ISSN: 2321-2152

IJMECE

International Journal of modern electronics and communication engineering

E-Mail
editor.ijmece@gmail.com
editor@ijmece.com

www.ijmece.com

ISSN2321-www.ijmece .com
Vol 8, Issuse.1 Jan 2020

Design of embedded systems with hybrid power estimate based on models

Mr. T. Gangadhar Rao, Mrs. B. Vijaya, Mr. R. Srinivas

Abstract

System-on-chip (SoC) power management is becoming more and more important as technology moves toward tighter integrated circuit architectures and faster performance. Power estimation now forms an essential element of the design process and calls for approaches at the electronic system level (ESL). The main goal of designing such specialized machinery is to increase the ratio of precision to speed. In this study, we propose a consumption prediction technique that may be applied early in system design by including consumption needs into cosimulation. This ground-breaking technique may be used with both solo power estimators and annotated power models to predict the energy consumption of white-box and black-box IPs. We performed system-level CABA (cycle accurate bit accurate) SystemC simulations to obtain the most accurate power estimates. We use a model driven engineering (MDE) method to automatically construct the simulated structures, which include standalone power estimators, making our strategy both quick and user-friendly. It is feasible to estimate consumption of the same architecture using both annotated power models and standalone power estimators simultaneously.

Introduction

While advancements in system-on-a-chip (SoC) integration have led to better computer performance, a major problem now is power loss. So, when exploring space for design, it is essential to consider power consumption. Reaching target time-to-market requires striking a compromise between power consumption and performance as early as possible in the design process. We want estimating methodologies that provide abstraction and automation in order to solve the power challenge without compromising design efficiency. Due to the extensive evaluation of the simulated system-on-chip (SoC) using low-level energy estimation techniques, the design time for complicated systems is substantially increased. Such approaches may have some accuracy, but they are much too sluggish to be practical. Consequently, we need novel theoretical approaches to outcome prediction. The cycle-accurate bit-accurate (CABA) level gives a more accurate description of a system than the register transfer level (RTL) [1]. It enables quicker simulation speeds compared to RTL. When transitioning from RTL to CABA, the processor side of the system is often shielded from the hardware implementation details, but the behavior at the clock cycle level is maintained. If components can communicate with one another via a binary system, then the term "bit-accurate" describes them well.

Assistant Professor^{1,2}, Associate Professor & HOD

Department of Electronics & Communication Engineering,

It is essential to model the system's behavior at the CABA level for precise power estimations. One may compromise between the simulation's speed and accuracy by adjusting the level of abstraction. Our simulations will operate at this degree of abstraction since that is the level that logic mandates.

Connected Tasks

How to most accurately predict power consumption in system-on-a-chip designs has been the subject of much study. They employ distinct estimation methodologies and work on different abstract levels. As it relies on electric currents flowing through the transistors, a description of the system-on-a-chip (SoC) down to the transistor level is necessary for layout-level power consumption estimation. It is possible to optimize the size and placement of the transistors based on the computing findings. Among them, SPICE [6] stands out for its ability to accomplish comparable goals. For more complex scenarios, this method requires a lot of time and resources, but it gets the job done. The use of this method becomes troublesome when applied to more complex systems because of these constraints. We use technology cell power models to evaluate consumption at the gate level. A cell's power consumption is proportional to its data rate. Of the several such parameters, the power supply's voltage and frequency are simply two examples. Consequently, in order to reduce the system's overall power consumption, a battery of tests has to be performed to optimize consumption characteristics. Synopsys' Power Gate is an example of a utility that functions at this level [7]. Such approaches nevertheless provide a very accurate estimate, even when dealing with the same quantity of data and simulation time limitations as at the layout level. The most advanced software-based logic simulator could still struggle to achieve even a few cycles per second with a design that has ten million gates [8]. There is a tenfold improvement in performance when using RTL compared to using Gate, and a hundredfold improvement when using layout (transistor) levels. It is simpler to express systems with more abstract components in RTL, which allows for faster simulation, such as adders and multipliers. You may estimate consumption in two ways. The first approach makes advantage of probabilistic inference [9]. **Gaspard2 Design Framework with Model-Driven Engineering**

Three main concepts are presented in MDE. of all kinds, including models, meta-transformations, and transformations. Every model is an abstract representation of some reality, but at its core are concepts and connections. A notion is a representation of the "things" it stands for, but a relationship is the "links" between actual items. One model in MDE might have several "views," or points of view. Reuse is made easier using the abstraction approach as it gets rid of the details. A metamodel is a collection of concepts and relations used to define models in a language for describing models; it also gives the syntax of a model. Like the bond between a piece of writing and the rules of its language, this one is inseparable. For a model to be considered consistent, it must adhere to its metamodel. Finally, MDE allows for the separation of concerns in various models, which makes the models more reusable and keeps them readable for humans. The development process of MDEs starts at a high degree of abstraction and moves through intermediate levels of abstraction via model transformations (MTs) in order to produce measurable results, such as executable models (or code) [32]. As they descend from higher to lower levels of abstraction, MTs improve, which helps maintain the different models in sync. At each stage, the MTs receive more detailed information on how they will be put into action. One way to transform the initial models into the final ones is using a compilation mechanism called an MT.

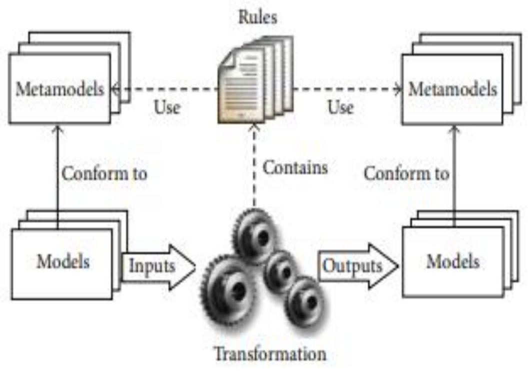

Figure 1: An overview of model transformations.

a more complicated model to an easier one. Figure 1 shows that when MT is performed externally, the source and target models follow different metamodels; however, when MT is performed internally, the source and target models follow the same metamodel. Typically, the intermediate, lower-level models will naturally offer technological conceptions, whereas the introductory, high-level models will solely include domain-specific ideas. To improve concepts in a destination metamodel, a metamodel transformation (MT) relies on a collection of rules, either declarative or imperative, that help to identify ideas in a source metamodel. An updated MT specific to a new model may be built by adding or changing rules. This technique has the advantage of allowing several model transformations to be defined at the same level of abstraction. These transformations may then be directed at different sets of lower-level abstractions, giving possibilities to target different technical platforms. You can do model transformations in either a one-way (where you can only change the source model and it will automatically generate the target model) or a two-way (where you can modify both the source and target models at the same time) fashion. The second case might lead to an issue with synchronizing the models [34]. As a standard for metamodel expression and query/view/transformation (QVT) [35], the OMG-proposed metaobject Facility (MOF) allows model transformations to be expressed.

A Mixed Methodology for Predicting Energy Use

At the CABA level, the sum of the energy consumptions of the system's subsystems yields the system's overall energy consumption. Consumption associated with read/write operations (dynamic consumption) and leakage currents (static consumption) are taken into account to offer an accurate calculation of total consumption. Static power consumption has long been overlooked in favour of its dynamic counterpart. This viewpoint shifted, however, with the introduction of new submicron technologies, which give equal weight to both forms of consumption. Since the related parts are straightforward, so too are the consumption models used in this study. The following equation serves as the foundation for our consumption models:

$$E = \sum_i N_i * C_i,$$

above, Ni is the component's total activity I or inactivity cycle realizations, and Ci is the activity I or inactivity cycle unit cost. The SoCLib library's central processing unit (CPU), cache, shared random access memory (SRAM), and interconnection network power consumption profiles were determined using this method [38]. In [5], you will find all the information you need on various consumption models. We were able to do it by quantifying the unit expenses of each component and identifying its main functions using simple instruments. We stick with the same power models for this project. To keep track of how many times something has happened, activity counters are used. As soon as the corresponding simulation event occurs, the corresponding counter is raised. Each cycle makes use of the whole architecture when using this strategy. The simulator's energy consumption models get data from the activity counters and use it to monitor the design's energy dissipation. The consumption simulator includes an energy model that reflects the technical details of each piece of gear. Figure 3 illustrates this approach. It is not possible to utilize the counters if the IP's code is unavailable. The source code of white-box IPs can have counters inserted into it, but until the source code of black-box IPs is made available, no one can monitor their actions.

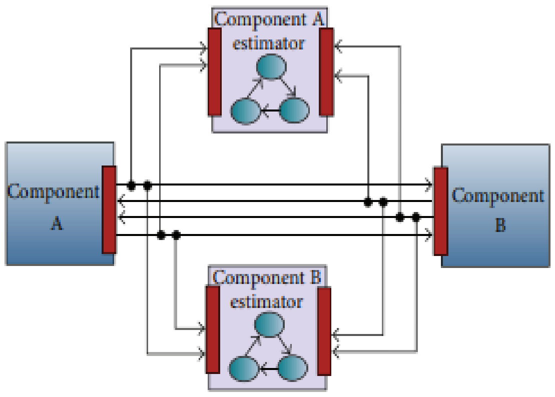

Figure 2: Power consumption estimators.

Independent modules linked to these IPs monitor events. In order to evaluate the consumption of a system that contains both white-box and black-box IPs, our hybrid method integrates these two approaches. Methods for estimating white-box and black-box IPs' usage are described below.

Experiment Outcomes

We determined the resource requirements of a 256x256 JPEG compression application to demonstrate the effectiveness of our technology. You may use this tool to convert between the BMP and JPEG picture formats. A number of simulated versions of the JPEG program were evaluated. The hardware implementations of these designs depend on micro networks, 16 KB SRAM memory, and MIPSR3000 CPUs. Both the instruction and data caches are independent, although they share an interface for communication. Caches that are directly mapped are just that. A write-through policy is in place for the data cache. You may use the same approach to analyze the IPs below as if they were black-box IPs, even though they are all white-box IPs (meaning their source code is clearly visible). Therefore, we integrate the counters into the program using independent estimators. In a simulated system, this enables us to use a white-box approach for certain IPs and a black-box one for the others. The simulations employed a two-processor configuration, as shown in Figure 16. In order for the estimator to function, it needs two interfaces, one for the left side and one for the right. Black box approach was used to construct the other IPs, but white box methodology was used to develop the cache memory in Figure 16. We were able to monitor the evolution of performance during the simulation by feeding an XML file that tracked the consumption of each architectural component into a reporting engine. This file then generated a consumption report. The report includes the percentage of overall consumption for each component across all kinds of simulations.

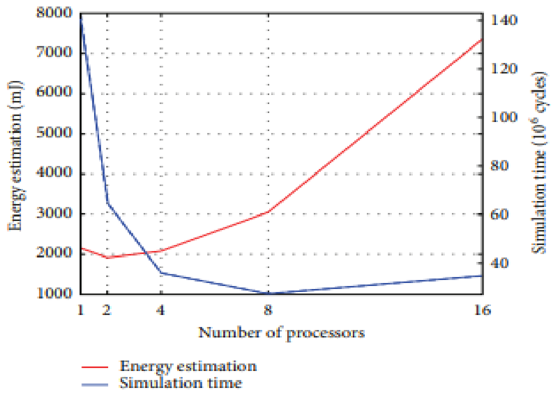

Figure 3: Performance and energy variation in terms of the number of processors.

cycles and information about the current cycle. It can also give details about a cycle chosen by the user and indicate the cycle where a consumption threshold fixed by the user was exceeded. This report gives a graphical view of the performance of the simulated architecture, which facilitates the design space exploration. To evaluate the impact of the number of processors on the performance and the total consumption of the system, we executed the JPEG application using systems with 1 up to 16 processors. The size of the instruction and data cache was set to 4 KB, and the MIPS frequency was set at 50 MHz All the processors execute the same JPEG application but on different image macroblocs. Figure 17 reports the execution time in cycles and the total energy consumption in mJ.

Table 1: Simulation time and energy consumption for combined estimation techniques.

Estimation approach				Simulation time (s)	Energy consumption (mJ)
Processor	Cache memory	Shared memory	Interconnect		
W	W	W	W	436	2080.4
W	W	W	B	550	2080.4
W	W	B	W	499	2080.4
W	W	B	B	578	2080.4
W	B	W	W	607	2077.4
W	B	W	B	731	2077.4
W	B	B	W	632	2077.4
W	B	B	B	756	2077.4
B	W	W	W	514	2080.4
B	W	W	B	598	2080.4
B	W	B	W	541	2080.4
B	W	B	B	627	2080.4
B	B	W	W	661	2077.4
B	B	W	B	781	2077.4
B	B	B	W	679	2077.4
B	B	B	B	828	2077.4

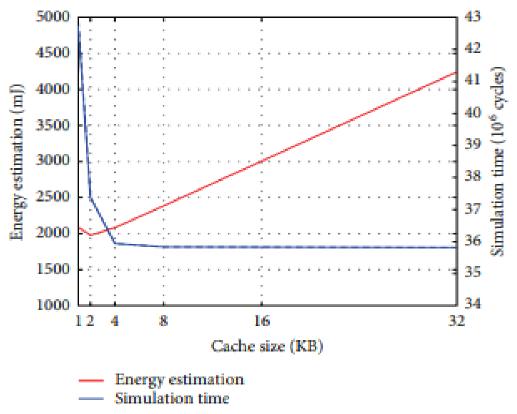

Figure 18: Variations in performance and energy consumption in terms of cache size with 4 processors.

Based on these results, it seems that adding more processors to a system could improve performance by decreasing execution time. There is non-linearity in the variation because activities share resources; processes can't always accomplish the same objective simultaneously, leading to waiting periods that reduce system performance. Down to a certain point, a system's total energy consumption drops as the number of execution cycles drops; beyond that, it tends to level off as the system's performance rises. Above a certain point, however, adding additional processors usually doesn't help much; doing so only causes more interconnect conflicts, which in turn causes more waiting cycles, which changes overall performances, especially power consumption. Using a four-processor configuration, we measured the effects on performance and power consumption of varying the sizes of the instruction and data caches. We executed the JPEG-parallelized technique with data and instruction cache sizes varying between 1 KB and 32 KB. We present our results in Figure 18. A higher overall energy consumption is indicative of a bigger cache. The amount of work or data being processed determines whether a larger cache improves system performance or not. Updating our cache from 4 KB to 8 KB resulted in a 14% increase in energy use but a 0.3% improvement in speed. Despite a 78% increase in power utilization, there was no discernible performance difference between the 8 KB and 16 KB caches. We built a 4-core system with 4 KB of cache to demonstrate that a system can use white-box and black-box approaches simultaneously. Changing our estimation methods for different parameters allowed us to run a large number of simulations. Because there are four different types of components in this case—the central processing unit (CPU), cache memory, shared memory, and connection—a total of sixteen individual simulations were performed, the results of which are shown in Table 1. Energy estimates are within a 0.15% margin of error across all conceivable permutations, according to the simulation findings. Because the signals between an architecture's components cannot reveal when cache FIFOs are being used, this discrepancy is caused by the invasive technique. It is possible to disregard the FIFO's consumption if it is little, and then we can say that the independent estimation modules provided trustworthy results by taking into consideration the most crucial and resource-intensive operations performed by

ISSN2321-www.ijmece .com
Vol 8, Issuse.1 Jan 2020

the components which were studied.

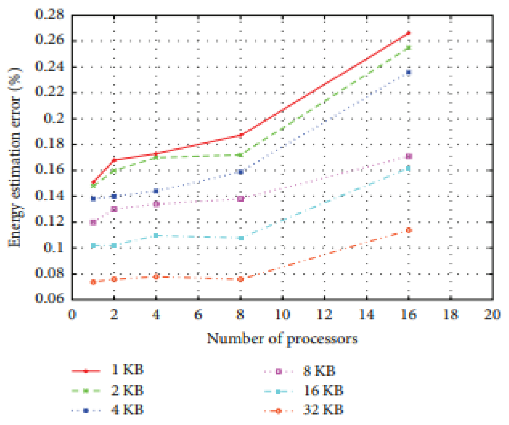

Figure 4: Variation in energy consumption estimation error in terms of cache size and number of processors.

Table 1 shows how the black-box method's extra modules cause a noticeable increase in simulation time. When the black-box method is used on every part of a system, a boost of up to 90% is achieved. When applied to the same system, combining the two methods speeds up simulations. Since the simulation at the CABA level is quick, even a doubling (the worst case) of the simulation time still allows for fast simulations, the increase may be deemed to be acceptable. We simulated systems with varying numbers of processors and cache sizes to evaluate the effectiveness of the black-box method. Two distinct kinds of simulations were run. Black-box approaches treat all parts of a system as opaque, whereas white-box approaches treat them as transparent. When comparing two systems with the same number of processors and cache size, but one using a white-box technique and the other using a black-box approach, the increase in simulation time was between 80% and 98%. Figure 19 demonstrates that when compared to the white-box method, the black-box method's estimate error does not go over 0.3%.

Conclusion

This paper presents a hybrid technique for estimating the energy consumption of systems-on-chip (SoC). This approach could work for white-box and black-box IPs. Our approach to white-box IP system simulation includes incorporating activity counters into IP codes to track the occurrence and consumption of activities. Through the signals they exchange while simulated, estimation modules are able to pick up on the behaviors of black-box IPs. An MDE technique was used to generate these modules. The Gaspard2 framework also used high-level models to automatically generate the simulated systems. We increased the framework's deployment level so that energy estimates could be included into the design process. The simulation's CABA implementation was done using SystemC. Because this degree of abstraction considers a fair amount of architectural elements, our simulations ran fast and produced accurate results. In order to test our approach in different environments, we used a combination of white-box and black-box methodologies. Even without access to an IP's source code, the simulation results showed that reliable consumption estimates could be derived. By analyzing the signals it provides and receives from other components of the system, we can easily determine the activity and its associated consumption cost. In future research, we want to expand our present work to include systems with more components, such as several CPUs and other interconnects. In order to directly compare the results of the simulations with those from real-world implementations of the systems under consideration, our future study may potentially include consumption data from these systems.

References

[1] "A fully static scheduling approach for fast cycle accurate systemc simulation of MPSoCs" (R. Buchmann and A. Greiner, 2007, pp. 101-104 in Proceedings of the International Conference on

Microelectronics (ICM '07), Cairo, Egypt, 2007.

"Portal of the model driven engineering community," MARTE: Modeling and Analysis of Real-Time Embedded Systems, 2009, published by OMG, is a UML Profile.

For reference, see DaRT's 2009 article "Gaspard2 framework" at http://www.gaspard2.org/.

[5] "Estimating energy consumption for an MPSoC architectural exploration," in Proceedings of the ARCS, vol. 3894 of Lecture Notes in Computer Science, pp. 298-310, Frankfurt, Germany, 2006, by R. Ben Atitallah, S. Niar, A. Greiner, S. Meftali, and J. L. Dekeyser.

(6) "Spice macromodel for power dmos transistors" by M. Andersson and P. Kuivalainen was published in 1992 in the Nordic Semiconductor Meeting proceedings in Finland.

referenced as "Synopsys low power solutions for asic design flow" in a 1998 technical report by Synopsys, available online at http://vada.skku.ac.kr/ClassInfo/ic/lowpower/.

(Reduced system-on-chip (SoC) simulation and development time) by C. Rowen, published in 2002 in IEEE Computer, volume 35, issue 12, pages 29–34.

Presented at the 1999 IEEE International Conference on Electronics, Circuits and Systems in Cyprus, "A probabilistic approach for rt-level power modeling" was written by J. Costa, J. Monteiro, L. M. Silveira, and S. Devadas.

[10] In an article published in 1998 in the IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Wu, Qiu, Pedram, and Ding discuss "Cycle-accurate macro-models for RT-level power analysis." The paper is located in volume 6, issue 4, and runs from pages 520 to 528.

[11] "Petrol approach to high-level power estimation" by R. P. Llopis and K. Goossens was published in August 1998 in the Proceedings of the International Symposium on Low Power Electronics and Design (pp. 130-132).