System and architecture-level power reduction of microprocessor-based communication and multi-media applications

Lode Nachtergaele Vivek Tiwari Nikil Dutt

IMEC, Leuven, Belgium Intel Corp., Santa Clara, CA U.C. Irvine, CA

1 Introduction

Current microprocessor architectures become more and more dominated by the data access bottlenecks in the cache, system bus and main memory subsystems. These also have a major influence on the system (board-level) power consumption. In practice this means lower energy consumption for a given throughput requirement.

In the booming domain of (largely embedded) cost-sensitive communication and multi-media applications, more and more implementations make use of microprocessor based platforms for flexibility reasons.

However, in order to provide sufficiently high data throughput at reasonable power consumption for these demanding applications, novel solutions for the memory access and data transfer will have to be introduced. These will have to be situated both at the processor architecture and the algorithm/compiler level.

The question we want to address in this paper and tutorial is what would these solutions look like. We will show that they will be based on processor architecture optimizations, on novel approaches in the application of compiler technology, and on exploiting the interface between the system hardware and software.

2 Architecture Optimizations

Due to the quadratic dependence of power on voltage, voltage reduction is the most favored method of reducing power. It has been shown that very aggressive voltage reductions are possible if architectural and algorithmic transformations are applied to the problem (pipelining & parallelism) to regain the loss in performance from voltage reduction [1]. This is a very powerful technique for throughput-oriented limited-function applications (e.g. digital filtering), and is an option for implementing part of the functionality of embedded multi-media applications.

Traditional CPU architecture research has focussed on the performance problems entailed by the increasing gap between CPU speed and memory bandwidth. Larger and more levels of caches and larger instruction scheduling windows are the approaches that have been adopted for increasing CPU performance. More recently, architectural optimizations aimed primarily at power reduction have become an active area of research.

The main ideas [2] can be classified under the following themes:

a) *module parameter tradeoffs* – the optimal size and configurations of micro-architectural modules such as caches [3], register-files etc. for the desired power/performance

b) *exploiting locality both for instructions and data* – e.g. mini-caches, mini-TLBs to avoid looking up the larger main cache [4] and *value locality* – where recent computations are saved to avoid re-computation [5]

c) *enabling more powerdown* – e.g. partitioning of caches to allow one the necessary bank to be powered up, and word-width wise partitioning of datapaths [6]

d) *speculation reduction* – dynamically reducing the speculation in the machine to reduce power – e.g. limiting instruction issue if the number of predicted branches exceeds a limit [7]

e) h/w hooks to allow for increased s/w control on power – e.g. a loop cache into which basic blocks are statically allocated by the compiler [8].

Optimizations such as the above are local to the CPU. Power reduction techniques of a wider scope are possible if the CPU is seen as a component of an overall system. For example, the CPU need not be fully-active if it is waiting for I/O. Similarly other system components need to be active or powered-up only when needed. This motivates the application of dynamic power management systems. The basic requirements for dynamic power management system are the availability of multiple power states for the various system components – CPU, memory, peripherals, mechanisms for detecting opportunities for power state transitions, and a control mechanism to coordinate state transitions. The detection and control functions can be distributed between the hardware and system software (O/S). For standard platforms such as the personal computers, there are standard power management specifications e.g. ACPI (Advanced Configuration and Power Interface Specification) [9]. For embedded applications, there are opportunities for additional flexibility and power management systems tuned for specific applications can be extremely efficient means for power reduction. Improved modeling of system behavior including stochastic models for user-behavior etc. has gained attention recently. Recently work has also focused on studying improved power management policies [10].

The power reduction from the power management discussed above comes from powering-down system components when not needed. This has gained widespread application due to its practicality and relative ease of implementation. An additional source of power efficiency comes from extending power management to include control on the CPU's voltage and performance. Dynamic voltage/freq has gained increased attention lately [11]. Realizing its full potential ideally requires a unified h/w-s/w approach. For general-purpose, open platforms, the various system components can come from different vendors, and thus, industry-wide standards and specifications will help faster adoption of these ideas. However, for embedded systems, the designer may have the flexibility of designing both the system h/w and s/w, as well as the end-application. In particular, multi-media applications are ideally suited for dynamic voltage/freq scaling since they often have regular activity patterns that can be pre-characterized.

3 Optimized Platform Mappings

In the domain of algorithm transformations and compilation technology for embedded data-dominated applications, there has been a lot of work for the traditional metrics of cost and performance. In recent years, significant progress has been made in targeting energy optimizations.

We will show that decisions made at this stage heavily influence the final outcome when the appropriate architectural issues of the embedded memories are correctly incorporated. This has to happen both at the ILP (instruction-level parallelism) compiler and in the preceding system compilation stages.

From the viewpoint of compilers, several opportunities for power reduction/management exist, particularly in the embedded multimedia and communication domains. Many of such applications are typically characterized by data-intensive computations operating on (multi-dimensional) arrayed data structures stored in off-chip memories. Thus compiler transformations that aim to reduce off-chip memory traffic often simultaneously improve performance, while reducing power.

3.1 System-level Code Transformations

Software specifications for multimedia systems are typically not optimized from a memory point of view. For example the software used during the development of the MPEG-4 video decoder uses several megabytes of memory while decoding a video of 352 by 288 pixels [12]. After systematic system level optimizations of the source code, only a few hundred kilobytes are actually required. Also the number of transfers from and to memory can be reduced by an order of magnitude. Both the reduction of size and number of transfers decrease both, at least linearly, the power consumption of the memory system while preserving behavior. Hence, exploration of Data Transfer and Storage (DTS) is an important pre-compilation step. A stepwise approach by a DTSE methodology [13], partially supported by tools, avoids a design time explosion. [14] gives a good overview of the general research in this domain.

The major principles of source-to-source transformations of the DTSE methodology are: a) Global data-flow transformation to avoid redundant transfers, b) Global loop and control flow transformations to increase locality of reference, c) Data reuse exploration to exploit the available memory hierarchy, d) SDRAM memory organization and e) Data layout decisions to reduce the memory size and to improve cache hit rates. These transformations are fully platform independent except the last two who are partially influenced by parameters of the target platform. This claim is supported by the results in [15]. This fact allows for a posteriori decisions about the target platform. A platform specific compiler, the subject in the next section, then optimizes platform dependent issues.

3.2 Platform Compiler Technology

Early experiments by Tiwari et al. [16] demonstrated reduced energy consumption (and higher performance) through improved register allocation, resulting in fewer spills to memory. Compiler techniques that improve data locality through coarse-grain transformations [13] and data layout optimization [17,18,19], result in significantly fewer cache misses, leading again to improved performance and lower power dissipation.

Similarly, instruction scheduling techniques to reduce instruction cache misses have been developed [20], resulting in reduced bus transitions per off-chip memory transfer. Recent work in memory-aware compilation [21,22,23] aims to better exploit memory access protocols of contemporary DRAMs for improving the memory bandwidth of applications.

The effects of such compiler optimizations (as well as many other contemporary compiler transformations) on power dissipation require a comprehensive measurement or simulation environment, since the relationship between performance and power or energy is not easily predictable. New efforts in building architectural power/energy-aware simulation [24,25,26] will help quantify the effects of compiler optimizations on power and energy. Finally, compiler-controlled power management techniques are beginning to appear [27,28], that dynamically tradeoff power for performance. The compiler, through a combination of static analysis, profile-driven data and feedback-driven optimization, can thus modify the power/performance characteristics of the target architecture, in consort with system-level power management schemes.

4 Conclusions

Microprocessor based platforms become more and more the choice for embedded solutions. To enable low power consumption platforms for upcoming demanding communication and multi-media applications solutions along three major axes are addressed: 1) architectural optimizations, 2) system level source-to-source transformations and 3) compiler technology.

5 Acknowledgements

We thank our colleagues at IMEC for their stimulating discussions and contributions to the DTSE methodology.

5 References

[1] A. Chandrakasan and R. Brodersen, "Low Power Digital CMOS Design," Kluwer Academic Press, 1998.

[2] IEEE Transaction on Computer Architecture Newsletter, special issue on "Interaction between Compilers and Computer Architectures'", June 1997.

[3] R. Bahar, G. Albera, and S. Manne, "Power and Performance Tradeoffs Using Various Caching Strategies," Proc. Intl. Symposium on Low-Power Electronics and Design, 1998.

[4] M. Kin and W. Mangione-Smith, "The Filter Cache: An Energy Efficient Memory Structure," Proc. Micro30, 1997.

[5] M. Azam, P. Franzon, W. Liu, and T. Conte, "Low Power Data Processing by Elimination of Redundant Computations," Proc. Intl. Symposium on Low-Power Electronics and Design, 1997.

[6] D. Brooks and M. Martonosi, "Dynamically Exploting Narrow Width Operands to Improve Processor Power and Performance," Proc. HPCA-5, 1999.

[7] S. Manne, A. Klauser and D. Grunwald, "Pipeline Gating: Speculation Control for Energy Reduction," Proc. ISCA-25, 1998.

[8] N. Bellas, "Architectural and Compiler Techniques for Energy Reduction in High Performance Microprocessors," Ph.D. Thesis, Univ. of Illinois at Urbana-Champaign.

[9] ACPI, http://www.teleport.com/~acpi/

[10] Y-H. Lu, E-Y. Chung, T. Simunic, L. Benini, and G. De Micheli, "Quantitative Comparison of Power Management Algorithms," Proc. DATE, 2000.

[11] T. Burd et. al., "A Dynamic Voltage Scaled Microprocessor System," Proc. ISSCC 2000.

[12] L. Nachtergaele, T. Gijbels, J. Bormans, F. Catthoor, M.Engels, "Power and speed-efficient code transformation of multi-media algorithms for RISC processors", IEEE Workshop on Multimedia Signal Processing, Los Angeles, California, USA, December 7-9, 1998, pp. 317-322.

[13] F. Catthoor, S. Wuytack, E. DeGreef, F. Balasa, L. Nachtergaele, and A. Vandecappelle, "Custom Memory Management Methodology," Kluwer Academic Press, 1998.

[14] L.Benini, G.De Micheli, "System-level power optimization techniques and tools", ACM Trans. on Design Automation for Embedded Systems (TODAES), Vol.5, No.2, pp.115-192, April 2000.

[15] K.Danckaert, F.Catthoor, H.De Man, "Platform independent data transfer and storage exploration illustrated on a parallel cavity detection algorithm", Proc. ACM Conf. on Par. and Dist. Proc. Techniques and Applications, PDPTA'99, Vol.III, pp.1669-1675, Las Vegas NV, June 1999.

[16] V. Tiwari, S. Malik, A. Wolfe, and T.C. Lee, "Instruction Level Power Analysis and Optimization of Software", Journal of VLSI Signal Processing Systems, Vol. 13, No. 2, August 1996.

[17] P. Panda, N. Dutt, and A. Nicolau, "Memory Issues in Embedded Systems-on-Chip: Optimizations and Exploration," Kluwer Academic Press, 1999.

[18] W. Shiue and C. Chakrabarti, "Memory Exploration for Low Power Embedded Systems," Proc. 36th Design Automation Conference, 1999.

[19] C. Kulkarni, F. Catthoor, H. De Man, "Advanced Data Layout Organization for Multi-media Applications," Proc. IPDPS Workshop on Parallel, Distributed Computing in Image Processing, Video Processing and Multimedia, 2000.

[20] H. Tomiyama, T. Ishihara, A. Inoue, and H. Yasuura, "Instruction Scheduling for Power Reduction in Processor-based System Design," Proc. Conference on Design, Automation, and Test in Europe, 1998.

[21] P. Grun, N. Dutt and A. Nicolau, "Memory-aware Compilation through Accurate Timing Extraction," Proc. 37th Design Automation Conference, 2000.

[22] P. Grun, N. Dutt and A. Nicolau, "MIST: An Algorithm for Memory Miss Traffic Management," Proc. International Conference on Computer-Aided Design, 2000.

[23] S. Rixner, W. Dally, U. Kapasi, P. Mattson and J. Owens, "Memory Access Scheduling," Proc. 27th International Symposium on Computer Architecture, 2000.

[24] D. Brooks, V. Tiwari and M. Martonosi, "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations," Proc. 27th International Symposium on Computer Architecture, 2000.

[25] N. Vijaykrishnan, M. Kandemir, M. Irwin, H. Kim, and W. Ye, "Energy-Driven Integrated Hardware-Software Optimizations using SimplePower," Proc. 27th International Symposium on Computer Architecture, 2000.

[26] M. Kandemir, N. Vijaykrishnan, M. Irwin, and W. Ye, "Influence of Compiler Optimizations on System Power," Proc. 37th Design Automation Conference, 2000.

[27] D. Marculescu, "Profile-Driven Code Execution for Low Power Dissipation," Proc. International Symposium on Low Power Electronics and Design, 2000.

[28] The COPPER Project: Compiler-Controlled Continuous Power-Performance Management, The Center for Embedded Computer Systems, University of California,Irvine. http://www.cecs.uci.edu/~copper