Integration of Power Saving Techniques in the UNISIM Simulation Framework through the Shadow Module design paradigm

Daniele Ludovici[1], Georgios Keramidas[2],
Georgi N. Gaydadjiev[1], and Stefanos Kaxiras[2]

[1] Delft University of Technology, Computer Engineering Lab, The Netherlands
{D.Ludovici,G.N.Gaydadjiev}@tudelft.nl
[2] University of Patras, Dept. of Electrical and Computer Engineering, Greece
{keramidas,kaxiras}@ee.upatras.gr

Abstract. Performance is no longer the only metric dominating modern system-level design. Power is emerging as major constraint to consider during system development. Consequently, several CAD tools with power estimation capabilities in addition to performance figures have been developed. Among these tools, UNISIM is a simulation framework that enables the architects to rapidly design a new system at different levels of granularity in systemC. It allows quick design space exploration and offers cycle accurate performance evaluation. We propose the *shadow module*, a new design paradigm for UNISIM that enables the framework to account for energy estimation. In this respect, we discuss the implementation of various cache power saving techniques. Experimental results utilizing realistic benchmarks show that power and performance figures can be easily identified with our enhanced analysis framework.

1 Introduction

Recent years have seen proliferation of cell phones and other mobile equipment, such as palmtops and laptops, in various aspects of our daily life. A common requisite for portable devices is a lengthy battery lifetime along with high performance due to more and more sophisticated applications. Furthermore, key requisite for a successful product is a limited design-time which consequently turns out in a shorter time-to-market. Advanced design tools play a prime role given the intrinsic complexity of the problem. The chance to quickly evaluate different options, at early stages of the design, represents a key factor that avoids incurring in erroneous design choices. Such choice would imply extra design cycles with a consequent additional cost in terms of development effort, money and design time. For this reason, considerable effort has been dedicated to develop tools for high level system design and simulation. Moreover, along with fast system development, these tools enable rapid power and performance estimation. UNISIM [1] is an example of framework for developing simulators at different levels of abstraction.

In this paper we present a new design paradigm for UNISIM, which enables clean separation between software describing system implementation and power estimation modeling code. Moreover, we discuss the integration of a power reduction technique, i.e. the MRU way predictor, showing its ease of development through the shadow module paradigm. Furthermore, we present results of power consumption as well as execution cycles for a normal cache utilization as well as sequential [2], phased [3] and MRU cache [4]. The rest of the paper is organized as follows. Section 2 gives an overview regarding CAD power estimation

tools. In Section 3 the UNISIM framework is presented and our improvements are discussed in Section 4. Section 5 presents the implementation of a MRU way predictor utilizing the shadow module design paradigm. and points out simulation results. Finally, Section 6 summarizes conclusions.

2 Related Work

In the Computer-Aided Design landscape, numerous power estimation tools working at different abstraction levels can be found. Transistor level simulators like SPICE [5] and PowerMill [6] enable accurate power consumption estimation requiring both a substantial development effort and rather long simulation times though. Such limitations pressed for new simulation infrastructures such as XTREM [7] and SimplePower [8] with higher abstraction level and shorter time to obtain accurate results. Further examples at this level are constituted by Wattch [9] and Hotleakage [10] that have contributed significantly to low-power research. Recently, many efforts of design space exploration infrastructures at system-level have been developed and an example is represented by the Sesame project [11]. Furthermore, the Asim project [12], along with its modular capabilities, constitutes a framework for creating performance models to be used in simulation infrastructures. Our work extends the UNISIM framework with power estimation capabilities. The inherent modularity and re-usability of UNISIM, along with the new power features enable designers to evaluate the impact that architectural modifications entail in terms of both performance and power.

3 The UNISIM Framework

UNISIM [1] is a simulation framework for developing simulators at different level of detail (transaction-level, cycle accurate, and more). In the development of a new simulator for an architecture, each hardware block is modeled as a software module. The entire system can be seen as composition of multiple hardware blocks mapped into different UNISIM modules. Modules communicate through interfaces. Moreover, UNISIM proposes a novel approach to facilitate code reusability. Commonly, most of the simulator code is devoted to model hardware control. The control code implemented within a module describes its interactions with other modules. Therefore, a module depends upon the one it communicates with. As result, control code needs to be rewritten if a module is replaced, thus decreasing re-usability. To address this issue, UNISIM defines a standard communication protocol that characterizes interaction between modules and hides this mechanism in the module communication interfaces of each module. Therefore, if a module implementing a particular cache strategy is replaced by another, no modifications are needed because the control part (and consequently the interaction with the other module) is hidden into the standardized interfaces of the modules itself. Another interesting aspect of the UNISIM framework is the possibility to easily attach an external *service* to perform a particular estimation (e.g., area, timing, power). The service (e.g., area estimator) is implemented as a module and is connected using a dedicated interface to the module that is being monitored (e.g., a cache). These modules communicate according to a client-server paradigm, i.e., the client must provide the server with the necessary information to enable the desired estimation. For instance, a cache module supplies the area estimator module with the number of cache lines, line widths in order to estimate area occupation for such memory structure.

4 Shadow Module: A New Design Paradigm for UNISIM

In order to build a modular and re-usable framework, different functionalities must be kept separate. The *shadow module* is a new design paradigm for the UNISIM framework. It enables the division between behavioral-code and power simulation-code. In Fig. 1 an example is shown. The shadow module analyzes ingress/egress traffic and collects statistics connecting to the communication channel (1). Whenever an operation is correctly executed by the ALU, the shadow module is notified using a dedicated interface (2). Power estimation is performed according to a configuration selected from the XML file (3) that determines the entry to lookup in the power characterization table (4). As previ-

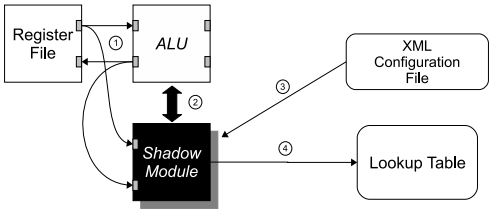
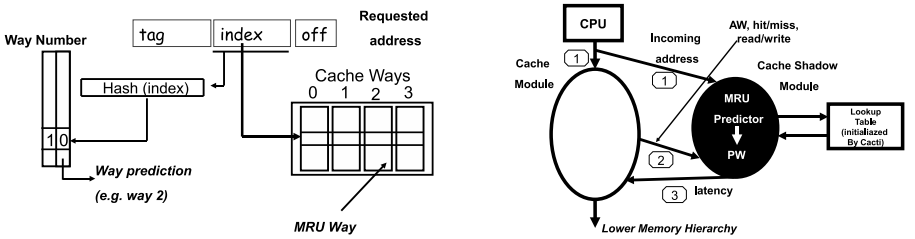


Fig. 1. The shadow module mechanism.

ously mentioned in Section 3, a hardware block in UNISIM is modeled through a software module and different modules are interconnected using communication channels. Channels are realized utilizing ports which permit data exchange according to a protocol defined by the framework. The shadow module is connected to input ports of the monitored module. Therefore, it can analyze ingress/egress traffic of the module that is shading and collect statistics (e.g., switching activity) to be utilized for power estimation. Shadow module and its counterpart communicate directly through a dedicated interface meant for synchronization. The necessity stems from the fact that the shadow module needs to be notified whenever an operation has been performed. The UNISIM communication protocol has been designed in such a way that modules have to perform a handshake using three wires: *data*, *accept* and *enable* in order to exchange data. Indeed, data could be sent many times if the target module does not accept it. In the example shown in Fig. 1, the ALU could be busy for various reasons (e.g., buffer full, more cycles to complete an operation, etc.) and thus not ready to accept data from the register file that would keep sending as long as succeed. From the shadow module point of view, this could bring to an erroneous evaluation because it has to collect statistics only when an operation has been really performed. Another interesting characteristic of this new design paradigm is the possibility to easily speed up design space exploration. The shadow module has been designed in order to select different configurations for the hardware block under investigation (e.g., for an arithmetic logic unit: implementation type, targeted technology, etc.). Different configurations correspond to different lookup table entries with their specific power characterization values. The configuration can be selected by modifying the XML file. Once the setup has been tailored, power consumption results for a different technology implementation of the same hardware unit can easily be acquired. The proposed approach allows rapid evaluation of various design choices permitting the selection of the most appropriate according to the power requirements to meet. Additionally, the shadow module has been envisioned also to support power optimization techniques such as MRU cache, sequential cache and phased cache, etc.

5 Power Saving Techniques Integration

In this section, the implementation through the shadow module design paradigm of typical cache level power reduction techniques (i.e., sequential cache, phased cache, and MRU cache) will be discussed. The aforementioned methods are the most popular for dynamic power reduction at the cache level. The underlying idea of all these techniques is to avoid (as much as possible) the associative search of a N-way set associative cache. To test and assess the versatility of the shadow model paradigm, we evaluated those power reduction techniques in the UNISIM framework without changing (or by not heavily change) the vanilla implementation of the system caches. Each of the above methods is implemented as a shadow module which monitors the ingress/egress traffic of the normal cache. However, compared to the previous example (ALU shadow model) two basic changes are needed. We will discuss those changes by employing the MRU way prediction technique. This technique aims at predicting a single way where a new reference might hit. A correct prediction results in a fast hit (the cache behaves as a direct map cache) and yields power benefits roughly proportional to the associativity (a single way out of N is accessed). On the other hand, a way misprediction results in no power savings and a slow hit, because the rest of the ways need to be searched and accessed. The MRU way predictor, that resides inside the shadow module, is implemented as a array of $log_2(N)$ bit binary numbers. The array has exactly the same number of sets as the cache and it is indexed by the same portion of the address that is used for cache indexing. Every entry of this array contains only the most recently accessed way of the corresponding cache set (this way number is used for the next prediction). Fig. 2(a) shows the general view of the way prediction scheme for a 4-way set associative cache. Compared to the ALU shadow module, a few changes are needed



(a) Way prediction scheme

(b) MRU Predictor implemented as Shadow Module

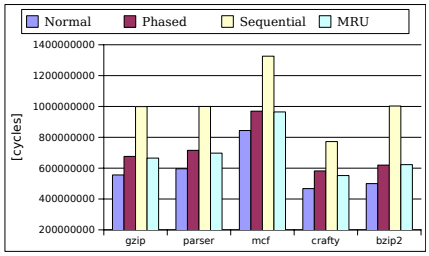
Fig. 2. The way prediction scheme and its implementation with the shadow module.

in order to accurately model this cache level power reduction technique. First of all, as showed for the ALU example, the cache shadow module is connected to the input/output ports of the monitored cache (shown in Fig. 2(b)). Therefore, the cache shadow model can snoop the next memory reference (from the import port) that is forwarded to the cache. A hash of this reference is used to index the way prediction array (this array is implemented in the cache shadow module). The indexed array entry contains the predicted way (from now PW). The behavioral code of the normal cache will perform the associative search in the cache array (no changes are needed so far in the normal cache code). At this point, the cache module and its shadow counterpart should communicate each other utilizing the services provided by the Unisim framework. The cache

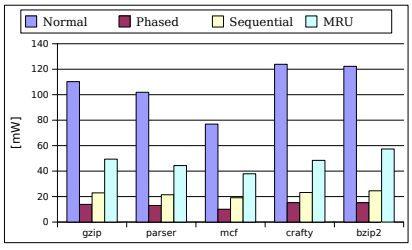
module will sent the hit/miss and the read/write information along with a way number (from now AW) to the shadow module. The AW way can be either the way where a hit just happened, or the way that is selected for replacement (in case of a miss). The cache shadow model will use this information to decide the latency of the current cache access (the information will be sent back to the normal cache module), to update the way predictor and to calculate the dynamic energy consumed by the cache during the current access. All the previous operations should happen for every new memory reference. The communication sequence is depicted in Fig. 2(b). The next step is to calculate the energy consumption of each cache access by consulting the power characterization lookup table initialized by Cacti [13]. For all the power reduction techniques, the cache shadow module is responsible to calculate the number of the ways (tag and data) accessed every time a new reference is inserted into the cache and account for the relative power consumption.

5.1 Experimental Results

In this section, we present both performance and power estimation results for a set of 5 benchmarks of the SPEC2000 suite. The benchmarks were compiled using the cross-compiling tool provided by the UNISIM project. For each benchmark, we skip the first 1Byte of instructions to avoid unrepresentative startup behavior and then we simulate 100MByte committed instructions using the reference inputs. The processor modeled is a PowerPC 405. The main cache module reflects a 64K, 8 way, write back, blocking, dual port L1 cache memory. In our implementation, we assume that the way predictor access proceeds in parallel with the cache decoder (the first pipeline stage of the cache), thus we do not account for the extra delay introduced by accessing the way predictor. The assumption has been verified with Cacti which shows that accessing a small predictor structure of few kilobits (way predictor array) is comparable in terms of latency to cache decode. Additionally, using Cacti we verify that the energy for accessing the way predictor is less than 0.5% of a full L1 access, consequently this extra energy is not considered. Furthermore, we assume a 2 cycles pipelined L1 cache (1 cycle for the decoder and 1 cycle for accessing the memory array, the sens. amps., the output driver, etc.) and we account for an extra cycle in order to quantify the negative impact of re-accessing the cache. Fig. 3(a) shows the results in terms of execution cycles, while Fig. 3(b) graph presents the power consumed by the L1 cache measured in mWatts.



(a) Execution cycle for several benchmarks deploying different power saving strategies.



(b) Total cache power consumption with different power saving techniques.

Fig. 3.

The obtained results provide evidence of the correctness of the implemented methods with respect to prior work and prove that the proposed shadow model approach can be easily utilized to better understand which power saving technique best meets the energy budget of the underdevelopment architecture.

6 Conclusions

In this paper we presented an augmented version of the UNISIM framework from the power perspective. We illustrated a new design paradigm describing the *shadow module* mechanism. Moreover, we showed an application describing the ease of implementation of classical power saving techniques leveraging the presented approach. Experimental results point out the effectiveness of the shadow module paradigm: useful insights concerning the power consumption along with performance figures can be easily gained with an affordable coding effort.

References

1. August, D., Chang, J., Girbal, S., Gracia-Perez, D., Mouchard, G., Penry, D., Temam, O., Vachharajani, N.: UNISIM: An Open Simulation Environment and Library for Complex Architecture Design and Collaborative Development. IEEE Computer Architecture Letters (CAL) **6**(2) (September 2007)
2. Kessler, R.E., Jooss, R., Lebeck, A., Hill, M.D.: Inexpensive implementations of set-associativity. SIGARCH Comput. Archit. News **17**(3) (1989) 131–139
3. Hasegawa, A., Kawasaki, I., Yamada, K., Yoshioka, S., Kawasaki, S., Biswas, P.: Sh3: High code density, low power. IEEE Micro **15**(6) (December 1995)
4. Inoue, K., Ishihara, T., Murakami, K.: Way-predicting set-associative cache for high performance and low energy consumption. In: Proceedings of the International Symposium on Low Power Electronics and Design. (1999) 273–275
5. Quarles, T., Pederson, D., Newton, R., Sangiovanni-Vincentelli, A., Wayne, C.: SPICE, Simulation Program with Integrated Circuit Emphasis `http://bwrc.eecs.berkeley.edu/Classes/IcBook/SPICE/`.
6. Huang, C.X., Zhang, B., Deng, A.C., Swirski, B.: The Design and Implementation of PowerMill. In: Proceedings of the 1995 International Symposium on Low Power Design (ISLPED '95), Dana Point, California, USA (1995) 105–110
7. Contreras, G., Martonosi, M., Peng, J., Ju, R., Lueh, G.Y.: XTREM: A Power Simulator for the Intel XScale Core. In: Proceedings of the 2004 ACM SIGPLAN/SIGBED conference on Languages, Compilers, and Tools for Embedded Systems (LCTES '04), Washington, DC, USA (2004) 115–125
8. Ye, W., Vijaykrishnan, N., Kandemir, M., Irwin, M.J.: The Design and Use of SimplePower: A Cycle-Accurate Energy Estimation Tool. In: Proceedings of the 37th Design Automation Conference (DAC-2000), Los Angeles, California, USA (2000) 340–345
9. Brooks, D., Tiwari, V., Martonosi, M.: Wattch: A Framework for Architectural-Level Power Analysis and Optimizations. In: Proceedings of the 27th International Symposium on Computer Architecture (ISCA 2000), Vancouver, Canada (2000) 83–94
10. Zhang, Y., Parikh, D., Sankaranarayanan, K., Skadron, K., Stan, M.: Hotleakage: A Temperature-aware Model of Subthreshold and Gate Leakage for Architects. Technical Report CS-2003-05, University of Virginia, Dept. of Computer Science (March 2003)
11. Erbas, C., Pimentel, A.D., Thompson, M., Polstra, S.: A framework for system-level modeling and simulation of embedded systems architectures. EURASIP J. Embedded Syst. **2007**(1) (2007)
12. Emer, J., Ahuja, P., Borch, E., Klauser, A., Luk, C.K., Manne, S., Mukherjee, S.S., Patil, H., Wallace, S., Binkert, N., Espasa, R., Juan, T.: Asim: A performance model framework. Computer **35**(2) (2002) 68–76
13. Reinman, G., Jouppi, N.P.: Cacti 2.0: An Integrated Cache Timing and Power Model. Technical report, HP Labs (2007)