Model-based hybrid power estimate for embedded system design

Dr M S Priyadarshini[1,] C N Arpitha[2] M Bhaskar Reddy [3], K Ramamohan Reddy[4]

[1,2,3,4] Associate Professor, Department of EEE, K. S. R. M College of Engineering(A), Kadapa

Abstract

As technology progresses toward larger density of integrated circuits and better levels of performance, system-on-chip (SoC) power management becomes increasingly crucial. Electronic system level (ESL) approaches are now necessary for power estimation throughout the design cycle. Improving the equilibrium between accuracy and speed is the major challenge in designing such specialized instruments. In this study, we provide a technique for predicting consumption that may be used at the preliminary stages of a system's design to include consumption criteria into the cosimulation. This is a revolutionary approach since it can be used to predict the energy consumed by both white-box IPs with the assistance of annotated power models and black-box IPs with the aid of standalone power estimators. We performed SystemC simulations at the CABA (cycle accurate bit accurate) level to get the most accurate power estimates possible. Our technique is fast and easy to use since we employ a model driven engineering (MDE) approach to automatically build the simulated architectures, which include standalone power estimators. Parallel use of annotated power models and standalone power estimators for estimating consumption of the same architecture is possible.

Introduction

Although improvements in computing performance have resulted from developments in system-on-a-chip (SoC) integration, power loss has emerged as a key concern. As a result, it is crucial to think about power usage as part of the design space exploration process. Achieving a balance between performance and power consumption early in the design cycle is critical for meeting time-to-market goals. The power problem can't be fixed while sacrificing design productivity, thus we need estimation methods that enable abstraction and automation. Because low-level energy estimate algorithms evaluate many aspects of the simulated SoC, they significantly lengthen the design time required for complex systems. While such methods could be accurate, they are much too slow to be effective. As a consequence, we need new theoretical methods for forecasting results. Compared to the register transfer level (RTL), the cycle-accurate bit-accurate (CABA) level provides a more precise description of a system [1]. It allows for faster simulation speeds than can be achieved using RTL. When moving from RTL to CABA, the implementation details of the hardware are often hidden from the processor side of the system but the behavior at the clock cycle level is preserved. The phrase "bit-accurate" implies that parts can exchange data utilizing a binary form of communication. In order to get accurate power estimates, it is necessary to simulate the system's behavior at the CABA level. The amount of abstraction used here allows for a trade-off between simulation speed and accuracy. Due to this Logic dictates that this is the abstraction level at which our simulations will run.

Connected Tasks

There has been a lot of research on how to best estimate power usage in SoC architecture. They operate on various abstract levels and use different estimating techniques. Because it is based on electric currents flowing through the transistors, estimating power consumption at the layout level requires a description of the SoC down to the transistor level. The results of the computations may then be used to fine-tune the transistor size and arrangement for optimal usage. One such tool is SPICE [6], which achieves similar results. Although this approach is quite accurate, it is time-consuming and resource-intensive for modeling more complicated situations. Due to these limitations, the application of this approach to increasingly complicated systems is problematic. In order to assess consumption at the gate level, technological cell power models are utilized. A cell's energy usage is directly related to the amount of data it receives. The voltage and frequency of the power supply are only two examples of many such factors. As a result, a battery of tests must be run to optimize consumption characteristics in an effort to cut the system's total power usage. One tool that operates at this level is Synopsys' Power Gate [7]. A similar amount of data and simulation time constraints exist as they do at the layout level, but the estimate is still very accurate using such methods. A design with 10 million gates may only be able to reach a few cycles per second even in the most sophisticated software-based logic simulator [8]. When compared to Gate and layout (transistor) levels, RTL offers speed increases of 10x and 100x, respectively. Systems with more abstract components, like as adders and multipliers, are easier to describe in RTL, allowing for speedier simulation. There are two methods for estimating consumption. Probabilistic inference is used in the first method [9].

Gaspard2 Design Framework with Model-Driven Engineering

There are three key ideas in MDE. transformations, meta-transformations, and models, naturally. Concepts and relationships are the backbone of every model, which is an abstract depiction of some reality. Relationships are the "links" between real-world entities, whereas concepts stand in for the "things" they represent. Multiple perspectives, or "views," can be taken on the same model in MDE. The abstraction method facilitates reuse by obviating the need to deal with specifics. A metamodel provides the syntax of a model and is a set of ideas and relations for defining models in a model description language. This connection may be compared to that between a text and its language's grammar. Every model is said to be consistent with its metamodel. Finally, MDE enables the concerns to be separated in multiple models, which promotes reusability and maintains the models' readability for humans. For tangible outcomes like an executable model (or code), the MDE development process begins at a high abstraction level and flows through intermediate levels of abstraction through model trans formations (MTs) [32]. To assist keep the various models in sync, MTs perform improvements as they go down from higher to lower levels of abstraction. The MTs get specifics on their actual implementation at each successive level. An MT is a compilation method that changes the original models into the final ones.

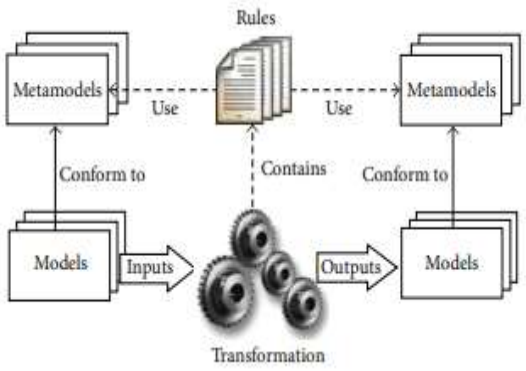

Figure 1: An overview of model transformations.

a simplified model to a complex one. As can be seen in Figure 1, in the case of an exogenous MT [33], the source and target models adhere to separate metamodels, but in the case of an endogenous transformation, the source and target models conform to the same metamodel. In most cases, only domain-specific ideas will be included in the introductory, high-level models, whereas technology notions will be presented naturally in the intermediate, lower-level ones. A metamodel transformation (MT) is predicated on a set of rules (declarative or imperative) that aid in the identification of ideas in a source metamodel for the purpose of creating enhanced concepts in a destination metamodel. By adding or altering rules, a new MT tailored to a new model may be created. One benefit of this method is that it enables the definition of many model transformations at the same abstraction level, each of which may be directed at a unique set of lower-level abstractions and therefore provide options to target alternative technological platforms. Both one-way (where only the source model is editable and the resulting target model is created automatically) and two-way (where the target model is also editable and the source model must be edited simultaneously) model transformations are possible. The second scenario raises the possibility of a model synchronization problem [34]. Model transformations may be specified using the OMG-proposed metaobject Facility (MOF) standard for metamodel expression and query/view/transformation (QVT) [35].

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:

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

where Ni is the total number of realizations of activity I or inactivity cycles for the component, and Ci is the unit cost of activity I or inactivity cycle. We used this technique to determine the power consumption profiles of the

CPU, cache, shared SRAM, and interconnection network that make up the backbone of the SoCLib library [38]. You can read everything about these consumption models in [5]. We achieved them by isolating the core operations of each component and measuring their unit costs using simple instruments. In the current work, we use the same power models. Activity counters are used to tally up how many times an event has occurred. Whenever the appropriate event happens in the simulation, the relevant counter is increased. Using this method, the whole architecture is used in each cycle. The energy dissipation of the design is tracked by sending the data from the activity counters to the energy consumption models included into the simulator. The technical specifications of each piece of hardware are reflected in the energy model included in the consumption simulator. This strategy is shown in Figure 3 If the IP's code is not available, then the counters cannot be used. Counters may be written into the source code of white-box IPs, but the activities of black-box IPs cannot be monitored until the source code is made public

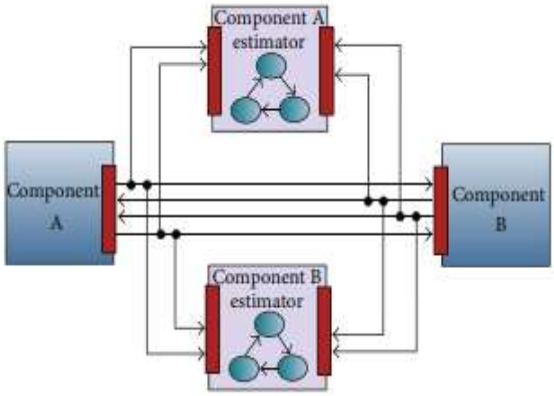

Figure 2: Power consumption estimators.

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

To prove the efficacy of our method, we calculated the resource needs of a 256x256 JPEG compression program. This program takes photos in the BMP format as input and produces images in the JPEG format as output. Several virtual implementations of the JPEG application were tested. These architectures rely on MIPSR3000 processors, 16 KB SRAM memory, and micro networks for their hardware implementations. There are two separate instruction and data caches in use, although they communicate with one another over the same interface. Directly mapped caches are just that. The data cache has a write-through policy. Even though all of the IPs below are white-box IPs (meaning their source code is easily readable), the same method may be used to analyse them as black-box IPs. As a result, we use independent estimators to incorporate the counters into the program. This allows us to adopt a white-box strategy for certain IPs in a simulated system while using a black-box method for the others. Figure 16 depicts one of the two-processor designs used in the simulations. The estimator requires a pair of interfaces, one for each direction of communication between the two halves. The cache memory in the Figure 16 example were developed using the white box methodology while the other IPs were developed using the black box methodology. The consumption of each component of an architecture was recorded in an XML file, which was fed into a reporting engine to produce a consumption report, allowing us to track the performance's development throughout the course of the simulation. Each component's share of total consumption across all simulation types is included in the report.

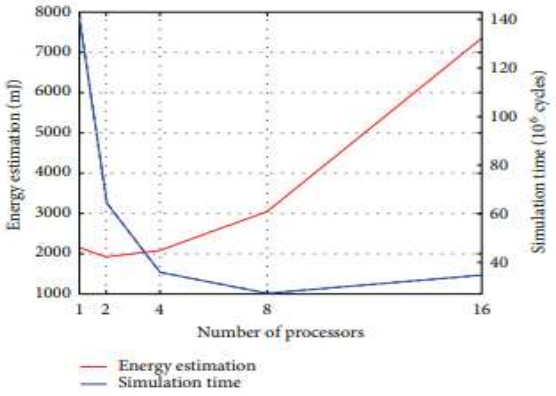

Figure 3: Performance and energy variation in terms of the number of processors.

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	489	2080.4
W	W	B	B	578	2080.4
W	B	W	W	687	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	504	2080.4
B	W	W	B	588	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

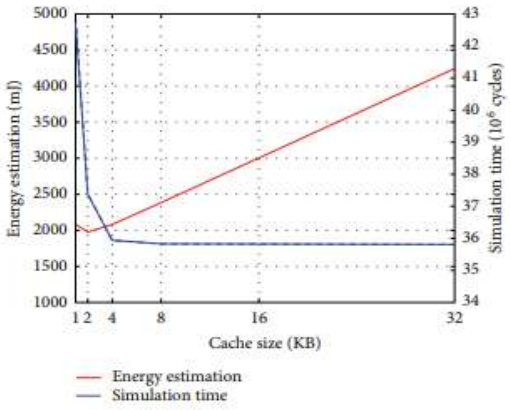

Figure 18: Variations in performance and energy consumption in terms of cache size with 4 processors.

These findings suggest that increasing the number of processors in a system might reduce execution time and hence boost overall performance. Due to resource sharing across the processes, there is non-linearity in the

variation; processes often cannot achieve the same goal at the same time, resulting in waiting cycles that degrade system performance. The overall energy consumption of a system reduces as the number of execution cycles is decreased, up to a certain number of processors, and then tends to stabilize as the system performance increases. However, above a certain threshold, increasing the number of processors often has little benefit, since it only introduces more conflicts at the interconnect, resulting in more waiting cycles, which modifies overall performances, notably in terms of power consumption. We utilized a 4-processor setup to analyse how changing the size of the instruction and data caches affected the overall system's performance and power consumption. We ran the JPEG-parallelized method with cache sizes ranging from 1 KB to 32 KB for both instructions and data. Figure 18 displays our findings. The larger the cache, the greater the system's total energy usage. The magnitude of the work at hand or the data being processed might affect how much of a difference a bigger cache makes to overall system performance. In our case, increasing the size of the cache from 4 KB to 8 KB increased energy usage by 14% but improved performance by 0.3%. There was no difference in performance between the 8 KB and 16 KB caches, although there was a 78% increase in power usage. We created a system with 4 processors and 4 KB of cache to show that both white-box and black-box techniques may be utilized concurrently for a system. We performed many simulations, each time changing how we estimated various factors. Table 1 displays the results of the 16 separate simulations that were run due to the presence of the 4 distinct kinds of components in this case (the CPU, the cache memory, the shared memory, and the connection). The simulation results reveal that for all possible permutations, the energy estimations are within a margin of error of less than 0.15 percent. Since the consumption of cache FIFOs is not detectable through the signals between the components of an architecture, this difference is attributable to the intrusive approach. However, if the FIFO's consumption is negligible, it can be ignored, and we may conclude that independent estimating modules allowed for reliable findings by accounting for the most important and resource-intensive processes carried out by the components under observation.

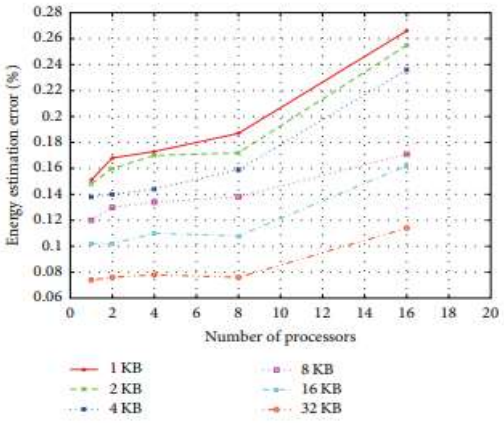

Figure 4: Variation in energy consumption estimation error in terms of cache size and number of processors.

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

Energy estimate for systems-on-chip (SoC) is presented, along with a hybrid method for doing so, in this study. Both white-box and black-box IPs may benefit from this method. When simulating a system with white-box IPs, our method involves inserting activity counters into the IP codes in order to identify the occurrences of the activities and to determine the consumption of these activities. Estimation modules are linked between black-box IPs to pick up on their actions through the signals they exchange while in simulation. These modules were created using an MDE strategy. In the Gaspard2 framework, the simulated systems were likewise automatically

produced from high-level models. We boosted the framework's deployment level to allow for the incorporation of energy estimates into the design flow. SystemC was used for the simulation's CABA implementation. Due to the reasonable number of architectural details that this abstraction level takes into consideration, we were able to run our simulations quickly while still getting quite accurate results. We employed a hybrid of white-box and black-box techniques for various setups to ensure the efficacy of our method. The simulation results demonstrated that accurate consumption estimates may be obtained even if an IP's source code is unavailable. To do this, we need simply identify the activity being performed and the associated consumption cost by analysing the signals it sends and receives from the other parts of a system. We want to extend our current work to accommodate systems with more moving parts, such as several CPUs and other kinds of interconnects, in future studies. Our future research may also take into account measurements of consumption from real-world implementations of the examined systems for direct comparison with simulation findings.

References

[1] R. Buchmann and A. Greiner, "A fully static scheduling approach for fast cycle accurate systemc simulation of MPSoCs," in Proceedings of the International Conference on Microelectronics (ICM '07), pp. 101–104, Cairo, Egypt, 2007.

[2] OMG, "Portal of the model driven engineering community," http://www.planetmde.org/.

[3] OMG, UML Profile for MARTE: Modeling and Analysis of RealTime Embedded Systems, 2009.

[4] DaRT, "Gaspard2 framework," 2009, http://www.gaspard2 .org/.

[5] R. Ben Atitallah, S. Niar, A. Greiner, S. Meftali, and J. L. Dekeyser, "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.

[6] M. Andersson and P. Kuivalainen, "Spice macromodel for power dmos transistors," in Proceedings of the Nordic Semiconductor Meeting, Finland, 1992.

[7] Synopsys, "Synopsys low power solutions for asic design flow," Tech. Rep., Synopsys, 1998, http://vada.skku.ac.kr/ClassInfo/ic/lowpower/.

[8] C. Rowen, "Reducing SoC simulation and development time," IEEE Computer, vol. 35, no. 12, pp. 29–34, 2002.

[9] J. Costa, J. Monteiro, L. M. Silveira, and S. Devadas, "A probabilistic approach for rt-level power modeling," in Proceedings of the 16th IEEE International Conference on Electronics, Circuits and Systems, Cyprus, 1999.

[10] Q. Wu, Q. Qiu, M. Pedram, and C. S. Ding, "Cycle-accurate macro-models for RT-level power analysis," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 6, no. 4, pp. 520–528, 1998.

[11] R. P. Llopis and K. Goossens, "Petrol approach to highlevel power estimation," in Proceedings of the International Symposium on Low Power Electronics and Design, pp. 130–132, August 1998