltd., Cadence Design Systems, Inc., and Synopsys, Inc. This work is also supported by Core Research for Evolutional Science and Technology (CREST) project of Japan Science and Technology Corporation (JST). We are grateful for their supports.

REFERENCES

[1] J. Flinn and M. Satyanarayanan, "Powerscope: a Tool for Profiling the Energy Usage of Mobile Applications", in Proc. of the 2nd IEEE workshop on Mobile Computing Systems and Applications, pp.2-10, February 1999.

[2] W. R. Hamburgen, D. A. Wallach, M. A. Viredaz, L. S. Brakmo, C. A. Waldspurger, J. F. Bartlett, T. Mann, and K. I. Farkas, "Itsy: Stretching the Bounds of Mobile Computing", IEEE Computer, vol. 34, pp.28-37, April 2001.

[3] V. Tiwari, S. Malik, and A. Wolfe, "Power Analysis of Embedded Software: A First Step towards Software Power Minimization", in Proc. of ICCAD, pp.384-390, Nov. 1994.

[4] M. T. C. Lee, V. Tiwari, S. Malik and M. Fujita, "Power Analysis and Low-Power Scheduling Techniques for Embedded DSP Software", in Proc. of the ISSS, pp.110-115, Sept. 1995.

[5] N. Chang, K. Kim, and H. G. Lee, "Cycle-Accurate Energy Consumption Measurement and Analysis: Case Study of ARM7TDMI", In Proc. of ISLPED, pp.185-190, Aug. 2000.

[6] C. Brandolese, W. Fornaciari, F. Salice, and D. Sciuto, "An Instruction-level Functionality-based Energy Eestimation Model for 32-bit Microprocessors," in Proc. of DAC, pp.346–351, June 2000.

[7] A. Sama, M. Balakrishnan, and J. F. M. Theeuwen, "Speeding Up Power Estimation of Embedded Software", in Proc. of ISLPED, pp.191-196, Aug. 2000.

[8] T. Sinha, and A. P. Chandrakasan, "JouleTrack – A Web Based Tool for Software Energy Profiling", in Proc. of DAC, pp.220-205, June 2001.

[9] A. Sinha, N. Ickes, A. P. Chandrakasan, "Instruction level and operating system profiling for energy exposed software", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol.11, no.6, pp.1044–1057, Dec. 2003.

[10] M. Sami, D. Sciuto, C. Silvano and V. Zaccaria, "Instruction-Level Power Estimation for Embedded VLIW Cores" in Proc. of 8th Int'l Workshop on Hardware/Software Co-design, pp.34-38, May 2000.

[11] Trimaran: http://www.trimaran.org.

[12] C. T. Hsieh, L. S. Chen, M. Pedram, "Microprocessor Power Analysis by Labeled Simulation," in Proc. of the Conference on DATE, pp.182–189, March 2001.

[13] W. Ye, N. Vijaykrishnan, M. Kandemir and M.J. Irwin, "The Design and Use of SimplePower: A Cycle-Accurate Energy Estimation Tool", Proc. of 37th DAC, pp.340-345, June 2000.

[14] D. Brooks, V. Tiwari, and M. Matonosi, "Wattch: A Framework for Architectural-Level Power Analysis and Optimization", in Proc. of ISCA, pp.83-94, June, 2000.

[15] J. T. Russell and M. F. Jacome, "Software power estimation and optimization for high performance, 32-bit embedded processors," in Proc. of ICCD, pp.328–333, Oct. 1998.

[16] S. Steinke, M. Knauer, L. Wehmeyer, and P. Marwedel, "An Accurate and Fine Grain Instruction-Level Energy Model Supporting Software Optimizations", in Proc. of Int'l Workshop on Power And Timing Modeling, Optimization and Simulation, pp.3.2.1-3.2.10, September 2001.

[17] T. Li and L. K. John, "Run-time Modeling and Estimation of Operating System Power Consumption", in Proc. of Int'l Conference on Measurements and Modeling of Computer Systems, pp.160-171, June 2003.

[18] G. Contreras and M. Martonosi, "Power Reduction for Intel XScale® Processors Using Performance Monitoring Unit Events", in Proc. of ISLPED, pp.221-226, Aug. 2005.

[19] W. L. Bircher, M. Valluri, J. Law, and L. K. John, "Runtime Identification of Microprocessor Energy Saving Opportunities", in Proc. of ISLPED, pp.275-280, Aug. 2005.

[20] T. K. Tan, A. Raghunathan, G. Lakshminarayana, N. K. Jha, "High-level Software Energy Macro-modeling", in Proc. of DAC, pp.605-610, June 2001.

[21] "The Micron System Power Calculator", *http://www.micron.com/products/dram/syscalc.html*