

An Adaptive Chip-Multiprocessor Architecture for Future Mobile Terminals

Mladen Nikitovic and Mats Brorsson

Department of Microelectronics and Information Technology,

Royal Institute of Technology, KTH

Electrum 229, SE-164 40 Kista, Sweden

email: {Mladen.Nikitovic, Mats.Brorsson}@imit.kth.se

ABSTRACT

Power consumption has become an increasingly important factor in the field of computer architecture. It affects issues such as heat dissipation and packaging cost, which in turn affects the design and cost of a mobile terminal. Today, a lot of effort is put into the design of architectures and software implementation to increase performance. However, little is done on a system level to minimize power consumption, which is crucial in mobile systems.

We propose an adaptive chip-multiprocessor (CMP) architecture, where the number of active processors is dynamically adjusted to the current workload need in order to save energy while preserving performance. The architecture is suitable in future mobile terminals where we anticipate a bursty and performance demanding workload.

We have carried out an evaluation of the performance and power consumption of the proposed architecture using previously validated high-level simulation models. Our experiments show that orders of magnitude in power consumption can be saved compared to a conventional architecture to a negligible performance cost. The method used is complementary to other power saving techniques such as voltage and frequency scaling.

Categories and Subject Descriptors

C.1.4 [Processor Architectures]: Parallel Architectures – *Mobile processors*.

General Terms

Measurement, Performance, Design, Experimentation.

Keywords

Mobile terminals, chip-multiprocessor (CMP), power consumption, energy-aware scheduling.

1. INTRODUCTION

Power consumption has played an increasingly important role in the field of computer architecture and will continue to do so in the future. As the number of available transistors on a chip increases, so does power consumption, which indirectly affects heat dissipa-

tion and packaging costs. This is currently a major problem for high-performance computers. Mobile terminals, such as personal digital assistants (PDAs) and mobile phones, depend on batteries as the main source of power. Unfortunately, the advances in battery technology have increased the available energy by only a few percent per year so far. Also, future mobile terminals will require more performance than most of today's embedded architectures can deliver. Considering the visions published by the Wireless World Research Forum, it is obvious that the performance demands on the computing platform in future mobile terminals will be very challenging [16].

The need for high performance will be highly dependent on the behavior of the load, which is anticipated to be bursty in these systems. For long periods, the performance demand will be negligible, which puts the terminal into standby-mode. At other times, such as during video encoding and/or relaying between different radio standards using software defined radio (SDR), the need for performance will be high, causing peaks in performance demand.

Common approaches to enhancing the performance of a computing platform involve increasing the clock frequency or utilizing some sort of parallelism. Increasing the clock frequency is effective but may require redesigning the circuits to deal with the shorter cycle time, but most importantly, it increases the dynamic power consumption (see equation 2 below). Since the supply voltage limits the clock frequency, it might be necessary to increase this factor as well, resulting in a cubic increase in power consumption. This is clearly not acceptable in an energy-effective mobile terminal.

A common way of using parallelism in high-performance computers is to utilize the available *instruction-level parallelism* (ILP) in the applications. This is a hardware-intensive approach to improved performance and gives diminishing returns making it unsuitable in embedded mobile systems [15]. A more promising approach to achieve high system performance is to utilize the available *thread-level parallelism* (TLP) which we believe will be common in systems we are considering. With several processors on one chip we can build a chip-multiprocessor (CMP) that can easily scale performance efficiently. However, the increased performance can cause the power consumption of the system to increase if no effort is put into making it energy-efficient.

A device uses energy by consuming power statically and dynamically. Dynamic power is consumed due to switching of transistors and short circuit currents during transistor switches (eq. 1). Static power is consumed due to leakage currents dissipated from the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CASES 2002, October 8-11, 2002, Grenoble, France.

Copyright 2002 ACM 1-58113-575-0/02/0010...\$5.00

device. Today, switching power accounts for about 90 percent of the total power consumption and leakage power for less than 10 percent. In this paper, we mainly tried to minimize the dynamic power. However, it is anticipated that power consumption due to leakage will increase to about 50 percent in the future with the advent of new process technologies.

Switching power depends on several parameters: voltage (V_{dd}), frequency (f), capacitive load (C_L) and the probability of switching (α) (eq. 2).

$$P_{\text{total}} = P_{\text{static}} + P_{\text{switch}} + P_{\text{short circuit}} \quad (\text{eq. 1})$$

$$P_{\text{switch}} \sim V_{dd}^2 \cdot f \cdot C_L \cdot \alpha \quad (\text{eq. 2})$$

From (eq. 2) we can see that a quadratic reduction in power consumption can be achieved by reducing the input voltage. Another possibility is to reduce the frequency. These techniques—voltage and frequency scaling—can reduce overall power consumption at the expense of performance. The performance cost comes from reducing the frequency, thus lowering the rate of execution. Another factor is the added delays when changing the voltage and frequency dynamically during execution.

As a complement to voltage and frequency scaling, we propose an adaptive chip-multiprocessor (ACMP) architecture that has two main goals: to satisfy the performance need of future mobile applications and at the same time minimize the energy consumed by the system. The proposed CMP architecture will minimize switching power by shutting off inactive processor cores dynamically. This is achieved by using the existing power modes of the embedded processor cores integrated on the CMP. Although several CMPs exist today as either commercial products or academic projects [2, 8, 9, 11, 12, 14], few have considered high-level power reduction techniques [6, 10] and we found no published results of their efficiency compared against competing systems. The proposed CMP architecture is adaptive in the sense that the available performance and power consumption can be altered dynamically. This is accomplished by using run-time operating system information to shut down inactive processors and thereby save energy. Several strategies are possible; one can maximize the computing power by activating as many processors as possible or deactivate them if the power consumption has a higher priority.

We have evaluated the proposed architecture using a high-level simulation technique and an example of a bursty multiprogrammed workload consisting of Commbench application programs [17]. The results show that the used energy can be reduced by an order of magnitude while still meeting performance demands using a four-processor ACMP running at 200 MHz as compared to a uni-processor running at 800 MHz.

The rest of the paper is organized as follows. Section 2 describes the proposed ACMP architecture. Section 3 describes the experimental methodology we have used for estimating the performance and power consumption. Section 4 then presents the results from simulation using different energy-saving techniques and compar-

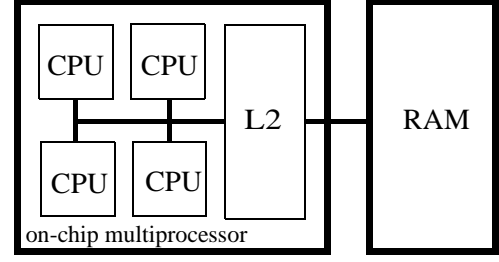


Figure 1. An ACMP with four embedded cores.

ing the proposed adaptive CMP architecture to a more traditional embedded uni-processor system. Section 5 discusses the results and draws some final conclusions.

2. ARCHITECTURE MODEL

Figure 1 shows the proposed architecture, which is a shared memory on-chip multiprocessor with a number of embedded processor cores. Each processor core has separate instruction and data caches and translation-lookaside buffers (TLBs). The processors share a common unified 2nd level cache on-chip through a shared bus.

This is in itself a rather conventional CMP model. The novel approach here lies in the capability of dynamically switching processors on and off as the need arises. This opens up a number of possibilities. The one that we are investigating in this paper is the ability to disable inactive processors when they are not needed for performance reasons. However, future work will look at the possibility of very fast context switches by activating processors in which a process context already resides in the registers and cache memories. The same technique can be used for ultra-fast interrupt handlings. However, in this paper we are only concerned with the capability of dynamically adjusting power and performance to the processing need.

We utilize the different power-saving modes often found in today's embedded cores [1]. The processors usually run in normal mode when using their full capability. In standby mode, the clock tree is disabled, which eliminates most of the dynamic power. Finally, in dormant mode the processor consumes minimal leakage power and no dynamic power. Table 1 also shows how many cycles are needed to reenter run mode from each state.

Table 1: Processor power modes

Mode	Description	Recovery Time
RUN	Processor executing instructions.	> 1 cycle
STANDBY	Processor clocks stopped.	> 10 cycles
DORMANT	Processor power rails off. Context need to be loaded from external memory.	> 100 cycles

As stated earlier, a processor module in dormant mode will consume minimal leakage power, making this mode effective in future processor technologies when static power consumption becomes increasingly significant. On the other hand, it requires more time to recover from such a state. The 100 cycles given in table 1 only refer to the recovery of processor state. In addition to this, the cache memory needs to be reloaded with relevant contents.

The ACMP architecture we are considering can be run in three different modes: *normal*, *adaptive*, and *perfect adaptive*. In normal mode, processors not executing any user applications will execute an idle process, thus still consuming power. In adaptive mode, processors assigned to execute the idle process will be deactivated to save energy. They can be deactivated in two ways: either they are put into standby mode, which shuts off the clock tree, or they are put into dormant mode, where leakage currents from the processor are minimized as well. When returning from dormant mode, the processor must load its context from the main memory, thus introducing a delay. In adaptive mode, we put half of the inactive processors into standby mode and the other half into dormant mode. We assume that a processor that is about to be deactivated initiates this algorithm which might lead to another processor moving from the standby mode to the dormant mode before putting itself in the standby mode.

In perfect adaptive mode, the processors are deactivated perfectly, which means that they can be deactivated or activated in one cycle, consuming zero power when deactivated. The motivation for this seemingly unrealistic mode is that it allows the optimal energy saving to be estimated this way.

3. EXPERIMENTAL METHODOLOGY

3.1 Simulation model

We have modified and augmented an existing simulator from the SimpleScalar simulation suite in order to estimate the performance and power saving effects of our proposed architecture [4]. The SimpleScalar suite is validated against existing processor models and is a common and accepted tool in the computer architecture community. The processor cores are modeled with a functional simulator assuming each instruction takes one clock cycle to execute. We augmented the original simulator, *sim-cache*, to be a multiprocessor simulator where the processor cores are scalar RISC processors running on a frequency of typically 200 Mhz at less than one volt. Each processor core has an instruction and data write-back cache and an instruction and data TLB. The processor cores share a unified level 2 on-chip write-back cache. Each core is connected to a shared bus which is arbitrated in a round-robin fashion. The modeled ACMP chip is finally connected to an off-chip RAM running at 133 Mhz.

The multiprocessor version of *sim-cache* was implemented by duplicating the processor core, the level 1 caches and the TLBs. Furthermore, the original inner loop in the simulator was modified to a state machine to be able to handle concurrent bus requests. To support execution of multiple processes on each processor and to enable time-sharing on the processor cores, a simple scheduler kernel with a period of 20 milliseconds was implemented. It was integrated into the simulator source code and should not be regarded as one of the user processes, thus it does not add any overhead in terms of performance or energy. The processes in the workload are also independent of each other so no cache coherence is required in this model. The simulator starts each simulation session by reading a workload file containing a list of release times and filenames of process-traces created by a trace-generation tool from the SimpleScalar suite, *sim-eio*. Processes are dynamically allocated to processors, thus the scheduler assigns a process to an idle processor or the processor with least amount load during run-time.

Finally, the ACMP simulator, *sim-acmp*, was augmented with power consumption models for the processor cores, caches, interconnect, off-chip bus and off-chip RAM memory. All on-chip models were configured to produce results according to 0.18 μ m process technology. The processor cores are functionally simulated and their power consumption is estimated by assigning an energy estimate, base cost, for each executed instruction. The base cost was derived from typical embedded processor specifications [5]. It has been shown that the power consumption of an embedded processor can be accurately measured by assigning a base cost for each executed instruction [13]. The cache hierarchy is simulated using existing performance models from SimpleScalar and power consumption is estimated using models from Wattch, which is a simulator based on one of the SimpleScalar simulators and extended with power consumption models [3]. The interconnect is modeled as a 10 mm, 512-bit shared bus with round-robin arbitration and its power consumption is estimated using wire capacitance models [18]. Also, the off-chip bus and DRAM memory is modeled using an existing power model [7].

All power models are assumed to consume maximum power when referenced, and zero dynamic power otherwise. This corresponds to the *cc1* clock gating mode in Wattch. Static power is only considered during simulation in adaptive mode, where processors turn off the clock tree, but still dissipate leakage currents. In that case, 10% of maximum power is approximated as leakage power and added to the total power.

Table 2 shows the various architecture configurations that have been simulated and evaluated.

3.2 Workload

The workload of mobile terminals is usually divided into smaller processes or threads and tend to have a bursty behavior due to unpredictable user interaction. This means that besides some periodical tasks, the workload is usually very modest. An user interaction with the terminal causes a number of processes to execute, forming workload peaks that demand high performance. Also, the applications are highly parallel by nature, which means that they can be easily distributed among the ACMP processors.

To model the bursty behavior, a workload scenario has been created using applications from the Commbench benchmark suite [17]. Figure 2 shows how the selected applications were organized to represent a workload scenario where processes are started at various time intervals. The workload consists of periodic and aperiodic processes. The periodic processes (FRAG and DRR) represent tasks that need to be done on a regular basis on a mobile terminal while the aperiodic processes are typically activated upon an external event like user interaction. We assume a situation where the performance demand of each separate program can be satisfied with a single processor in the ACMP running at 200 MHz. Figure 2 shows a situation where each process can run on its own processor. With the previously explained assumption and the given workload scenario it can be seen that at most five processors are needed to satisfy the performance demand. When fewer than five processors are used, some programs will be executed using time-sharing. A process always finishes execution on the processor to which it was allocated; thus no load balancing or process migration is utilized.

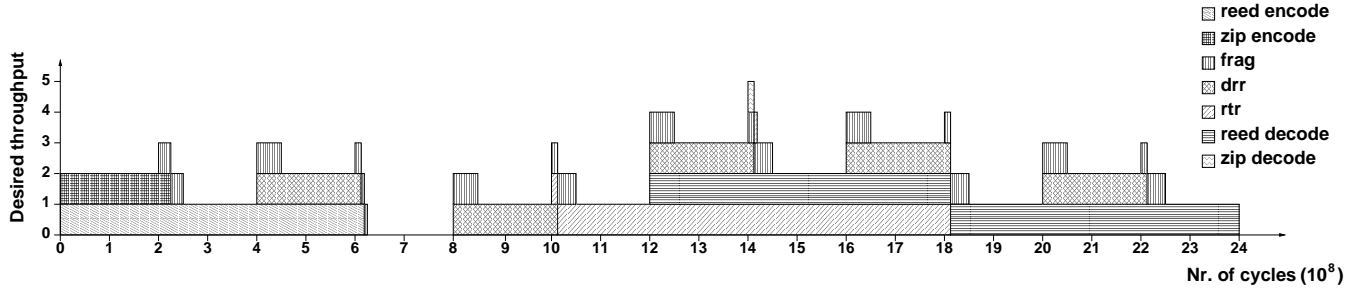


Figure 2. Workload scenario using the Commbench applications.

Table 2: ACMP configurations used in the experiments.

Model	Power-saving modes	Processor parameters
ACMP _x _N	No adaption. All processors execute all the time.	200 MHz, 0.7 V, x processors, 16/ x kB instruction and data L1 caches. 64 kB unified L2 cache. 0.275 nJ per inst.
ACMP _x _A	Adaption where an idle processor is put into the standby mode. Half of the idle processors are put into dormant mode.	
ACMP _x _PA	Perfect adaption into dormant mode without execution time overhead.	
SP_N	No power-saving mode. Total running time set to the shortest ACMP execution time.	800 MHz, 1.65 V, single-processor, 16 kB instruction and data L1 caches. 64 kB unified L2 cache. 1.125 nJ per inst.
SP_PA	Perfect power-saving mode when idle. Total running time set to the shortest ACMP execution time.	
SP_S	Scaled performance to match the need. Never idle.	450 MHz, 1.09 V, single-processor, 16 kB instruction and data L1 caches. 64 kB unified L2 cache. 0.556 nJ per inst.
SP	No power-saving mode. Total execution time set to when program finishes all tasks.	800 MHz, 1.65 V, single-processor, 16 kB instruction and data L1 caches. 64 kB unified L2 cache. 1.125 nJ per inst.

The Commbench benchmark suite consists of a number of applications aimed at producing a workload typical in telecommunication environments. The applications can be distinguished with respect to how data is processed. The header-processing applications (HPA) only process packet headers whereas the payload-processing applications (PPA) process the entire dataset as a single packet. A performance and energy characterization of the selected programs from Commbench listed below can be seen in tables 4 and 5. In this characterization, each application was executed using the ACMP simulator with a single processor configuration according to table 3. The processor is assumed to execute one instruction per cycle if memory is not referenced. Otherwise, memory stall cycles are added to the execution time. The power consumption of each module is estimated using known models or figures from existing components [3, 5, 7, 18]. The results from the energy characterization show that applications with high memory stall-rate have a low average power consumption. This is because the modeled second-level cache and the off-chip memory are very energy efficient but slow upon access; thus the energy is consumed over a long period of time, which results in a low average power consumption.

The header-processing applications perform operations on a per-packet basis independent of the size and packet payload type. The Radix-Tree Routing (RTR) application performs a table lookup on every packet in a datagram-based network, and on every connection in a connection-based network. In FRAG, packets are split into

Table 3: Processor configuration for benchmark characterization.

Module	Performance model	Power (mW)
CPU	600 Mhz @ 1.3 Volt (0.18 μ m), 1 inst/cycle	450
L1 I-cache (IL1)	4 kB, write-back, directly mapped, 32 B line, 6 cycle miss penalty	154
I-TLB	32 entries, 2-way associative, 4 kB pages, 1000 cycle miss penalty	28
L1 D-cache (DL1)	4 kB, write-back, 4-way associative, 32 B line, 6 cycle miss penalty, 32-entry	411
D-TLB	32 entries, 2-way associative, 4 kB pages, 1000 cycle miss penalty	57
L2 unified cache (UL2)	16 kB, write-back, direct mapped, 64 B line, 50 cycle miss penalty	91
Off-chip bus	64 bit wide	154
Off-chip memory	256 MB DRAM, 133 Mhz @ 3.3 V	352

multiple fragments and additional operations like header adjustment and checksum computation are performed. The Deficit Round Robin (DRR) fair scheduling algorithm is commonly used in switches.

Table 4: Performance characterization of selected Commbench applications.

Application	Dataset	# cycles (10 ⁶)	# inst (10 ⁶)	miss rates
RTR	32,239 entries of routing prefixes and IP addresses.	1,414	811	IL1 : 1.66% DL1: 0.34% UL2: 8.72%
FRAG	100,000 packets of 20 bytes size, maximum fragment size: 567	58	43	IL1 : 0.02% DL1: 1.98% UL2: 19.14%
DRR	1,000,000 packets, quantum size: 10	425	213	IL1 : 0.03% DL1: 6.72% UL2: 4.40%
ZIP_ENC	Default html code, compression level 6	2,399	227	IL1 : 0.00% DL1: 19.59% UL2: 62.11%
ZIP_DEC	Encoded html code	132	40	IL1 : 0.08% DL1: 2.87% UL2: 24.17%
REED_ENC	Default html code	761	623	IL1 : 0.02% DL1: 0.16% UL2: 7.50%
REED_DEC	Encoded html code	1,428	1,205	IL1 : 0.03% DL1: 0.15% UL2: 10.41%

Table 5: Energy characterization of selected Commbench applications.

Application	# mem ref (10 ³)	Memory stall rate	Power (mW)	Energy (J)
RTR	Loads: 245,966 Stores: 58,046	19%	490	1.15
FRAG	L: 7,637 S: 3,921	7%	593	0.06
DRR	L: 104,484 S: 21,836	18%	507	0.36
ZIP_ENC	L: 47,560 S: 15,104	87%	147	0.59
ZIP_DEC	L: 8,512 S: 1,846	62%	252	0.06
REED_ENC	L: 100,773 S: 33,639	0.4%	608	0.77
REED_DEC	L: 146,876 S: 68,455	0.4%	612	1.457

Payload processing applications usually consist of an encoding and decoding section that operates on a stream of packets. ZIP is a data compression application based on the Lempel-Ziv (LZ) algorithm. The Reed-Solomon Forward Error Correction scheme is implemented in the REED application. It adds redundancy to data transmissions.

4. EXPERIMENTAL RESULTS

Figure 3 shows a comparison of the performance as normalized execution time for each ACMP and uniprocessor configuration (see Table 2) running the example workload scenario. The graph is normalized to the fastest uniprocessor configuration (SP).

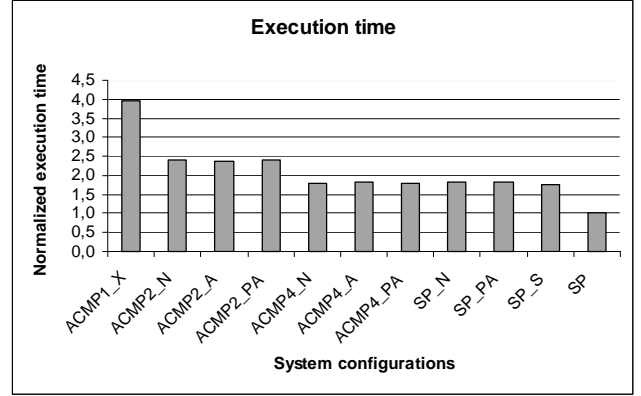


Figure 3. The performance of the different models.

The ACMP architecture is varied in the number of processors (1, 2, 4), cache sizes, and previously described power-saving modes (normal, adaptive, perfect adaptive). The uniprocessor is executed in normal mode (SP_N), then in dormant when inactive (SP_PA) and finally voltage and frequency is scaled to 450 Mhz at 1.09 V to save energy while providing just about the same processing power as the ACMP with four processors (SP_S). For the sake of this study, we assume that the necessary computing power is satisfied with an ACMP with four processors. The execution time of the ACMP with four processors will serve as a reference for how long the uniprocessor will be running. Furthermore, we assume that the amount of chip area is limited. This means that the first level caches sizes are decreased as more processors are added to the architecture. The single-CPU ACMP has data and instruction caches of 16 kB each and a 64 kB level 2 cache. When another processor is added, the level 1 caches will be halved to 8 kB each and a four-processor ACMP has 4 kB caches.

As expected, the ACMP configuration with one processor running at 200 MHz is four times slower than the reference processor. When we add one processor to the ACMP configuration, we almost get a linear speedup and the execution time is further reduced using four processors. Since the workload processes are totally independent, this is not a controversial result.

There are several reasons for not achieving perfectly linear speedup. First, the ACMP configurations could perform better using load balancing, which is not implemented here. As cache sizes decrease, their miss rate increases, which also degrades performance. Since the workload is bursty, the ACMP processor utilization decreases when the idle time increases, which contributes to the performance degradation. Another factor is that there are applications that require the processor computing power for a long time, thereby keeping the processors occupied while others might be inactive. Examples of such applications are reed_encode, rtr, and reed_decode.

The results also show that the performance overhead when running in adaptive mode as compared to perfect adaptive mode is very small. In a two-processor ACMP, the overhead is non-existent because both processors are occupied during the majority of the simulation and therefore have few opportunities to enter power-saving modes. In a four-processor configuration, more processors

are inactive and therefore able to enter power-saving mode. However, this configuration did not introduce a significant performance degradation. The estimated performance overhead amounted to only 3 percent, which is due to several possible factors. First, the state transition time is quite low considering the total simulation time. Second, the performance of the processors does not seem to decrease significantly even though their cache contents need to be reloaded from main memory when recovering from dormant mode. This could indicate that the applications have a relatively small data set, which could be reloaded rapidly upon context loss.

One of the prerequisites in our experiments is that the performance of a four-processor CMP is sufficient for the workload needs. In order to make a fair comparison with a uniprocessor configuration we scaled the clock frequency to 450 MHz which is represented by the SP_S configuration in figure 3. As seen, it has about the same performance as the four-processor CMP models. The energy cost per instruction and voltage level for the scaled processor was interpolated from previously known values [5].

Any power reduction technique is useless if it reduces the performance more than it improves power consumption. Not only is it annoying for the user with low performance, but it might the final energy consumption the same as before, or even higher, unless the power consumption is reduced more than the performance is reduced. In relation to the executions times of each process as seen in figure 2, the overheads of adapting the number of processors are very low and we do not anticipate any significant effect on the execution times. This can be verified in figure 3 where we can see that the adaptive models (ACMPx_A) have only a slight performance degradation due to the adaptation process. We will now see the effect of the adaptive algorithm on the power consumption which we hope will be more substantial.

Figure 4 depicts the energy consumption of the different architecture configurations, broken down in energy consumed in the various system components. The energy bars are normalized against the energy of the uniprocessor without any energy-saving techniques (SP).

First of all we can note that the dominating amount of energy is spent in the CPU and the level-one data and instruction caches. The rest of the memory system and interconnect is not very energy-consuming. The reason for this is that the level-one caches are accessed on every clock cycle and since the miss-rates with one exception (see table 4) are very low, the energy spent in the rest of the memory system becomes almost negligible.

Moreover, the results show that the total energy of the ACMP architecture in normal mode first tends to decrease when going from a single processor to a two-processor configuration. The decrease in energy consumed is small. There are several reasons why the reduction is so modest. The two-processor configuration is first of all almost twice as fast as the single-processor configuration. One could then argue that the total energy should also decrease to about half of the single-processor configuration. The reason is that there are cases when an application is executed on one of the processors while the other processor is inactive. The inactive processor is executing the idle process, which is very energy-consuming because its high hitrate causes high stress to the instruction cache and the CPU pipeline. When moving to a four-

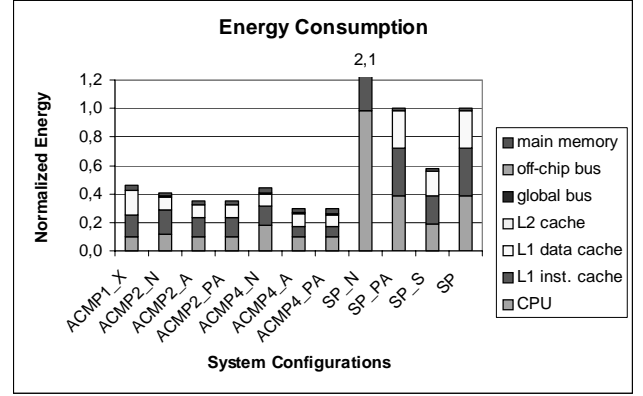


Figure 4. Energy distributions in the different models.

processor configuration, the total energy increases to a level higher than that of the single-processor configuration. Although this configuration is faster than the previous configurations, the number of processors executing the idle process is higher, which results in even more wasted energy.

In adaptive mode (ACMPx_A), power is saved by putting the processor in either standby or dormant mode. This reduces almost all of the wasted energy because dynamic power, the source of the majority of energy consumption today, is eliminated in both standby and dormant mode. In adaptive mode half of the idle processors are kept in standby mode and the rest in dormant mode. At least one processor must be active to be able to activate or deactivate the other processors. When using two processors, one processor is held dormant when possible and the other is kept in active state. This removes the majority of the idle process execution and introduces a small performance degradation since very few state transitions are made due to high processor utilization. The energy savings can be noted in the CPU and instruction cache. These components suffer the most when running the idle thread. The four-processor configuration saves even more energy by utilizing the available parallelism and thereby executing faster. The energy overhead compared to perfect adaptive mode was 2 percent.

The perfectly adaptive configuration (ACMPx_PA) shows how much energy could be saved in the optimal case. As the idle process time increases with the number of processors, so do the energy savings in the optimal case, since idle processors are optimally deactivated. One can see that a four-processor configuration consumes about a third of the energy than the reference uniprocessor. Another observation is that there is a very small energy overhead in the adaptive case compared to the perfect adaptive case.

The voltage and frequency scaled processor (SP_S) uses less energy than the reference processor because it can run on a lower operating voltage. As stated earlier, this has a quadratic effect on power savings. However, the reduced voltage level also reduces the maximum operating frequency. This results in a longer execution time than the reference processor, which degrades its energy-efficiency. The energy savings and performance degradation results in a energy-efficiency almost equal to the reference processor.

Figure 5 depicts the energy-efficiency of each configuration with respect to its performance. The efficiency is measured using the Energy-Delay product where the energy from figure 4 is multiplied by the execution time in figure 3. As expected, the single-processor ACMP configuration is not very energy-efficient mainly due to its poor performance. When adding a processor, both execution time and energy consumption are reduced, resulting in about the same EDP values as the reference processor. The adaptive ACMP architecture with four processors proves to be about twice as energy-effective compared to the reference processor and orders of magnitude more efficient than the uniprocessor with no special power modes

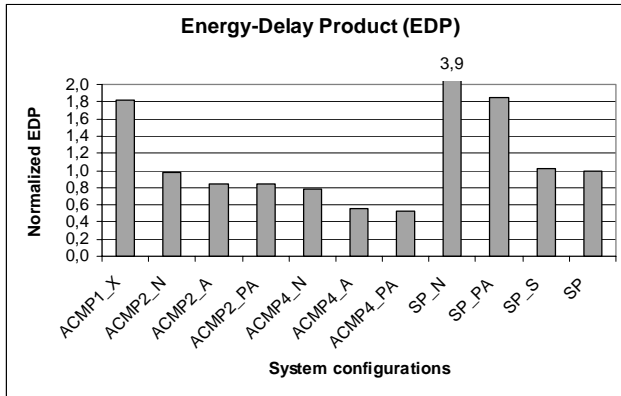


Figure 5. The Energy-Delay Product of the different models.

5. CONCLUSIONS

We have proposed an adaptive chip-multiprocessor architecture suitable for future mobile terminals with high processing demands and bursty workloads. The idea is to adapt the number of processors dynamically to the current workload needs. To our knowledge, this is the first study to report on performance and energy effects of this kind of architecture under these premises.

Using high-level simulation models we have shown that an adaptive CMP can save an order of magnitude in energy as compared to a uniprocessor system with no power reduction methods. Even if adaption is not used, we have shown that a CMP architecture is more energy-efficient than the powerful uniprocessor architecture which would have been needed to cope with the peak performance demands. We have shown that considerable energy can be saved by deactivating processors whose computing power is not utilized.

We have in this study used a simulation based algorithm to decide when processors should go to a power-saving state. In reality, the decision should, of course, be taken by the operating system. Future studies will investigate operating system algorithms for this purpose, and we will then also have the possibility to study more elaborate scheduling policies.

We will also look at the possibility to switch off parts of a processor to share cache memory state for fast context switching, or interrupt servicing, without reloading cache state. Another important issue is to look at representative workloads with real-time constraints and with inter-process communications.

6. ACKNOWLEDGMENTS

The research in this paper is partly supported by the Swedish Foundation for Strategic Research in the INTELECT programme.

7. REFERENCES

- [1] ARM Ltd, *ARM1022ETM Technical Reference Manual*, ARM DDI 0237 A, <http://www.arm.com>
- [2] L. A Barroso et al., Piranha: A Scalable Architecture Based on Single-Chip Multiprocessing, in *Proceedings of the 27th International Symposium on Computer Architecture*, 2000, pp. 282-293.
- [3] D. Brooks, V. Tiwari, M. Martonosi, Wattch: A Framework for Architectural-level Power Analysis and Optimizations, in *Proceedings of the 27th International Symposium on Computer Architecture*, 2000, pp. 83-94.
- [4] D. Burger, T. Austin. *The SimpleScalar Tool Set, Version 2.0*, Technical Report CS-TR-97-1342, University of Wisconsin, June 1997.
- [5] L. T. Clark, An Embedded 32-b Microprocessor Core for Low-Power and High-Performance Applications, in *IEEE Journal of Solid-State Circuits*, volume 36, issue 11, pp. 1599-1608, November 2001.
- [6] M. Edahiro, S. Matsushita, M. Yamashina, N. Nishi, A Single-Chip Multiprocessor for Smart Terminals, in *IEEE Micro*, July-August 2000, Volume 20 Issue 4, pp. 12-30.
- [7] R. Fromm et al., The Energy Efficiency Of Iram Architectures, in *Proceedings of IEEE International Symposium on Computer Architecture*, pp. 327-337, 1997.
- [8] L. Hammond et al., The Stanford Hydra CMP, in *IEEE Micro*, March-April 2000, Volume 20, Issue 2, pp. 71-84.
- [9] A. Kowalczyk et al., The first MAJC Microprocessor: a Dual CPU System-on-a-Chip, in *IEEE Journal of Solid-State Circuits*, Volume 36, Issue 11, pp. 1609-1616. November 2001.
- [10] T. Koyama, K. Inoue, H. Hanaki, M. Yasue, E. Iwata, A 250-MHz Single-Chip Multiprocessor for Audio and Video Signal Processing, in *IEEE Journal of Solid-State Circuits*, Volume 36, Issue 11, pp. 1768-1774, November 2001.
- [11] L. Sang-Won, S. Yun-Seob, K. Soo-Won, O. Hyeong-Cheol, H. Woo-Jang, RAPTOR: A Single Chip Multiprocessor, *The First IEEE Asia Pacific Conference on ASICs*, pp. 217-220, 1999.
- [12] U. Schmidt, K. Caesar, Datawave: A Single-Chip Multiprocessor for Video Applications, in *IEEE Micro*, Volume 11, Issue 3, pp. 22-25, pp. 88-94, June 1991.
- [13] A. Sinha, A.P Chandrakasan, JouleTrack-a Web Based Tool for Software Energy Profiling, in *Proceedings of Design Automation Conference*, pp. 220-225, 2001.
- [14] J. M Tedler, S. Dodson, S. Fields, H. Le, B. Sinharoy, *Power4 System Architecture, Technical White Paper*, October 2001.
- [15] D. Wall, Limits of Instruction-level Parallelism, in *Proceedings of the Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ACM, pp. 176-188, 1991.
- [16] Wireless World Research Forum, *Book of Visions 2001*, <http://www.wireless-world-research.org>
- [17] T. Wolf, M. Franklin, CommBench-A Telecommunications Benchmark for Network Processors, in *IEEE International Symposium on Performance Analysis of Systems and Software*, pp. 154-162, 2000.
- [18] Z. Yan, M. J Irwin, Energy-Delay Analysis for On-Chip Interconnect at the System Level, in *Proceedings of IEEE Computer Society Workshop On VLSI*, pp. 26-31, 1999.