Impact of Source Code Optimizations on Power Consumption of Embedded Systems

David A. Ortiz and Nayda G. Santiago
Electrical and Computer Engineering Department
University of Puerto Rico, Mayagüez Campus
Mayagüez, PR 00681-9042
e-mail:{david.ortiz, nayda.santiago}@ece.uprm.edu

***Abstract*— Power consumption is an important constraint in the design of battery-operated embedded systems. Minimizing power dissipation may be handled in terms of hardware or software optimizations. Source code-level optimization techniques have been used as an alternative to achieve low power consumption when programming embedded systems. However these techniques should be analyzed with statistical sound methods in order to reach strong conclusions about their real impact on power consumption. In this work, source code optimizations were applied on a set of representative benchmarks for embedded processors (MiBench) to analyze whether the techniques have or not an effect on power dissipation of a set of microprocessor-based platforms. Design of experiments techniques (DOE) and analysis of variance (ANOVA) were used to achieve statistical sound conclusions. Results showed that not all optimizations have a significant effect on power consumption, moreover some techniques depend on the target platform where they are run.**

I. Introduction

Power consumption is one of the main design constraints for devices in embedded systems such as wireless sensors, computer systems, and biomedical devices. The main reasons for analyzing power consumption in these systems is due to limited battery life time, heat dissipation, size constraints, and costs [1], [2]. 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 [3]. There is a considerable amount of work done in hardware power optimization, however these techniques are only applied in early design steps [4].

Embedded software transformations are another way to reduce power consumption, since it 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 compilers optimization [5] where several techniques have been created and incorporated to compilers [6].

Source code and instruction-level optimizations appear as an alternative in low power consumption analysis [7]–[9]. Although instruction-level optimizations have shown good results with respect to low power consumption, source code optimizations have advantages in terms of portability, readability, and maintenance [10], [11]. Some studies done in embedded software optimization have shown that source code optimization techniques tend to diminish power consumption [12].

In this work, we investigated the actual impact of source code-level optimizations on power consumption of different embedded platforms. We have selected a set of benchmarks to perform our study, and design of experiments (DOE) methods were used to be able to track cause and effect on the data obtained. We have shown that some source code-level optimizations have an effect on power consumption of embedded systems, however it is important to base the evaluation of the effects on statistical methods in order to attain robust conclusions. This document is organized as follows: in section 2, related work in low power techniques for microprocessors in terms of embedded software is discussed. Section 3 illustrates the methodology implemented and section 4 shows the results obtained and the analysis performed. Finally, conclusions are presented.

II. Related Work

The reduction of power due to the embedded software running on the microprocessor has been addressed by compiler, instruction, or source code-level optimizations.

A. Compiler Optimizations

Compiler optimizations have been reported by [5], [13], [14]. Esakkimuthu *et al.* [13] found that better results are obtained in terms of memory energy savings when optimizing compiler vs. hardware optimizations. Zambreno *et al.* [5] analyzed the effect of performance optimizations on memory power consumption, and observed that a good performance optimization technique may not give the best results in terms of power reduction. Ravindran *et al.* [14] proposed an approach for compiler-directed dynamic placement of instructions into a low-power code cache. Ravindran showed that when applying dynamic placement techniques, energy savings may be achieved on the WIMS microcontroller platform.

B. Instruction-Level Optimizations

Instruction-level optimizations can alter power consumption in embedded systems [2], [8], [15].

The first works in the area of power consumption at instruction level were studies done by Tiwari *et al.* [15], where an instruction-level power model was developed to

978-1-4244-2332-3/08/$25.00 © 2008 IEEE

measure power consumption of embedded processors. This work showed power reductions when applying assembly-level optimizations.

Another significant study in this area is the work of Russell and Jacome [2]. Their work is focused on a statistical analysis to know if a parameter can model power consumption. In their study, they performed an instruction-level energy estimation model for embedded processors. A final study that is important to mention is the work done by Oliver *et al.* [8]. Here, the authors analyzed some factors at the instruction-level that have effect on power consumption, such as branches.

C. Source Code-Level Optimizations

Source code-level optimizations for execution time have been studied extensively by Leupers [10]. Moreover programming style may have effect on power consumption of embedded systems. These optimizations were analyzed by [7], [11], [12].

There are important works in terms of source code-level optimization. Leupers [10] and Sharma [11] classified source code optimizations in machine-independent and machine-dependent.

In terms source code optimization for power reduction, Simunic *et al.* [12] classified code optimization techniques in algorithmic, data, and instruction-flow optimizations. In our case we used algorithmic optimizations since they do not take into consideration the target platform. Dalal *et al.* [7] studied software-dependent components such as arithmetic circuits, data busses, and memories, as a way to lower power consumption in embedded applications. Sharma and Ravikumar [11] presented a study of the implementation of the ADPCM codec benchmark. In this work, optimization techniques applied at the source code-level were classified into structural and machine-dependent optimizations.

III. Methodology

In contrast to previous work found in literature, we used statistical sound methods to study whether there is a significant effect on power reduction when applying source code-level optimization techniques on programming of embedded systems.

The methodology is divided into six parts. First, a selection of three target platforms for studying the impact of source code optimizations was done. The platforms chosen for this study were an Intel 8051, a Motorola HC12, and an ARM7TDMI. Second, a set of benchmarks for embedded processors called MiBench [16] were chosen for comparison purposes. These benchmarks represent six groups of embedded systems applications: automotive, telecommunications, office, networking, consumers and security. After that, a profile of the benchmarks was performed in order to know the critical code structures where more percentage of time was spent. Next, a set of loop-oriented optimization techniques originally designed to improve performance were analyzed in terms of power dissipation of an Intel 8051 platform, for identifying potential source code-level optimization techniques [17]. Afterwards, three source code-level optimizations were chosen [12], and design of experiment techniques (DOE) [18] were used to analyze their impact on power consumption. Finally, instrumentation of the target platforms was done, and analysis of variance (ANOVA) was used to make a statistical analysis.

A. Target Platforms

To perform the analysis of power consumption on embedded systems, three of the most representative embedded processors were chosen. Some features of the selected platforms are described in this section.

The Intel 8051 is an 8-bit CISC core developed for embedded devices. A special feature of the Intel 8051 is a boolean module that allows to perform logic operations. The Motorola HC12 is a 16-bit CISC core, and the ARM7TDMI is a 32-bit RISC core, both for high performance purposes. The three platforms selected, are used depending on the target application. The Intel 8051 is used in automotive and control processes, while the Motorola HC12, and the ARM7TDMI have been used in literature for data processing purposes. In terms of power consumption, all the selected platforms present low power features, that make them appropriate when designing systems with power constraints.

B. Benchmarks

In order to analyze power consumption of the three platforms selected, a set of representative benchmarks for embedded applications had to be chosen. Guthaus *et al.* [16] developed a group of benchmarks called MiBench with the purpose of measuring performance on embedded processors. Since most of traditional benchmarks are designed to make performance analysis on desktop computers, MiBench were developed to measure performance and other metrics on microprocessor-based systems.

Mibench is composed of 35 applications, divided in six groups that represent commercial applications of embedded systems: automotive, consumers, office, network, security, and telecommunications. Since these benchmarks are written in C language, they have features of portability and readability, making them adaptable to any embedded platform. For the purposes of this work a subgroup of MiBench was chosen. These benchmarks are shown in table I.

C. Profiling

To be able to apply source code-level optimization techniques on the benchmarks selected, it was important to perform

TABLE I
BENCHMARKS SELECTED FROM MIBENCH

Group	Benchmarks
Automotive	Basicmath, Bitcount, Qsort
Office	Stringsearch
Dijkstra	Network
Telecommunications	ADPCM Coder, ADPCM Decoder, FFT

a profile of each program in order to identify critical code structures of the benchmarks. This characterization was done with a profiler provided by the *gcc* compiler. Some statistics achieved with the profiler are shown in table II.

D. Source Code-Level Optimizations

Source code-level optimizations can be classified in algorithmic optimizations [11], [12], loop transformations, and function inlining methods [10]. Following is a definition of the three techniques used in this study.

Loop unrolling is an optimization technique where the body of a loop is copied several times. In function inlining, the body of the function is inserted directly in the code structure where it is used. Variables declaration is used to replace variable types by others which tend to lower power consumption.

Since loop unrolling has shown good results in terms of low power consumption for the Intel 8051 platform in previous literature [17], this transformation was one of the three optimization techniques chosen to optimize the proposed benchmarks. Also, function inlining and variable declarations techniques were implemented, which have also been demonstrated in literature [5], [12].

E. Instrumentation

Power consumption of an embedded system can be calculated evaluating the supply voltage and the average current drawn by the platform [15]. This calculation is given by equation 1.

$$P_{system} = V_{cc} I \tag{1}$$

where P_{system} is the power consumption of the embedded system, I is the average current and V_{cc} is the supply voltage. In order to determine the power consumption, we measured the current through the embedded system. This measurement was performed connecting an ammeter between the power supply and the platform, while running non-optimized and optimized versions of the benchmarks within an infinite loop.

TABLE II
PROFILE OF THE BENCHMARKS

Benchmark	Function	% Time
Basicmath	usqrt	98.01
Bitcount	bit_shifter	31.10
	bit_count	26.83
	ntbl_bitcnt	12.50
	bitcnt	11.59
Qsort	qsort	88.52
	init_search	27.78
	str_search	25.00
Dijkstra	dijkstra	62.50
	enqueue	25.00
ADPCM Coder	adpcm_coder	98.60
ADPCM Decoder	adpcm_decoder	88.12
FFT	fft	97.53

F. Design of the experiment and Statistical Analysis

An experiment is a test where a set of variations are performed on a system in order to observe its response under certain conditions [18]. In DOE, an analysis of a test is done to chose an appropriate design of experiment, depending on the factors that are going to be studied in a process. The goal when applying DOE techniques on a test, is to reach valid conclusions about the impact of the design factors on the behavior of the system under analysis.

In this work, the objective was to analyze the actual effect of the source code-level optimizations selected on power consumption of different platforms, without biased interpretations about results. In our case, the factors of the experiment were the optimization phase of each technique, and the benchmarks. After that, power consumption of the platforms selected, was recognized as the variable under study. Then, a two-factor factorial design, was chosen to study the effect of the design factors on the response variable. Finally, the experiment was performed, and the statistical analysis of the data was done for the three optimization techniques considered in this study. Table III shows the two-factor factorial design for loop unrolling technique on each platform selected.

IV. EXPERIMENTAL RESULTS

The analysis of variance (ANOVA) is a statistical test used to evaluate the effect of a set of factors on a response variable. It is analyzed depending on the significance level or *p-value*. The *p-value* is a measure of how much evidence exists to accept or reject an hypothesis. The hypothesis we have is that the factors of the experiment do not have any effect on the outcomes. The significance level is selected according to the type of problem. In this case a *p-value* of 0.05 was chosen, hence there is a 5% of probability to reject the initial hypothesis. In this study ANOVA was used to know whether power reduction on the platforms selected was due to the source code-level optimization techniques used or to random factors.

TABLE III
TWO-FACTOR FACTORIAL DESIGN FOR LOOP UNROLLING TECHNIQUE ON EACH PLATFORM SELECTED: POWER CONSUMPTION *(mW)*

Optimization phase (8051 platform)	Benchmarks		
	Qsort	Dijkstra	Stringsearch
Non-Optimized	93.42	91.44	95.4
Optimized	92.23	80.66	92.18

Optimization phase (HC12 platform)	Benchmarks		
	Qsort	Dijkstra	Stringsearch
Non-Optimized	668.97	667.89	668.88
Optimized	667.71	666.81	667.53

Optimization phase (ARM7TDMI platform)	Benchmarks		
	Qsort	Dijkstra	Stringsearch
Non-Optimized	1300.05	973.17	1292.94
Optimized	1268.01	956.88	1268.19

Table IV summarizes the results obtained from the ANOVA about the actual impact of each optimization technique on power consumption when analyzing the three platforms.

TABLE IV
SIGNIFICANT IMPACT OF OPTIMIZATION TECHNIQUES ON EACH PLATFORMS: *p-value*

Optimization Technique	Platforms		
	Intel 8051	Motorola HC12	ARM7TDMI
Loop Unrolling	No impact $p = 0.225$	Impact $p = 0.004$	Impact $p = 0.033$
Function Inlining	No impact $p = 0.196$	No impact $p = 0.464$	No impact $p = 0.184$
Variable Types	No impact $p = 0.275$	Impact $p = 0.011$	No impact $p = 0.201$

V. CONCLUSIONS

We observed that the three techniques selected in this study tend to diminish power consumption of the three platforms considered, however the statistical analysis performed showed that only loop unrolling and variable types declaration had a significant impact on the power dissipated by the Motorola HC12 and ARM7TDMI. Also, the statistical analysis showed that depending on the architecture, an optimization technique may have or not impact on power consumption. Based on the results obtained, the techniques applied on the Intel 8051 platform did not show a significant impact, while when applied on the Motorola HC12 and the ARM7TDMI, the optimization techniques showed a significant effect on power reduction of the system.

ACKNOWLEDGEMENT

This work has been supported by the Engineering Research Centers Program of the National Science Foundation under Award Number ERC-9986866.

REFERENCES

[1] A. Chatzigeorgiou and G. Stephanides. Energy issues in software design of embedded systems. *2nd WSEAS International Conference on Applied Informatics, Rethymnon, Crete, Greece*, Jul. 2002.

[2] J.T. Russell and M.F. Jacome. Software power estimation and optimization for high performance, 32-bit embedded processors. *International Conference on Computer Design: VLSI in Computers and Processors*, pages 328 – 333, Oct. 1998.

[3] I. Hong, D. Kirovski, Qu Gang, M. Potkonjak, and M.B. Srivastava. Power optimization of variable-voltage core-based systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18(12):1702 – 1714, Dec. 1999.

[4] N.K. Jha. Low power system scheduling and synthesis. *IEEE/ACM International Conference on Computer Aided Design*, pages 259 – 263, Nov. 2001.

[5] J. Zambreno, M.T. Kandemir, and A. Choudhary. Enhancing compiler techniques for memory energy optimizations. *Embedded Software. Second International Conference, EMSOFT 2002*, 2491:364 – 381, 2002.

[6] H. Mehta, R. Owens, M. Irwin, R. Chen, and D. Ghosh. Techniques for low energy software. *ISLPED - International Symposium on Low Power Electronics and Design*, pages 72 – 75, 1997.

[7] V. Dalal and C.P. Ravikumar. Software power optimizations in an embedded system. *Fourteenth International Conference on VLSI Design*, pages 254 – 259, Jan. 2001.

[8] J. Oliver, O. Mocanu, and C. Ferrer. Energy awareness through software optimization as a performance estimate case study of the MC68HC908GP32 microcontroller. *4th International Workshop on Microprocessor Test and Verification: Common Challenges and Solutions*, pages 111 – 116, May. 2003.

[9] Y. Yingbiao, Y. Qingdong, L. Peng, and X. Zhibin. Embedded software optimization for MP3 decoder implemented on RISC core. *IEEE Transactions on Consumer Electronics*, 50(4):1244 – 1249, Nov. 2004.

[10] R. Leupers. Code optimization techniques for embedded processors. *Kluwer Academic Publishers*, 2000.

[11] A. Sharma and C.P. Ravikumar. Efficient implementation of ADPCM codec. *Thirteenth International Conference on VLSI Design*, pages 456 – 461, Jan. 2000.

[12] T. Simunic, L. Benini, and G. de Micheli. Energy-efficient design of battery-powered embedded systems. *IEEE Transactions on Very Large Scale Integration Systems*, 9(1):15 – 28, Feb. 2001.

[13] G. Esakkimuthu, N. Vijaykrishnan, M. Kandemir, and M.J. Irwin. Memory ystem energy: Influence of hardware-software optimizations. *Proceedings of the 2000 International Symposium on Low Power Electronics and Design, 2000. ISLPED '00*, pages 244 – 246, Dec. 2000.

[14] R.A. Ravindran, P.D. Nagarkar, G.S. Dasika, E.D. Marsman, R.M. Senger, S.A. Mahlke, and R.B. Brown. Compiler managed dynamic instruction placement in a low-power code cache. *International Symposium on Code Generation and Optimization*, pages 179 – 190, Mar. 2005.

[15] V. 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.

[16] M.R. Guthaus, J.S. Ringenberg, D. Ernst, T.M. Austin, T. Mudge, and R.B. Brown. MiBench: A free, commercially representative embedded benchmark suite. *IEEE International Workshop on Workload Characterization, 2001. WWC-4. 2001*, pages 3 – 14, Dec. 2001.

[17] D.A. Ortiz and N.G. Santiago. High-level optimization for low power consumption on microprocessor-based systems. *50th IEEE International Midwest Symposium on Circuits and Systems (MWSCAS'07)*, pages 1265 – 1268, Aug. 2007.

[18] D.C. Montgomery. Design and analysis of experiments. *Wiley, New York, 6th edition*, 2004.