XEEMU: An Improved XScale Power Simulator

Zoltán Herczeg[1], Ákos Kiss[1], Daniel Schmidt[2], Norbert Wehn[2], and Tibor Gyimóthy[1]

[1] University of Szeged, Department of Software Engineering
Árpád tér 2., H-6720 Szeged, Hungary
{zherczeg,akiss,gyimothy}@inf.u-szeged.hu

[2] University of Kaiserslautern, Department of Electrical Engineering
Erwin-Schrödinger-Straße, D-67663 Kaiserslautern, Germany
{schmidt,wehn}@eit.uni-kl.de

Abstract. Energy efficiency is a top requirement 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. Power simulators offer flexibility and allow a detailed view on the sources of power consumption. In this paper we present XEEMU, a fast, cycle-accurate simulator, which aims at the most accurate modeling of the XScale architecture possible. 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 energy consumption estimation, respectively.

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 mobile phones, mp3 players and PDAs [1]. 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 knobs for minimizing energy and power like clock gating, power supply and frequency scaling, leakage control, etc. But the actual power and energy consumption of a microprocessor is determined by the application running on it. However the interrelation between the software and hardware w.r.t. power/energy consumption and optimization is very challenging. 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.

Several possibilities exist to evaluate the energy and power consumption, the most obvious being measurements on real hardware. This is seldomly 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.

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 is the key to energy efficient embedded software design.

In this paper we present XEEMU, a fast, cycle-accurate simulator for the XScale architecture [2], one of the most widespread, low-level RISC architectures in the embedded domain. In contrast to many other existing power simulators, as [3,4], 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 [5] in terms of runtime and energy estimation, due to its improved pipeline and power model and the cycle accurate simulation of the SDRAM subsystem. 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 simulation. 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 methodology is a general approach that can be used for a wide range of different processors. Our case-study, which yielded in the creation of XEEMU, shows the pitfalls in creating precise pipeline models and power models. It points out aspects that have to be taken into account when creating power simulators targeted for other architectures as well.

The rest of the paper is organized as follows: Section 2 presents related work done so far on power and energy simulation and measurement, Section 3 describes how XEEMU was developed from an existing simulator by correcting the identified problems and Section 4 shows the results of the improvements. Finally, Section 5 concludes the gathered experiences and gives directions for future work.

2 Related Work

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.

Spice [6] is the de-facto industrial standard general circuit simulator. While it offers a high accuracy it requires 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 [7,8] 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 as the XScale the execution times of instructions may vary strongly. This is especially true for external memory accesses, but also for stalls due to mispredicted branches.

The existence of a nearly cycle-accurate instruction level simulator for the XScale architecture is claimed in [9]. However, this tool is not available to the public. Furthermore, it was not compared to other existing simulators so the results can neither be verified nor exploited.

To model exact, cycle-accurate behavior of a pipeline, simulation on the micro-architectural level is mandatory. SimpleScalar [10] is an open source, configurable, generic processor core simulator. It 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 [11] 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 [3] (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.

Wattch [4] 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 [5] 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. Since this approach targets our intentions we will investigate this tool in more detail in the next sections.

3 XTREM – An In-Depth Review

3.1 Experimental Setup and Benchmarks

To validate XTREM, we use the ADI 80200EVB, XScale-based evaluation board from ADI Engineering. The board is equipped with an Intel 80200 XScale processor [12] and 32 Mbytes of SDRAM. Input clocks provided by the board are 66MHz for CLK and a 100MHz MCLK for the peripheral bus controller and the SDRAM. CLK is used to generate the XScale internal core clock CCLK, which we set to 600MHz, to achieve the best MIPS/power ratio [2]. The core operating voltage is 1.3V. The memory and peripheral bus controller is hosted in a Xilinx FPGA [13].

The board provides a jumper to measure core power consumption. Using a Tektronix TPS 2014 digital storage oscilloscope connected to a PC we sampled the power consumption with a 0.1Ω resistor in line with the processor core. The core voltage during operation remained constant. To eliminate noise in the measurements we calculated the average value for each sample by measuring seven runs of each benchmark.

The results of simulation and measurement are compared for 10 test programs selected from the CSiBE benchmark environment [14]. 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.

All these programs are written in standard C and are compiled with the GCC-based Wasabi cross compiler tool chain [15] to stand alone binaries, using optimization option -O3. 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, thus eliminating a source for side effects.

3.2 Performance Validation

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.

We configured XTREM to the same frequency and voltage as the hardware, i.e. 600MHz core clock. The memory clock in XTREM is, however, not configurable, but instead it assumes a constant memory access latency. As due to cost

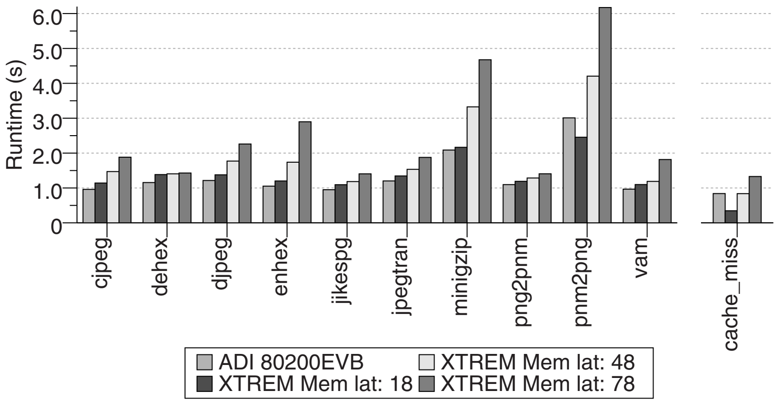

Fig. 1. Simulated runtime for different values of memory latency in XTREM compared to measurements on the board

limitations DRAM dominates for larger memories in embedded systems, this assumption is invalid. Depending on the state of the memory controller and the memory banks, and on the clock frequency of the memory subsystem memory access times may vary heavily. Using the processor internal performance counters we observed cache miss costs varying from 78 to 126 CCLK cycles for the chosen setup. To reflect these effects a cycle accurate model of the memory controller and the memory banks in the memory clock domain is mandatory.

However, setting the memory latency in XTREM to 78 CCLK cycles yields in a runtime overestimation for all test programs, by more than a factor of 2 for the memory intensive pnm2png. The optimal correlation of benchmark run times as simulated by XTREM and measured on the processor was found with a fixed memory access latency of only 18 CCLK cycles, as is shown in Fig. 1. 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.

An in-depth investigation of XTREM shows a lot of other inaccuracies in the modeling of the caches and the pipeline, which is shown in Fig. 2. 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

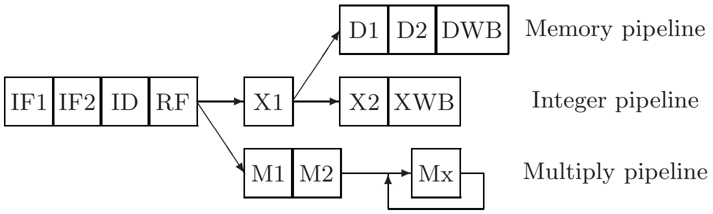

Fig. 2. The pipeline of XScale

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 the XScale.

To achieve cycle-accurate simulation of the XScale processor we extended the simulator with a cycle-accurate model of the SDRAM subsystem and fixed all the detected pipeline related problems. The BTB and the caching strategies were modified according to the documentation of the XScale architecture.

3.3 Power Modeling

The implementation of the power model of XTREM (inherited from Wattch) is based on activity counters 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. Besides the mentioned issue, 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.

Instead of developing a new power model, we adopted and fine-tuned the versatile power model of the Sim-Panalyzer tool. 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.

4 Experimental Results

The modifications made to XTREM in response to the problems described in Section 3 resulted in a new improved power and performance simulation tool,

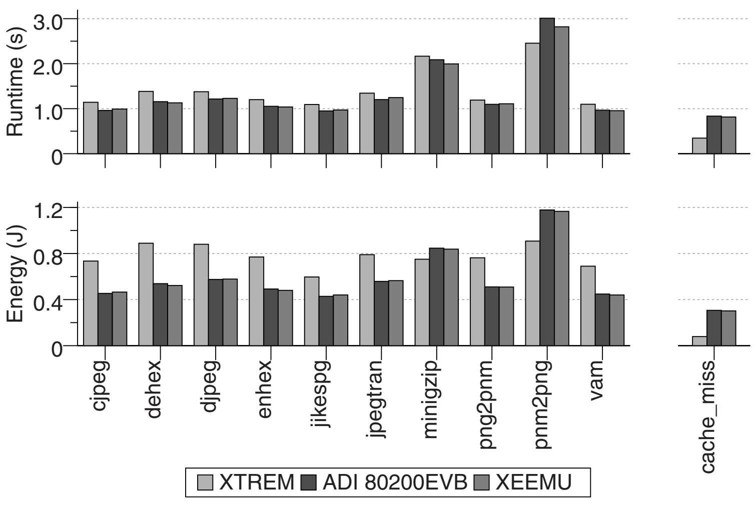

Fig. 3. Runtime and energy consumption

called XEEMU. We compared run times and energy consumptions for our chosen benchmark set and synthetic benchmarks on XTREM, XEEMU, and the evaluation board, as described in Section 3.1. Results are shown in Fig. 3. As mentioned earlier, XTREM was configured to a memory latency of 18 CCLK cycles, even though the synthetic benchmark cache_miss clearly indicates a higher memory latency. 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%, respecitvely. XEEMU also works very well on all synthetic benchmarks.

The same applies to the 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. Due to the much more accurate pipeline model, the improved simulation of the memory subsystem, and the fine-tuned power model, the energy estimates provided by XEEMU are accurate within 4.5% in worst case and 1.6% in the average case.

In Fig. 4 the instantaneous power dissipation of two test programs is shown. Since the real and simulated run times 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

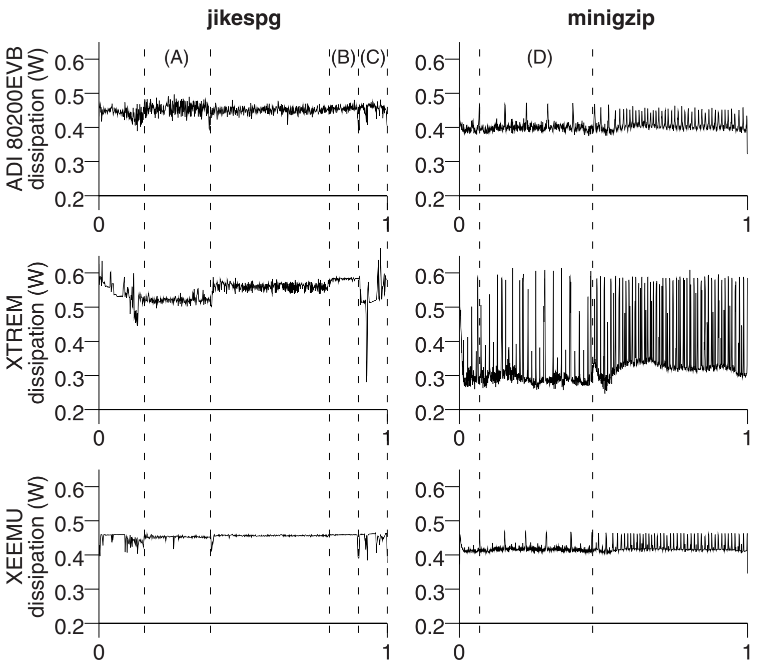

Fig. 4. Power dissipation of two programs with normalized runtime

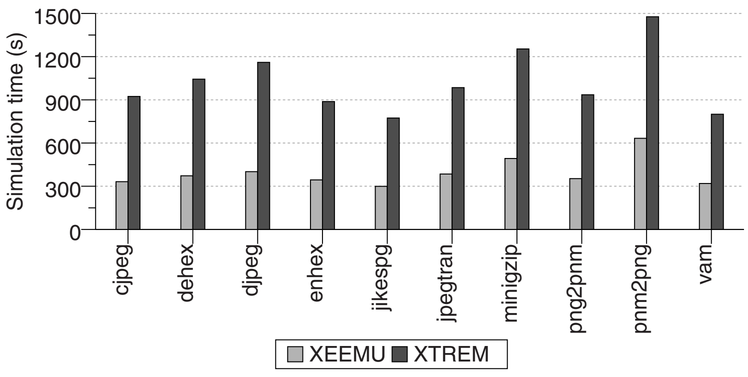

Fig. 5. Comparison of simulation times for XEEMU and XTREM

bus increase. Because of the inaccuracy of the power model of XTREM, this yields an unwanted increase in the 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 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 to the problems observed with XTREM, the dissipation graphs of XEEMU show the accuracy of its power model, pipeline model, and memory model.

Another important benefit of XEEMU over XTREM is the improved runtime, which we measured on a standard PC (DualCore AMD 2.2GHz, 4GB RAM) running Debian Linux, Kernel 2.6.19.2. As shown in Fig. 5, XEEMU simulates the benchmarks on average 2.5 times faster than XTREM.

5 Conclusions and Future Work

In this paper we presented XEEMU, a new power and energy simulation tool for XScale processor cores. XEEMU extends the architectural model of XTREM with a precise, cycle-accurate model of the memory subsystem and corrects many errors we found in the simulation of the pipeline. It has been validated using measurements on real hardware and shows a high accuracy for runtime, instantaneous power, and total energy consumption estimation. With a low average error of only 3.0% for runtime and only 1.6% for energy consumption estimation it clearly outperforms XTREM in all test cases. Using a different and less computationally complex power model than XTREM, XEEMU also offers a more than 2-fold increase in simulation speed.

XEEMU was designed for maximum simulation accuracy while still offering high flexibility. In contrast to XTREM, it already offers two completely asynchronous, configurable clock domains for the core and memory clock, as well as a cycle accurate SDRAM memory subsystem simulation. Currently, we extend the simulation model of the memory subsystem with a power model, making XEEMU one of the first realistic system-simulators for cycle accurate performance and power evaluation available. Additionally we will implement features for dynamic voltage scaling offering a valuable tool for compiler and embedded software designers. XEEMU will be released to the public for a wider use.

Acknowledgement. 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. ARTEMIS Strategic Research Agenda Working Group: Strategic research agenda, 1st edn. (2006)
2. Clark, L.T.: An embedded 32-b microprocessor core for low-power and high-performance applications. IEEE Journal of Solid-State Circuits 36(11), 1599–1608 (2001)
3. Mudge, T., Austin, T., Grunwald, D.: The SimpleScalar-Arm power modeling project, `http://www.eecs.umich.edu/~panalyzer/`
4. Brooks, D., Tiwari, V., Martonosi, M.: Wattch: A framework for architectural-level power analysis and optimizations. In: Proceedings of the 27th Annual International Symposium on Computer Architecture (June 2000)
5. Contreras, G., Martonosi, M., Peng, J., Ju, R., Lueh, G.Y.: XTREM: a power simulator for the Intel XScale core. SIGPLAN Not. 39(7), 115–125 (2004)
6. Rabaey, J.M.: The Spice Home Page `http://bwrc.eecs.bekeley.edu/Classes/IcBook/SPICE/`
7. Tiwari, V., Malik, S., Wolfe, A.: Power analysis of embedded software: a first step towards software power minimization. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 2(4), 437–445 (1994)
8. Tiwari, V., Malik, S., Wolfe, A., Lee, M.: Instruction-level power analysis and optimization of software. VLSI Signal Processing (13), 223–238 (1996)
9. Varma, A., Debes, E., Kozintsev, I., Jacob, B.: Instruction-level power dissipation in the Intel XScale embedded microprocessor. In: Sudharsanan, S., Bove Jr., V.M., Panchanathan, S. (eds.) Proceedings of SPIE, Embedded Processors for Multimedia and Communications II, San Jose, California, USA, vol. 5683, pp. 1–8. SPIE (2005)
10. Austin, T., Larson, E., Ernst, D.: Simplescalar: an infrastructure for computer system modeling. computer 2(35), 59–67 (2002)
11. Ye, W., Vijaykrishnan, N., Kandemir, M., Irwin, M.J.: The design and use of SimplePower: A cycle-accurate energy estimation tool. In: Proc. of 37th DAC, Los Angeles, California, pp. 340–345 (2000)
12. Intel corporation: Intel 80200 Processor based on Intel XScale Microarchitecture: Developer's Manual. Order Number: 273411-003 (March 2003)
13. Intel corporation: High Performance Memory Controller for the Intel 80200 Processor. Order Number: 273494-001 (March 2001)
14. Department of Software Engineering, University of Szeged: GCC code-size benchmark environment (CSiBE) `http://www.csibe.org/`
15. Wasabi Systems Inc.: Wasabi Systems GNU tools version 031121 for Intel XScale microarchitecture `http://www.intel.com/design/intelxscale/dev_tools/031121/wasabi_031121.h tm`