A Low Power Strategy for Future Mobile Terminals

Mladen Nikitovic and Mats Brorsson
Department of Microelectronics and Information Technology,
Royal Institute of Technology
Electrum 229, SE-164 40 Kista, Sweden
{mladen,matsbror}@imit.kth.se

Abstract

In this paper, we have investigated the efficiency of two power-saving strategies that reduces both static and dynamic power consumption when applied to a chip-multiprocessor (CMP). They are evaluated under two workload scenarios and compared against a conventional uni-processor architecture and a CMP without any power-aware scheduling. The results show that energy due to static and dynamic power consumption can be reduced by up to 78% and that further 8% energy can be saved at the expense of response-time of non-critical applications.

Furthermore, a small study on the potential impact of system-level events showed that system calls can contribute significantly to the total energy consumed.

1. Introduction

The functionality of mobile terminals such as Personal Digital Assistants (PDAs) and cell phones has increased rapidly in recent years and the trend is to integrate even more functionality in the future. This results in an increased performance demand on the underlying computer architecture design. Currently, it is common to use an uni-processor system with high enough frequency to satisfy the performance need. However, one cannot increase the frequency indefinitely without paying the penalty of increased power consumption. Therefore, a more efficient architecture is needed.

We believe that a multi-processor system implemented on a single chip i.e. chip-multiprocessor (CMP) can satisfy that demand because it can extract both instruction-level parallelism (ILP) and thread-level parallelism (TLP), whereas an uni-processor only extracts ILP. This way, the CMP can provide performance without having to run in high frequencies, resulting in low power consumption if intelligently managed.

In this paper, we propose two power-saving scheduling strategies applied to a CMP architecture. In contrast to our previous work [8], we analyze potential savings in both dynamic and static power consumption using a refined workload model to better reflect non-deterministic user interaction. It is common that only user-level events are captured in simulator statistics. Therefore, a small study is done on system-level events such as system calls and taskswitches and their potential contribution to the overall energy consumption.

Our experimental results show that up to 78% of both static and dynamic power can be reduced using our proposed strategies compared to both uni-processor and CMP without any power-aware scheduling. Furthermore, it showed that system calls can contribute significantly to the total energy consumed.

Very few have designed CMPs for mobile devices with low power consumption in mind. Although [6] published a CMP with the ability to activate and de-activate sections of the chip, no study have been done showing its efficiency.

2. Adaptive CMP Architecture

Our proposal is based on a CMP architecture consisting of simple single-pipelined processors with private instruction and data caches. The processors are interconnected through an atomic bus through which they can access the shared unified second level cache. Coherency between caches is maintained using MESI protocol.

The adaptivness of the CMP comes from the OS scheduler's ability to dynamically activate and de-activate processors. To be able to do that, the CMP has to consist of processors that can be put in power-saving modes. We implemented a process scheduler that does normal process scheduling according to Earliest Deadline First (EDF) algorithm and also schedules the activity of individual processors according to simple strategies.

The scheduler is utilizing a power-saving mode in the processor that disables its clock-tree and power lines. This way, both dynamic and static power consumption is minimized. The cell state in the caches is preserved using drowsy caches [4], meaning that data can still be fetched if needed with small performance cost, which is useful when sharing coherence state.

We evaluated two simple processor scheduling strategies. The first strategy, S1, prioritizes performance while the second strategy, S2, prioritizes low power consumption while still meeting deadlines. Both strategies impacts the scheduler's way of doing process allocation and process migration. In the first strategy, applications are allocated to already running processors to reduce the overhead with waking up from power-saving mode. In the second strategy, non-critical applications are allocated on the same single processor to reduce power consumption at the expense of longer response time.

3. Workload Model

We created two multi-programmed workload scenarios consisting of selected applications from the Mediabench benchmark suite [7]. Applications in the workloads were either time-critical, thus they had deadlines or they were executed according to a best-effort strategy, thus they had low scheduling priority.

Each application is independent, thus there is no inter-process communication.

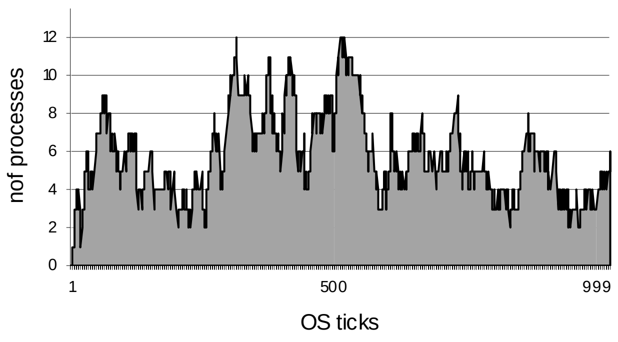

Figure 1. A heavy workload scenario.

In order to evaluate a scheduling strategy, we cannot execute applications in isolation but need some way to aggregate them in a workload. Previous use of multi-programmed workload models have been trivial in nature and did not exhibit any of the burstiness that we are expecting. For that reason, we have chosen to use a model, *b-model*, derived from studies on traffic with self-similar patterns in web, video, and disk environments [11]. Our approach is to map each data point in the generated traffic graph into a release of a benchmark application. In our study we are considering two workload scenarios; one where the workload is light, consisting of 64 applications, thus poses a modest load on the system, and a heavy load, consisting of 256 applications, where the system is more utilized (see Figure 1.). Both workloads are aggregated over 1024 OS ticks, where each tick is a 20 ms period.

4. Evaluation Methodology

We use simulation to estimate performance and power consumption of our chosen architecture configurations. Timing is estimated using SimpleScalar models [2]. Our simulator is based on a functional cache simulator, *sim-cache*, and extended to a multiprocessor system. We assume that it takes one cycle to execute an instruction if no memory stalls are experienced.

The power consumption of the simulated architectures is estimated using Wattch [1]. It provides models for caches, TLBs, and the global clock-tree. Also, it is capable of simulating several clock-gating strategies. We use the base cost model proposed by Sinha et al. [10] for modeling processor power consumption. As for the global bus, the power consumption is estimated using capacitance models of simple wires [9]. The activity factor of the bus is set to 50%. The power consumption of off-chip bus and main memory is modeled using [5]. All on-chip models are based on 0.18 µm process technology figures.

Most studies only analyze user-level activities. Events such as system calls and taskswitches become unaccounted for. We are using a simple pessimistic model for such events. We assume that 100 instructions are enough to perform a taskswitch and 1,000 instructions are needed to perform a system call. We assume a 80% cache miss-ratio for both events.

Table 1: Energy consumed by system-level events.

Workload	# syscalls	# taskswitches	energy ratio	energy ratio
Heavy	37 012	8 956	3.1%	0.003%
Light	9 337	4 236	0.7%	0,007%

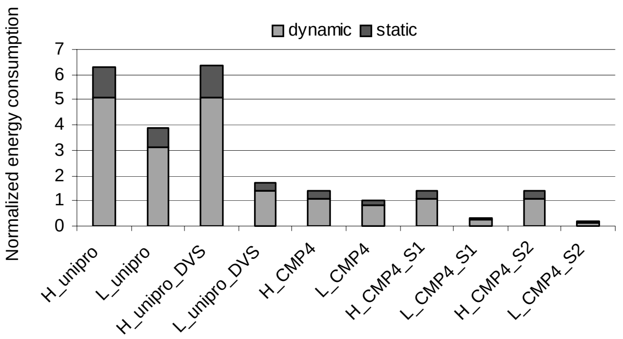

Figure 2. Estimated energy consumption.

5. Results and Conclusions

Figure 2 shows the dynamic and static energy consumed by architectures we considered during heavy and light workload scenarios (H/L prefix), thus uni-processor with and without DVS, 4-way CMP without scheduling strategies, 4-way CMP with S1 or S2 strategy. The bars are normalized against the 4-way CMP without any strategy. Using DVS on the uni-processor only decreased the energy consumed during the light workload. We can see that the 4-way CMP without any strategy consumes as much as a uni-processor with DVS during a light workload and several times less energy during the heavy workload. Using S1 strategy, the 4-way CMP reduced its energy consumption by 78% during the light workload. Energy consumption was further reduced using S2 strategy by 8% at the expense of response-time of non-critical applications while deadlines of time-critical applications are still met.

Table 1 shows how many system calls and taskswitches were performed and their energy consumption compared to overall energy consumption. Results show that system calls can contribute significantly to the energy consumed during heavy workloads whereas taskswitches were efficient at all times.

References

[1] D. Brooks et. al., Wattch: A Framework for Architectural-level Power Analysis and Optimizations. In *Proc. of International Symp. on Computer Architecture*, pp. 83-94, 2000.
[2] D. Burger et. al., *The SimpleScalar Tool Set, Version 2.0*. CS-TR-97-1342, University of Wisconsin, 1997.
[3] L. T. Clark et al, An Embedded 32-b Microprocessor Core for Low-Power and High-Performance Applications, IEEE Solid-State Circuits, Volume 36, Issue 11, pp. 1599 -1608.
[4] K. Flautner et. al., Drowsy caches: simple techniques for reducing leakage power, In Proceedings of International Symposium on Computer Architecture, pp. 148-157, 2002.
[5] R. Fromm et al., The Energy Efficiency Of Iram Architectures. In *Proceedings of IEEE International Symposium on Computer Architecture*, pp. 327-337, 1997.
[6] M. Edahiro et. al., *A Single-Chip Multiprocessor for Smart Terminals*. IEEE Micro, Vol. 20 Issue 4, pp. 12-30, 2000.
[7] C. Lee et. al., Mediabench: A tool for evaluating and synthesizing multimedia and communications systems. In *Proc. of International Symp. on Microarchitecture*, pp. 330-335, 1997.
[8] M. Nikitovic and M. Brorsson. An adaptive chip-multiprocessor architecture for future mobile terminals. In *Proc. of CASES'02*, pp. 43-49, 2002.
[9] J.M. Rabaey and M. Pedram, *Low Power Design Methodologies*. 1996, ISBN 0-7923-9630-8.
[10] A. Sinha and A.P Chandrakasan, JouleTrack-a Web Based Tool for Software Energy Profiling. In *Proc. of Design Automation Conference*, pp. 220-225, 2001.
[11] M. Wang et. al., Data Mining Meets Performance Evaluation: Fast Algorithms for Modeling Bursty Traffic. In *Proc. of International Conf. on Data Engineering*, pp. 507-516, 2002.