Global Journal of Researches in Engineering
Electrical and Electronics Engineering
Volume 12 Issue 7 Version 1.0 July 2012
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 2249-4596 & Print ISSN: 0975-5861

Power Profiling and Analysis of MI-Benchmarks Using Xscale Power Simulator (XEEMU)

By Nishant Kumar & Rahul Johari
Guru Gobind Singh Indraprastha University, India

Abstract - It is often required to compute the power consumed by an application to avoid later surprises of low battery life, or high temperature a device reaches while running the application; potentially damaging the device. There are three ways to find power consumed by a system for executing a particular application. 1) Run application on a reference board and measure the power dissipated. 2) Simulate the power consumed by the target system by running standard applications on it, but simulations can be time consuming. 3) Use heuristic formulas based on pre-analyzed simulation data. In this paper we come with some empirical formulas to calculate the power of a particular application, So that we do not need to run extensive simulations.

Keywords : *VLSI, Xscale, XEEMU.*

GJRE-F Classification : *FOR Code: 090699*

Strictly as per the compliance and regulations of:

Power Profiling and Analysis of MI-Benchmarks Using Xscale Power Simulator (XEEMU)

Nishant Kumar [α] & Rahul Johari [σ]

Abstract - It is often required to compute the power consumed by an application to avoid later surprises of low battery life, or high temperature a device reaches while running the application; potentially damaging the device. There are three ways to find power consumed by a system for executing a particular application. 1) Run application on a reference board and measure the power dissipated. 2) Simulate the power consumed by the target system by running standard applications on it, but simulations can be time consuming. 3) Use heuristic formulas based on pre-analyzed simulation data. In this paper we come with some empirical formulas to calculate the power of a particular application, So that we do not need to run extensive simulations.

Keywords : *VLSI, Xscale, XEEMU.*

I. Introduction

Power consumed by a particular application running on a device defines:

1) Battery life of the device
2) Maximum temperature device can reach while running the application

Power dissipated in embedded systems can be reduced with multiple hardware optimization techniques, such as transistor resizing, low-voltage design techniques and frequency control methods. There is a considerable amount of work done in hardware power optimization; however these techniques are only applied in early design steps such as VLSI design and synthesis. Embedded software transformations are another way to reduce power consumption, since software is responsible for driving the circuits and components of the system. In terms of software optimization techniques, power dissipation can be reduced with compiler, instruction-level, and source code-level optimization methods. Most of the work done to reduce power consumption has been oriented to compiler optimization where several techniques have been created and incorporated into compilers. Source code and instruction-level optimizations are an alternative in low power consumption analysis. Although instruction-level optimizations give excellent results with respect to low power consumption, source code optimizations have advantages in terms of portability, readability, and maintenance. Some studies done in embedded software optimization have shown that source code optimization techniques tend to diminish power consumption.

II. Instruction Level Power Analysis

The increasing popularity of power constrained mobile computers and embedded computing applications drives the need for analyzing and optimizing power in all components of a system. Software constitutes a major component of systems where power is a constraint. In light of this, there is a clear need for considering the power consumption in systems from the point of view of software. Software impacts the system power consumption at various levels of the design. However, in order to systematically analyze and quantify this impact, it is important to start at the most fundamental level – the instruction level. Just as logic gates are the fundamental units of computation in digital hardware circuits, instructions can be thought of as the fundamental unit of software. Accurate modelling and analysis at this level is the essential capability needed to quantify the power costs of higher abstractions of software. Traditional power analysis tools are not suited for power analysis of software. Accurate tools exist only for the lower most levels of the design, but they can be slow and cumbersome to use, and often these cannot even be applied, since the lower level information is not available to the software or embedded system designer. These problems can be overcome if the current being drawn by the CPU during the execution of a program is physically measured. An instruction level power analysis technique based on this idea has recently been developed. This technique helps in formulating instruction level power models that provide the fundamental information needed to evaluate the power cost of entire programs.

The average power, ***P,*** consumed by a microprocessor while running a program is given by:

$$P = I \times VCC, \quad (1)$$

Where, ***I*** is the average current and *Vcc* is the supply voltage.

The energy, E, consumed by a program is further given by:

$$E = P \times N \times r, \quad (2)$$

Author α σ : Department of Electronics and Communication Engineering, University school Of Information Technology, Guru Gobind Singh Indraprastha University, India.
E-mails : nishant.usit@gmail.com, rahuljohari@hotmail.com

Where, N is the number of clock cycles taken by the program and T is the clock period.

Thus, the ability to measure the current drawn by the CPU during the execution of the program is essential for measuring its power/energy cost.

A simple and practical technique has been developed to measure this current. The power supply connections to the CPU are isolated from the rest of the system. The program under consideration is put in an infinite loop and executed on the CPU. The current waveform will now be periodic and an average value can be visually obtained from a standard off the shelf digital ammeter. [3]

III. MI-Benchmarks

A benchmark is a program used as reference to make comparisons about the performance of a system. Benchmarks represent real applications which are run on computing systems. Although the concept of benchmarking is widely used in the area of computing systems performance, it can also be used when measuring other metrics such as power consumption. A benchmark has three important features:

1. Benchmark programs are easy to use.
2. Small size.
3. May run on different platforms.

These benchmarks were divided in six groups that represent embedded systems market:

- Automotive
- Consumers
- Office
- Networking
- Security
- Telecommunications [5]

IV. Selected Benchmarks

For the purposes of this work, a subgroup of the benchmarks explained before was chosen. The benchmarks used in this study are shown in table

Table 1 : Selected from Mi-Benchmarks for the study of Power Consumption

BENCHMARK	GROUP
Jpegtran	Consumer
Basicmath1	Automotive
Basicmath3	Automotive
Basicmath4	Automotive
QSORT	Automotive
String search	office
Susan	Automotive

V. Simulation Tool Used

We have used XEEMU (Xscale power simulator) to find the power consumed by a software program. XEEMU is a fast, cycle-accurate simulator for the XScale architecture. In contrast to many other existing power simulators, which simulate power and performance of theoretical microprocessor architectures, XEEMU aims at the most accurate modelling of the XScale architecture possible, trading off flexibility for much more reliable results. XEEMU proves to be more accurate than the other power simulators in terms of runtime and energy estimation, due to its improved pipeline and power model and the cycle accurate simulation of the SDRAM subsystem. It offers a high flexibility through freely and independently configurable frequencies for the core clock and the memory. [6]

VI. Methodology

Let the total energy consumed by a program is E, energy consumed by a particular type of instruction is I_j, and N_j is the number of times the instruction consuming I_j energy is executed in the program. We can write following formula:

$$\sum I_j * N_j = E \qquad (3)$$

To solve the problem, we will use:

1) A power simulator, to find the total energy consumed (E) by a software program
2) An instruction trace generator to find
   a. Types of instructions executed (to derive the correct range of j)
   b. Total number of times, a particular type of instruction is executed (N_j)

We will run power simulator and instruction trace generator on standard programs to come up multiple equations; we can decide to run 6 programs to generate 6 equations in 1 variable). We will then solve the available equations and find power consumed by each instruction (I_j). Once we know the power consumed by instruction, I_j, we need not run time consuming simulation to find E. Instead, we can compute the approximate Energy consumed from the instruction trace. We took 6 Mi-benchmark programs and calculated the power of each.

VII. Analysis

In this analysis, we assume that all instructions consume equal amount of energy to execute. Since, we have 6 MI-Benchmarks; we have 6 equations in 1 variable.

Table 2 : Number of Instructions

Program	Number of instructions
Jpegtran	6129891
Basicmath1	515034
Basicmath3	327723
Basicmath4	29329196
QSORT	10878448
String search	11564494

Table 3 : Energy of programs from Simulator

Program	Energy
Jpegtran	0.0083
Basicmath1	0.0005
Basicmath3	0.0004
Basicmath4	0.0288
QSORT	0.0134
String search	0.0129

Table 4 : Test programs Data

Programs	Instructions	Energy
Susan	32308776	0.0340
Matrix add	10172676	0.0104
Matrix multiplication	12138715	0.0127

VIII. Experimental Results

We used GNU Octave to solve Over-determined case of Equations to get the following results.

ALL_INSTRS = [6129891; 515034; 327723; 29329196; 10878448; 11564494];

SIMULATED_ENERGY = [0.0083; 0.0005 ; 0.0004; 0.0288; 0.0134 ; 0.0129];

Energy_per_inst = ALL_INSTRS\SIMULATED_ENERGY = 1.0354e-009

test_program_total_instr1 =32308776;

SIMULATED_ENERGY1 = 0.0340;

empirical_energy1 = Energy_per_inst*test_program_total_instr1= 0.0335

test_program_total_instr2=10172676

SIMULATED_ENERGY2 =0.0104

empirical_energy2 = Energy_per_inst*test_program_total_instr2=0.010532

test_program_total_instr3=12138715

SIMULATED_ENERGY3= 0.0127

empirical_energy3 = Energy_per_inst*test_program_total_instr3=0.012568

IX. Error Calculation

%Error=[(Theoretical Value-Experimental Value)/Theoretical Value] * 100

1) **E1** = [(0.0340-0.0335)/ 0.0340] x 100

   E1= 1.4705

2) **E2** = [(0.0104-0.010532)/ 0.010532] x 100

   E2=-0.0132

3) **E3** = [(0.0127-0.012568)/ 0.0127] x 100

   E3 = 1.0393

X. Future Work

We will perform different analysis by dividing the instructions into multiple categories:

1) Memory instructions.
2) Non memory instructions.
3) ALU instructions.
4) Branch instructions.
5) Register Transfer instructions.

Now we will calculate the energy consumed by each type of instruction. We have multiple equations in 2,3,4,5 variables. By solving them we get the energy consumed by one instruction of each type. By this information we can calculate the energy of a particular program only by the knowledge of number of instructions executed of each type.

References Références Referencias

1. Tiwari, S. Malik, and A. Wolfe. Power analysis of embedded software: A first step towards software power minimization. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2(4):437 – 445, Dec. 1994.
2. Tiwari and M. Tien-Chien Lee. Power analysis of a 32-bit embedded microcontroller. Design Automation Conference. Proceedings of the ASP-DAC '95/CHDL '95/VLSI '95. IFIP International Conference on Hardware Description Languages; IFIP International Conference on Very Large Scale Integration., Asian and South Pacific, pages 141 – 148, Aug., Sept. 1995.
3. Tiwari, S. Malik, A. Wolfe, and M.T. Lee. Instruction level power analysis and optimization of software. Proceedings of 9th International Conference on VLSI Design, Bangalore, India, pages 326 – 328, Jan. 1996.
4. M. Lee, V. Tiwari, S. Malik, and M. Fujita. Power analysis and low-power scheduling techniques for embedded DSP software. Fujitsu Scientific and Technical Journal, vol.31:215-229, 1995, 1995.
5. Matthew R. Guthaus, Jeffrey S. Ringenberg, Dan Ernst, Todd M. Austin, Trevor Mudge, Richard B. Brown -MiBench: A free, commercially representative embedded benchmark suite
6. Zolt´an Herczeg1, ´Akos Kiss1, Daniel Schmidt2, NorbertWehn2, and Tibor Gyim´othy1 -XEEMU: An Improved XScale Power Simulator
7. Wasabi software development tools user's guide for Intel's Xsacle micro- architecture. Version 1.0, March 2004
8. David Brooks, Vivek Tiwari, Margaret Martonosi-Wattch: A Framework for Architectural-Level Power Analysis and Optimizations

9. http://www.mathworks.in/help/techdoc/math/f4-983672.html#f4-2064
10. http://en.wikipedia.org/wiki/Instruction_set_simulator