The Emerging Power Crisis in Embedded Processors: What can a (poor) Compiler do?

L. N. Chakrapani
College of Computing
Georgia Institute of Technology
Atlanta, Georgia, 30332

P. Korkmaz, V. J. Mooney III, K. V. Palem, K. Puttaswamy
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia, 30332

W. F. Wong
Dept. of Computer Science
National University of Singapore
Singapore 117543

Abstract

It is widely acknowledged that even as VLSI technology advances, there is a looming crisis that is an important obstacle to the widespread deployment of mobile embedded devices, namely that of power. This problem can be tackled at many levels like devices, logic, operating systems, micro-architecture and compiler. While there have been various proposals for specific compiler optimizations for power, there has not been any attempt to systematically map out the space for possible improvements. In this paper, we quantitatively characterize the limits of what a compiler can do in optimizing for power using precise modeling of a state-of-the-art embedded processor in conjunction with a robust compiler. We provide insights to how compiler optimizations interact with the internal workings of a processor from the perspective of power consumption. The goal is to point out the promising and not so promising directions of work in this area, to guide the future compiler designer.

1. Introduction

The widespread deployment of embedded processors in mobile devices promises to open up new frontiers in applications. However, an important barrier that can severely limit this development is the issue of power consumption. An energy efficient device, i.e. one that consumes low amount of power over time, can potentially last longer and would require less bulky power supply units. In most devices, the computing element accounts for a high portion of the energy consumption [1]. This is a well-known issue and is being attacked from various fronts.

CASES'01, November 16-17, 2001, Atlanta, Georgia, USA.

First is the design of low power VLSI devices and logic. It is well known that these make the most significant contribution to saving power. Next is the use of novel micro-architecture techniques such as voltage scaling. The third front is in the software runtime system like the operating system, which can utilize power saving micro-architectural features and schedule tasks accordingly. An obvious trick the operating system can play, for example, is to put the processor in a low power mode during idle periods. Lastly, given an application, a compiler can attempt to optimize it for power. It is this last issue, which this paper will address.

There have been a number of studies on compiler optimizations for power, for example [4]. However, as far as we know, there are no systematic studies that address the following question: "what phenomena in the interactions of the compiler, the application and the micro-architecture of a processor gives rise to energy savings?"

The main contributions of this paper are as follows:

- We present an integrated infrastructure consisting of a compiler and a gate-level model for a state-of-the-art embedded processor, namely an ARM-like processor [7], which we used to study precise energy consumption patterns. Actual power measurements on a StrongARM development board are also used to support our claims.
- We quantitatively characterize the subtle interactions that accounts for power savings achieved by compiler optimizations.
- We classify compiler optimizations into broad categories of how they achieve power savings and in doing so; point out the promising directions of research in this area.
- We point out the limitations compilers face in existing embedded processors when optimizing for power.

It is hoped that this exploration will serve as a map for compiler writers wishing to tackle this important issue.

The paper is organized as follows: In Section 2, we present the detailed model of our experiment infrastructure. In Section 3, we address a number of fundamental questions regarding compiler optimizations for power by means of carefully crafted experiments. These experiments were performed using our precise processor model simulator as well as actual measurements on a development board. In Section 4, we propose a taxonomy of compiler optimizations for power and examine how previous proposals can be fitted in our classification system. In Section 5, we propose a number of recommendations arising from the insights we gained from the experiments. This is followed by a conclusion.

2. The Anatomy and Physiology of a Processor

The twin issues of modeling and optimization of power consumption have to be addressed at several layers for an accurate and thorough result. Previous compiler optimization research in this area has either relied on actual power measurements of the processors [2] or has relied on architectural simulation [3], mathematical techniques and regression to model the behavior of the hardware. Actual measurement of the power consumed by a processor is difficult and error prone. Even if achieved, detailed information and insight regarding the power consumed by each of the architectural subsystems might not be available. On the other hand mathematical techniques, regression and architectural simulation might fail to provide accurate and reliable results.

Our power research infrastructure consists of an optimizing compiler infrastructure called Trimaran [6]. A backend of Trimaran, called *Triceps* has been developed to generate code, which targets the ARM [7] architecture. The second part consists of a Verilog model of an ARM-like RISC processor developed by the University of Michigan [5]. The RTL core has been synthesized with the Synopsys Design Compiler [8] targeted towards a 0.25 micron TSMC library. The synthesized gate level netlist is placed and routed using Cadence Silicon Ensemble and Cadence Virtuoso.

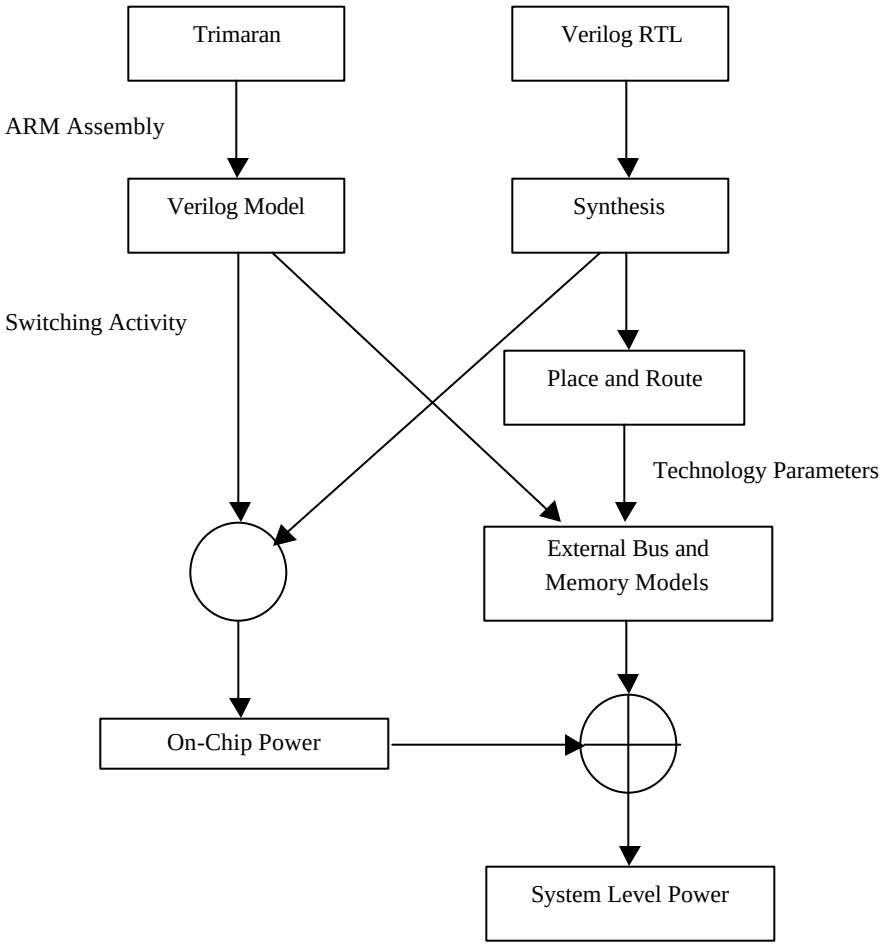


Fig 1. Infrastructure for Power measurement

Synopsys Power Compiler is used as the power estimation and analysis tool. This tool has two interfaces, one to the simulation environment and the other to the synthesis environment. The switching activity of the various functional units of the ARM-like[1] processor for a particular benchmark assembly generated by Trimaran is obtained from the simulation environment. Then the switching activity is annotated onto the synthesis environment to provide measurements of static and dynamic power. External models for bus and main memory are fed with the switching activity on the bus to obtain a measure of power dissipated by the bus and main memory.

Thus our infrastructure provides an accurate and detailed estimate of the gate level power consumption while functioning as a seamless infrastructure for detailed and accurate study of compiler optimizations, architectural innovations and their impact on power and performance. The experiments are organized into studies of the ALU subsystem, the Register file and the Instruction and Data cache.

In addition to the Trimaran–Verilog RTL infrastructure described above, there is a StrongARM SA 110-based development board called "Skiff" from the Compaq Western Research Laboratories. The processor on the board has a power supply that is distinct from that used by the other components on the board. Power consumed by the processor is measured by means of a sensitive measurement device measuring the current drawn by the processor. Experiments are run on this board and the results are crosschecked with those produced by the Trimaran–Verilog RTL infrastructure. Since the latter measures actual power consumption albeit at the granularity of the entire SOC processor while the former only models the core processor, measurements taken by both methods do not agree in absolute terms. However, what is important is whether the correlation of the results are preserved as this serves as an important verification of the model's results.

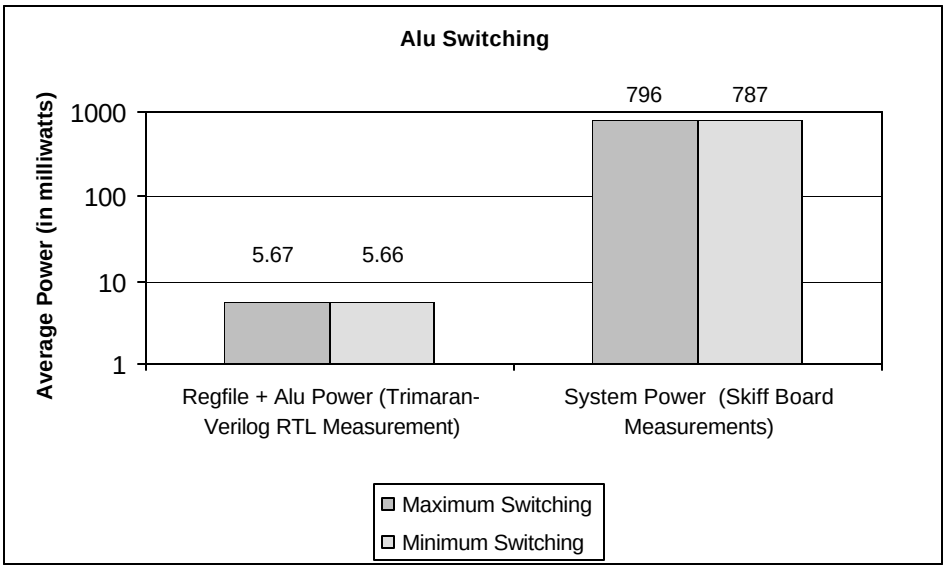
3. The Key Insights

The experiments were conducted on the power measurement infrastructure with an aim of answering specific questions to get an insight into power dissipation and energy consumption.

I. The ALU Subsystem

A) Does reduction in switching activity reduce power?

It was proposed in [9] that reduction in switching activities of the ALU translates to energy savings. To explore this, segments of code which compute the same value, but one which is optimized for minimal switching of the inputs and the other for which is modified for maximum switching were run on the model and power numbers collected. The Hamming distance, which is a count of the number of bit flips between two values, was considered as a measure of switching.



[1] Since the actual details of the internal ARM micro-architecture is proprietary information, the Verilog model we used is that of a RISC approximation that executes the ARM instruction set.

It can be seen that there is virtually no improvement in power dissipation and energy consumption. This is in accordance with earlier studies on other architectures [2]. Hence the applicability of this technique should be further investigated before being pursued as a viable optimization for power and energy.

B) Do all type of ALU instructions consume the same amount power?

To explore this different types of instructions were run in a loop and the power numbers collected. It is observed that logical and add / subtract instructions consume the same amount of power and multiply consumes higher amount of power as well as takes a longer number of cycles to complete, thereby consuming more energy. Hence strength reduction by replacing multiplies with a series of additions and shifts would be beneficial. Further, since multiply takes only about 30% more power than adds and subtracts, care should be taken so that the number of instructions replacing multiply is no more than 30% of the total number of cycles taken by multiply instruction. Otherwise, the penalty incurred due to a larger cycle count would have a negative impact on energy even though the instructions themselves have a positive impact on power.

II. The Register File

A) Does the number of accesses to the register files play a role in power consumption?

Two experiments, one that accesses values from registers and another that uses immediate operands were run on the Skiff board as well as on the Verilog model.

	ALU + Reg File Power (Trimaran-Verilog) in mW	System Power (Skiff Board) in mW
Register Operands	4.784	776
Immediate Operands	4.784	760

It can be observed that while the system power measured from the board shows a difference, the Verilog model does not. This is because in the model for both immediate as well as register operands the register file is accessed and in the appropriate operand is multiplexed into the ALU. In general, we can assume that number of accesses to the register file matters. Hence aggressive copy propagation and constant propagation should be done. Wherever possible immediate addressing should be used. The effectiveness of these optimizations is architecture dependent.

B) Does the value accessed from the register file affect power?

To explore this question, examples were constructed that access values from the register with maximum, intermediate and minimum amount of switching. Hamming distance was taken as a measure of switching. It is observed that more the switching of the value accessed from the register file more is the reported power consumption. The combined ALU and Register file power consumption is observed to increase by about 12% between minimum switching and maximum switching. This suggests a possible course of optimization through instruction scheduling and code transformation so as to minimize switching in register file access.

	Regfile + Alu Power (Trimaran-Verilog) in mW	System Power (Skiff Board) in mW
Maximum Switching	5.573	769
Intermediate Switching	5.105	736
Minimum Switching	4.978	708

III. The Cache subsystem

A) Do the number of accesses to the cache affect power?

Three sections of code were run on the Trimaran-Verilog model and on the skiff board and the power numbers were collected. The code accessed the data cache 100%, 50% and 0% of times respectively. It is observed that there is about 24% more power consumption between no access to the data cache and full access to the data cache.

	Data Cache Power (Trimaran-Verilog) in mW	System Power (Skiff Board) in mW
Maximum Access	152.168	1.150
Intermediate Access	141.976	1.040
Minimum Access	129.577	0.930

B) Does the value accessed from the cache matter?

To determine if heavy switching in the value accessed from the cache plays a role in power and energy consumption, experiments were setup where in one instance values were accessed alternately such that all 32 bits of the value

switched. In the latter case no bits switched in the cache access. The measured power numbers are:

	Data Cache Power (Trimaran-Verilog) in mW	System Power (Skiff Board) in mW
Maximum Switching	113.11	888
Minimum Switching	113.11	867

Whereas the Trimaran-Verilog model does not show any variation the actual system power shows about 2.5% decrease in power. From this we can conclude implementation details of the cache play a role and power savings in case of no switching is perceptible but very less. One optimization that can be performed is to optimize cache accesses so that the accessed values don't switch as much.

4. The Taxonomy

Based on our study, we would like to propose taxonomy of compiler optimizations from the perspective of power. We can classify all known compiler optimizations [10] into one of three classes.

Class A Optimizations

These optimizations benefit energy due to an improvement in performance. For a program, the energy consumed is a product of the average power dissipation per cycle and the number of cycles taken for completion. Any reduction in the number of cycles in the completion time would automatically translate into improvements in energy consumption [4]. Most of the current optimizations for power fall under this category. Reductions in the number of loads and stores, procedure cloning, loop unrolling, procedure inlining, loop transformations, partial redundancy elimination are a few examples.

Class B Optimizations

These optimizations benefit energy while having no impact or a decrease in performance. Innovations in instruction scheduling, register pipelining, innovations in code selection to replace high power dissipating instructions with other instructions are a few examples.

Class C Optimizations

These optimizations are the ones that are bad for power dissipation and energy consumption. Typically the optimizations that have a negative impact on performance also have a negative impact on energy consumption.

5. Recommendations

5.1. To the Compiler Designer

Based on the results presented above it is clear that the highest impact on energy consumption is by improving the performance of the code. Using the current day architectures the only optimizations for power that can be done is scheduling register accesses so that the register file switching is minimized and proper code selection and aggressive but careful strength reduction to replace power hogging instructions with other less power consuming instructions.

5.2. To the Computer Architect

Based on the measurements above it is clear that with current day architectures, *novel* compiler optimizations that target power are few. However there are several architectural innovations that can be exposed to the compiler so that many more Class B optimizations are facilitated. Examples would be a bit width sensitive ALU that can derive hints from the compiler, compiler controlled voltage and frequency scaling among other things.

6. Conclusion

In this paper, we explored the limits of a compiler in optimizing for power for existing processor architectures. Using a precise model of a popular embedded microprocessor, supported by actual measurement on a development board, experiments carefully crafted to test specific aspects of the micro-architecture were performed. Our study reveals that to a great extent compiler optimizations for locality and performance translate to optimizations for power. This is in agreement with other similar studies. We have also identified a few opportunities for novel optimizations such as reduction in register file switching do exist. However, our experiments have shown that in order to obtain substantial gains in energy saving, innovating micro-architectural features and exposing them to the compiler is necessary.

References

1. Marc A. Viredaz and Deborah A. Wallach, "Power Evaluation of a Handheld Computer: A Case Study," *Research Report 2001/1 Compaq Western Research Lab*.
2. Vivek Tiwari, Sharad Malik and Andrew Wolfe, "Compilation Techniques for Low Energy: An Overview," In *1994 Symposium on Low-Power Electronics, San Diego, CA, October 1994.*
3. David Brooks, Vivek Tiwari and Margaret Martonosi, "Wattch - A Framework for architectural-level power analysis and optimizations," In *Proc. Of The 27th Annual International Symposium on Computer architecture, 2000,* Pages 83 – 94.

4. Madhavi Valluri and Lizy John, “Is Compiling for Performance == Compiling for Power?” *The 5th Annual Workshop on Interaction between Compilers and Computer Architectures (INTERACT-5), Monterrey, Mexico*, January 20, 2001.

5. ARM Verilog Model, http://www.eecs.umich.edu/~tnm/power/verilog_arm.html, *University of Michigan, Electrical Engineering and Computer Science.*

6. *Trimaran, http://www.trimaran.org*.

7. ARM Ltd., http://www.arm.com.

8. Synopsys Design Compiler, *http://www.synopsys.com/products/logic/design_comp_ds.html*.

9. Su, C., Tsui, C., and Despain, A. “Low Power Architecture Design and Compilation Techniques for High-Performance Processors”. In Digest of Papers: *Spring COMPCON 94*, pp. 489--498, February 1994.

10. Steven S. Muchnick, *Advanced Compiler Design and Implementation*. Morgan-Kaufmann Publishers. 1997.