國立交通大學

資訊科學與工程研究所

碩 士 論 文

以硬體協助之多核心嵌入式系統
效能與耗能評估工具

Hardware-Assisted Performance/Energy Evaluation Tool for Multi-core Embedded System

研 究 生：林俊瑋

指導教授：曹孝櫟　博士

中 華 民 國　九 十 八 年 八 月

以硬體協助之多核心嵌入式系統效能與耗能評估工具

Hardware-Assisted Performance/Energy Evaluation Tool for Multi-core Embedded System

研 究 生：林俊瑋　　Student: Jyun-Wei Lin

指導教授：曹孝櫟　　Advisor: Shiao-Li Tsao

國 立 交 通 大 學
資 訊 科 學 與 工 程 研 究 所
碩 士 論 文

A Thesis
Submitted to Institute of Computer Science and Engineering
College of Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in

Computer Science

August 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年八月

以硬體協助之多核心嵌入式系統效能與耗能評估工具

學生：林俊瑋　　　　　　　　指導教授：曹孝櫟

國立交通大學 資訊科學與工程研究所 碩士班

摘要

有效的效能與耗能評估是嵌入式系統設計時期的關鍵技術。然而，傳統的評估方法難以兼具快速以及精確。此外在多核心逐漸應用於嵌入式系統的今日，傳統的評估方法將面臨更為複雜及困難的挑戰。有鑑於此，本文提出一種以硬體協助且適用於多核心嵌入式系統的效能與耗能評估工具，並予以實現。所提之工具提供程式執行時期硬體事件的監控，並藉此推算出元件耗能，其可避免軟體取樣(Sampling)所造成的額外負擔，以便能呈現系統原始的行為與特性。實驗結果顯示，所提之方法可於 100 MHz 的四核心仿真環境下進行，精度可達微秒以下，其兼具快速、精細且真實的特性將有助於複雜多核心嵌入式系統設計時期之細部評估與分析。

Hardware-Assisted Performance/Energy Evaluation Tool

for Multi-core Embedded System

Student: Jyun-Wei Lin Advisor: Shiao-Li Tsao

Institute of Computer Science and Engineering

National Chiao Tung University

Abstract

Effective performance and energy evaluation of embedded systems is one of the critical issues during design phase. However, conventional approaches suffer from difficulties to provide fast and accurate evaluation of the system, especially for those embedded systems using multi-core technology. In this thesis, we propose and realize a hardware-assisted performance and energy evaluation tool for a multi-core embedded system. Our approach provides hardware monitor for runtime programs, and uses these monitor information to estimate the system energy consumption without introducing extra software sampling overhead. The experimental results show that our approach can work at a 100 MHz quad-core emulation platform. The profiling granularity is higher than microsecond. It provides fast and fine-grained evaluation of the multi-core embedded system during the design phase.

誌謝

能順利完成本論文，首先要感謝指導教授曹孝櫟博士，在碩士兩年的學習過程中使我們能以正面樂觀的態度面對所遭遇的各種挑戰，並提供良好的實驗設備及適時的指導。另外也感謝口試委員施吉昇教授、范倫達教授、曾建超教授針對論文提出許多建議，使本論文能更趨完善。

實驗方面感謝網路上許多開放原始碼的社群及組織，提供無論是硬體及軟體的程式碼及討論。並感謝國科會「系統層級之多核心嵌入式系統發展與驗證環境」計畫的支持及相關子計畫的協助與合作。

最後要感謝實驗室學長、同學及學弟妹，無論在學業、研究、計畫及日常生活上的幫忙與照應，使我能順利度過這充實的兩年。

Contents

摘要 i
Abstract ii
誌謝 iii
Contents iv
List of Tables v
List of Figures vi
Chapter 1. Introduction 1
Chapter 2. Related Work 3
2.1. Performance Evaluation 3
2.1.1. Software Simulation 3
2.1.2. Software Profiling 4
2.1.3. Runtime Hardware Monitor 6
2.2. Energy Evaluation 8
2.2.1. Circuit Simulation 8
2.2.2. Hardware Measurement 8
2.2.3. Architecture-Level Evaluation 8
Chapter 3. Performance/Energy Evaluation Tool for Multi-core Embedded Systems 11
3.1. Performance and Energy Evaluation Methodology 12
3.2. Proposed Evaluation Architecture 13
3.3. Energy Model 15
3.4. Advantages of Our Approach 17
Chapter 4. Implementation 18
4.1. Emulation Platform 19
4.1.1. GRLIB Open Source IP Library 19
4.1.2. LEON3 MPSoC Emulation Environment 20
4.1.3. DE3 Development and Education Board 21
4.1.4. SnapGear Embedded Linux Distribution for LEON3 21
4.2. Evaluation of LEON3 Processor 23
4.2.1. Runtime Performance Monitor 23
4.2.2. Energy Calculation 24
Chapter 5. Experimental Results 26
5.1. Case Study: SPLASH-2 FFT 29
Chapter 6. Conclusions 31
References 32

List of Tables

Table 1. Register descriptions....................14
Table 2. LEON3 processor configurations....................20
Table 3. LEON3 processor events for REALprof monitors....................23
Table 4. LEON3 processor energy consumption....................25
Table 5. Feature comparison of performance and energy evaluation approaches....................27
Table 6. StratixIII 340 FPGA resource usage....................28

List of Figures

Figure 1. OProfile profiling overhead 5

Figure 2. SnoopP architecture 5

Figure 3. Airwolf profiler architecture 5

Figure 4. MAMon monitoring system 7

Figure 5. Bhattacharjee’s approach 9

Figure 6. Ghodrat’s approach 10

Figure 7. Suggested development and evaluation flow for embedded systems 11

Figure 8. Emulation platform infrastructure 13

Figure 9. REALprof architecture 14

Figure 10. Overview of our emulation platform 18

Figure 11. Block diagram of LEON3 processor 19

Figure 12. Block diagram of LEON3 MPSoC 20

Figure 13. DE3-340 development and education board 22

Figure 14. A stack of DE3 board - more powerful emulation system 22

Figure 15. Flow of gate-level power analysis 24

Figure 16. Overhead comparison with different number of HPCs 26

Figure 17. Overhead comparison with different sampling frequencies 27

Figure 18. Hardware performance counters sampling of CPU_0 29

Figure 19. Energy evaluation of CPU_0 30

Figure 20. Energy evaluation of each core 30

Chapter 1. Introduction

Multi-core has become a trend of processor design today. According to recent researches, multi-core is one of efficient ways to use transistors and increase performance, it also provides better energy efficiency and thermal management. Embedded system grows dramatically in recent several years. It has become more powerful and more complex. Multi-core technology is thus applied to some embedded systems in order to offer better performance and improve energy efficiency. For example, some high-end application processors [1][2][3] use ARM MPCore [4] multi-core technology for MID (Mobile Internet Device) products.

Embedded systems are designed for specific applications. They are usually battery-driven devices with limited operating time. To meet system requirements and extend operating time, it is very important to evaluate and optimize performance and energy consumption of embedded systems in design stage. However, conventional approaches suffer from difficulties to provide fast and accurate evaluation of the system, especially for those multi-core embedded systems with complex hardware and software. Therefore, fast and accurate profiling technology for multi-core embedded systems is a critical issue.

Reconfigurable device provides more flexible and relatively lower performance than conventional ASIC (application specific integrated circuit). FPGA (field programmable gate array) is the most popular reconfigurable device in recent several years. Due to improvement of manufacturing technology, capacity and performance of FPGA increases very fast. Today, it is possible to perform a complex SOPC (system-on-a-programmable-chip) on a FPGA. Therefore, some previous researchers have taken advantage of FPGAs to assist system profiling [5][6][7][8].

According to above discussion, this thesis proposes REALprof, a hardware-assisted performance/energy evaluation tool for multi-core embedded systems. Our approach is fast and effective for system analysis, and helps designers in adjusting hardware configuration and improving software in design stage. It provides hardware monitor for runtime programs, and uses these monitor information to estimate energy consumption of the target system. Besides, it is possible to perform different profiling granularities from coarse-grained to fine-grained using our approach. We also provide some tools to generate system-level performance and energy report of the target system.

Our approach is different from conventional simulation approaches. It uses FPGA to accelerate evaluation speed and performs full system emulation with operating system. Comparing to conventional software profiling approaches, our approach uses hardware to sample runtime hardware information without extra software sampling overhead. Therefore, our approach reserves the original target system behavior without inserting sampling instructions into the target software.

The remainder of this thesis is organized as follows: Chapter 2 discusses related work. Chapter 3 describes methodology and architecture of our proposed performance/energy evaluation tool, REALprof. Chapter 4 describes implementation and experimental environment in detail. Chapter 5 presents experimental results and a case study using our approach. Finally, Chapter 6 concludes the thesis.

Chapter 2. Related Work

In this chapter, we review previous works of performance and energy evaluation. In general, performance evaluation (Section 2.1) can be categorized into three categories: software simulation (Section 2.1.1), software profiling (Section 2.1.3), and runtime hardware monitor (Section 2.1.3). Energy evaluation (Section 2.2) can be categorized into three categories: circuit simulation (Section 2.2.1), hardware measurement (Section 2.2.2), and architecture-level evaluation (Section 2.2.3).

2.1. Performance Evaluation

2.1.1. Software Simulation

It is popular to use software simulators for system development, hardware behavior verification, and performance evaluation. ISS (instruction set simulator) is common used for processor simulation, such as SimpleScalar [9]. In recent years, system-level virtual platform is very popular to perform simulation of a full system or a SoC (System-on-a-Chip), such as CoWare Platform Architect [10]. For multi-core simulation, some previous researchers proposed several simulators [11][12][13][14][15].

Software simulation is convenient for developers in early design stage. It is easy to modify the system behavior and doesn’t need the real target hardware. However, software simulation speed is slower than real hardware execution. It is very difficult and time-consuming to perform effective simulation for multi-core systems with complex hardware and software. Because simulation speed is slow, it is inefficient to simulate huge software such as operating system. In general, only a small and critical portion of the program performs simulation, and operating system is usually ignored. However, multi-core systems usually need operating system to manage each core. Besides, multi-core simulators must consider synchronization between each core to perform the real system behavior. Otherwise, the simulation may be different from the real system execution.

2.1.2. Software Profiling

Software profiling is popular for program performance evaluation. Profiling tool statistics or inserts some instructions into the target program or the operating system to measure the program behavior at runtime. For example, Intel VTune Performance Analyzer [16] is a statistical profiler which profiles the program at runtime for Intel processors. OProfile [17] is a system profiler for Linux. GNU profiler, gprof [18] is a part of GNU Binutils (binary utilities) for inserting some profiling instructions into the target program at compile time.

Software profiling is useful and convenient for software development. However, it is inaccurate because profiler inserts some extra instructions into the target program or system, and it takes some overhead. To minimize the profiling overhead, we must reduce the extra inserted instructions. However, the more information you want to gain from profiling, the more extra instructions you need to insert, and the more profiling overhead you need to pay. It is a trade-off between profiling information and accuracy. Figure 1 illustrates OProfile profiling overhead which executes on HP Netserver with two 400 MHz Pentium II processors. It takes about 10% overhead when the profiling frequency is 10,000 times per second with separate threads mode. The default profiling frequency is 2,000 times per second to perform acceptable low-overhead profiling. It is useful for millisecond level profiling but difficult to perform accurate profiling in higher granularities.

Besides, the original system behavior may be changed due to the inserted profiling instructions. For multi-core systems, it may cause different process schedule.

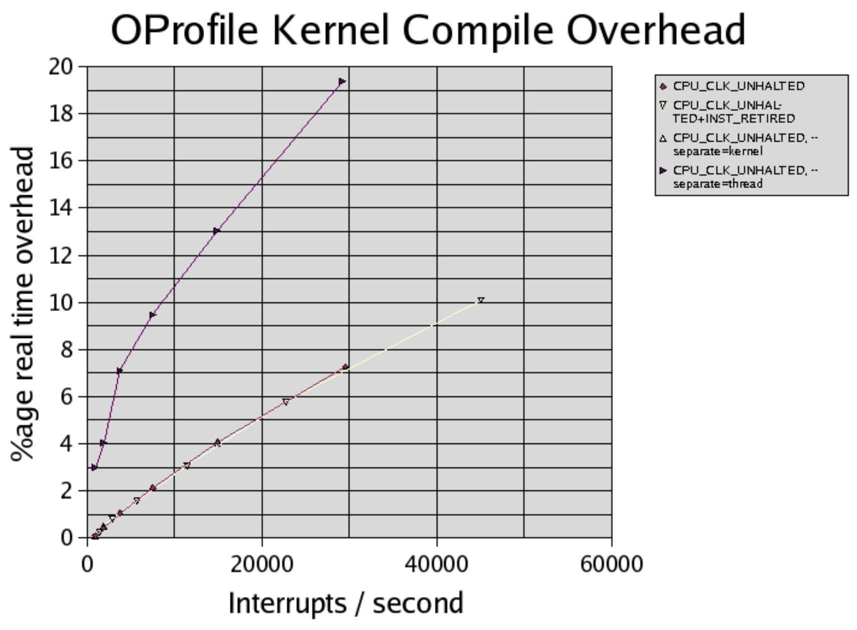
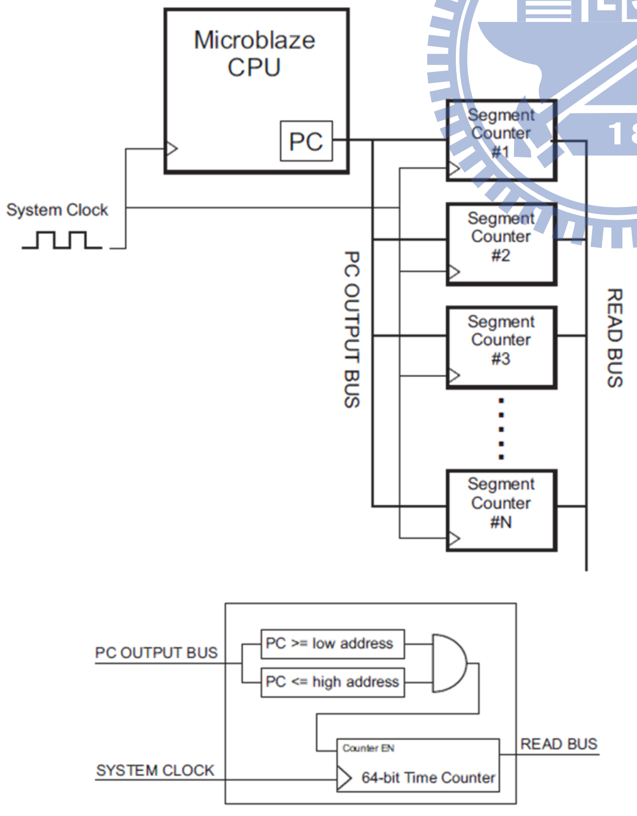


Figure 1. OProfile profiling overhead



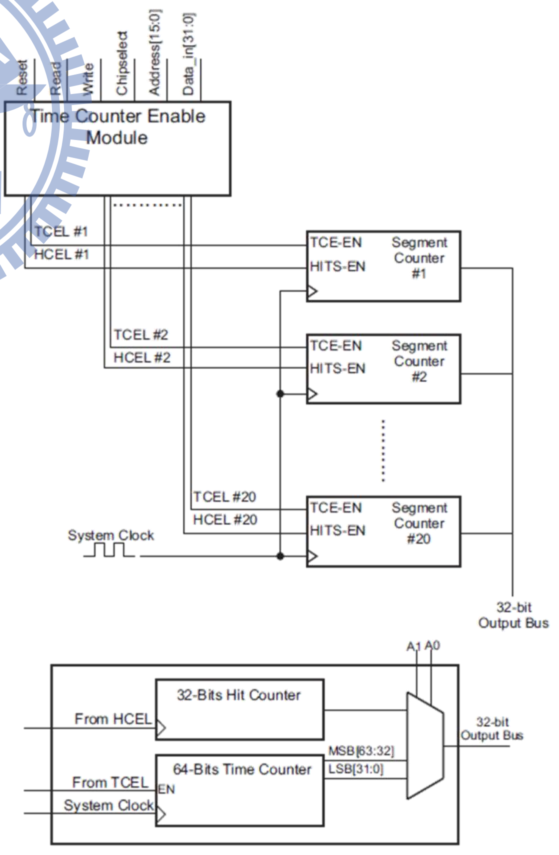


Figure 2. SnoopP architecture

Figure 3. Airwolf profiler architecture

To improve profiling accuracy, some previous approaches use specific hardware to assist performance evaluation. For example, SnoopP [5] (Figure 2) and Airwolf profiler [6] (Figure 3) are FPGA-based profilers. They proposed specific hardware logic to calculate execution time of each function call. The accuracy of these approaches is better than gprof software profiler.

2.1.3. Runtime Hardware Monitor

Some processors include performance counters to monitor the specific, low-level, and hardware-related activities at runtime. These performance counters provide meaningful sampling and reflect some important information of the target system performance, such as cache miss count, pipeline stall cycle. For example, ARM processor family provides several performance counter registers in control coprocessor CP15 [19]. Programmer setups PMNCs (performance monitor control registers) to control variant performance monitor facilities of performance counter registers by using coprocessor instructions. However, different processors have different performance counters and different access methods. Software with performance counters support causes poor portability. Browne [20] proposed a general performance counters application program interface - PAPI to enhance the software portability with performance counters support. Programmer can use PAPI to perform profiling conveniently.

Performance counters perform fast and effective performance evaluation. However, the real target processor is need, and it must support the performance counters which you need. It lacks for flexibility because it is impossible to modify the existing hardware design after fabrication. It is only useful for software performance evaluation and optimization. Besides, we must consider performance counters access overhead. As mentioned before, it is a trade-off between profiling information and accuracy.

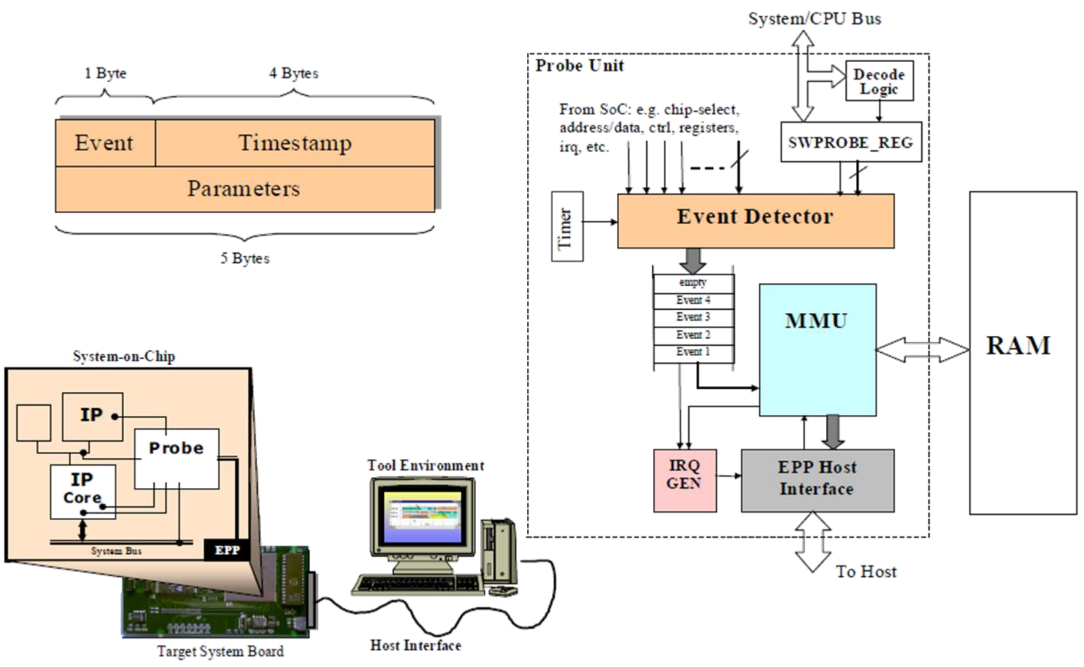


Figure 4. MAMon monitoring system

MAMon [21] (Figure 4) is one of hardware approaches for accurate runtime system monitor which is different from conventional performance counters approaches. A specific monitor hardware records timestamp and some parameters when dedicated events are triggered on the target system. These data are managed by a specific memory management unit, and transmitted to the host computer with specific hardware controller, interface, and protocol. The host computer constructs an event database using these data. Finally, designers can retrieve meaningful event information from the event database and analyze the target system. MAMon is good for event-based profiling. However, the infrastructure of MAMon is complex. Because of limited transmission bandwidth, it is difficult to scale MAMon to monitor huge number of events in detail.

2.2. Energy Evaluation

2.2.1. Circuit Simulation

Energy consumption is related to manufacturing technology, transistor character, circuit architecture, and other low-level hardware-related factors. Circuit simulation is a common energy evaluation approach. For example, Synopsys HSPICE [22] is a circuit simulator with power estimation for full-custom design, and Synopsys PrimeTime PX [23] is a cell-based design tool for gate-level power analysis. These tools are useful for hardware designers to estimate energy consumption at early design stage, but the simulation speed is extremely slow for ultra large scale integration circuit today. For multi-core systems with complex hardware and software, these kinds of approaches are not feasible.

2.2.2. Hardware Measurement

Device energy consumption can be measured by power meter directly [24]. It is fast and true, but it needs measurement environment and the real target hardware with essential measurement points. Besides, synchronization between the measurement host and the target device must be considered for detail evaluation.

Some power management chips provide runtime voltage and current sampling of the target system. Programmers can read sampling registers to retrieve present voltage and current data and calculate energy consumption.

2.2.3. Architecture-Level Evaluation

Architecture-level energy evaluation analyzes architecture of the target design and constructs energy models for each component. Wattch [25] is one of the well-known previous works using this approach. It categorizes processor components into four categories: array structures, fully associative content-addressable memory, combinational logic and wires, and clocking. Power models of each component are based on capacitance equations with interface to SimpleScalar. Architecture-level energy simulation [26][27] is faster than circuit simulation, but it is still time-consuming for complex systems today.

Contreras [28] proposed another architecture-level approach for Intel XScale processor. The concept of this approach is that energy consumption is close related to some hardware activities. Intel XScale processor provides some performance counters for monitoring some hardware activities, and the mechanism is similar to ARM processor family. Contreras constructs processor power models by using these available hardware performance counters. It performs a quick energy evaluation with acceptable accuracy. Because it is based on runtime hardware monitor, the real target hardware is required, and it must provide essential performance counters which are related to the energy consumption of each component. This approach can be also applied to XScale software simulator to perform energy evaluation [29].

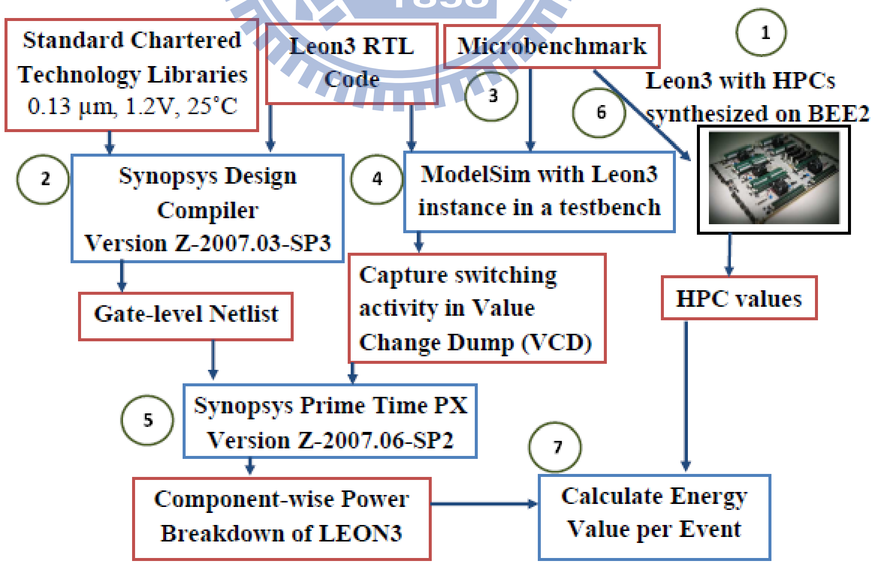

Standard Chartered Technology Libraries 0.13 µm, 1.2V, 25°C
Leon3 RTL Code
Microbenchmark
1
Leon3 with HPCs synthesized on BEE2
3
6
2
Synopsys Design Compiler Version Z-2007.03-SP3
4
ModelSim with Leon3 instance in a testbench
Capture switching activity in Value Change Dump (VCD)
HPC values
Gate-level Netlist
5
Synopsys Prime Time PX Version Z-2007.06-SP2
7
Component-wise Power Breakdown of LEON3
Calculate Energy Value per Event

Figure 5. Bhattacharjee's approach

Bhattacharjee [7] proposed a FPGA-based power emulation approach (Figure 5). The concept is similar to Contreras's approach. They add some component-specific event counters to the target processor design and use these counters to construct power models. Because FPGA is reconfigurable, this approach is more flexible than Contreras's approach. Although FPGA emulation speed is slower than the real target hardware, it is extremely faster than circuit simulation and software architecture-level energy evaluation. The energy evaluation approaches of Contreras and Bhattacharjee are based on runtime hardware monitor. As mentioned before, performance counters access introduces some extra software overhead. It is useful for millisecond level profiling but difficult to profile in higher granularities with acceptable accuracy.

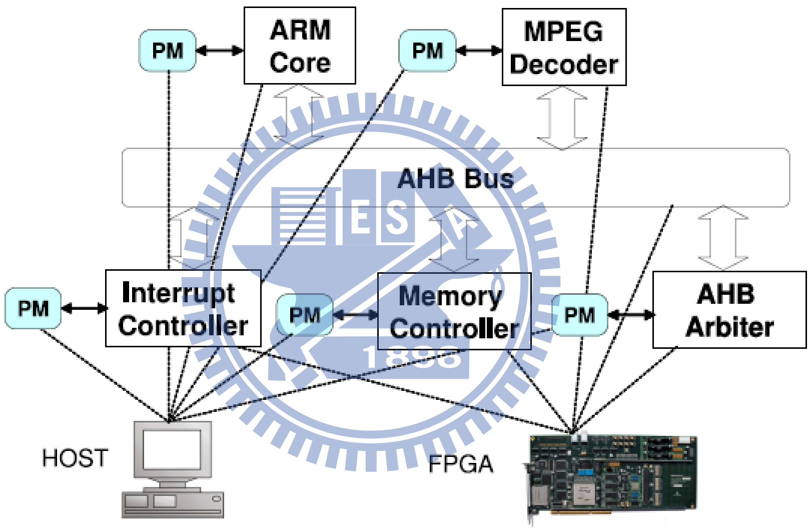


Figure 6. Ghodrat's approach

Ghodrat [8] proposed a hybrid approach for system energy evaluation (Figure 6). Some of the components use software simulation and some of the components use FPGA emulation. This approach reduces the simulation time and has more flexible than pure FPGA emulation. However, the communication between software simulation and FPGA emulation introduces some overhead, and it needs interfaces and protocols between the host simulation computer and the target FPGA board.

Chapter 3. Performance/Energy Evaluation Tool for Multi-core Embedded Systems

In this chapter, we discuss our proposed performance and energy evaluation solution for multi-core embedded systems in detail. Section 3.1 introduces methodology and evaluation flow. Section 3.2 describes our proposed tool - REALprof. Section 3.3 discusses how to construct component energy models in early design stage. Section 3.4 describes the advantages of our approach.

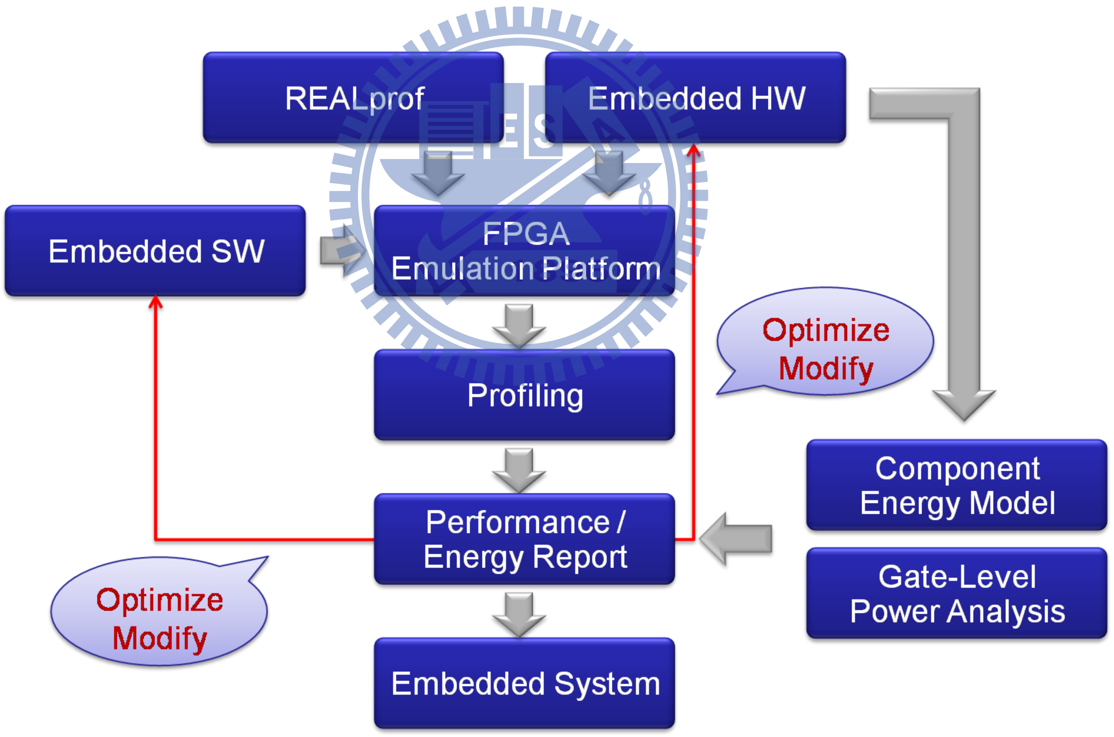


Figure 7. Suggested development and evaluation flow for embedded systems

3.1. Performance and Energy Evaluation Methodology

Our approach uses FPGA to perform fast performance and energy evaluation. Figure 7 illustrates our suggested development and evaluation flow for embedded systems. Hardware of the embedded system uses powerful FPGA to construct a complex SoC emulation platform. It includes several synthesizable soft intelligent properties and uses FPGA to perform fast emulation. Software of the embedded system then executes on the emulation platform. Designer debugs on such prototype system and evaluates the system. To meet the requirements of the embedded system, designer makes heavy efforts to optimize the hardware and software.

Our proposed tool, REALprof is based on FPGA. It uses hardware to assist system evaluation. Designer adds REALprof to the target hardware design, and builds emulation platform using FPGA. For embedded software, REALprof provides runtime hardware monitor for the target program without software sampling overhead. Designer can use REALprof to perform fast, fine-grained, and real profiling. The generated profiling report includes performance and energy information. Performance information is retrieved from REALprof, it includes many useful data about runtime hardware monitor. Energy information is based on performance information. Designer constructs component energy models using gate-level power analysis or component datasheets. The generated performance and energy report provides designers a system-level view of the target embedded system, and helps designers to optimize the hardware and software in design stage. After several iterations of the evaluation and optimization, designer can remove REALprof from the target design, and the final well-optimized embedded system is done.

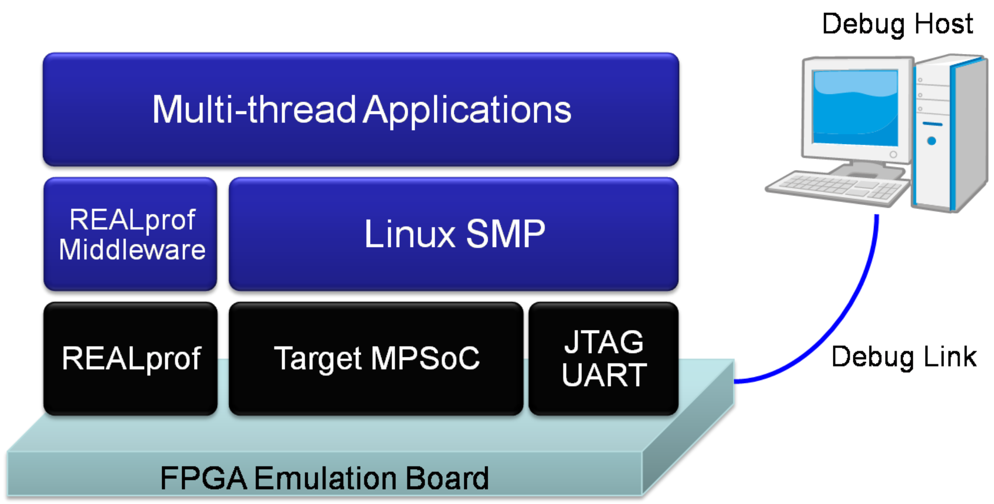


Figure 8. Emulation platform infrastructure

Figure 8 illustrates a typical emulation environment for a multi-core embedded system using our approach. Designer integrates the target MPSoC (Multi-Processor System-on-a-Chip) design with debug interface cores (JTAG, UART…) and REALprof, then uses FPGA to construct a target emulation platform. Embedded software executes on the target emulation platform, it includes several multi-thread embedded application programs and Linux with SMP (Symmetric Multi-Processing) support. Programmer uses our provided middleware to control the REALprof hardware and perform performance and energy evaluation. The host computer communicates with the target emulation platform using debug link and retrieves monitor information from REALprof.

3.2. Proposed Evaluation Architecture

As mentioned before, conventional software profiling approaches insert profiling instructions into the target software. It destroys the original target system behavior and causes extra software overhead. In order to improve profiling accuracy for embedded systems. This thesis proposes a reconfigurable hardware-assisted log profiler - REALprof. The behavior of REALprof is similar to conventional software profilers which records hardware performance counters periodically at runtime. However, our approach uses hardware to record runtime information automatically. It doesn't cause extra software sampling overhead and reserves the original system behavior.

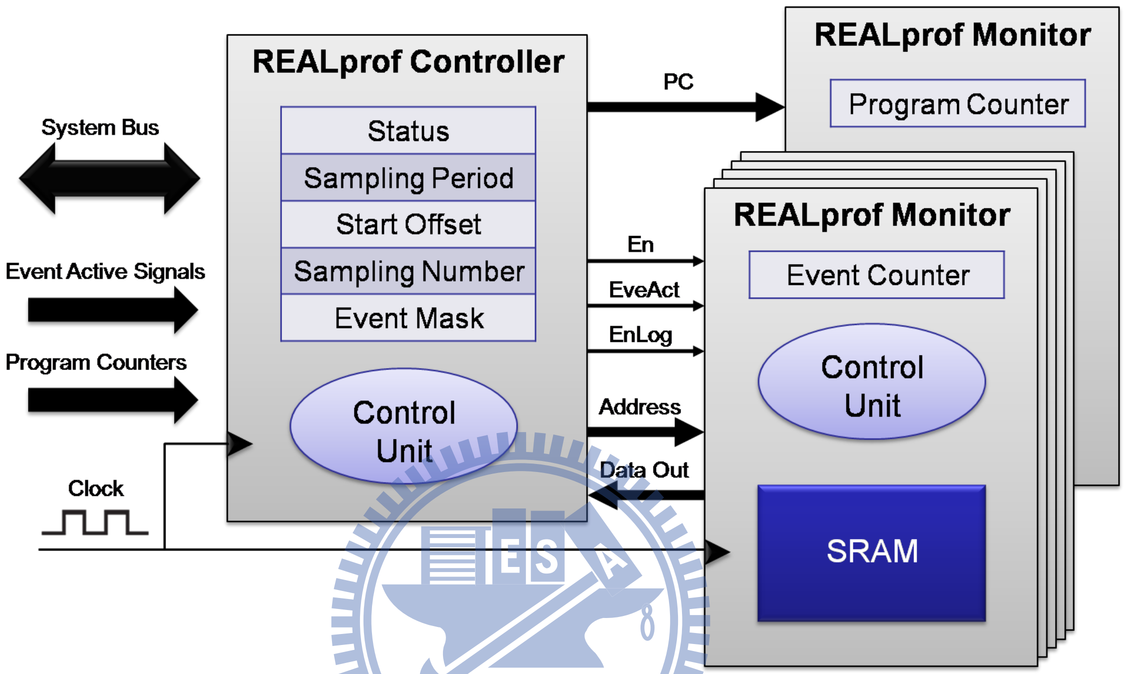


Figure 9. REALprof architecture

Table 1. Register descriptions

Memory Map	Register	Description
Base + 0x00000000	Status	Control status of the controller and monitors.
Base + 0x00000004	Sampling Period	Sampling period of REALprof.
Base + 0x00000008	Start Offset	Start profiling with specific delay cycles.
Base + 0x0000000C	Sampling Number	Sampling number of each event monitor.
Base + 0x00000010	Event Mask	Enable/Disable of each event monitor.

Figure 9 illustrates the architecture of REALprof. REALprof is designed as a bus slave component. Programmer can use memory-map I/O to access REALprof. It consists of a controller and several monitors. REALprof controller contains several control registers. Table 1 lists the detail register descriptions. Programmer indicates status, sampling period, start offset, and event mask by setting these registers. Several event active signals are connected to REALprof controller, such as pipeline stall and cache miss. Each event active signal is connected to EveAct signal of an individual REALprof monitor. REALprof monitor snoops the EveAct signal and increases the event counter once the event is active. En signal is true when REALprof is active, it indicates whether REALprof monitors snoop signals or not. EnLog signal is similar to timer IRQ (Interrupt ReQuest) signal. The period is indicated by Sampling Period register. When EnLog is triggered, each REALprof monitor stores the current value of event counter to the SRAM and resets the event counter to zero. At the same time, REALprof controller increases Sampling Number register to indicate that number of samples is collected in each REALprof monitor SRAM.

In order to connect the profiling results with the source code, program counter of each processor is also connected to REALprof. Each program counter is connected to an individual REALprof monitor with a similar mechanism to record the program counter. After profiling is done, programmer accesses REALprof to retrieve runtime monitor information from SRAMs. According to these monitor information, designer can understand runtime program behavior in detail without extra software profiling overhead.

3.3. Energy Model

Component energy models can be constructed in several ways. For existent components, we can reference datasheets or measure components directly. For components in early design stage, power analysis is a feasible solution.

Gate-level power analysis is popular for digital designs using CMOS manufacture technology in early design stage, such as Synopsys PrimeTime PX. It calculates power dissipation using the target design netlist, cell library power models, signal activities, and capacitance effects.

Equation (1) describes that CMOS power dissipation (P) consists of dynamic power ($P_{dynamic}$) and leakage power ($P_{leakage}$):

$$P = P_{dynamic} + P_{leakage} \tag{1}$$

Dynamic power is related to signal activities. Equation (2) describes that dynamic power dissipation due to switching power ($P_{switching}$: charge/discharge of load capacitances) and short circuit power ($P_{short\_circuit}$: switching causes both NMOS and PMOS are partially on):

$$P_{dynamic} = P_{switching} + P_{short\_circuit} \tag{2}$$

Equation (3) describes that leakage power dissipation due to reverse-biased ($P_{reverse\_biased}$) and sub-threshold ($P_{sub\text{-}threshold}$) conduction:

$$P_{leakage} = P_{reverse-biased} + P_{sub-threshold} \tag{3}$$

Equation (4) describes that energy consumption (E) is the integral of the instantaneous power dissipation:

$$E = \int P \cdot dt \tag{4}$$

According to above equations, foundry can develop their cell library with gate-level power models. We can perform gate-level power analysis to evaluate the component energy consumption in early design stage.

In order to offer system-level energy evaluation, we construct high-level component energy models according to the gate-level power analysis results. Equation (5) shows our high-level component energy model:

$$E = P_{idle} \cdot T_{total} + \sum E_{event\_active} \cdot C_{event\_active} \tag{5}$$

Total energy consumption (E) consists of idle energy and active energy. Idle energy is the product of idle power (P_{idle}) and total execution time (T_{total}). Idle power reflects leakage power and power of modules which are always active. Active energy is the product of event active energy (E_{event_active}) and event active count (C_{event_active}). Active energy reflects energy consumption variations of different hardware events. The event active count is based on performance monitor information which can be retrieved from REALprof, such as pipeline stall, cache miss.

3.4. Advantages of Our Approach

Our approach provides both performance and energy evaluation for embedded systems. It is useful for designers to optimize system and make a trade-off between performance and energy efficiency in design stage. Because we use FPGA to construct reconfigurable emulation platform, our approach is more flexible than ASIC approach. The evaluation speed is faster than software simulator, and it is possible to evaluation program with operation system and complex software.

Our approach is different from conventional software profiling. We use hardware to sample performance counters and program counters periodically. It avoids software sampling routine, and the target program doesn’t need to insert extra profiling instructions. Therefore, our approach doesn’t cause heavy software profiling overhead and reserves the original behavior of the target system.

In order to reduce profiling overhead and provide acceptable accuracy, conventional software profiling usually works in millisecond level granularities. Without software sampling overhead, our approach can provide different profiling granularities. It is possible to perform cycle sampling using our approach. In design stage, designer can do several profiling iterations from coarse-grained to fine-grained to understand the program behavior in detail.

Chapter 4. Implementation

Figure 10 illustrates overview of our emulation platform. We construct the target MPSoC design using GRLIB open source IP library with our proposed tool - REALprof. Then, we download the target design to DE3 FPGA board to construct an emulation platform. We use SnapGear Linux distribution with SMP support and multi-thread applications to construct the embedded software. Embedded software executes on the emulation platform and performs runtime hardware monitor using REALprof. After profiling done, our tools generate performance and energy reports with diagrams for system analysis.

The rest of this chapter is organized as follows: Section 4.1 introduces hardware and software of the emulation platform. Section 4.2 describes REALprof implementation in detail.

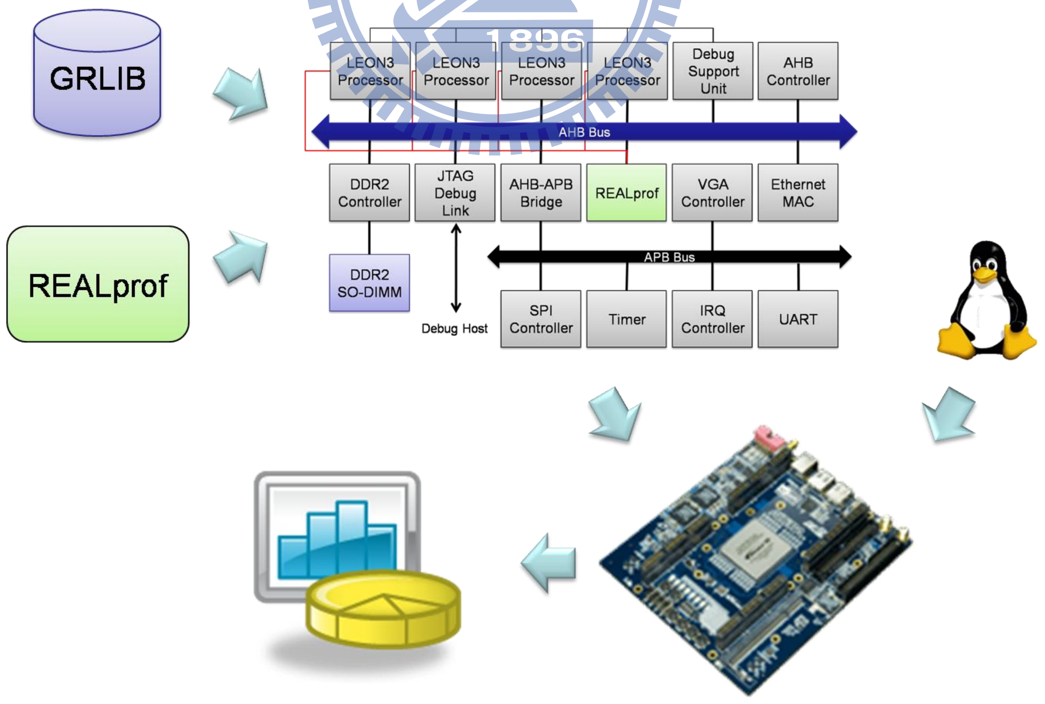


Figure 10. Overview of our emulation platform

4.1. Emulation Platform

4.1.1. GRLIB Open Source IP Library

In this thesis, we use GRLIB [30] IP library to construct the target MPSoC platform. GRLIB is an integrated set of reusable IP cores provided by Aeroflex Gaisler. It is provided under dual license mode, GNU GPL license or commercial license. It includes LEON3 soft-core processor and abounding peripheral cores. Each core of GRLIB is interfaced using AMBA 2.0 protocol and uses synthesizable VHDL model. LEON3 is a 32-bit soft-core processor based on the SPARC V8 architecture. It has 7-stage pipeline with L1 Harvard architecture cache. Data cache of LEON3 supports snoop protocol. It is possible to perform SMP system using LEON3 processors. To provide debug environment for the target system, GRLIB also includes a debug support unit core for LEON3 processor with GRMON [31] debug monitor software support. Figure 11 illustrates the block diagram of LEON3 processor. The original design of LEON is developed by ESA (Europe Space Agency) for space applications.

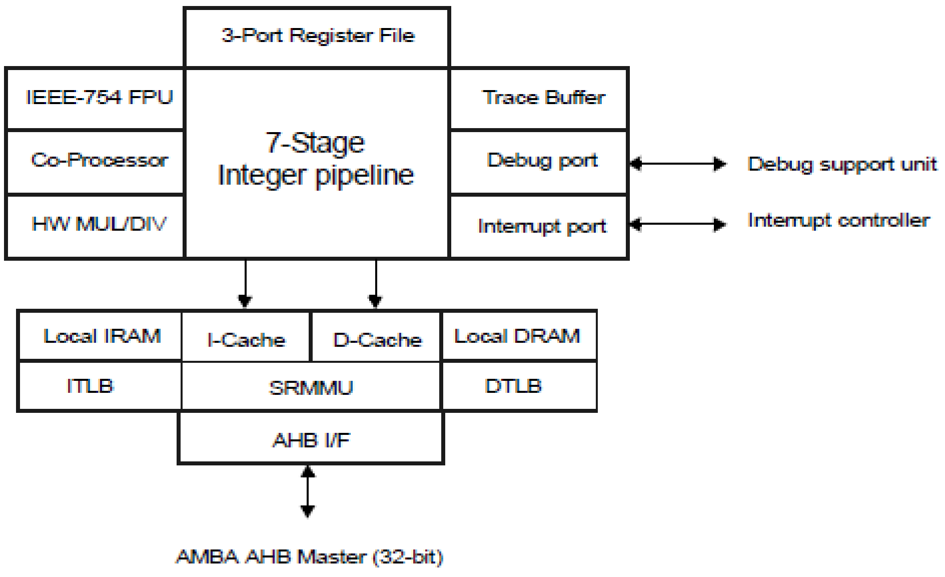


Figure 11. Block diagram of LEON3 processor

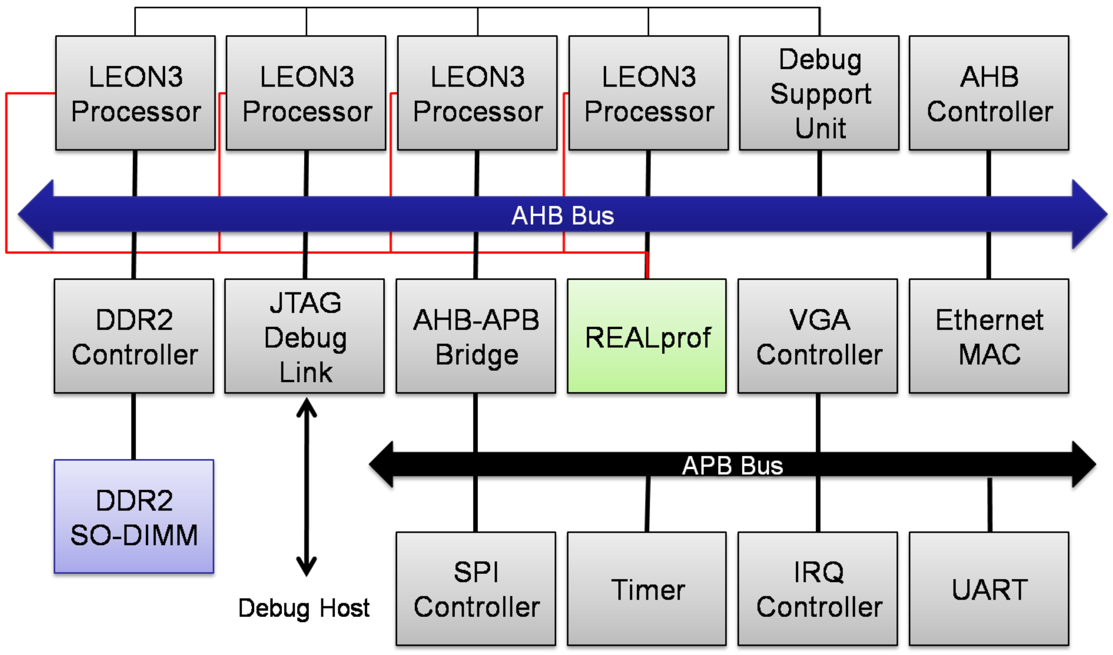


Figure 12. Block diagram of LEON3 MPSoC

Table 2. LEON3 processor configurations

Module	Configurations
Integer Unit	8 register windows, 1-cycle load delay, SPARC V8 MUL/DIV support, 2-cycle multiplier latency, power-down mode support
Floating Point Unit	only netlist is available for FPGA
L1 Instruction Cache	4-way set-associative, total 16 KB, 32 bytes/line, LRU
L1 Data Cache	4-way set-associative, total 16 KB, 32 bytes/line, LRU, AHB fast snoop and separate snoop tags
MMU	Separate instruction/data TLB, fast write buffer, LRU 32 entries for each instruction/data TLB

4.1.2. LEON3 MPSoC Emulation Environment

Figure 12 shows our target MPSoC design. It includes LEON3 quad-core, abounding peripherals, and our proposed tool - REALprof. Four LEON3 processors snoop data on AHB (Advanced High-performance Bus). Configurations of each processor are shown in Table 2.

REALprof is wrapped as an AHB slave. Each processor connects its program counter and event signals to REALprof. Detail of performance events and energy calculation will describe in section 4.2.

4.1.3. DE3 Development and Education Board

In this thesis, we use DE3-340 [32] development and education board to construct prototype emulation platform, as shown in Figure 13. DE3 is developed by Terasic Technologies. It consists of Altera Stratix III 340 FPGA, DDR2 SO-DIMM socket, and several peripheral components. Figure 14 shows that multiple DE3 boards can be joined together to increase gate count using HSTC (High Speed Terasic Connectors) interface. Altera Stratix III 340 FPGA uses TSMC 65-nm process to provide high performance and high density. It contains 338000 equivalent logic elements and rich on-chip embedded memory. We use Quartus II 8.0 design suite [33] to compile our design and download the target design to DE3 FPGA.

4.1.4. SnapGear Embedded Linux Distribution for LEON3

SnapGear Linux distribution [34] is a full open source package for embedded systems with boot loader, device drivers, Linux kernel, libraries, and abounding applications. Aeroflex Gaisler provides LEON3 port of SnapGear Linux 2.6.21.1 with SMP support for multi-core embedded systems. We add middleware for REALprof access includes device drivers and some utilities. Multi-thread applications can be integrated to SnapGear Linux package and use REALprof to perform performance and energy evaluation on the target multi-core emulation platform.

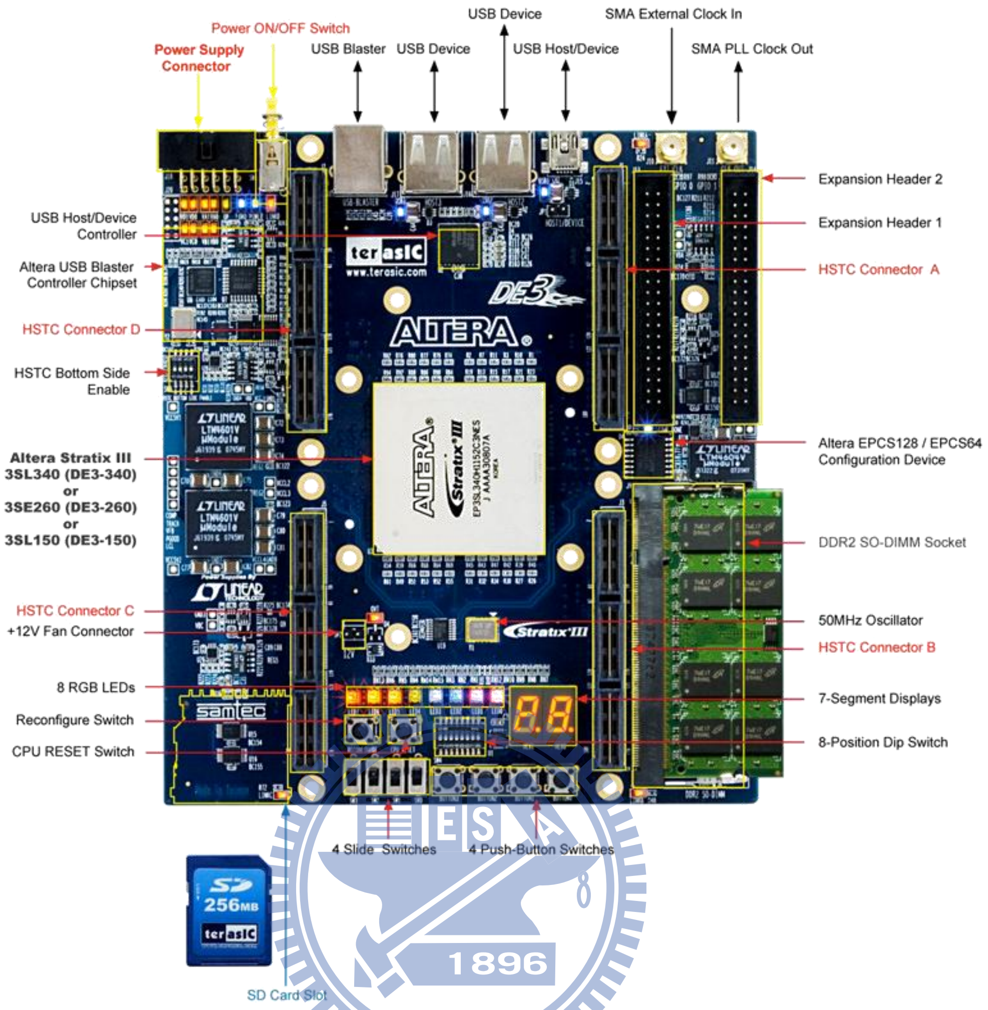


Figure 13. DE3-340 development and education board

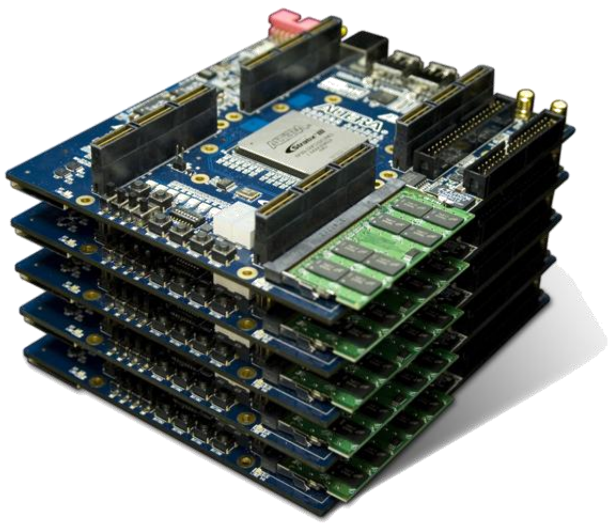


Figure 14. A stack of DE3 board - more powerful emulation system

4.2. Evaluation of LEON3 Processor

4.2.1. Runtime Performance Monitor

In order to evaluate performance and energy of LEON3 processor, we monitor several hardware events of each processor using REALprof. Table 3 lists these events in detail.

Table 3. LEON3 processor events for REALprof monitors

Event number	Event description
0	Program counter
1	Pipeline stall because of instruction cache miss
2	Pipeline stall because of data cache miss
3	Multiplication operation
4	Division operation
5	Instruction cache hit
6	Data cache read hit
7	Data cache write hit
8	Instruction cache miss
9	Data cache read miss
10	Data cache write miss
11	Cache flush
12	Instruction TLB miss
13	Data TLB miss
14	Power down cycle
15	Single register access
16	Double register access

4.2.2. Energy Calculation

We use runtime hardware monitor information to calculate component energy consumption. Figure 15 illustrates gate-level power analysis flow that describes in section 3.3. We use Design Compiler [35] to synthesis the target design with Faraday cell-based design kit for UMC 90 nm 1P9M process. In typical condition (25˚C, 1.0 Volt), the target design operates at 400 MHz. After synthesis, we use ModelSim [36] to perform gate-level simulation for generated netlist and software. The generated waveform in VCD (Value Change Dump) format then feeds to PrimeTime PX for dynamic power calculation. PrimeTime PX performs hierarchical power report and FSDB (Fast Signal DataBase) waveform according to design netlist, cell library, cell delay, and VCD waveform. Finally, we construct component energy models using power report and waveform.

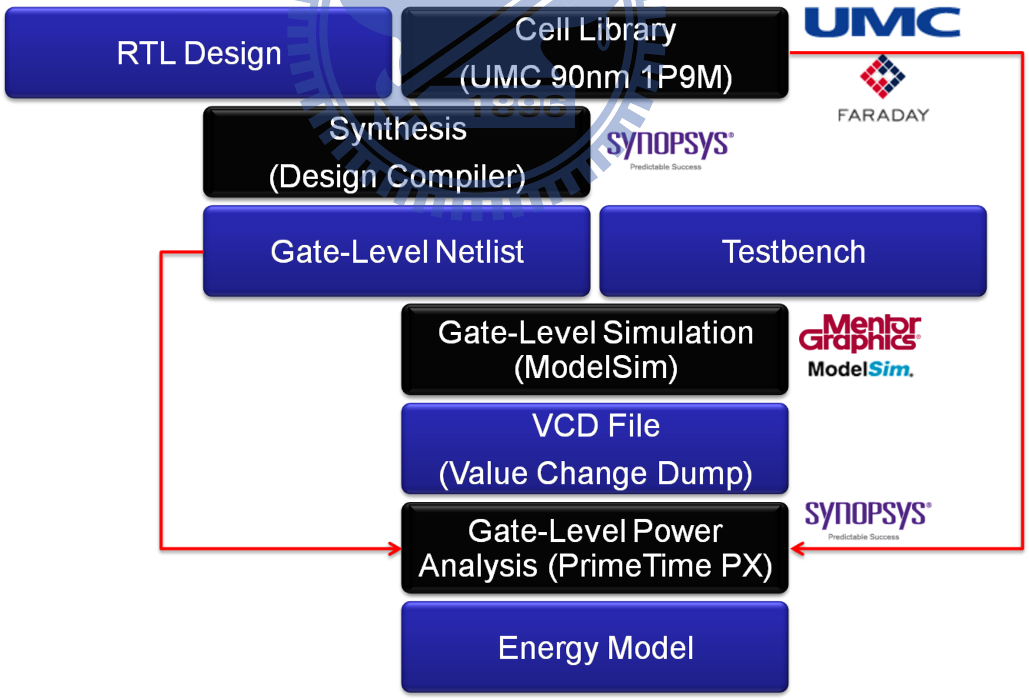


Figure 15. Flow of gate-level power analysis

Table 4 lists event energy consumption of LEON3 processor according to the gate-level power analysis report. In typical case, data cache dominates processor energy variances.

Table 4. LEON3 processor energy consumption

Component	Event	Energy/Power
Pipeline	Power down	16.1 mW
	Idle	49.2 mW
	Normal	58.2 mW
	Multiplication operation	32 pJ
	Division operation	218.75 pJ
	Instruction TLB miss	4.25 pJ
	Data TLB miss	4.25 pJ
Instruction cache	Power down	0.4084 mW
	Idle	49.1 mW
	Flush	16.403 nJ
	Hit	126.575 pJ
	Miss	123 pJ
Data cache	Idle	0.601 mW
	Flush	25.869 nJ
	Hit	314 pJ
	Miss	952.95 pJ
Register file	Idle	6.28 mW
	Single access	9.65 mW
	Double access	12.9 mW

Chapter 5. Experimental Results

Our approach uses hardware to assist sampling automatically. Programmer sets REALprof registers to start sampling before the target program section and stop sampling after the section. It doesn't need extra software overhead to sample hardware performance counters periodically. Figure 16 shows that overhead of conventional software sampling for hardware performance counters increases linearly with number of counters. The more information you want to get by sampling counters, the more overhead and inaccurate samples you get. However, our approach offers constant and negligible extra software overhead. It reserves the original target program behavior.

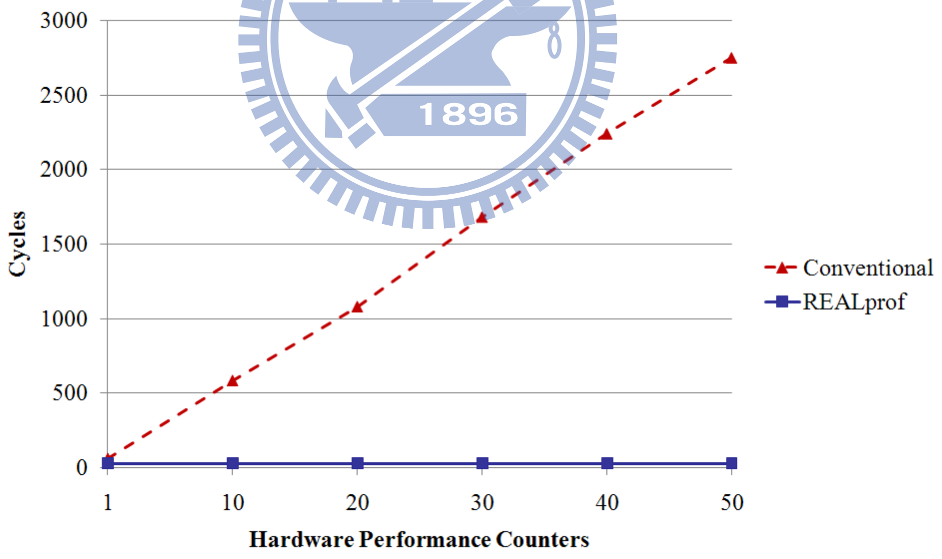


Figure 16. Overhead comparison with different number of HPCs

Figure 17 shows that software sampling overhead increases with profiling frequency. Sampling overhead is only negligible in millisecond granularity. It is difficult to perform acceptable profiling results in microsecond granularity using conventional approach.

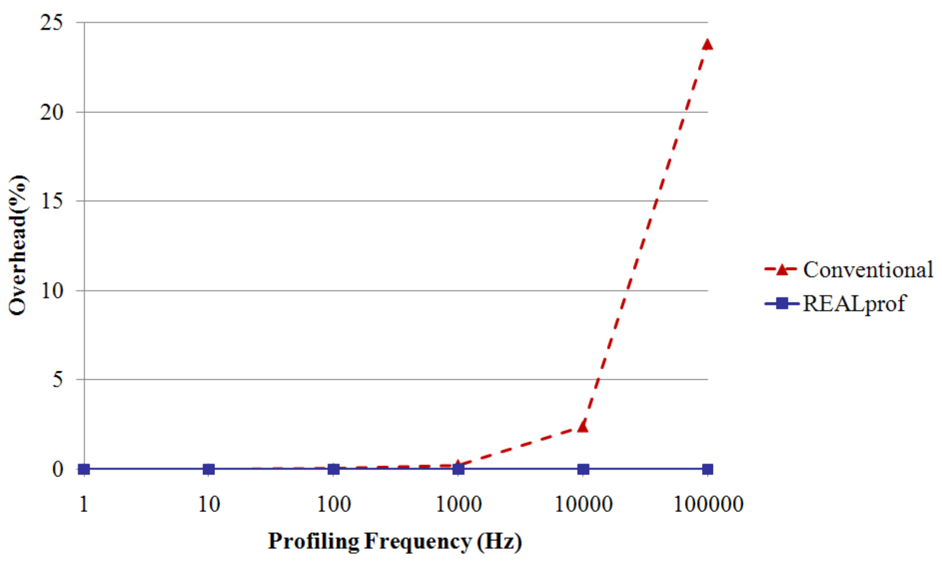


Figure 17. Overhead comparison with different sampling frequencies

Table 5 compares our approach with other conventional approaches for performance and energy evaluation. Our approach offers fast emulation about 100 MHz. Software overhead is negligible and it provides fine-grained profiling. It is possible to profile target software with operating system and scalable for complex multi-core systems.

Table 5. Feature comparison of performance and energy evaluation approaches

	REALprof	HPC profiling	uArch simulator	Circuit simulation
Method	HW emulation	directly execute	SW simulation	SW simulation
Speed	~ 100 MHz	real speed	KIPS ~ MIPS	extremely slow
Flexibility	reconfigurable	N	Y	Y
SW overhead	negligible	Y	N/A	N/A
Granularity	> cycle	~ ms	N/A	great
HW resource	HPC + BRAM	HPC	N	N
Program behavior	keep	destroy	N/A	keep
Operating system	Y	Y	usually skip	skip
Multi-core support	scalable	VTune, OProfile	SESC, GEMS	Y

Our approach uses reconfigurable hardware resource to record the sampling information. Table 6 shows the resource usage report of our emulation FPGA. Each REALprof monitor uses one M9K memory block to provide up to 256 records of each 32-bit hardware performance counter. For StratixIII 340 FPGA, the target design with REALprof only uses about 30% logic and 12% memory blocks. REALprof only uses about 2% logic and 4% memory blocks for 1 controller and 68 monitors. Therefore, it is possible to use REALprof for more complex embedded systems.

Table 6. StratixIII 340 FPGA resource usage

	Combinational ALUTs	Logic registers	DSP block	Block memory bit	Number
LEON3 processor	12581	8451	4	343936	4
DDR2 controller	781	601	0	4096	1
peripherals	2600	1269	0	16384	N/A
REALprof	4826(2%)	2091(1%)	0(0%)	557056(4%)	1 ctrl 68 mon
Total	58531(22%)	37765(14%)	16(3%)	1953280(12%)	

5.1. Case Study: SPLASH-2 FFT

This section demonstrates our approach for a multi-core embedded system. We use SPLASH-2 [37] FFT as the target multi-thread application program. Figure 18 illustrates the hardware performance counters sampling of CPU_0. As shown in figure, our approach provides fine-grained sampling in microsecond level. Designer can use REALprof to observe the variation of each counter in detail. According to these runtime hardware information and energy models, we can estimate energy consumption of each component. Figure 19 illustrates energy evaluation of CPU_0 components. As shown in figure, data cache behavior dominates the variation of the processor. This phenomenon is match with the gate-level power analysis result. Figure 20 illustrates energy evaluation of each core. Designer can use the evaluation result to analysis and optimize the target design. In this case, we can observe that SPLASH-2 FFT is a well parallelized multi-thread program.

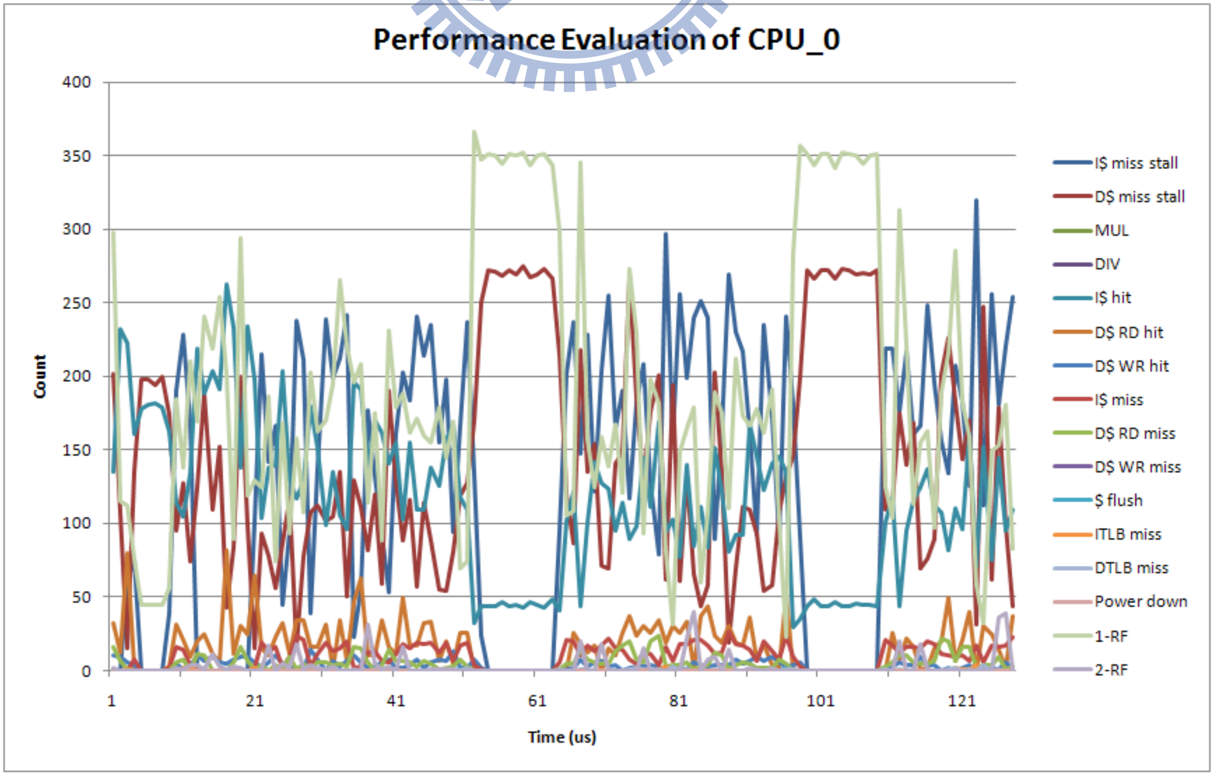


Figure 18. Hardware performance counters sampling of CPU_0

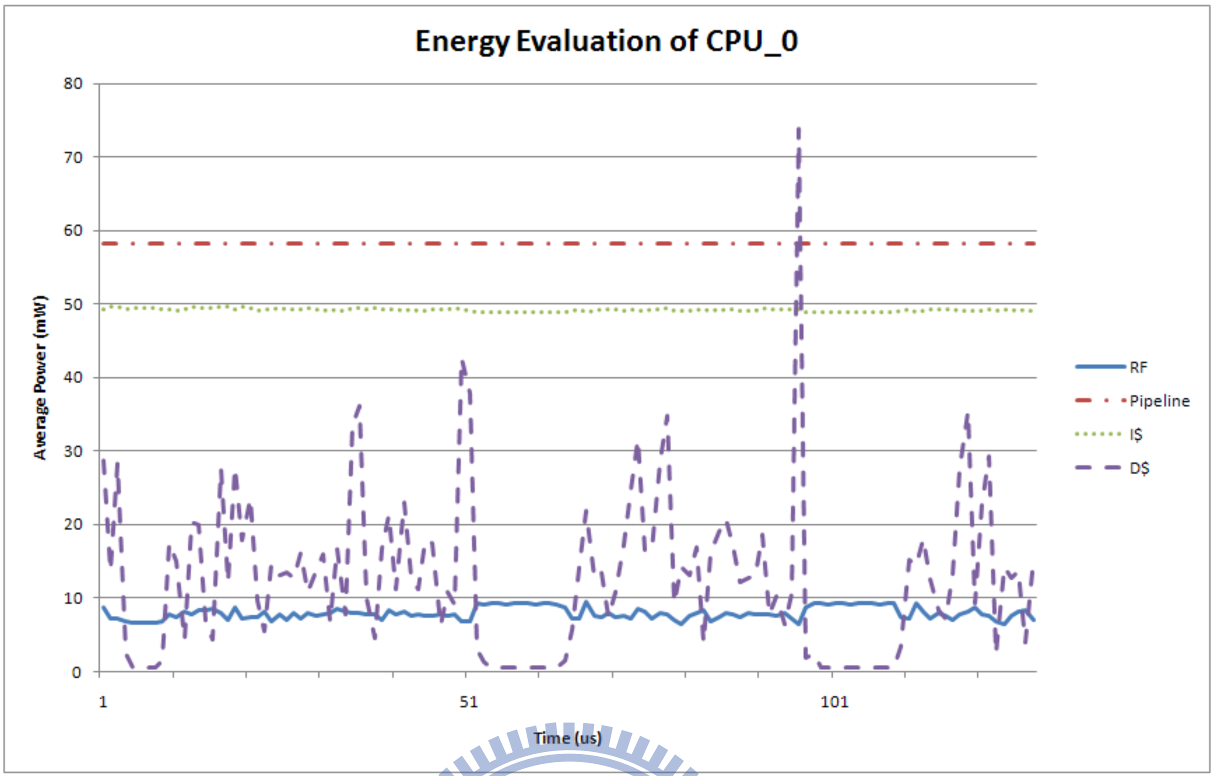


Figure 19. Energy evaluation of CPU_0

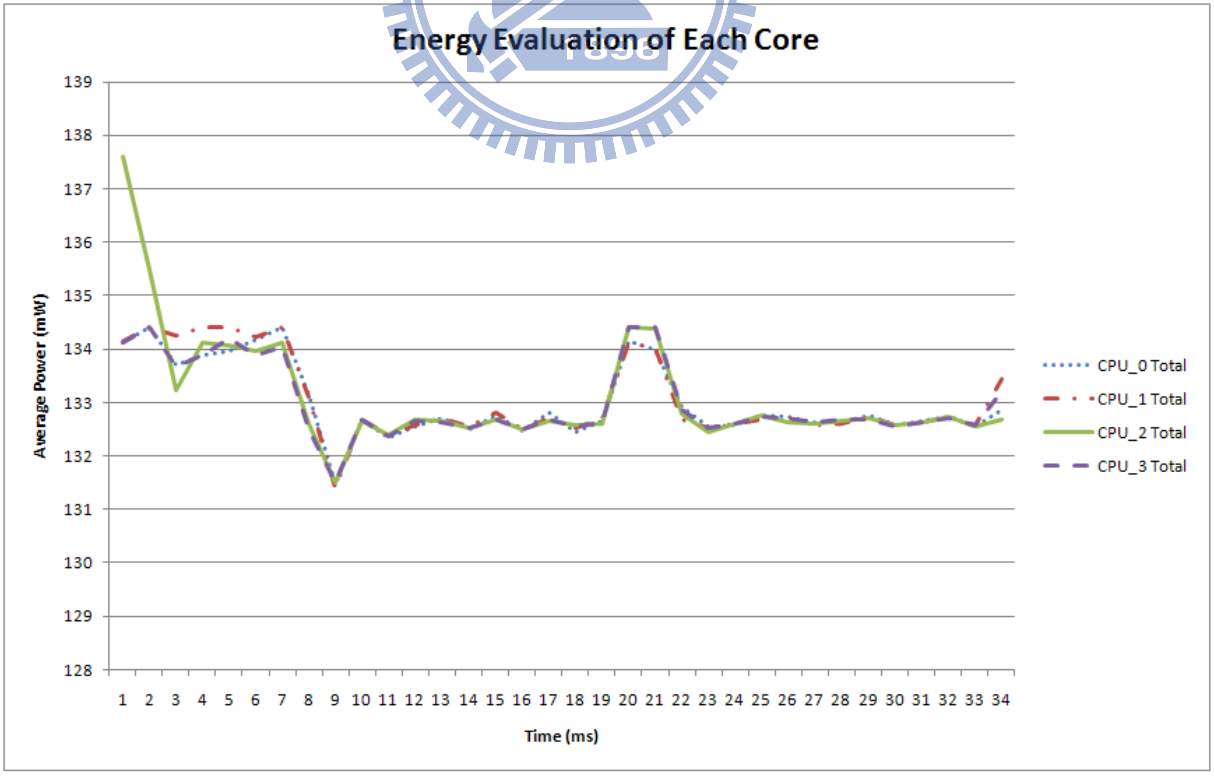


Figure 20. Energy evaluation of each core

Chapter 6. Conclusions

In this thesis, we propose a hardware-assisted performance and energy evaluation tool - REALprof. Our approach uses FPGA to provide fast full system emulation, and it is scalable for complex multi-core embedded systems. The experimental results show that it can work at a 100 MHz quad-core emulation platform. Our approach uses hardware to sample runtime hardware information without extra software sampling overhead. It provides different granularities from coarse-grained to fine-grained. As shown in case study, multi-core embedded system with Linux operating system can use REALprof to perform effective fine-grained profiling in microsecond level. The generated system-level performance and energy report is useful for designers to optimize hardware and software in design stage.

In order to provide experimental environment of a multi-core embedded system. We construct a powerful LEON3-based quad-core MPSoC emulation platform on DE3 board. Besides, we construct energy models of LEON3 soft-processor using gate-level power analysis with UMC 90 nm cell library. Finally, we perform a case study of profiling a SPLASH-2 multi-thread program on the target multi-core embedded system.

References

[1] NVIDIA Tegra.
http://www.nvidia.com/page/handheld.html

[2] Qualcomm Snapdragon.
http://www.qctconnect.com/products/snapdragon.html

[3] Texas Instruments OMAP 4 Platform.
http://www.ti.com/omap4_platform

[4] ARM11 MPCore.
http://www.arm.com/products/CPUs/ARM11MPCoreMultiprocessor.html

[5] L. Shannon and P. Chow, "Using Reconfigurability to Achieve Real-Time Profiling for Hardware/Software Codesign," Proceedings of the 12th International Symposium on Field Programmable Gate Arrays, pp. 190-199, Monterey, California, February 2004.

[6] J. G. Tong and M. A. S. Khalid, "Profiling Tools for FPGA-Based Embedded Systems: Survey and Quantitative Comparison," Journal of Computers, Vol. 3, No. 6, pp. 1-14, June 2008.

[7] A. Bhattacharjee, G. Contreras, and M. Martonosi, "Full-System Chip Multiprocessor Power Evaluations Using FPGA-Based Emulation," Proceedings of the 13th International Symposium on Low Power Electronics and Design, pp. 335-340, Bangalore, India, August 2008.

[8] M. A. Ghodrat, K. Lahiri, and A. Raghunathan, "Accelerating System-on-Chip Power Analysis Using Hybrid Power Estimation," Proceedings of the 44th Design Automation Conference, pp. 883-886, San Diego, California, June 2007.

[9] T. Austin, E. Larson, and D. Ernst, "SimpleScalar: An Infrastructure for Computer System Modeling," IEEE Computer, Vol. 35, No. 2, pp. 59-67, February 2002.

[10] CoWare Platform Architect.
http://www.coware.com/products/platformarchitect.php

[11] M. M. K. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, and D. A. Wood, "Multifacet's General Execution-driven Multiprocessor Simulator (GEMS) Toolset," ACM SIGARCH Computer Architecture News, Vol. 33, No. 4, pp. 92-99, November 2005.

[12] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi, and S. K. Reinhardt, "The M5 Simulator: Modeling Networked Systems," IEEE Micro, Vol. 26, No. 4, pp. 52-60, July 2006.

[13] J. Renau, B. Fraguela, J. Tuck, W. Liu, M. Prvulovic, L. Ceze, S. Sarangi, P. Sack, K. Strauss, and P. Montesinos, SESC: Cycle Accurate Architectural Simulator, January 2005. http://sesc.sourceforge.net

[14] L. Benini, D. Bertozzi, A. Bogliolo, F. Menichelli, and M. Olivieri, "MPARM: Exploring the Multi-Processor SoC Design Space with SystemC," Journal of VLSI Signal Processing, Vol. 41, No. 2, pp. 169-182, September 2005.

[15] J. Cong, K. Gururaj, G. Han, A. Kaplan, M. Naik, and G. Reinman, "MC-Sim: An Efficient Simulation Tool for MPSoC Designs," Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design, pp. 364-371, November 2008.

[16] Intel VTune Performance Analyzer.
http://www.intel.com/cd/software/products/asmo-na/eng/vtune/239144.htm

[17] OProfile - A System Profiler for Linux.
http://oprofile.sourceforge.net

[18] GNU gprof.
http://www.cs.utah.edu/dept/old/texinfo/as/gprof.html

[19] ARM11 MPCore Processor Technical Reference Manual, ARM Limited, 2008.

[20] S. Browne, J. Dongarra, N. Garner, G. Ho, and P. Mucci, "A Portable Programming Interface for Performance Evaluation on Modern Processors," International Journal of High Performance Computing Applications, Vol. 14, No. 3, pp. 189-204, April 2000.

[21] M. E. Shobaki and L. Lindh, "A Hardware and Software Monitor for High-Level System-on-Chip Verification," Proceedings of the 2nd International Symposium on Quality Electronic Design, pp. 56-61, San Jose, California, March 2001.

[22] Synopsys HSPICE.
http://www.synopsys.com/Tools/Verification/AMSVerification/CircuitSimulation/HSPICE/Pages/default.aspx

[23] Synopsys PrimeTime PX.
http://www.synopsys.com/Tools/Implementation/SignOff/Pages/PrimeTime.aspx

[24] D. Shin, H. Shim, Y. Joo, H.-S. Yun, J. Kim, and N. Chang, "Energy Monitoring Tool for Low-Power Embedded Programs," IEEE Design and Test of Computers, Vol. 19, No. 4, pp. 7-17, July 2002.

[25] D. Brooks, V. Tiwari, and M. Martonosi, “Wattch: A Framework for Architectural-Level Power Analysis and Optimizations,” Proceedings of the 27th International Symposium on Computer Architecture, pp. 83-94, Vancouver, British Columbia, Canada, June 2000.

[26] W. Ye, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, “The Design and Use of SimplePower: A Cycle-Accurate Energy Estimation Tool,” Proceedings of the 37th Design Automation Conference, pp. 340-345, Los Angeles, California, June 2000.

[27] R. Ben Atitallah, S. Niar, and J.-L. Dekeyser, “MPSoC Power Estimation Framework at Transaction Level Modeling,” Proceedings of the 19th International Conference on Microelectronics, pp.245-248, Cairo, Egypt, December 2007.

[28] G. Contreras and M. Martonosi, “Power Prediction for Intel XScale Processors Using Performance Monitoring Unit Events,” Proceedings of the 10th International Symposium on Low Power Electronics and Design, pp. 221-226, San Diego, California, August 2005.

[29] G. Contreras, M. Martonosi, J. Peng, G.-Y. Lueh, and R. Ju, “The XTREM Power and Performance Simulator for the Intel XScale Core: Design and Experiences,” ACM Transactions on Embedded Computing Systems, Vol. 6, No. 1, February 2007.

[30] J. Gaisler, E. Catovic, M. Isomäki, K. Glembo, and S. Habinc, GRLIB IP Library User’s Manual Version 1.0.20, Aeroflex Gaisler, February 2009.

[31] GRMON User’s Manual Version 1.1.35, Aeroflex Gaisler, March 2009.

[32] ALTERA DE3 Development and Education Board User Manual, Terasic Technologies.

[33] Altera Quartus II.
http://www.altera.com/products/software/quartus-ii/subscription-edition

[34] D. Hellström, Manual: SnapGear Linux for LEON Version 1.38.0, March 2009.

[35] Synopsys Design Compiler.
http://www.synopsys.com/Tools/Implementation/RTLSynthesis/Pages/DesignCompilerGraphical.aspx

[36] Mentor Graphics ModelSim.
http://www.mentor.com/products/fv/modelsim

[37] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, “The SPLASH-2 Programs: Characterization and Methodological Considerations,” Proceedings of the 22nd International Symposium on Computer Architecture, pp. 24-36, Santa Margherita Ligure, Italy, June 1995.