Energy and Performance Analysis for Multi-threaded Tasks on A Multi-Processor System

Blind

ABSTRACT

Multiprocessor systems offer superior performance and potentially better energy-reduction than single-processor systems. It all depends, however, on how well the application can be mapped onto the architecture. Indeed, a careful tradeoff of energy and performance requires a thorough understanding of the energy consumption pattern across the architecture, both in hardware and software. This paper proposes MultiPo-Sim, a cycle-accurate simulator to estimate performance and energy consumption in multiprocessor systems. MultiPo-Sim can profile the energy per hardware module, and can also analyze the behavior of the application running on the multiprocessor. Several applications illustrate the capabilities of MultiPo-Sim, including a fingerprint minutiae detection program and a data-flow image encoder. The fingerprint detection program, which is data parallelized, consumes most energy on the computation of the algorithm (88%~99%). The data-flow image encoder, which is task parallelized, consumes most energy on inter-process communications (51%~69%). By using MultiPo-Sim, designers can easily evaluate the system characteristics and evaluate if the parallelism of the algorithm matches the parallelism of the architecture.

1. Introduction

Multiprocessor systems-on-chip (MPSOC) have been proposed as a way to achieve high performance as well as low energy consumption [1][2]. Multiple cores on the chip offer higher parallelism and thus potentially higher performance. In addition, by lowering the supply voltage and operating frequency for processor cores, the energy consumption of the system can be reduced significantly. However, in order to design an energy efficient multiprocessor system while still maintaining a given performance, a thorough understanding of the energy consumption patterns is required.

There are three design issues of concern when evaluating the energy-efficiency of a multiprocessor system. Each of these issues is specific for multiprocessor systems, and does not play a significant role in single-processor systems. The first one is the use of hardware synchronization hardware modules in the multiprocessor system. These synchronization modules help in the resource management among multiple processor units, and they have impact on both performance and energy consumption of multiprocessor systems. The second issue is the use of resource management software in application program running on the multiprocessor system, such as for example semaphores or mutexes in multithreaded programs. This software also causes extra execution cycles as well as energy consumption. The third issue is the partitioning of the application over the processors into parallel, communicating tasks. Applications can be partitioned according to their major data streams, or according to their composing subtasks. This gives rise to two different parallelizing strategies, called data- and task parallelizing. Each strategy has a different impact on the application characteristics and the additional overhead caused on the multiprocessor architecture. We will have a further discussion on these issues in section 5 and 6.

This paper proposes a cycle-accurate framework, MultiPo-Sim, to analyze both the performance and energy consumption of a multiprocessor system. Besides cycle count, MultiPo-Sim can evaluate the energy consumption of each module in the multiprocessor system, such as cores, caches and the central bus interconnect. Moreover, MultiPo-Sim can also trace the program running on each core and profile the energy consumption of a specific sub-function or primitive. This feature can be used to extract the energy spent on the synchronization among processor cores. MultiPo-Sim also supports processor cores with voltage/frequency scaling, and returns the impact of scaling on performance as well as energy consumption. The modular configuration of MultiPo-Sim also enables to simulate different multiprocessor architectures easily. MultiPo-Sim can simulate 331,500 cycles per second for a four-processor system on a 3GHz, 512MByte Fedora-2 PC. Two multi-threaded applications, the fingerprint minutiae detection program and a data-flow image encoder, are used as the drivers to explore the multiprocessor system architecture. With different parallelization strategies, the two applications result in different energy and performance characteristics.

This paper is organized as follows. Section 2 briefly discusses the related research efforts on the estimation of energy consumption of microprocessor systems. Section 3 introduces the power models used in the paper. Section 4 gives a more detail discussion on the proposed multiprocessor system and also explains how the MultiPo-Sim has been implemented. Two applications with different parallelizing schemes are introduced in section 5. The performance and energy results are shown in section 6. Section 7 will draw the conclusion.

2. Prior Art

The estimation of energy-consumption and power modeling of processor-based systems has been widely studied. Due to the large size and high complexity of processor-based systems, it is not practical to use the low-level power estimation tools such as PowerMill and QuickPower. Several techniques are being used to abstract the power model and speed up the power estimation process. Instruction-based power analysis uses the instructions or instruction-pairs and associates them to power models [3]. Micro-architectural power models [4][5] are proposed to provide a more accurate power model. Power macro models are constructed based on the knowledge of the internal micro-architectures. Because it models the micro-architecture in more detail, the simulation time is longer. Macro-modeling is proposed to connect the architectural power model to the application software. The program behavior, such as sub-routine calls[6] and system calls[7] are used to estimate the energy consumption.

Several power estimation platforms have also been developed by extending cycle-accurate simulators with power models [8][9][10]. A cycle-accurate simulator provides detailed behavior of a processor-based system, including the internal pipelining, the cache access behavior, and the communication to the external system. Usually the cycle-accurate behavior will be ported into power models for different components, such as data-path logic, memory, and interconnect.

Although a lot of research effort has been done on power modeling for processor-based systems, most of them are focusing on the single processor system. M.Loghi et. al [11] proposed a cycle-accurate power analysis for a multiprocessor SoC. By using the cycle-accurate behavior, the authors combine the power and performance models of different components. However, they can only breakdown the energy consumption for different hardware modules.

In this paper, we propose the MultiPo-Sim, which is a cycle-accurate energy/performance simulation platform for multiprocessor systems. In addition to profiling hardware modules, MultiPo-Sim can also analyze software functions, which gives the designers insight into the energy consumption behavior of the multiprocessor system.

3. Power Model

Fig.1 shows an instance of the multiprocessor. A shared memory architecture [13] is chosen to reuse as much as possible from existing single processor systems. There are four ARM cores with private instruction and data caches. Each ARM has a specific voltage/frequency(V/f) module which controls the operating clock frequency and the supply voltage to achieve energy scaling. Besides ARM processors, there are two other components in the system: a hardware test-and-set lock to support inter-process communication and synchronization, and a memory interface to access off-chip memory. The system uses a central bus as the medium to connect the modules.

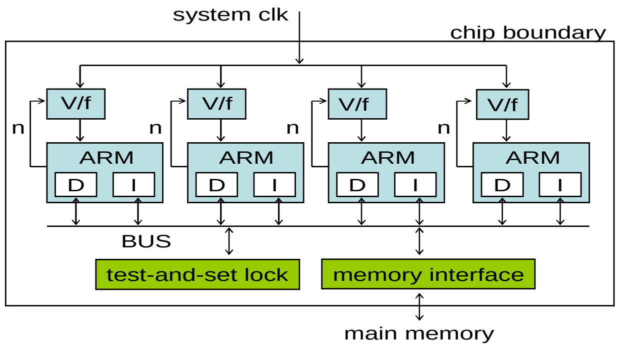


Fig.1: A shared memory multiprocessor system with energy scaling

Based on this multi-processor architecture, we categorize the power model into three different components: processor cores, caches, and synchronization modules. We apply different models for each component. Based on the cycle-accurate behavior of each component, the power model will estimate the energy consumption. The power models are developed for 0.18um CMOS technology.

3.1 Processor Core (without Cache)

The processor core used in MultiPo-Sim is a 5-stage pipelined StrongARM microarchitecture. While we have a cycle-accurate model of the ARM core [23], developing a detailed power model for the core itself is not trivial, especially when energy scaling techniques are supported. Therefore, MultiPo-Sim uses the average power consumption to determine the energy consumption of the processor core. We choose a processor core from ARM Corp., ARM966E-S[23], which is similar to the ISS of MultiPo-Sim in both architecture and performance. We use the performance characteristics of the processor core provided by ARM Corp., which shows the nominal operating frequency and the average power consumption of ARM966E-S core without cache are 200MHz and 0.70 mW/MHz respectively.

The V/f units in Fig.1 provide different voltage/frequency operation modes, or power modes, for each ARM processor. The power modes are controlled by the application. In order to model the energy scaling capability of the processors, we use similar voltage/frequency scaling characteristics as the LART platform[14]. The LART platform uses SA-1100 processor core, which is also a StrongARM-based processor. The SA-1100 runs at 251MHz with the supply voltage of 1.65V, and 59MHz with the supply voltage of 0.79V. The energy scaling ratio (V^2f ratio) of the high frequency mode and low frequency mode is 18.5. These characteristics are mapped to the processor cores used in MultiPo-Sim. The ARM core in MultiPo-Sim supports two steps of energy scaling. We use the same supply voltages for high/low power modes as in the LART, and keep energy scaling ratio the same by adjusting the operating frequency. The energy scaling characteristics are shown in Table-1.

Table-1: Power characteristics of the processor cores

	Pouwelse[12]	Energy-Scaled ARM Core in MultiPo-Sim
Processor	StrongARM	StrongARM
V/f high power (V/MHz)	1.65 / 251	1.65 / 200
V/f low power (V/MHz)	0.79 / 59	0.79 / 47
Frequency(f) ratio	4.25	4.25
V^2f ratio (high/low)	18.5	18.5

Besides dynamic energy consumption, there is also static energy consumed by the system. A lot of techniques have been proposed at circuit and architecture level to reduce the static energy consumption [14][15]. In this paper we are using 0.18um technology, thus the static energy does not occupy a significant portion of the total energy consumption of the system. The static energy of a 0.18um CMOS circuit is typically less than 1% of the total energy dissipation. This paper will focus on the dynamic energy consumption. However we do believe that the static energy would play an important role when more advanced semiconductor technology is used.

3.2 Cache

In both single processor and multiprocessor systems, caches usually consume a significant portion of the total energy [11][16]. Therefore we need a cache power model which is accurate enough to reflect the energy consumption of the caches. Thanks to the cycle-accurate simulation provided by MultiPo-Sim, we can trace the access behavior to the caches as precise as every clock cycle.

Sim-Panalyzer[9] is a energy estimation tool for the ARM architecture. It is an augmentation to the SimpleScalar performance simulator [17] attaching the power models for the components and using the cycle-accurate behavior provided by SimpleScalar as the input of the power model. The latest version of Sim-Panalyzer provides very detailed energy estimation models for the components which account for a large portion of the total energy of the processor core, such as L1 caches and clock trees. We port the cache model used in Sim-Panalyzer into our platform. By providing the access behavior of the cache, the power model of the cache can return the energy consumption.

The Sim-Panalyzer cache model is divided into two parts, the tag model and the memory bank model. These two models are similar but with different parameters, e.g. size of the memory bank. Based on the data of the 0.18um technology, the Sim-Panalyzer cache model first calculates the effective capacitances of switching (C_{switch}), internal ($C_{internal}$) and leakage ($C_{leakage}$). The size of the effective capacitances will change based on the parameters such as the size of the memory, number of the bit lines, number of the ports and which type of the port (read, write, or read/write), etc. Due to the cycle-accurate simulation, the cache power model uses the actual access data from the processor as the inputs. The number of the switchings on the cache lines is calculated by actually counting the value differences on the cache line. The total energy consumption within a certain clock cycle can be estimated by:

$$\{(\#\text{ of switchings}) * C_{switch} + C_{internal} + C_{leakage}\} * SV^2 \qquad (1)$$

where SV^2 is the square of the supply voltage.

3.3 Central Bus and Synchronization Modules

In addition to the processor cores and caches, a multiprocessor system also requires a medium as the interconnect of the system and modules to handle the synchronization among processors. In this paper, the interconnect is implemented with a central bus architecture, and multiprocessor synchronization is implemented with a test-and-set lock (as shown in Fig.1).

Central Bus. An energy model for bus-based interconnect is not a trivial matter. Different bus architectures and techniques used on the bus raise the complexity to analyze the energy consumption on the bus, and consequently also the complexity to model it. Since we use relatively simple bus scheme, the energy is mainly consumed by the switching activity on the bus wires. We therefore model the bus as a long wire on the chip, and use Berkeley Predictive Technology Model (BPTM) [18] to model the wire capacitance of the bus interconnect. The length of the bus is estimated as the following:

$$\text{Bus Length} = 2 * \text{Sqrt}(A_{total\_core} + A_{total\_cache}) \qquad (2)$$

Equation (2) reflects the length of the sum of the width and height of the chip die where A_{total_core} and A_{total_cache} are the area of the processor cores and the caches respectively. Using the accumulated area as the die size somewhat underestimates the impact of the inefficiency of the placement and the bus-interconnect. However, recent semiconductor technology usually has multiple metal layers which can mitigate the area increased by global interconnect. Moreover, modern processor systems, including multiprocessor systems, are highly customized design in terms of performance and area optimization. Therefore we believe equation (2) can provide an appropriate first-order estimation.

We assume the bus uses inverter-based drivers and receivers (Fig.2(b)), which will add a load of 0.05pF to 0.1pF according to TSMC 0.18um technology. In this paper, we use 0.1pF as the load for each node on the bus. Thus more processors connected to the bus will increase the total load on the bus, which consequently raises the energy consumption for each switching on the bus.

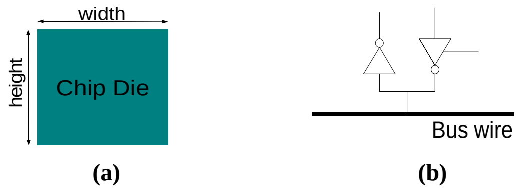


Fig.2 : (a) The height and width of the chip die (b) Driver and receiver connected to the bus wire

External memory interface. The external memory interface supports a connection to an off-chip memory system. It is composed of a control logic unit and data registers. Due to its low complexity, the energy consumption will be dominated by the large receiver- and driver buffers connected to the bus. Therefore we use the energy consumption of the buffer to represent the power model of the external interfaces.

Test-and-set lock. The test-and-set module manages the access to the locks of the resources. It contains 16 registers to store the value of the lock (1 means lock and 0 means free) for a specific resource and control logic to handle the test-and-set protocol. Same as the external interface, the simple functionality of test-and-set lock makes its bus buffer the dominant energy consumption component. We use the energy consumption of the buffer to represent the power model of the test-and-set lock.

4. Multiprocessor Platform

4.1 Multiprocessor Architecture

In this paper, we use a shared memory multiprocessor architecture (Fig.1). Each processor core has its own data and instruction cache. The caches implement a write-through policy. Whenever the cache writes on the bus, the transaction will also be observed by all the other caches. Therefore, the cache coherency has been sustained by snooping on the write-transactions. Each processor has a 4K instruction cache and a 4K data cache.

A hardware test-and-set lock is implemented to provide the synchronization function of the system. This is used as a semaphore and mutexes for the shared resources. Whenever a processor wants to access a shared resource, e.g. shared memory, it has to acquire the lock for this resource. The lock will be released after it has finished accessing the resource. The so-called spin-lock scheme[19] has been used as the protocol for the processor to grab the lock: when the processor wants to acquire the lock, it will repeatedly check the lock until the processor owns it. It currently supports 16 different locks to handle the resources synchronization in the system.

The hardware modules are connected by a central bus. The central bus uses a master-slave transaction-based scheme. Each master can initiate read- or write transactions to the slave. Only one transaction can be conducted on the bus at a time, and there is no support for split transaction.

The multithread programming model is based on the QuickThreads cooperative multithreading library [20]. The main thread will first create other threads and then calls start to initiate the execution of multithreading. After threads are created, thread context information is stored in a queue. When a processor is idle, it will check the queue and take one thread if there is any thread needs to be executed.

The multiprocessor system supports two energy scaling modes, a high power mode and a low power mode as explained in section 3. The switching between different power modes is controlled by the application. Note that this energy scaling control scheme is orthogonal to the other energy scaling techniques. It can work with other energy scaling control techniques in different levels as well, e.g. the energy scaling control by a embedded OS[25].

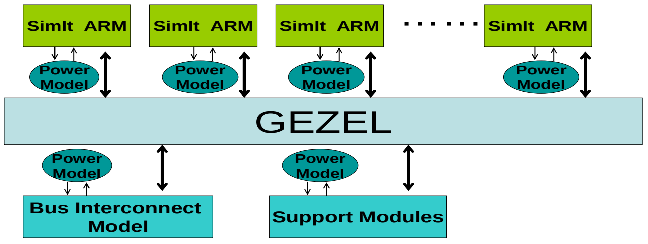


Fig.3: The platform architecture of MultiPo-Sim

4.2 Multicore-Power Simulator (MultiPo-Sim)

We construct a cycle-accurate simulation platform, called MultiPo-Sim, which can analyze both the performance and energy consumption of a shared-memory multiprocessor architecture shown in Fig-1. MultiPo-Sim is developed by combining SimIt-ARM[21] and GEZEL[22]. SimIt-ARM is an ISS (Instruction Set Simulator) for the ARM ELF instruction set. It models a 5-stage pipeline ARM micro-architecture. The parameters of the instruction/data caches, such as associativity, block size, cache line size and etc, can also be reconfigured. GEZEL is an environment which facilitates the exploration, simulation and implementation of domain-specific micro-architectures. It uses finite-state-machine-datapath (FSMD) semantics, which allows designers to capture the datapath and control operations of hardware models independently.

The simulator architecture of the MultiPo-Sim is illustrated in Fig.3. GEZEL provides a common platform which can co-simulate different numbers of processors, bus interconnect and other system modules. Each processor is modeled by one SimIt-ARM ISS. The power models are attached to each module in the MultiPo-Sim. The energy consumption is estimated based on the cycle-accurate behavior of each module. Due to the cycle-accurate simulation feature, the MultiPo-Sim profiles the performance of the system in terms of cycle count. In addition, MultiPo-Sim can also trace the application to evaluate the cycle count and the energy consumption spent on a specific function of the program. Later on we will use this capability to profile the energy consumption spent on the test-and-set lock of the multiprocessor systems.

MultiPo-Sim uses a modular configuration. Each module in the system can be added or modified individually. This feature makes the MultiPo-Sim a very flexible platform. For example, MultiPo-Sim can easily simulate the system with different numbers of processor cores by only changing parameters in the configuration file. The multithread programming model does not assume a detailed knowledge on the system architecture. Therefore, the same application can be run on different multiprocessor systems without changing the program. The multiprocessor architecture shown in Fig.1 is an instance which can be simulated on MultiPo-Sim and demonstrate energy and performance characteristics of a multiprocessor system.

5. Applications

In order to take advantages of the computation power provided by the multiprocessor system, applications need to be parallelized. There are basically two schemes to parallelize the application, data parallelization and task parallelization.

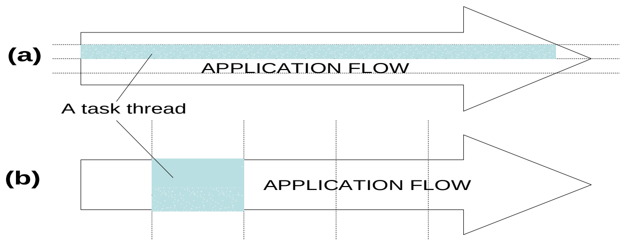


Fig.4: (a) Data and (b) task parallelization

We define the application flow as the flow of information from the input to the output of the system. Given an application flow from left to right, the process of the application can be partitioned in parallel with the application flow, which we call data-parallel processing (Fig.4(a)). Each partition is implemented as a task thread and can be executed by a processor core. Usually the task threads in a data parallelized system are independent of each other and do not require the synchronization very often. The other way is to parallelize the process of the application according to the individual tasks in the application flow, which we call a task-parallel processing (Fig.4(b)). Each task thread can be executed by any processor core. However, threads depend on the data passed by the other thread before them. These two parallel system architectures have different impact on the energy consumption and performance. We use two applications, a fingerprint minutiae detection and a data-flow image encoder, to demonstrate the characteristics of a data-parallel system and a task-parallel system respectively.

Fingerprint minutiae detection (Data-Parallel). The minutiae detection algorithm takes a 256 by 256 gray scale fingerprint image as the input. Multiple phases of the image processing have been conducted on the fingerprint and the minutiae are generated as the output of the algorithm. The reference algorithm is a fixed-point version of the NIST fingerprint software [24]. The multithreaded fingerprint minutiae detection program partitions the fingerprint image into four different sections. As shown in Fig.5, each image section has 144 by 144 pixels of the fingerprint image, and will be initiated as an individual thread.

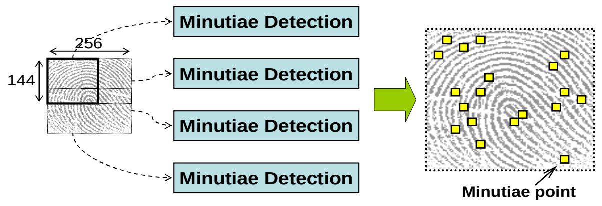


Fig.5: Multithreaded fingerprint minutiae detection

Data-flow Image Encoder (Task Parallel). The second application is a high throughput image encoder which is implemented as a data-flow system. Fig.6 illustrates an instance of a data-flow system. It is consists of actors of different operations. The actors are communicating through the intermediate queues, and these queues are located in the external memory of the multiprocessor systems. Thus every access to the queues is translated as the traffic on the bus. In the data-flow image encoder, each actor is implemented as a thread and can be executed by any processor core in the system. There are total 32 actors in the system.

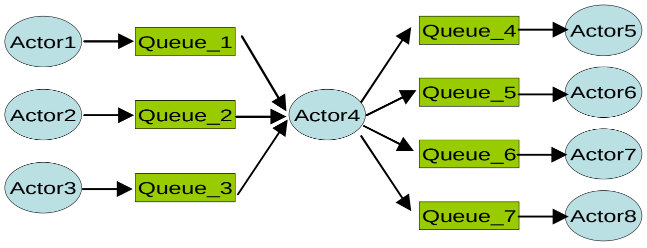


Fig.6: Data-flow application

6. Energy and Performance Analysis

The MultiPo-Sim can trace both energy consumption and performance of specific components of hardware as well as software. The two applications have been applied onto different processor schemes, including the system with single-processor, dual-processor and quad-processor. Different power modes are also used to evaluate the amount of energy saving and the impact on the performance. We use different labels to indicate the configurations. For example, 2HL represents the dual-processor architecture with one processor in high-power mode and another one in low-power mode, while 2HH means a dual-processor architecture with both processor cores running at high-power mode.

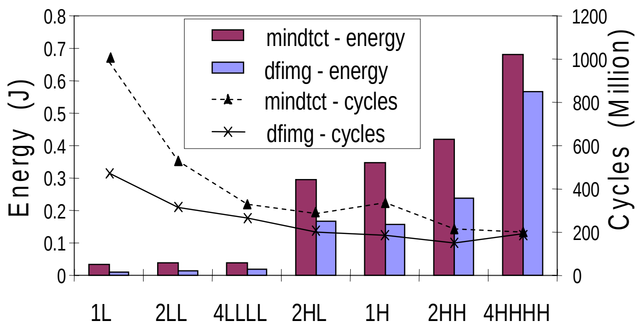


Fig.7: Energy consumption and execution cycles on different processor schemes

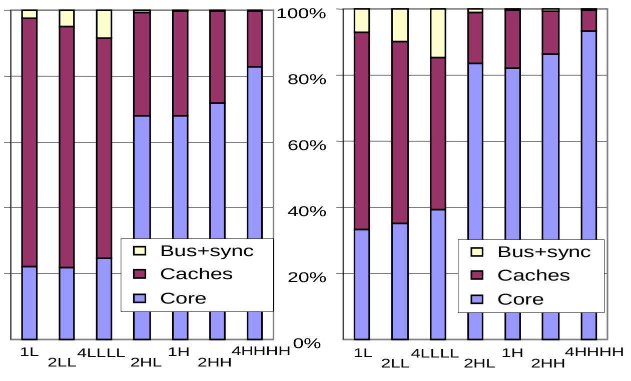
6.1 System Performance Enhancement

Fig.7 shows the execution cycles (lines) and energy consumption (bars) of the minutiae detection (mindtct) and the data-flow image encoder (dfimg). The basic trend shows that increasing the number of processor cores and using the high power mode will enhance the overall system performance. However, the speed-up does not linearly reflect the number of processor cores. For example, in mindtct, the 2HH scheme is 36% faster than 1H scheme, while 4HHHH scheme has only 40% speed-up compared to 1H. In dfimg, the 4HHHH scheme is even slower than the 2HH and the 1H scheme. The main reason is that the bus and the test-and-set lock can only allow the access from one processor at a time. If more than one processor would like to access them, the processors have to line up in order to access these resources. The sequential properties of the bus and the test-and-set compensate the potential performance enhancement provided by multiple cores.

6.2 Energy Scaling

The energy consumption of different processor schemes basically shows a reverse trend as the execution cycles. The increasing of the number of processor cores and using the high power mode will consume more energy. The energy scaling technique applied on the multiprocessor system reduces the total system energy consumption significantly. However, due to the additional parallelism and computation power provided by multiple cores, the overall system performance has been sustained (or even better) when comparing to a single processor system. For example, in mindtct, 2HL scheme is 15% faster than the single processor nominal case (1H) while consuming only 86% of the 1H. With a slightly 2% faster than 1H scheme, the 4LLLL consumes even only 12% of the total energy consumption of the 1H. This proves that the multiprocessor system can provide energy reduction as well as faster execution.

Fig.8 shows the normalized energy consumption breakdown for different components in the system. For the processor schemes with high power mode, the processor cores consume the most energy, 67% to 82% for mindtct and 82% to 93% for dfimg. The caches dominate the energy consumption in the schemes with low power mode, 66% to 75% for mindtct and 46% to 60% for dfimg. Compared to the cores and the caches, the central bus and the synchronization modules do not consume too much energy (0.8% to 8%).



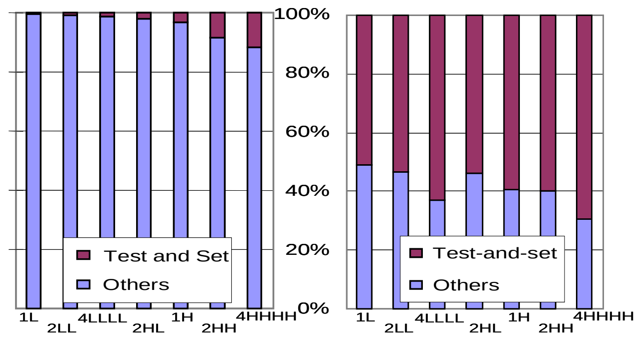
(a) mindtct **(b) dfimg**

Fig.8: Normalized energy consumption of different HW components for two different applications

6.3 Synchronization Overhead

Processor cores are synchronized using a inter-process communication function, which is implemented as a test-and-set function in the multiprocessor architecture used in this paper. MultiPo-Sim traces the execution of the application and profiles the energy spent on the test-and-set operation for each module. Fig.9 shows normalized energy of the test-and-set operations. Note that in this figure the test-and-set operation represents the energy spent on the inter-process communication for each module in the system, including processor cores, caches, bus interconnect, memory interface as well as the hardware test-and-set module.

As we mentioned before, the task threads of data parallelized applications are almost independent from other task threads. Therefore, there is a low amount of synchronization in the system. As illustrated Fig.9(a), the energy consumption for the test-and-set function in the mindtct occupies 0.4% to 12% of the total energy consumption. However, in data parallelized applications, the actors need to pass and receive the data to(from) other actors. Therefore the synchronization is happening frequently. Fig.9(b) shows that out of 51% to 69% of the total energy in dfimg is consumed by the test-and-set function.



(a) mindtct **(b) dfimg**

Fig.9: Normalized energy consumption of test-and-set operation for two different applications

Given the energy characteristics of applications with different parallel strategies, designers can adapt the application model towards the underlying architecture. For instance, a data parallelized application can benefit more by running on a bus-based multiprocessor system as shown in this paper. Or vice versa, designers can optimize the multiprocessor architecture to execute a given multithreaded application. For both cases, MultiPo-Sim can provide the necessary information to the designers.

7. Conclusion

In this paper, we propose a cycle-accurate simulator, MultiPo-Sim, which can profile both the performance and energy consumption of hardware modules as well as software functions. MultiPo-Sim also supports energy scaling of processor cores. In addition to high simulation speed, the modular configuration also enables MultiPo-Sim to simulate different multiprocessor architectures easily. With these features, MultiPo-Sim returns the thorough understanding of the performance and energy characteristics of a multiprocessor system.

Given a data parallelized application, the task threads are independent from each other and result in a low amount of synchronization in the system. Therefore the processor cores and caches consume the most of the total energy. However, the synchronization happens frequently for a task parallelized application, which results in a large portion of the total energy consumed by test-and-set functions. By using MultiPo-Sim, designers can easily evaluate the system characteristics and choose the appropriate architecture to achieve energy efficiency and high performance.

References

[1] L.Hammond, B.A.Nayfeh, K.Olukotun, "A Single-Chip Multiprocessor," *Proc. of IEEE*, pp.79-85, Sept. 1997.

[2] A.Jerraya, W.Wolf, "Multiprocessor Systems-on-Chips," Morgan Kaufmann, Sept 2004, ISBN 0-12-385251-X.

[3] V.Tiwari, S.Malik, A.Wolfe, "Power Analysis of Embedded Software: a First Step Towards Software Power Minimization," *IEEE VLSI Systems*, Vol. 2, no. 4, pp.437-445, Dec. 1994.

[4] D. Brooks et al., "Power-Aware Micro-Architecture: Designand Modeling Challenges for Next-Generation Microprocessors," *IEEE Micro*, Vol. 20, no. 6, pp. 24-44,Nov.-Dec. 2000.

[5] N. Vijaykrishnan, M. Kandemir, M. Irwin, H. Kim, W. Ye, D.Duarte, "Evaluating Integrated Hardware-Software Optimizations using a Unified Energy Estimation Framework," *IEEE Trans. on Computers*, Vol. 52, no. 1, pp. 59-76,Jan. 2003.

[6] M. Lajolo, A. Raghunathan, S. Dey, L. Lavagno, "Cosimulation-Based Power Estimation for System-on-Chip Design," *IEEE VLSI Systems*, Vol. 10, no. 3, pp. 253-266, June 2002.

[7] R. Dick, G. Lakshminarayana, A. Raghunathan, N. Jha,"Analysis of Power Dissipation in Embedded Systems using Real-Time Operating Systems," *TCAD*, Vol. 22, no. 5, pp. 615-627, May 2003.

[8] D.Brooks, V.Tiwari, M.Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations," ISCA, pp.83-94, 2000.

[9] Sim-Panalyzer Project, http://www.eecs.umich.edu/~panalyzer/

[10] G.Contreras, M.Martonosi, J.Peng, R.Ju, G.-Y.Lueh, "XTREM:A Power Simulator for the Intel Xscale Core," LCTES, pp115-125, 2004.

[11] M.Loghi, M.Poncino, L.Benini, "Cycle-Accurate Power Analysis for Mutliprocessor System-on-a-chip," GVLSI, pp.401-406, Apr. 2004.

[12] J. Pouwelse, K. Langedoen, H. Sips, "Application-directed voltage scaling," IEEE Trans. on VLSI Systems, 11(5):812—826.

[13] HP bookJ. Hennessy, D. Patterson, "Computer Architectures: A quantitative approach," MKP Publishers, 2002.

[14] Kao, J., S. Narendra, A. Chandrakasan. "Sub-threshold Leakage Modeling and Reduction Techniques," *ICCAD 2002* (Embedded Tutorial) , pp. 141-148, San Jose, California, November 2002.

[15] S. Martin, et. al "Combined Dynamic Voltage Scaling and Adaptive Body Biasing for Lower Power Microprocessors under Dynamic Workloads," ICCAD 2002, pp.721-725.

[16] J. Montanaro, et al, "! 160-MHz, 32-b, 0.5-W CMOS RISC Microprocessor" IEEE JSSC, pp1703-1714, 1996.

[17] Simple Scalar, http://www.simplescalar.com/

[18] Berkeley Predictive Technology Model (BPTM), http://www-device.eecs.berkeley.edu/~ptm/

[19] G. Andrews, "Concurrent programming - principles and practice", 102—105, Benjamin Cummings Publ. 1991.

[20] D. Keppel, "Tools and Techniques for Building Fast Portable Threads Packages," UWCSE 93-05-06, U. Washington, 1993.

[21] SimIt-ARM, http://simit-arm.sourceforge.net/

[22] GEZEL Project, http://www.ee.ucla.edu/~schaum/gezel/

[23] ARM Corp, http://www.arm.com/

[24] S.Yang, K.Sakiyama, I.Verbauwhede, "A Compact and Efficient Fingerprint Verification System for Secure Embedded Systems," 37th Asilomar Conference, Nov. 2003.

[25] Energy Aware Linux/RK, http://www-2.cs.cmu.edu/~rtml/