

Energy simulation of embedded XScale systems with XEEMU

Zoltán Herczeg^a, Daniel Schmidt^{b,*}, Ákos Kiss^a, Norbert Wehn^b and Tibor Gyimóthy^a

^aUniversity of Szeged, Department of Software Engineering, Árpádtér 2, H-6720 Szeged, Hungary

^bUniversity of Kaiserslautern, Department of Electrical Engineering, Erwin-Schrödinger-Straße, D-67663 Kaiserslautern, Germany

Abstract. Energy efficiency is key in embedded system design. Understanding the complex issue of software power consumption in early design phases is of extreme importance to make the right design decisions. Here, not only the CPU but also the external memory plays a very important role. Power simulators offer flexibility and allow a detailed view on the sources of power consumption. However, many simulators lack accuracy and focus only on the CPU core without considering the memory subsystem. In this paper, we present XEEMU, a fast, cycle-accurate simulator, which aims at accurately simulating the power consumption of an XScale-based system including its memory subsystem. It has been validated using measurements on real hardware and shows a high accuracy for runtime, instantaneous power, and total energy consumption estimation. The average error is as low as 3.0% and 1.6% for runtime and CPU energy consumption estimation, respectively.

Keywords: XScale, power dissipation, energy consumption, simulation, SDRAM, DVFS

1. Introduction

Due to advances in microelectronics and communication technology, complex information processing can be efficiently embedded in an increasing range of portable products like smart mobile phones, and PDAs [3]. Low cost, energy efficiency and fast time to market are the top requirements in embedded system design. Typical portable appliances are microprocessor-centric architectures. Power and energy optimization of hardware is a well investigated discipline. Hence, modern microprocessor architectures are equipped with many techniques for minimizing energy and power at design and run-time, like clock gating, power supply and frequency scaling, leakage control, etc. But the actual power and energy consumption of a microprocessor system is finally determined by the application running on it. Thus, the interrelation between the software and hardware w.r.t. power/energy

consumption is very important and its optimization is a hot research topic, e.g. [14].

Understanding the complex issue of software power consumption in combination with the underlying processor system in early design phases of embedded systems is of extreme importance to make the right design decisions. Dynamic voltage and frequency scaling (DVFS) has emerged as the standard technique for software-based energy reduction. However, often the hardware models underlying the proposed DVFS schemes are based on simplifications that do not reflect the real energy consumption and the complete CPU system, which consists of more than just the CPU core.

Several possibilities exist to evaluate the energy and power consumption of such a system, the most obvious being measurements on real hardware. This is seldom possible, however, as often the platform is not fully defined in the early phases of embedded systems design. Hardware measurements on a specific platform do not allow to evaluate the effect of different architectural parameters, like different processors, cache models and sizes, as well as the effects of different system

*Corresponding author. Tel.: +49 (0) 631 205 3513; Fax: +49 (0) 631 205 4437; E-mail: schmidt@eit.uni-kl.de.

parameters, especially the size and type of the memory system.

Power simulators offer more flexibility and overcome the problem of noise and side-effects that influence measurements. With simulation tools it is possible to gather large amounts of data automatically. They also allow a much more detailed view on the sources of power consumption than power measurement. Hence, the availability of *accurate* power simulation tools covering a complete CPU system is key to energy efficient embedded software design.

In this paper we present XEEMU, the XScale energy emulator, a fast, cycle-accurate simulator for the XScale architecture [6], one of the most widespread, low-level RISC architectures in the embedded domain. In contrast to many other existing power simulators, as [5, 16], which simulate power and performance of theoretical microprocessor architectures, XEEMU aims at the most accurate modeling of the XScale architecture possible, trading off flexibility for much more reliable results.

XEEMU proves to be more accurate than the well-known and freely available XScale simulator XTREM [7] in terms of runtime and CPU energy estimation, due to its improved pipeline and power model and the cycle-accurate simulation of the SDRAM subsystem. More importantly, it also allows the investigation of memory energy consumption, which makes it a unique tool for the evaluation of embedded CPU system energy optimization schemes. It offers a high flexibility through freely and independently configurable frequencies for the core clock and the memory. Nonetheless, XEEMU reaches more than twice the simulation speed compared to XTREM.

The methodology we used to create XEEMU is an iterative two-step process, combining measurements and performance and energy modeling. As the total energy consumption is heavily dependent on the runtime of a program, it is mandatory to model the behavior cycle-accurate first, before in the second step the power model is created. The creation of both the power and the behavioral pipeline model is done iteratively with a validation against measurements on an evaluation board at the end of each iteration. Where publicly available documentation is not sufficient for the creation of models, synthetic benchmarks that stress one certain effect are used to isolate and test individual parts of a model. The refinement of a model is done in two ways: extension and correction if the observed effects are not considered in the model, or parameter fitting when the effects are considered but over- or underestimated.

The rest of the paper is organized as follows: Section 2 explains the different possible abstraction levels for simulators and the differences between CPU and system simulation. Section 3 investigates how system energy consumption of a real system running at different voltage levels and frequencies can behave drastically different from common models. This motivates the need for a system simulator that takes into account the energy consumed not only by the CPU but also by the memory. Related work on energy models and power simulators is presented in Section 4. In Section 5, we describe how XEEMU was developed from an existing simulator by correcting the identified problems and including behavioral and power models for a configurable memory subsystem. Simulation results on runtime and power simulation are presented in Section 6. Finally, Section 7 concludes the gathered experiences and gives directions for future work.

2. Background

2.1. Abstraction levels

The simulation of the behavior as well as the power dissipation of microprocessors can be done on various abstraction levels, namely circuit level, architectural level, and instruction level.

While general circuit simulators like Spice [17] offer very high accuracy, they require a precise description of the hardware on transistor level. Due to the complexity of a microprocessor, this yields in unacceptably low simulation speed. Moreover, the circuit-level layout of a processor is usually not available to the public.

Much greater simulation speeds can be achieved at the instruction level. Tiwari et al. proposed a methodology [19,20] to characterize the power consumption of individual instructions by hardware measurements. It requires measurement of benchmarks for every single instruction as well as additional benchmarks for inter-instruction dependencies and data dependencies. The accuracy of this approach is limited by the fact that in the highly optimized pipelines of today's microprocessors execution times of instructions may vary strongly. This is especially true for external memory accesses, but also for stalls due to mispredicted branches.

Hence, to model exact, cycle-accurate behavior of a pipeline, simulation on the micro-architectural level is mandatory. The simulation speed of these kinds of simulators is still much higher than that of a circuit simulator. Additionally, it is not necessary to know the circuit-

level layout or the specifics of the production process to get meaningful results for the power simulation. Also, the necessary information about the number and functionality of the pipeline stages of a microprocessor is usually publicly available. Hence, power models and behavioral models can be created much quicker on this abstraction level. It offers the best trade-off of modeling effort, simulation speed, and simulation accuracy, both for simulation of runtime behavior as well as power consumption of a microprocessor.

2.2. Scope

The scope of the simulation is another important issue. It can range from pure microprocessor core simulation to full system simulation.

Many simulators focus mainly on the CPU neglecting those effects that arise from communication between CPU and memory, or CPU and peripheral devices. Often, the latency for external memory accesses is considered a constant number of clock cycles. However, external memory is typically based on SDRAMs which show large variations in access latency depending on the current access. Hence, this constant latency model is not only a false assumption but it also makes it hard to accurately assess the effects of frequency scaling. As in today's embedded systems, where CPU, memory, and other peripherals are often clocked at independent frequencies, changing the operating frequency of the CPU will result in a different memory access latency. Thus, a computationally intensive task will benefit more from an increased CPU frequency, while the runtime of a memory-bound process will not increase linearly with a higher CPU frequency.

Another, perhaps even more important issue is the simulation of the power consumption of the other components of a system. Many schemes for the dynamic adaptation of operating frequency and voltage have been proposed to reduce the energy consumption for processing a given task or set of tasks, e.g. [9]. The basic idea is based on the well-known formula which relates the switching power P of a CMOS circuit to the operating voltage V and the clock frequency, according to

$$P \propto fV^2. \quad (1)$$

Under the additional assumption that the execution time is inversely proportional to the frequency, the total energy E for the task is then

$$E \propto V^2. \quad (2)$$

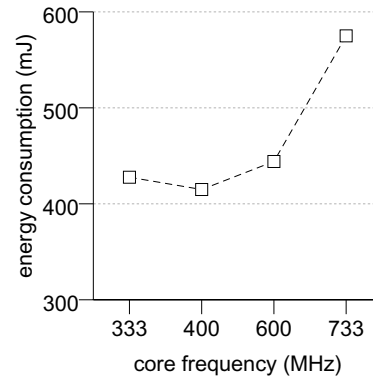


Fig. 1. Measured CPU energy consumption for a computation dominant benchmark.

However, as already mentioned above, the assumption of an inverse proportional relationship between execution time and frequency does not hold for today's embedded systems. Additionally, Eqs (1) and (2) completely neglect static and leakage power consumption as well as the portion of the power consumption accounted for by the other system components. However, in today's embedded systems the CPU power and memory power are more or less on the same level [18].

3. Motivation

The importance of taking into account all components of a system rather than focusing on a CPU can clearly be seen, when studying the energy needed for the completion of a task at different voltage levels and frequencies. For our tests, we used an ADI 80200EVB evaluation board [2], which is equipped with the Intel XScale 80200 processor [12] and 32 MB SDRAM. The processor can be clocked at different CCLK frequencies, ranging from 333 MHz to 733 MHz in 66 MHz steps. The clock frequency (MCLK) of the front-side bus and the 32 MB of SDRAM is 100 MHz. The board was modified such that the core CPU voltage could be varied in three steps: 1.215 V (for up to 400 MHz), 1.32 V (for up to 600 MHz), and 1.5 V (for up to 733 MHz). For all measurements presented, the core voltage is always set to the lowest possible value for the according CPU frequency.

A computation dominant benchmark (vam), which fits completely into the XScale's caches serves as an example used throughout this chapter. In Fig. 1, the measured CPU energy consumption for this benchmark at different core frequencies are compared. As men-

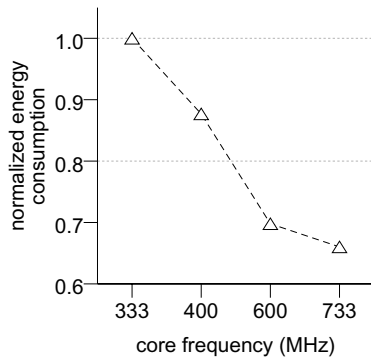


Fig. 2. Measured system energy consumption for a computation dominant benchmark.

tioned, for every frequency the lowest of the three possible core voltages was chosen, the points at 333 MHz and at 400 MHz were both obtained at 1.215 V. It can be seen that the higher of the two frequencies leads to a lower energy consumption, in contradiction to the assumption that leads to Eq. (2). The reason is the non-negligible static power consumption of the XScale CPU, which is independent from the frequency but of course not from the runtime of the benchmark.

In Fig. 2, the normalized energy consumption of the *complete* system is shown as a function of the CPU frequency. The memory and other system components are not affected by voltage and frequency scaling. While the conclusion from Eq. (1), that lower operating voltages lead to lower energy consumption, holds to some extent for the CPU energy consumption, it is rendered completely invalid for the system energy consumption. In complete contradiction to this, the lowest energy is consumed at the highest frequency and, thus, at the highest possible CPU voltage. This is only a surprising result at first sight. At higher frequencies the completion of the task takes less time. Thus, the energy consumed by those components of the system that do not benefit from the voltage scaling of the CPU, is reduced. These savings outweigh the penalty introduced by the higher CPU energy consumption.

As embedded systems usually can not be completely turned off after having completed a task, it is a more fair comparison to switch the system to idle mode after computation has finished. Let us assume we have a periodic task with a period just equal to the execution time at 333 MHz. Then the energy consumed in one period equals the energy consumed during the computation plus the energy consumed in idle mode, waiting for the end of the period. Figure 3 shows the energy consumption during one period which is normalized to

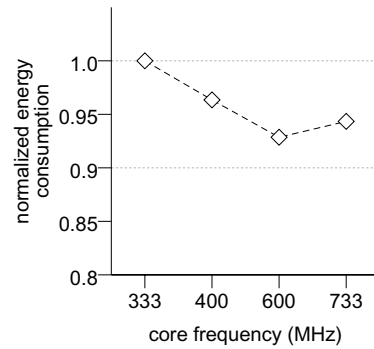


Fig. 3. Measured system energy consumption, padded to equal execution time for all frequencies.

the execution time of the task at 333 MHz. At 333 MHz, of course, the idle time is 0. In this case it can be seen that the energy optimal voltage and frequency setting is neither the lowest, nor the highest.

It is evident that DVS schemes based on the unrealistic assumptions of Eqs (1) and (2) may lead to higher energy consumptions rather than to lower energy consumptions. It can be seen that the energy optimal voltage and frequency setting for the system is different than that for just the CPU. It will usually also depend on the type of benchmark (i.e., limited by the speed of the memory or limited by the CPU), on the number and type of system components (especially the size and type of memory), and on the energy consumption in idle state.

Results similar to these have also been found by Miyoshi et al. [15] and by Snowdon et al. [18]. Snowdon et al. designed an XScale-based board specifically for the purpose of evaluating the energy consumption of CPU and memory under various voltage and frequency settings.

These observations show the need for the joint simulation of the energy consumption of the CPU and a memory subsystem, which is the most common denominator of every embedded system. However, still today, most simulators focus on the CPU only. Hence, we decided to develop a system-level power simulator for the XScale platform that is able to accurately reproduce the measurements on a realistic system. The goals for this simulator were:

- cycle-accurate simulation of the XScale core,
- cycle-accurate simulation of the SDRAM subsystem,
- accurate simulation of two independent clock domains for memory and CPU,
- a configurable memory system layout,

- a precise power model for the CPU, capable of accurate simulation of DVFS, and
- a precise power model for the simulation of the memory subsystem.

These design goals led to the development of XEEMU, which was first presented in [11].

4. Related work

A lot of related work exists in the field of processor simulation. The most wide spread of these are based on SimpleScalar [4].

SimpleScalar is an open source, configurable, generic processor core simulator. It works on the micro-architectural level and, thus, is capable of cycle-accurate pipeline simulation covering all internal effects, like stalling. It offers high flexibility and enables designers to evaluate architectural optimizations. SimplePower [23] extends SimpleScalar with power estimation functions for each of the pipeline stages. In every simulated clock cycle the functions calculate the power consumption for every functional unit, based on analytical power models and look-up-tables. Because the pipeline organization differs heavily from that of the XScale architecture, results are not comparable.

Sim-Panalyzer [16] (formerly named PowerAnalyzer) is a power estimation tool based on SimpleScalar. It interprets ARM instructions, just like XScale processors, but as it does not alter the generic simulation core it still cannot be used to simulate the XScale core accurately. The memory subsystem is not simulated by this tool.

Wattch [5] is a parameterized power model of common structures present in modern microprocessors, which has also been used to extend SimpleScalar, but it is flexible and can also be integrated into other architectural simulators. It is based on mathematical formulas but focuses only on dynamic power consumption, completely omitting the growing effect of leakage.

Based on the Wattch power model, Contreras et al. created XTREM [7] an architectural power simulator tailored for the XScale core. It differs from the previous mentioned works in that it focuses on one specific architecture aiming at a more accurate power and performance simulation at the cost of less architectural flexibility. As the implementation of XEEMU is based on XTREM, we will investigate this tool in more detail in the following sections.

Miyoshi et al. [15] were among the first who studied the power and energy consumption of complete com-

puter systems rather than focusing on the CPU of the system, only. In their experiments on two different platforms, they found that the lowest-performance setting does not necessarily optimize the energy consumption for a given task. They also propose a methodology to find the optimal setting.

In [18], Snowdon et al. present measured energy consumption data of an XScale-based platform with 64 MB of SDRAM. They show that at lower CPU frequencies, memory power is the dominating factor in this system. They also show results similar to those presented in Section 3 contradicting common models of system energy consumption.

Fan et al. [10] evaluated the synergy of low-power states available in mobile SDRAM and DVFS. For their experiments they extended the well-known simulator PowerAnalyzer with an energy consumption model of mobile SDRAM. This extended simulator however, was never made available to the public. As mentioned earlier, PowerAnalyzer and its successors do not accurately model a real existing CPU, and in this work only one specific memory configuration was examined.

In [21], Wang et al. presented a configurable DRAM simulator, DRAMsim. Just like XEEMU's memory power model, it is based on the work by Janzen [13]. While DRAMsim is a stand alone simulator for the energy consumption of memories it offers an interface for integration into existing processor simulators. However, there are no system simulators available on the web that integrate DRAMsim to enable system simulation.

5. XEEMU

When we started the work on our system-level power simulator, we used the most accurate power and performance simulator available for XScale at that time, i.e., XTREM as basis. However, XTREM did not meet our requirements: it was not able to perform system simulation but core simulation only, the power model was not capable of accurately simulating voltage and frequency scaling, and even the core simulation showed some inaccuracies when compared to real hardware measurements. To get more accurate results both at core and systems levels, we analyzed the behavioral and the power models of XTREM to find the sources of inaccuracies.

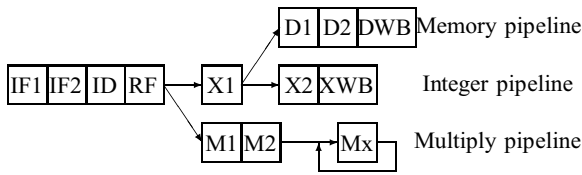


Fig. 4. The pipeline of XScale.

5.1. Behavioral model

It is very well-known that there is a strong correlation between runtime and energy consumption. Thus, cycle-accurate simulation is necessary before starting to create a power model. Therefore, we first validated the runtime model of XTREM.

An in-depth investigation of XTREM shows a lot of inaccuracies in the modeling of the caches and the pipeline, which is shown in Fig. 4. The problem with the cache handling of XTREM is that, while the XScale processors support both write-through and write-back policies, it supports the write-through policy only, and even that is handled incorrectly. Moreover, XTREM does not simulate the fetch buffers between the caches and the main memory, which are used in the processor. As for the pipeline, in the Instruction Fetch 1 (IF1) stage XTREM uses a branch target buffer (BTB) of different size and indexing algorithm for predicting branches than the true XScale processor. XTREM also does not support queuing of fetched instructions between the pipeline stages IF2 and Instruction Decode (ID), while the queue is used in the XScale architecture so that IF1 and IF2 can still operate when later stages of the pipeline are stalling. In XTREM, some functionalities are implemented in the Execute 1 (X1) stage, while they reside in earlier stages in XScale: the flow generator, which translates complex, CISC-like instructions (like block data transfers) to micro operations (μops) works in ID, and the shifter unit is in the Register File (RF) stage. These differences cause improper stalling behavior. Even more important is that the operation of XTREM's flow generator is incorrect as well, since it generates almost twice as much μops for data transfer instructions as XScale in reality.

We investigated not only the above mentioned core functionalities, but the implementation of the memory subsystem as well. We found that the memory clock in XTREM is not configurable, but instead it assumes a constant memory access latency. As already discussed in Section 2.2, this assumption is invalid. Depending on the state of the memory controller and the memory banks, and on the clock frequency of the memory sub-

system, memory access times may vary heavily. Using the processor's internal performance counters, we observed cache miss costs varying from 78 to 126 CCLK cycles at 600 MHz CCLK and 100 MHz MCLK.

However, setting CCLK to 600 MHz in XTREM and configuring the memory latency to 78 CCLK cycles yields a runtime overestimation for all test programs (see Section 6), by more than a factor of 2 for the memory intensive `pnm2png` benchmark. The optimal correlation of benchmark runtimes, as simulated by XTREM and measured on the processor, was found with a fixed memory access latency of only 18 CCLK cycles, as shown in Fig. 5. Obviously, this number is much smaller than the measured memory latency of at least 78 CCLK cycles. This is due to the fact that XTREM does not consider that by pipelining up to 4 outstanding memory requests the memory latency can be partially hidden, thus preventing the XScale pipeline to stall in reality. To counterbalance this effect, the memory latency in XTREM has to be much smaller. In the synthetic benchmark (`cache_miss`), which was designed to produce a high number of cache misses, latency hiding is less effective, yielding in a higher average stall rate. Here, XTREM has to be configured with a memory latency of 48 CCLK cycles to estimate the runtime of this benchmark correctly.

To achieve cycle-accurate simulation of the XScale processor, we developed XEEMU, a derivative of XTREM, where we fixed all the detected pipeline-related problems, and modified the BTB and the caching strategies according to the documentation of the XScale architecture. We also realized that a cycle-accurate model of the memory controller and the memory banks in the memory clock domain is mandatory. Thus, we extended our simulator with a completely new, cycle-accurate model of the SDRAM subsystem.

5.2. Power model

The implementation of the power model of XTREM (inherited from Wattch) is based on activity counters of the core and assumes that inactive functional units consume almost no power because of effective clock gating. However, our measurements have shown that the power dissipation decreases only a small amount (about 20%) on core stalls. Additionally, the power model does not support dynamic frequency (CCLK) and voltage scaling. Hence, this power model is not suitable for the accurate power modeling of XScale.

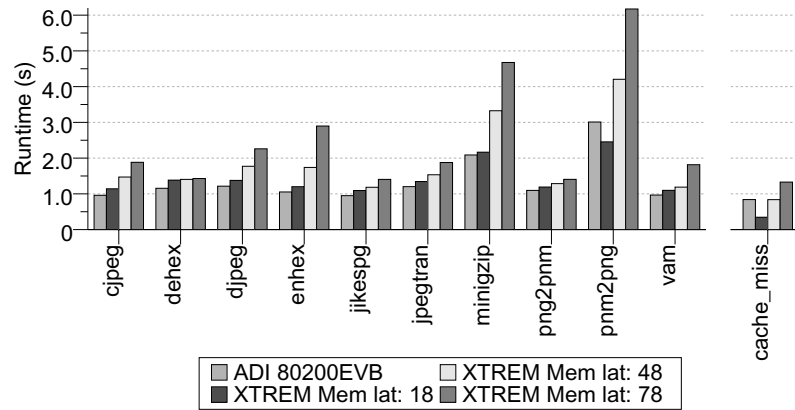


Fig. 5. Simulated runtime for different values of memory latency in XTREM compared to measurements on the board.

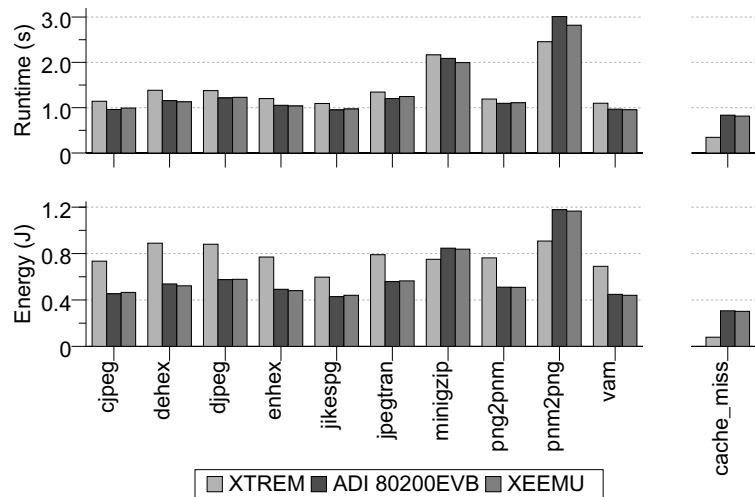


Fig. 6. Runtime and CPU energy consumption at 600 MHz/1.3 V.

Instead of developing a new core power model, we adopted and fine-tuned the versatile power model of the Sim-Panalyzer tool in XEEMU. During the iterative process of fine-tuning we compared measurements and simulation and fitted the parameters of the power model of each functional unit or pipeline stage using synthetic benchmarks that stress certain features of the pipeline.

We extended the power model by SDRAM memory to allow not only their behavioral but power consumption simulation as well. The implementation is based on power models published by the SDRAM manufacturer Micron and the formulas presented in [13]. The model we use is in many ways configurable and even the support for SDRAM low-power modes is incorporated. As we cannot directly measure the SDRAM power consumption on our board, the finished prototype is validated against the results measured by Snowdon et al. [18].

6. Experimental results

We selected 10 test programs from the CSiBE benchmark environment [8] to evaluate XEEMU. The selection contains various command line tools such as data and image compressors, converters and parsers. These programs not only test the overall accuracy of the simulator but exploit the special characteristics of the architecture as well. The JPEG compressor (cjpeg) utilizes many shift operations. The hex encoder-decoder pair (enhex, dehex) executes many conditional block data transfer instructions. The VSL abstract machine (vam) is a computation dominant program, i.e., it rarely accesses the memory and fits in the caches. In contrast to vam, minigzip and the PNG encoder (pnm2png) are memory dominant programs, causing a high number of data cache misses.

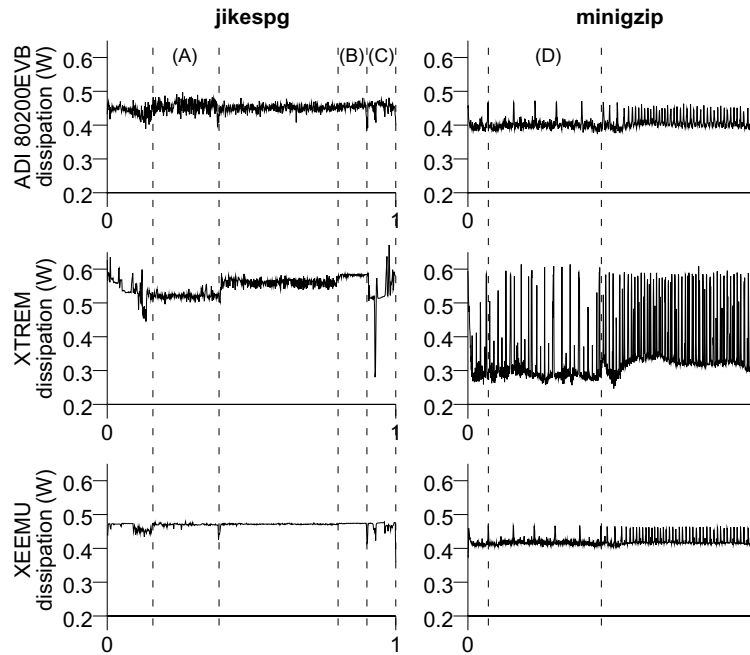


Fig. 7. CPU power dissipation of two programs with normalized runtime.

All these programs are written in standard C and are compiled with the GCC-based Wasabi cross compiler tool chain [22] to stand alone binaries, using optimization option `-O3`. To eliminate a source for side effects on the hardware and in the simulator, the programs are executed with no underlying operating system and all I/O operations mapped to main memory.

For our benchmark set, we compared runtimes and CPU energy consumptions on the evaluation board and on XEEMU, and additionally on its ancestor XTREM, all configured to 600 MHz core clock and operating voltage of 1.3 V for best MIPS/power ratio [6]. We configured XTREM to a memory latency of 18 CCLK cycles, since as mentioned earlier, this is the optimal configuration for XTREM for the chosen setup, even though the synthetic benchmark cache `_miss` clearly indicates a higher memory latency. The results of the comparison are shown in Fig. 6. It can be seen that the runtime of memory dominated benchmarks is underestimated by XTREM, while it overestimates the runtime of all other benchmarks. The average error (not including the synthetic benchmark) is 13.0%, the maximum error 19.7%. XEEMU shows a much more accurate prediction of the runtime in every single benchmark and improves the average and maximum errors to 3.0% and 6.4%, respectively. Moreover, XEEMU also works very well on all synthetic benchmarks.

The same applies to the core energy consumption as well. XTREM overestimates the effectiveness of clock gating in the case of pipeline stalls and, thus, underestimates the energy consumption of memory dominant programs. The energy estimates for the CPU core consumption provided by XEEMU are accurate within 4.5% in worst case and 1.6% in the average case, due to the much more accurate pipeline model, the improved behavioral simulation of the memory subsystem, and the fine-tuned CPU core power model.

In Fig. 7, the instantaneous power dissipation of the CPU is shown for two test programs. Since the real and simulated runtimes of the programs differ, we normalized them to allow comparison. The first program, `jikespg`, was chosen as it has several different regions of execution (delimited by dashed vertical lines on the charts). Region A is a computation dominant part of the program. In this region the inaccurate pipeline model of XTREM results in non-existing stalls during the simulation, which are the cause for the mispredicted relative decrease in dissipation. In region B, the number of read and write accesses to the external bus increases. Because of the inaccuracy of the power model of XTREM, this yields an unwanted increase in the power dissipation. Memory read instructions are frequent in region C, as well, but contrary to region B, they cause low external bus activity, i.e., it is mostly the cache that serves the requests. The improper modeling of the

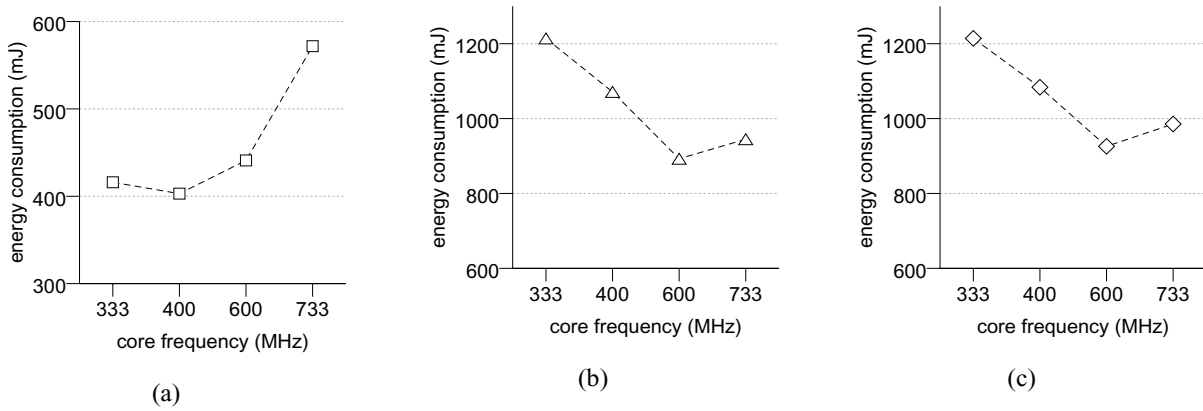


Fig. 8. Simulated energy consumption of the vam benchmark for various frequencies: (a) CPU consumption, (b) system consumption, and (c) system consumption, padded to equal time.

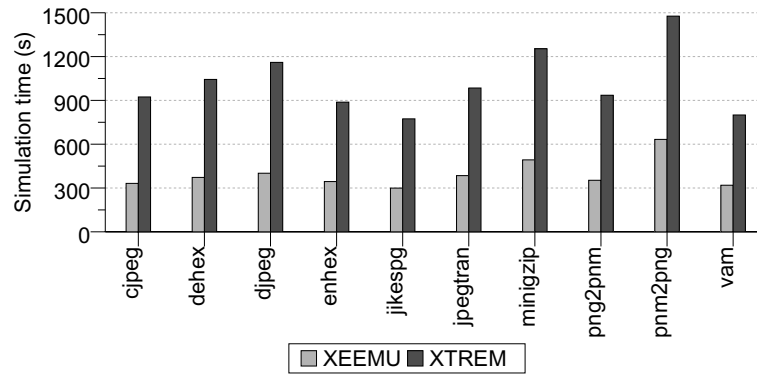


Fig. 9. Comparison of simulation times for XEEMU and XTREM.

external bus results in the drop in the dissipation graph of XTREM.

The other example is minigzip, a heavily memory dominant program, which causes a huge amount of cache misses, accounting for 56% of the runtime. This is especially true for region D, where the core usually stalls, except for the peaks found on the dissipation graphs of the board and XEEMU. In the graph of XTREM, the extra peaks are caused by the wrong memory access model and memory latencies. On the contrary, the dissipation graphs of XEEMU show the accuracy of its power and behavioral models.

More important than the correct simulation of the energy consumption of the CPU at one specific frequency and voltage setting is XEEMU's capability of simulation of DVFS for the CPU and the SDRAM. Figure 8(a) shows exemplarily the simulated CPU energy for the vam benchmark in different configurations, which matches the measured energy consumption with a maximum error of only 2.9%. Similar results were obtained for all other benchmarks, as well.

Since XEEMU is extended with behavioral and power models for memory, we can simulate the joint CPU and memory energy consumption as well. However, the energy consumption of the memory on the ADI evaluation board can not be measured directly. Thus, our system energy consumption measurements were influenced not only by the CPU and memory, but also other peripheral components and the on-board voltage regulation. It can be observed that the simulation results generated with XEEMU for the system simulation (see Figs 8(b) and (c)) show the same general trend as the measurements shown in Section 3. These results are in line with the findings presented in [18]. The only obvious difference is in Fig. 8(b) at 733 MHz, which can be attributed to the slight difference between the measurement and simulation setups as described above.

Another important benefit of XEEMU over XTREM is the improved runtime, which we measured on a standard PC (DualCore AMD 2.2 GHz, 4 GB RAM) running Debian Linux, Kernel 2.6.19.2. As shown in

Fig. 9, XEEMU simulates the benchmarks on average 2.5 times faster than XTREM.

7. Conclusions and future work

In this paper, we presented XEEMU, a new power and energy simulation tool for XScale-based systems. It has been validated using measurements on real hardware and shows a high accuracy for runtime, and instantaneous power and total energy consumption estimation of the CPU core. With a low average error of only 3.0% for runtime and only 1.6% for CPU energy consumption estimation, it clearly outperforms its ancestor XTREM in all test cases. Using a different and computationally less complex power model than XTREM, XEEMU also offers a more than 2-fold increase in simulation speed.

Additionally, XEEMU offers two completely asynchronous, configurable clock domains for the core and memory clocks. This, together with a precise, cycle-accurate behavioral model and a manufacturer-published-data-based power model of the memory subsystem makes XEEMU one of the first realistic system-simulators for cycle-accurate performance and power evaluation available.

Moreover, XEEMU has some interesting features that are not described in this paper in detail but point out directions for future research. The internals of XEEMU have been modified so that it supports dynamic voltage and frequency scaling, which makes experimentation with DVFS easy. We have plans for experimenting with offline DVFS approaches, where the voltage and frequency-switching points are determined at build time of the application to be optimized, as well as with online techniques, which are usually implemented as part of an operating system. To support the latter techniques, we have ported Linux to XEEMU.

With all these features, XEEMU is a unique tool to investigate all power and energy relevant effects of system configuration, OS-based or application-based DVFS, exploitation of low-power DRAM modes, and other software optimizations. With the belief that XEEMU can be a valuable tool for compiler and embedded software designers, we released it to the public for a wider use [1].

Acknowledgment

This work was (partially) supported by NKTH (Hungarian National Office for Research and Technology)

and BMBF (German Federal Ministry of Education and Research) within the framework of the Bilateral German-Hungarian Collaboration Project on Ambient Intelligence Systems (BelAmI), and by the Péter Pázmány Program (no. RET-07/2005).

References

- [1] XEEMU homepage. <http://www.inf.u-szeged.hu/xeemu/>.
- [2] ADI Engineering. 80200EVB Evaluation Board Hardware – Users Guide, December 2001. Revision 1.1.
- [3] ARTEMIS Strategic Research Agenda Working Group. Strategic research agenda – first edition, 2006.
- [4] T. Austin, E. Larson and D. Ernst, SimpleScalar: an infrastructure for computer system modeling, *Computer* 2(35) (2002), 59–67.
- [5] D. Brooks, V. Tiwari and M. Martonosi, *Wattch: A Framework for Architectural-Level Power Analysis and Optimizations*, In Proceedings of the 27th Annual International Symposium on Computer Architecture, June 2000.
- [6] L.T. Clark, An embedded 32-b microprocessor core for low-power and high-performance applications, *IEEE Journal of Solid-State Circuits* 36(11) (November 2001), 1599–1608.
- [7] G. Contreras, M. Martonosi, J. Peng, R. Ju and G.-Y. Lueh, XTREM: a power simulator for the Intel XScale core, *SIGPLAN Notices* 39(7) (2004), 115–125.
- [8] Department of Software Engineering, University of Szeged. GCC code-size benchmark environment (CSiBE). <http://www.csibe.org/>.
- [9] G. Dhiman and T.S. Rosing, Dynamic voltage frequency scaling for multi-tasking systems using online learning. In ISLPED '07: Proceedings of the 2007 International Symposium on Low Power Electronics and Design, ACM, New York, NY, USA, 2007, pp. 207–212.
- [10] X. Fan, C.S. Ellis and A.R. Lebeck, *The Synergy between Power-Aware Memory Systems and Processor Voltage Scaling*, In PACS, pages 164–179, 2003.
- [11] Z. Herczeg, Á. Kiss, D. Schmidt, N. Wehn and T. Gyimóthy, XEEMU: An improved XScale power simulator, in: *PATMOS*, N. Azmard and L.J. Svensson, eds, volume 4644 of Lecture Notes in Computer Science, Springer, 2007, pp. 300–309.
- [12] Intel corporation. Intel 80200 Processor based on Intel XScale Microarchitecture: Developer's Manual, March 2003. Order Number: 273411–003.
- [13] J. Janzen, Calculating memory system power for DDR SDRAM, *Designline* 10(2) (2001).
- [14] A.A. Jerraya, S. Yoo, D. Verkest and N. Wehn, *Embedded Software for Soc*, Kluwer Academic Publishers, Norwell, MA, USA, 2003.
- [15] A. Miyoshi, C. Lefurgy, E. Van Hensbergen, R. Rajamony and R. Rajkumar, *Critical Power Slope: Understanding the Runtime Effects of Frequency Scaling*, In ICS '02: Proceedings of the 16th international conference on Supercomputing, pages 35–44, New York, NY, USA, 2002. ACM Press.
- [16] T. Mudge, T. Austin and D. Grunwald, *The SimpleScalar-Arm Power Modeling Project*, <http://www.eecs.umich.edu/~panalyzer/>.
- [17] J.M. Rabaey, *The Spice Home Page*, <http://bwrc.eecs.berkeley.edu/Classes/IcBook/SPICE/>.
- [18] D.C. Snowdon, S. Ruocco and G. Heiser, *Power Management and Dynamic Voltage Scaling: Myths and Facts*, In PARC '05:

- Proceedings of the 2005 Workshop on Power Aware Real-time Computing, September 2005.
- [19] V. Tiwari, S. Malik and A. Wolfe, Power analysis of embedded software: a first step towards software power minimization, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* **2**(4) (1994), 437–445.
- [20] V. Tiwari, S. Malik, A. Wolfe and M. Lee, Instructionlevel power analysis and optimization of software, *VLSI Signal Processing* (13) (1996), 223–238.
- [21] D. Wang, B. Ganesh, N. Tuaycharoen, K. Baynes, A. Jaleel and B. Jacob, DRAMsim: a memory system simulator. SIGARCH, *Computer Architecture News* **33**(4) (2005), 100–107.
- [22] Wasabi Systems Inc. Wasabi Systems GNU tools version 031121 for Intel XScale microarchitecture. http://www.intel.com/design/intelxscale/dev_tools/031121/wasabi_031121.htm.
- [23] W. Ye, N. Vijaykrishnan, M. Kandemir and M.J. Irwin, *The design and use of Simplepower: A Cycle-Accurate Energy Estimation Tool*, In Proc. of 37th DAC, pages 340–345, Los Angeles, California, 2000.