Energy and Performance Analysis of Mapping Parallel Multi-threaded Tasks for An On-Chip Multi-Processor System

Bo-Cheng Charles Lai
EE Department
UCLA
CA 90095-1594
bclai@ee.ucla.edu

Patrick Schaumont
ECE Department
Virginia Tech.
VA 24061
schaum@ee.ucla.edu

Wei Qin
ECE Department
Boston University
MA 02215
wqin@bu.edu

Ingrid Verbauwhede
EE Dept. UCLA, CA
and
ESAT, K.U.Leuven, BE
Ingrid@ee.ucla.edu

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 of the application across the architecture. We develop a simulation platform, MultiPo-Sim, which returns the cycle-accurate performance and energy consumption of a multiprocessor system, for both hardware components and software primitives. On the hardware level, energy scaling techniques can be modeled and each processing core can operate at different energy modes. MultiPo-Sim achieves 331K cycles per second simulation speed for a four-processor system on a 3GHz, 512MByte Fedora-2 PC. On the software level, data parallelizing and task parallelizing are two common models of multi-thread programming. By using MultiPo-Sim, we show that they show different energy and performance characteristics when mapping onto a multi-processor system.

1. Introduction

Multiprocessor systems-on-chip (MPSOC) have been proposed as a way to achieve high performance as well as low energy consumption [1]. 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. The contributions of this paper are two fold. The first contribution is that we develop a cycle-accurate multiprocessor simulation platform, MultiPo-Sim, which enables the simulation of multi-threaded tasks on a multiprocessor system and returns both performance and energy consumption. The second contribution is to explore the energy and performance characteristics of the different models of multi-threaded tasks on a multiprocessor system.

This paper is organized as follows. Section 2 gives the prior art about the estimation of energy consumption of processor-based systems. Section 3 discusses the data and task parallelizing models. Two applications are introduced to represent each model. Section 4 introduces the multiprocessor system and simulator used in this paper. Section 5 explains the power models used in the MultiPo-Sim. Section 6 shows the performance and energy consumption results. The conclusion will be drawn in section 7.

2. Prior Art

The estimation of energy-consumption and power modeling of processor-based systems have been widely studied. Cycle-accurate simulators are used to develop the power estimation platforms so that both energy and performance can be evaluated[2][3]. However, most of them are focusing on the single processor systems rather than multiprocessor systems. M.Loghi et. al[4] 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, their work can only breakdown the energy consumption for different hardware modules and does not support energy scaling for processing cores. The main contributions of the proposed multiprocessor simulator in this paper, the MultiPo-Sim, are simulation of energy scaling of individual processor cores in addition to the ability to trace the energy/performance of the software components.

3. Multi-threading Models

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.

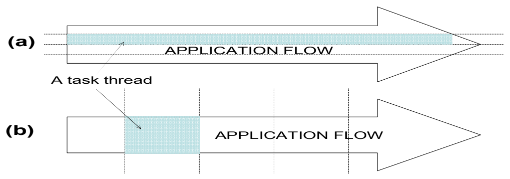


Fig.1: (a) Data and (b) task parallelization

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.1(a)). Each partition is implemented as a task thread and can be executed by a processor core. 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.1(b)). These two parallel system architectures have different impact on the energy consumption as well as performance. We use two applications, a fingerprint minutiae detection and a high throughput image encoder, to demonstrate the characteristics of a data-parallel model and a task-parallel model respectively.

The fingerprint minutiae detection algorithm is a fixed-point version of the NIST fingerprint software [11]. The multithreaded fingerprint minutiae detection program partitions the fingerprint image into four different sections. As shown in Fig.2(a), each image section has 144 by 144 pixels of the fingerprint image, and will be initiated as an individual thread. The second application is a high throughput image encoder which is implemented as a data-flow system. Fig.2(b) illustrates an instance of a data-flow system. It consists of actors of different operations. The actors are communicating through the intermediate queues, and each actor is implemented as a thread. There are a total of 26 actors in the system.

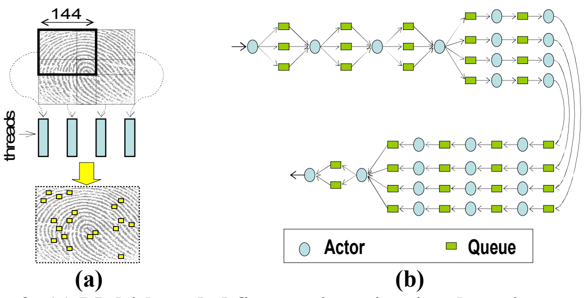


Fig.2: (a) Multithreaded fingerprint minutiae detection (b) Data-flow image encoder application

4. Multiprocessor Platform and MultiPo-Sim

In this paper, we use a shared-memory multiprocessor architecture. Fig.3 shows an instance of the multiprocessor system with four ARM processors. Each processor core has a data and instruction cache. For each processor, there is 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 different hardware modules. Note that the bus interfaces, the memory interfaces and the test-and-set lock are always operating at nominal speed.

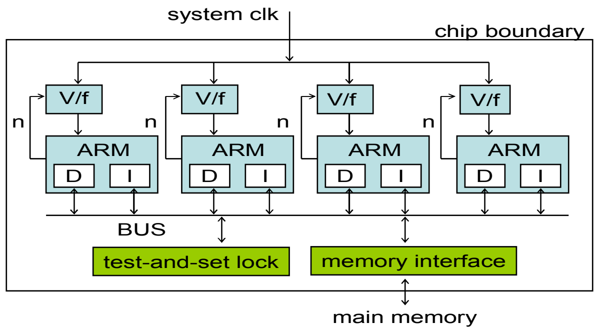


Fig.3: A shared memory energy-scaled multiprocessor system

We develop a cycle-accurate simulation platform, called MultiPo-Sim, which can analyze both the performance and energy consumption of a shared-memory multiprocessor architecture. MultiPo-Sim is developed by combining SimIt-ARM[8] and GEZEL[9]. 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 hardware module. The energy consumption is estimated based on the cycle-accurate behavior of each module. In addition, MultiPo-Sim can also trace the application to evaluate the cycle count and the energy consumption spent on a specific function of a program.

5. Power Model

We categorize the power model into three different hardware components: processor cores, caches, and synchronization modules. The power models are developed based on 0.18um CMOS technology. Note that MultiPo-Sim does not model the off-chip energy consumption, such as off-chip memory and interconnect.

5.1 Processor Core (without Cache)

Since we are modeling the energy consumption of a complete multiprocessor system, the breakdown of the energy consumption of each module is more important than profiling the detailed energy consumption of a single core. Therefore MultiPo-Sim chooses the average power consumption to estimate the energy consumption of the ARM processor core. We use a processor core from ARM Corp., ARM966E-S[10], which is similar to the ISS of MultiPo-Sim in both architecture and performance. We choose 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.

In order to model the energy scaling capability of the processors, we use similar voltage/frequency scaling characteristics as the LART platform[5]. 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 ARM cores used in MultiPo-Sim to support two steps of energy scaling.

Table-1: Power characteristics of the processor cores

	Pouwelse[5]	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

This paper focuses on the dynamic energy consumption. We realize that the static energy will play an important role when the system size is growing larger and more advanced semiconductor technology is used. With an adequate power model, this can be integrated in MultiPo-Sim and is in our future research plan.

5.2 Cache

In both single processor and multiprocessor systems, caches usually consume a significant portion of the total energy [6]. Sim-Panalyzer[12] is a energy estimation tool for the ARM architecture. The latest version of Sim-Panalyzer provides very detailed energy estimation models for caches, which we port to our platform. By using the cycle-accurate cache access behavior, such as the number of switchings on the cache line, from the processor as the inputs, the power model returns the energy consumption of the cache.

5.3 Central Bus and Other Modules

Central Bus. Due to the moderate system clock frequency, 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 the Berkeley Predictive Technology Model (BPTM) [7] to model the wire capacitance of the bus interconnect. We approximate the length of the bus as the following:

$$\text{Bus Length} = 2 * \text{Sqrt}(A_{total\_core} + A_{total\_cache}) \qquad (1)$$

Equation (1) 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. We assume the bus uses inverter-based drivers and receivers, which will add a load of 0.05pF to 0.1pF according to TSMC 0.18um technology.

Test-and-set lock and Memory interface. Due to their low complexity, the energy consumption of these two components 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.

6. Detailed Energy and Performance Analysis

Fig.4 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. The energy consumption of different processor schemes basically shows a reverse trend as the execution cycles. Increasing 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.

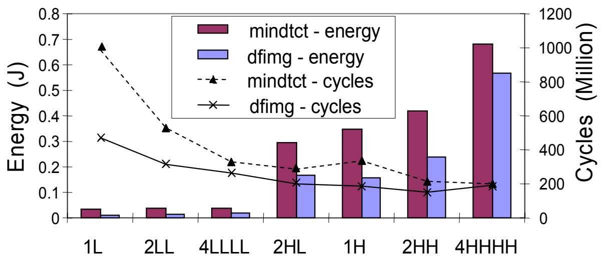
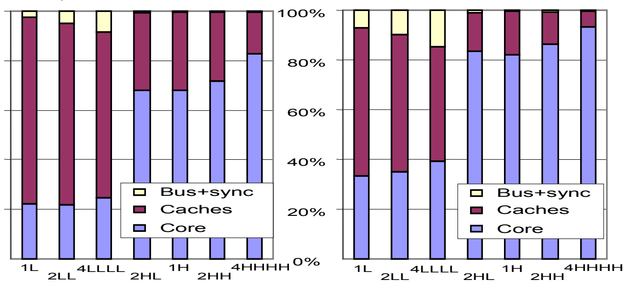


Fig.4: Energy consumption and execution cycles on different processor schemes

Fig.5 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%).

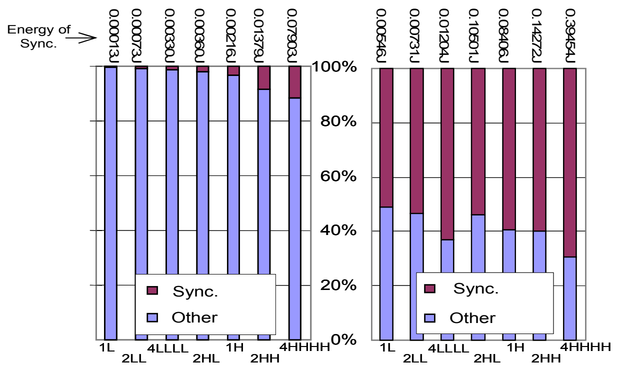


(a) mindtct **(b) dfimg**

Fig.5: Normalized energy consumption of different HW components for two different applications

Processor cores are synchronized using a test-and-set function. Fig.6 shows the normalized energy of the synchronization operations. Note that, in this figure, the synchronization 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.

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 (0.4% to 12%). However, in data parallelized applications, the actors need to pass and receive the data to(from) other actors. The synchronization is happening frequently and consumes 51% to 69% of the total energy in dfimg application.



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

Fig.6: Normalized energy consumption of synchronization operation for two different applications

7. Conclusion and Future Work

MultiPo-Sim, a multiprocessor simulator, profiles both the performance and energy consumption of hardware modules as well as software functions. Given a data parallelized application, the task threads result in a low amount of synchronization in the system. The synchronization happens frequently for a task parallelized application, which results in a large portion of the total energy consumed by synchronization operations.

8. Acknowledgement

The authors gratefully acknowledge the support of NSF (Grant CCR 0310527) and SRC (Grant SRC-2003-HJ-1116).

References

[1] L.Hammond, B.A.Nayfeh, K.Olukotun, "A Single-Chip Multiprocessor," *Proc. of IEEE*, pp.79-85, Sept. 1997.
[2] D.Brooks, et. al "Wattch: a framework for architectural-level power analysis and optimizations," ISCA, pp.83-94, 2000.
[3] Sim-Panalyzer Project, http://www.eecs.umich.edu/~panalyzer/
[4] M.Loghi, M.Poncino, L.Benini, "Cycle-Accurate Power Analysis for Mutliprocessor System-on-a-chip," GVLSI, pp.401-406, Apr. 2004.
[5] J. Pouwelse, K. Langedoen, H. Sips, "Application-directed voltage scaling," IEEE Trans. on VLSI Systems, 11(5):812—826.
[6] J. Montanaro, et al, "160-MHz, 32-b, 0.5-W CMOS RISC Microprocessor" IEEE JSSC, pp1703-1714, 1996.
[7] Berkeley Predictive Technology Model (BPTM), http://www-device.eecs.berkeley.edu/~ptm/
[8] SimIt-ARM, http://simit-arm.sourceforge.net/
[9] GEZEL Project, http://www.ee.ucla.edu/~schaum/gezel/
[10] ARM Corp, http://www.arm.com/
[11] S.Yang, K.Sakiyama, I.Verbauwhede, "A Compact and Efficient Fingerprint Verification System for Secure Embedded Systems," 37th Asilomar Conference, Nov. 2003.
[12] Sim-Panalyzer Project, http://www.eecs.umich.edu/~panalyzer/