World Academy of Science, Engineering and Technology
International Journal of Computer and Systems Engineering
Vol:5, No:11, 2011

System-Level Energy Estimation for SoC based on the Dynamic Behavior of Embedded Software

Yoshifumi Sakamoto, Kouichi Ono, Takeo Nakada, Yousuke Kubo, and Hiroto Yasuura

***Abstract*—**This paper describes a system-level SoC energy consumption estimation method based on a dynamic behavior of embedded software in the early stages of the SoC development. A major problem of SOC development is development rework caused by unreliable energy consumption estimation at the early stages. The energy consumption of an SoC used in embedded systems is strongly affected by the dynamic behavior of the software. At the early stages of SoC development, modeling with a high level of abstraction is required for both the dynamic behavior of the software, and the behavior of the SoC. We estimate the energy consumption by a UML model-based simulation. The proposed method is applied for an actual embedded system in an MFP. The energy consumption estimation of the SoC is more accurate than conventional methods and this proposed method is promising to reduce the chance of development rework in the SoC development. ∈

***Keywords*—**SoC, Embedded Sytem, Energy Consumption, Dynamic behavior, UML, Modeling, Model-based simulation

I. Introduction

RECENTLY, the energy consumption requirements for each SoC (System-on-a-Chip) are stricter each year. During the SoC development process, it is difficult to determine whether the energy consumption satisfies the system requirements before the completion of the design or pilot phase, which often results in developmental rework. Such rework causes schedule delays and significantly increases development costs. Accurate estimates of energy consumption at early stages of the SoC development can significantly reduce the risks. Energy-saving technologies can be used in the SoC, but they must consider the SoC architecture and the physical layout design at early stages of the development.

Prior work uses spreadsheets to estimate the energy consumption in the early stages of the SoC development, but the accuracy has been poor. The standard analytical methods for energy consumption estimation using CPF (Common Power Format) or UPF (Unified Power Format) are usually used only in the final stages of SoC development.

In particular, the energy consumption of an SoC used in an embedded system is strongly affected by the dynamic behavior of its software. To reduce the energy consumption of the SoC, it is effective to use energy-saving technologies such as power gating and DVFS (Dynamic Voltage and Frequency Scaling) in the SoC design. To effectively apply these energy-saving technologies in an SoC design, the voltage and frequency changes and the ON-OFF timing of the power supply must be controlled by the dynamic behavior of the software. This dynamic behavior must be considered when estimating the energy consumption.

To address these problems, we devised a method to estimate the energy consumption of the SoC based on the dynamic behavior of the software. In the early stages of the SoC development process, modeling with a high level of abstraction is required for the dynamic behavior of the software and for the behavior of the SoC, so that the energy consumption of the SoC can be estimated. We use UML (Unified Modeling Language) model-based simulations. UML is a model description language standardized by OMG (Object Management Group) [1], and is widely used for software architecture designs in the development of enterprise or embedded systems.

The novelty of our method is to estimate the energy consumption of the SoC based on the dynamic behavior of the software. In this paper, the model describing the dynamic behavior of the software is called a dynamic behavior model and the model describing the behavior of the SoC is called an SoC behavior model.

The dynamic behavior model is created from the existing embedded systems utilizing a reverse modeling method. The SoC behavior model is created from an architectural design of the SoC, the energy consumption information of the IP cores, and the behavior of the IP cores with the energy saving technologies. The dynamic behavior model is a high-level model that can describe the dynamic behavior of the software.

We use a reverse modeling method [2, 3, 4] to create the dynamic behavior model. In the reverse modeling, an existing system is analyzed using three analytical technologies, design document analysis, static analysis, and dynamic behavior analysis. Based on the results of these analyses, the existing system can be described with a behavior model at a high level of abstraction.

The SoC behavior model calculates the energy consumption of the whole SoC and the delay time taking individual behaviors of the software, which are feed from the dynamic behavior model, into account. The dynamic behavior model feeds the individual behaviors every 1ms. The energy consumption of the whole SoC is calculated with reference to the energy consumption of the IP cores' specifications and the architectural design of the SoC. The IP cores are circuit blocks to configure the SoC, such as processors, buses, memory

Yoshifumi Sakamoto is with Graduate School of Information Science and Electrical Engineering, Kyushu University, 388, Enpukuji-cho, Muromachi-dori Oike Sagaru, Nakagyo-ku, Kyoto-shi, Kyoto 604-8175 Japan (email: sakay@jp.ibm.com)

Kouichi Ono is with IBM Research – Tokyo, Kanagawa, Japan (email: onono@jp.ibm.com)

Takeo Nakada is with IBM Research – Tokyo, Kanagawa, Japan (email: nakada@jp.ibm.com)

Yousuke Kubo is with IBM Japan Services Company Ltd., Tokyo, Japan (email: youk@jp.ibm.com)

Hiroyo Yasuura is with Kyushu University, Fukuoka, Japan (email: yasuura@c.csce.kyushu-u.ac.jp).

controllers, or accelerators. The delay time is the time for operations as affected by the energy-saving technologies used in the SoC. By feeding the delay time from the SoC behavior model back into the dynamic behavior model, the dynamic behavior model will hold the next feed for a duration specified by the delay time. Estimating the energy consumption in the model-based simulation is done with trace-driven simulations [5, 6].

We evaluated the validity by applying this method for a MFP (Multi Function Peripheral/Printer), a typical embedded-system product. The energy-saving technologies evaluated included Clock Gating and Dynamic Power Gating.

Here is the structure of this paper. Related work is covered in Section 2. Modeling appears in Section 3. Model-based simulation for estimating the energy consumption is described in Section 4. Validity verification by comparing to conventional method is in Section 5. Finally, our conclusion and future work appears in Section 6.

II. Related Work

The methods to analyze the energy consumption with CPF (Common Power Format) or UPF (Unified Power Format) [7] use a netlist. Because the netlist is a deliverable in the SoC development, such methods are only suitable for the final stages of SoC development, but are not suitable to estimate the energy consumption in the early stages while the architecture is still described at a high level of abstraction. The sleep control method [8] is a course-grained method that uses IP core unit or module unit in the SoC, but this is different from a simulation method that focuses on the dynamic behavior of the system.

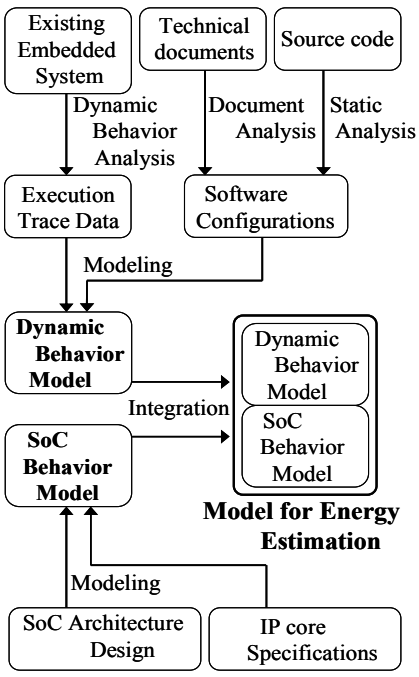

Fig. 1 Modeling flow

Spreadsheet-based methods [9] can estimate the energy consumption in the early stages of SoC development. They use both the static energy consumption, which is estimated from the specification of each IP core, and the dynamic energy consumption, which is estimated from the Activity Factor. The Activity Factor is a percentage of the switching frequency.

Our new method in this paper is similar to the spreadsheet-based methods, because it is using the static energy consumption, which is estimated from the specification of each IP core. However, the proposed method in this paper is different from the spreadsheet-based methods because is uses the dynamic behaviors of the system, not the Activity Factor.

A method using UML model to estimate the energy consumption of a cache memory in SoC has been proposed [10]. This uses the design data from the UML model. The proposed method in this paper differs because it is referring to the SoC architecture and describing its behavior as a UML model.

III. Modeling

A. Modeling Flow

The dynamic behavior model describes the dynamic behavior of the software. The SoC behavior model describes the energy consumption and the delay time. Fig. 1 shows flows to construct these models.

By applying the reverse modeling method to an existing embedded system, its dynamic behavior model is constructed as an executable UML model that reproduces the dynamic behavior of the software. It is necessary to acquire execution trace data with timing information while running the actual software in the embedded system. The execution trace data is acquired using a system observation technology [11] that insures the data describing the dynamic behaviors is not affected by the observations. The execution trace data includes the timestamps for function calls and returns, identifiers for the threads that execute the function calls, and the input values of any arguments.

In the dynamic behavior analysis, the software executes in defined execution scenarios based on the results of the design document analysis and static analysis. The execution scenarios express the focal behaviors of the embedded system as inspection objects and collect the execution trace data.

Next we refer to the system and control structures obtained from the design document analysis and static analysis, which were created from the execution trace data, to create the dynamic behavior model.

The SoC behavior model is constructed to be an executable UML model that reproduces the energy consumption and the delay time. We refer to the energy consumption of the IP cores specifications and architectural design of SoC to create an SoC behavior model. The behaviors of the SoC behavior models are dependent on the individual energy-saving technologies being used. The energy evaluation model combines the SoC behavior model with the dynamic behavior model to simulatie the energy consumption of the SoC.

B. Dynamic Behavior Model

The dynamic behavior model has two modules: The task module and task manager module (Fig. 2). The task module is a

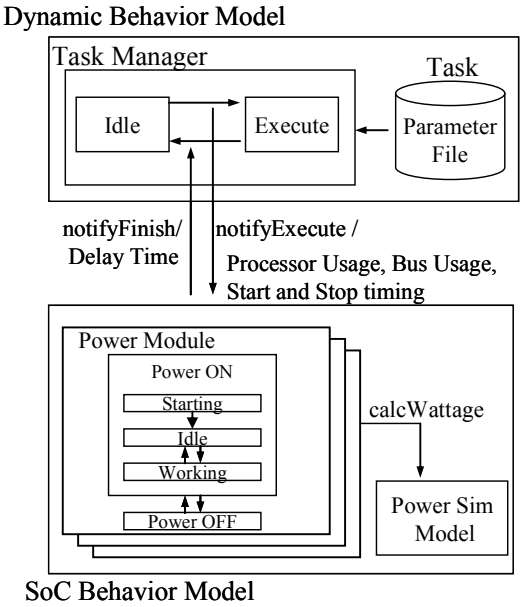

Fig. 2 Model for Energy Estimation

parameter file that configures the data obtained by analyzing the execution trace data. The task manager module describes the behaviors of the software based on the parameters read from the task module file.

The task module includes the timestamps for function calls and returns, processor usage per unit time, and the sizes of the data transfers via the buses. The task module tracks changes of the behavior of the software as the software runs. This module is loaded for a simulation scenario in a model-based simulation by the task manager module.

The task manager module controls the SoC behavior model, as described by a sequence diagram. The task manager module sends the time to start and stop operations of the IP cores to the SoC behavior model. These times are calculated from the timestamps for function calls and returns from the task manager module. The bus usage is calculated from the sizes of the data transfers per unit time and the bandwidth of each bus. The processor usage and the bus usage are also sent to the SoC behavior model.

C. SoC Behavior Model

The SoC behavior model is composed of a power module and a power sim model.

The power sim model accumulates the energy consumption of each power module and calculates the total power consumption of the SoC. The power module controls the state of each IP core and calculates its energy consumption. It tracks the end of each process and the delay time of each behavior for the dynamic behavior model. There is a power module for each IP core, which is a unit of the system structure. This is the model that describes the state transitions of the energy consumption of each IP core. It doesn't have any information about the functions or internal structures of the IP cores. Fig. 3 shows a state diagram of the power module. There is a power on state and a power off state. The power on state has internal states for

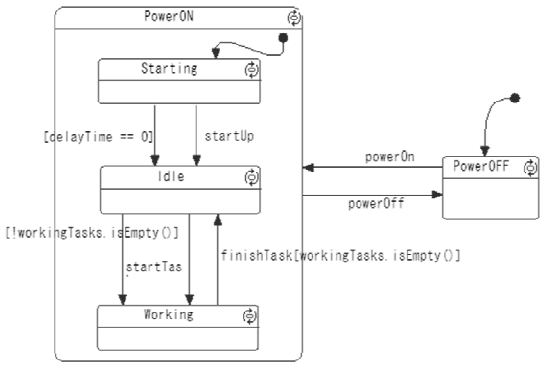

Fig. 3 State Diagram of Power Module

Starting, Idle, and Working. Starting is the state that shows the setup period for an energy-saving technology. Idle is a state that doesn't perform any data processing for some clock ticks. Working is the state that does data processing during some number of clock ticks.

D. Energy Evaluation Model of MFP

Execution trace data from the actual MFP is collected to create a dynamic behavior model. This execution scenario calls for four pages continuous printing. The images are print quality evaluation images from JEITA (Japan Electronics and Information Technology Industries Association) [12]. We selected these two images (Fig. 4) for the energy consumption simulation, which requires different processing times for the internal expression generation (IEG) and for the internal expression processing (IEP). The task module was created from the execution trace data. The task manager module was described based on the printing function of the MFP. The printing process involves these steps:

1. Print description language data is received from the host.
2. Internal Expression Generation (IEG): The data is converted into internal expressions suitable for processing in the MFP.
3. Internal Expression Processing (IEP): The internal expressions are rasterized into image data.

Next the SoC behavior model of the SoC of the MFP is described. Fig. 5 shows the organization of the SoC, based on its IP cores that are active in printing operations. For simplicity, the Configuration is limited to the processor, memory controller (Mem Ctl), bus, and two hardware accelerators (Fig.5 (a)). Accelerator A (Acc. A) and Accelerator B (Acc. B) are hardware accelerators used in the IEP. This is the baseline

Fig. 4 Execution scenario – print images

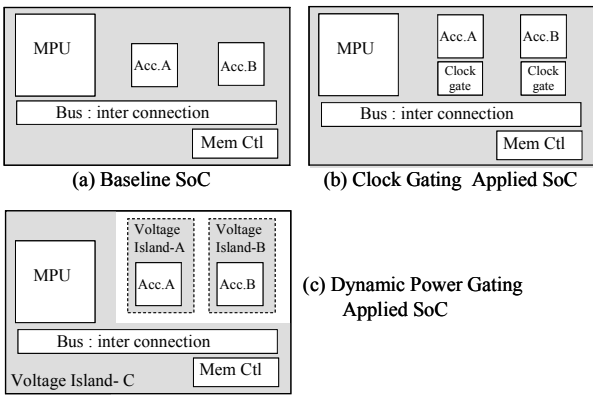

Fig. 5 SoC block diagrams

SoC.

We ran the SoC behavior model for several configurations. The energy-saving technologies adopted in new SoCs are Clock Gating and Dynamic Power Gating, which are applied to both Accelerator A and Accelerator B. In an SoC with clock gating, additional components are inserted to suspend clock to the accelerator (Fig. 5(b)). In an SoC in which Dynamic Power Gating is used, the power planes of the accelerators are isolated. Accelerator A is in Voltage Island A (VI.A), Accelerator B is in Voltage Island B (VI.B), and the other components are in Voltage Island C (VI.C). Each voltage island can cut its power supply independently (Fig. 5(c)). We used five power modules in our SoC behavior model.

This is how the dynamic behavior and the SoC behaviors were modeled. The energy evaluation model uses those models.

E. Calculation of Energy Consumption

The energy consumption of each IP core in the SoC is tracked with its dynamic behavior model. The energy consumption of the SoC is estimated based on each behavior, using the parameters for energy consumption by each IP core as shown in Table I. These energy consumptions are based on the specifications of the IP cores. The energy consumption of the processor is calculated using this equation (1), with the processor usage coming from the dynamic behavior model.

$$P_{\text{mpu\_rate}} = (P_{\text{mpu\_max}} - P_{\text{mpu\_min}}) \cdot \text{MPU usage}(\%) + P_{\text{mpu\_min}} \tag{1}$$

The energy consumption of a memory controller and a bus is calculated using Equations (2) and (3), where the memory transfer size comes from the dynamic behavior model. We use 2,240MB/s as an effective memory bandwidth. U is the usage of memory bandwidth.

$$U = \frac{\text{MemoryTransferSize}}{\text{EffectiveMemoryBandwidth}} \tag{2}$$

$$P_{\text{memc}\cdot\text{bus\_rate}} = \{(P_{\text{mem\_max}} - P_{\text{mem\_min}}) + (P_{\text{bus\_max}} - P_{\text{bus\_min}})\} \cdot U + P_{\text{mem\_min}} + P_{\text{bus\_min}} \tag{3}$$

The energy consumption at time t for Accelerator A or Accelerator B are given by $P(t)$ for discrete times. T in Equation (4) denotes the set of discrete times when the accelerator is in operation. The V in the equation denotes the set of discrete times when the accelerator is not running. The start time and the running time of each accelerator are given by the dynamic behavior model. The energy consumption of an accelerator is calculated using these equations:

$$P_{\text{acc\_}X} = \begin{cases} \sum_{t\in T} P_{\text{acc\_op}}(t) + \sum_{t\in V} P_{\text{acc\_fr}}(t) & (\text{BaselineSoC}) \\ \sum_{t\in T} P_{\text{acc\_op}}(t) + \sum_{t\in V} P_{\text{acc\_cg}}(t) & (\text{ClockGating}) \\ \sum_{t\in T} P_{\text{acc\_op}}(t) + \sum_{t\in V} P_{\text{acc\_pg}}(t) & (\text{DynamicPowerGating}) \end{cases} \tag{4}$$
$$(X = \text{A,B})$$

Here is the equation of the total energy consumption of the system in the energy evaluation:

$$P = P_{\text{mpu\_rate}} + P_{\text{memc}\cdot\text{bus\_rate}} + P_{\text{acc\_A}} + P_{\text{acc\_B}} \tag{5}$$

IV. Simulation

We evaluated the energy consumption of three different SoC designs with model-based simulations. One was the baseline SoC and the other two SoCs used energy-saving technologies. The simulation scenarios are the same as in the dynamic behavior analysis of the execution scenarios, with two print images and each of them is four pages continuous print. We used the model execution feature in IBM Rational Rhapsody [13] for the model simulations.

It is important to insure the changes of the energy consumption are driven by the system behavior in the simulation. Fig. 6 shows the simulation results showing the changes in the energy consumption over time. The SoC model of the clock gating reduces the energy consumption by 6.5% compared to the baseline SoC model. The SoC model of the dynamic voltage gating reduces the energy consumption by 15% compared to the baseline SoC. However, the delay time affect to the overall operating time. The operating time is by 10,390 microseconds longer than the baseline SoC. The delay time is startup waiting time of 10 microseconds. The startup waiting time is a duration defined from a point when a process start notice is released by the dynamic behavior model to a

TABLE I
ENERGY CONSUMPTION FOR EACH IP CORE IN THE SOC

IP Core	Condition	Name	Energy Consumption(mJ)
Processor	Working(Max.)	Pmpu_max	120
	Idle	Pmpu_min	615
Memory Controller	Working(Max.)	Pmem_max	53
	Idle	Pmem_min	35
Bus and Inter connections	Working(Max.)	Pbus_max	37
	Idle	Pbus_min	20
Accelerator –A And Accelerator-B	Working(Max.)	Pacc_op	165
	Idle	Pacc_fr	140
	Clock Gating	Pacc_cg	84
	Dynamic Power Gating	Pacc_pg	9

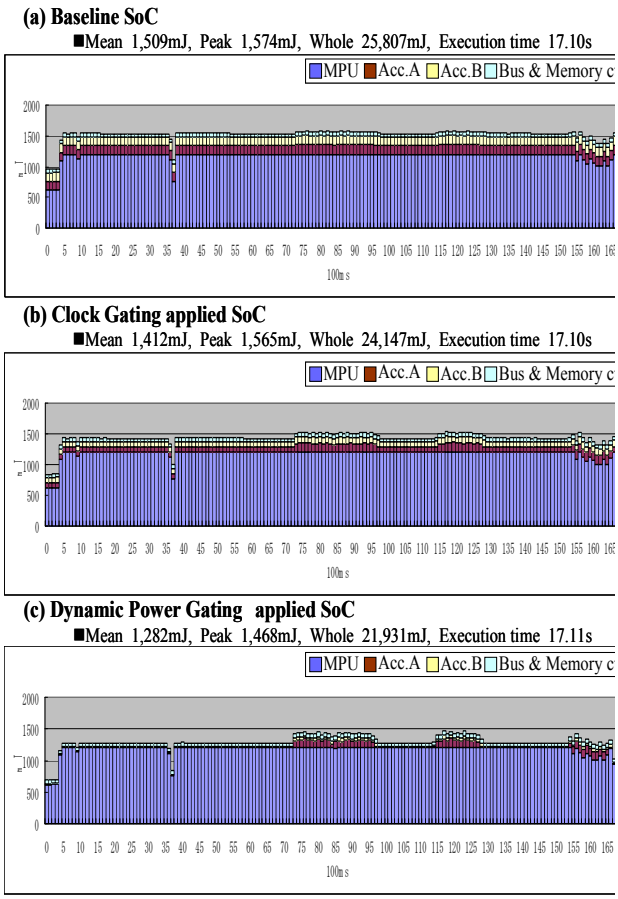

Fig. 6 Simulation results of energy simulation, continuous printing of four pages

point when the power for the voltage island switches on.

Next, we simulated the tradeoff for dynamic power gating in the application. One tradeoff is between the startup waiting time and the energy consumption. The other is the tradeoff between the energy consumption caused by the surge current and the extension of the operating time.

The startup waiting time is swept incrementally to find a trade off point (Fig. 7). When the startup waiting time is more than 5,500 microseconds, the total energy consumption is larger than the baseline SoC, and so there no net energy savings from the dynamic power gating. We believe this is because of the increase in the total energy consumption due to the longer processor operating time from the longer setup waiting and delay times.

The energy consumption by the surge current at the time of power supply startup has to be added to the model. Also, we must consider the time from the end of execution of the accelerators to the time the power is cut for the voltage island (Fig. 8). The energy consumed by the surge current is five times the normal value, and the surge lasts 500 microseconds. When the hold time is set at 3,000 microseconds, the overall energy consumption is minimized. In addition, if the hold time is set at 29,300 microseconds or longer, there is no net energy reduction. This indicates that the periods when the accelerators are in operation are not evenly distributed.

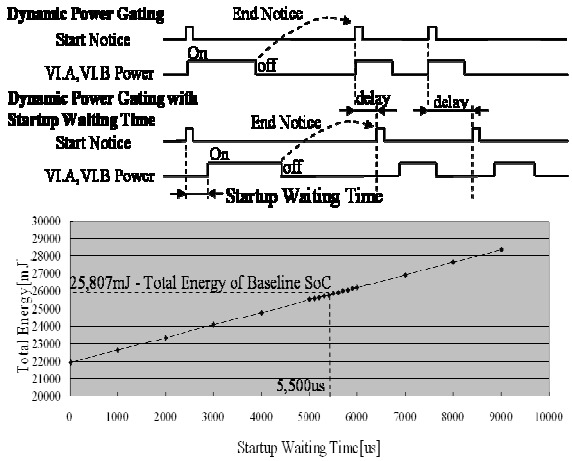

Fig. 7 SoC energy consumption and startup waiting time

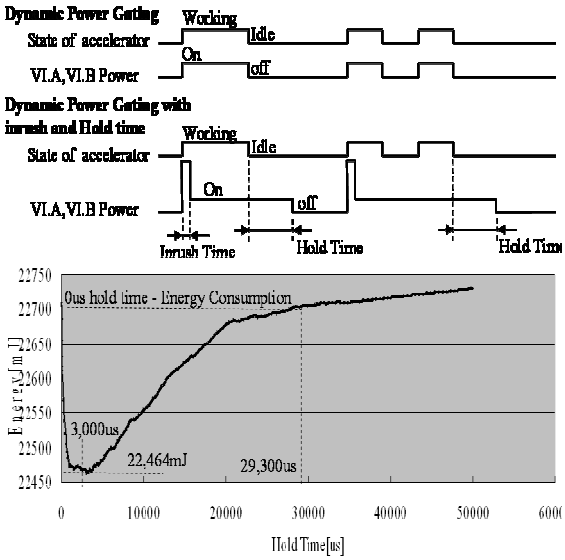

Fig. 8 SoC energy consumption and hold time

V. Validity Verification

To verify the validity of the proposed method, we compared the estimated energy consumption with the spreadsheet and the proposed method. The verification subjects were the baseline SoC, a clock-gated SoC, and dynamic-power-gated SoC. Currently, the spreadsheet-based estimation method for energy consumption is generally used. In addition, the energy consumption was measured for the actual SoC.

The results of these comparisons appear in TABLE II. TABLE III shows the settings for the activity factor in the spreadsheet-based method. The activity factor in the actual SoC design was 0.070. The estimated conditions were a junction temperature of 105°C and process rules of 90 nm. The ratio of execution time per page for Accelerator A was 18.6% and Accelerator B was 10.3%. These ratios came from the operating logs of the actual software. These ratios were reflected in the

TABLE II
COMPARISON OF ENERGY CONSUMPTION

SoC type	(a) Proposed Method (mJ)	(b) Spread sheet (mJ)	(c) Actual SoC (mJ)	Error (a) vs (c) (%)	Error (b) vs (c) (%)
Baseline	1,509	1,620	1,361	10.9	19.0
Clock Gating Applied	1,412	1,488	1,250	13.0	19.0
Dynamic Power Gating Applied	1,282	1,377	1,113	15.2	23.7
Average	-	-	-	13.0	20.6

TABLE III
ACTIVITY FACTOR FOR ENERGY ESTIMATION SPREADSHEET-BASED ENERGY ESTIMATION

IP Core	Baseline SoC	Clock gating / Dynamic Power Gating SoC
Processor	0.070	0.070
Memory Controller	0.070	0.070
Bus and Inter connections	0.070	0.070
Accelerator A	0.070	0.013
Accelerator B	0.070	0.007

activity factors of each accelerator. We calculated the energy consumption of the same configuration as the model of the SoC from the measurements of energy consumption during normal operations of the actual SoC and from measurement of the energy consumption with power gating of the actual SoC. The average error of the proposed method and the actual SoC was 13.0% and the average error of the spreadsheet-based method and the actual SoC was 20.6%. These results show the proposed method is appropriate to estimate the energy consumption of the SoC in the early stages of SoC development. The proposed method had higher accuracy than the standard spreadsheet-based method.

VI. CONCLUSION AND FUTURE WORK

In this paper, we described a system-level SoC energy consumption estimation method based on the dynamic behavior of the embedded software. We represented the dynamic behavior of the software and the energy consumption behavior of the SoC using UML models. We estimated the energy consumption with a model-based simulation using those UML models. From the comparisons with conventional method and the actual SoC, the proposed method accurately estimated the energy consumption of the SoC in the early stages of the SoC development. Our future works will involve two directions. One is to improve the accuracy of the simulation, which requires more accurate modeling of the processor that consume the most energy. The other direction is to consider how to add other IP cores in the SoC to the model.

ACKNOWLEDGMENT

The authors would like to thank the members of this project for their comments and support.

TRADEMARK

IBM, Rational, and Rhapsody are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

REFERENCES

[1] Object Management Group, http://www.omg.org/
[2] K. Ono, M. Toyota, R. Kawahara, Y. Sakamoto, T. Nakada, and N. Fukuoka, "A modeling method for performance analysis in model-driven development", Proceedings of the *13th Design, Automation and Test in Europe* (DATE 2010), 2010, pp.1337-1340.
[3] K. Ono, M. Toyota, R. Kawahara, Y. Sakamoto, T. Nakada, and N. Fukuoka, "A Model-based Method for Evaluating Embedded System Performance by Abstraction of Execution Traces", Proceedings of *6th European Conference on Modeling Foundations and Applications* (ECMFA 2010), Springer (2010), pp.233-244.
[4] Y. Sakamoto, T. Nagano, T. Nakada, K. Ono, K. Hisazumi, and A. Fukuda, "Development of Embedded Systems Using Reverse Engineering and Model-based Performance Evaluation", Proceedings of *the 5th International Conference on Project Management* (ProMAC2010), 2010, pp.160-168.
[5] J. M. Hsu and P. Banerjee. "Performance measurement and trace driven simulation of parallel CAD and numeric applications on a hypercube multicomputer". *IEEE Transactions on Parallel and Distributed Systems,* Vol. 3, No. 4, pp. 451.464, July 1992.
[6] C. A. Prete, G. Prina, and L. Ricciardi, "A trace-driven simulator for performance evaluation of cache-based multiprocessor systems.", *IEEE Transactions on Parallel and Distributed Systems,* Vol. 6, No. 9, pp. 915-929, September 1995.
[7] IEEE Std.1801, Standard for Design and Verification of Low Power Integrated Circuits, http://standards.ieee.org
[8] Y. Kanno, "Hierarchical Power Distribution with 20 Power Domains in 90-nm Low-Power Multi-CPU Processor", *Solid-State Circuits Conference,* 2006. ISSCC 2006. Digest of Technical Papers. IEEE International
[9] Cadence InCyte Chip Estimator, http://www.cadence.com/products/ld/chip_estimator/pages/default.aspx
[10] A. G. Silva-Filho, R. F. Cordeiro, C. Cristiano, Ara ´ ujo, A. Sarmento, M. Gomes, E. Barros, and M. E. Lima, "An ESL Approach for Energy Consumption Analysis of Cache Memories in SoC Platforms", *Hindawi Publishing Corporation, International Journal of Reconfigurable Computing,* Volume 2011, Article ID 219497,
[11] N. Ohba and K. Takano, "Hardware debugging method based on signal transitions and transactions", Proceedings of *the 11th Asia and South Pacific Design Automation Conference* (ASP-DAC 2006), 2006, pp. 454-459.
[12] JEITA - Japan Electronics and Information Technology Industries Association, http://www.jeita.or.jp/english/
[13] IBM Rational Rhapsody, http://www-01.ibm.com/software/awdtools/rhapsody/