High-Level Optimization for Low Power Consumption on Microprocessor-Based Systems

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

Abstract— **Power consumption is an important constraint in the design of battery-operated embedded systems. The problem of minimizing power dissipation may be handled in terms of hardware or software optimizations. High-level language optimization techniques appear as an alternative to achieve low power consumption when programming embedded systems. In this work, software optimization techniques were applied to a set of code segments in a high-level language, in order to analyze the effect that source code optimizations have on the power dissipation of microprocessor-based systems. Design of experiments (DOE) techniques were used in order to reach statistical sound conclusions about the actual impact that software optimization techniques have on power consumption.**

I. INTRODUCTION

Power consumption is one of the main designing constraints for embedded electronic 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 lifetime, heat dissipation, size constraints, and costs [1], [2]. The power dissipated in embedded systems can be reduced with multiple hardware optimizations, such as transistor resizing, low-voltage design techniques, and frequency control methods [3], [4], among others. There is a large body of work done in hardware power optimizations, but these techniques are applied only at early design steps, making difficult to reduce power consumption in later implementation stages [5], [6].

Power consumption is affected by the program running on a microprocessor. In terms of software optimization techniques, power dissipation can be reduced with compiler, low-level and high-level language optimization methods. Most of the work done to reduce power consumption has been oriented to embedded systems compilers optimization [4], [7], therefore several techniques have been created and incorporated in the compiler design [8], [9].

Low and high-level language optimization techniques appear as an alternative in low power consumption analysis [10], [11]. Although low-level language optimizations have shown to give good results with respect to power consumption, high-level language optimizations present advantages in terms of portability, readability, and maintenance [12]. Some studies done in software optimization have shown that high-level language optimization techniques tend to diminish power consumption [13].

In this study, certain segments of embedded software were optimized for performance and analyzed using design of experiments (DOE) methods. The purpose of this study is to evaluate the impact of a set of high-level language optimization techniques on the power consumption of embedded systems. Initial results show that those performance optimization techniques studied do not have a significant impact on power consumption.

This document is organized as follows. In section 2, related work in low power techniques for embedded systems in terms of software optimizations is discussed. Section 3 illustrates the methodology implemented. Section 4 explains the target microprocessor platform, and section 5 shows the results and the analysis. Finally, conclusions and future work are presented.

II. RELATED WORK

The reduction of power consumption due to the software running on a microprocessor can be addressed in different levels: compiler, low-level, or high-level language. All of these mechanisms have advantages and disadvantages depending on the target processor and architecture.

At compiler level, Ravindran *et al.* [9] proposed an approach for compiler directed dynamic placement of instructions into a low-power code cache. Ravindran showed that applying dynamic placement techniques, energy savings can be achieved on the WIMS microcontroller platform. Zambreno *et al.* [7] presented the importance that compiler optimizations have on memory power consumption. Since memory is one of the main sources of power dissipation, they made a comparison between performance and memory power optimization. Metha *et al.* [8] proposed a compilation technique for lower power consumption based on the energy consumed by the instruction register (IR) and register file decoder. Leupers [12] analyzed different techniques to design power-efficient compilers, and presented a set of software optimization techniques for compilers code generation.

In terms of low-level language, Tiwari *et al.* [14] analyzed the problem of power consumption from the software running on a microprocessor. An instruction-level method was designed, allowing the examination of the current and power generated by each instruction on the processor. Oliver *et al.* [11] proposed optimization of assembly code for a MC68HC-908GP32 microcontroller. They implemented instruction-level techniques and classified microprocessor instructions depending on the usage frequency, and number of cycles per instruction. Russell and Jacome [2] performed a similar work to Tiwari [14]. They analyzed the assembly code generated by an optimizing compiler assigning a power cost for each instruction. Russell concluded that minimizing execution time, the power consumption in the system will be reduced.

In relation to high-level language, Chatzigeorgiou and Stephanides [1] showed the importance of software optimization for low power consumption, presented hardware components dependent of embedded software, and analyzed software effects in terms of power. They compiled the full search motion estimation algorithm using the *C* compiler of the ARM, and used a profiler to obtain the energy consumed by the processor. The authors concluded that large energy savings are obtained when the power is addressed in software design. Ravikumar *et al.* [10] performed a study of source code and its effect on embedded systems power consumption. They optimized high-level code, and verified code size, execution speed and power dissipation in an ARM processor for the ADPCM speech compression algorithm. Yingbiao *et al.* [15], showed software optimization techniques for low power consumption applied to a RISC core MP3 decoder. They achieved high performance and memory space optimization, designing efficient software running on the microprocessor. Simunic *et al.* [13] proposed source level code optimization techniques for a MPEG decoder on an ARM microprocessor, and compared them with the ARM compiler results. They concluded that power savings achieved with source level code optimizations gave better results than those from compiler optimization tools.

In contrast to the previous work in literature, this work addresses high-level language optimizations to evaluate the power dissipation of a microprocessor-based system. In order to make this analysis, we designed an experiment for understanding the effect that software optimizations designed for time reduction have on power consumption.

III. Methodology

Since the objective of this work is to evaluate the effect that high-level language optimizations has on power consumption in embedded systems, we followed an experimental approach for this study. First, an experiment was designed using design of experiments (DOE) techniques. A set of loop-oriented optimizations were selected and applied to certain segments of code. Power consumption on the microprocessor board was measured while running the embedded software. Finally, obtained results were analyzed using statistical analysis applicable to DOE techniques.

A. High-Level Language Optimization Techniques

Certain optimization techniques originally designed for improving performance were analyzed in terms of the power dissipation of the microprocessor-based system. The techniques selected for our experiment were the following loop-oriented techniques: statement reordering, loop unswitching, loop peeling, scalar expansion, loop fusion, loop alignment, loop fission, nested loops, loop reversal, loop unrolling, and loop interchanging [16]. All these methods cannot be applied together in the same scenario due to the different structures of each code. Therefore, we used different fragments of code in order to test the selected optimization techniques.

B. Power Consumption Measurements

The power consumption of an embedded system can be calculated evaluating the supply voltage and the average current [14]. This calculation is given by (1).

$$P_{system} = V_{cc} I \qquad (1)$$

where P_{system} is the power consumption of the embedded system (*mW*), I is the average current (*mA*) and V_{cc} is the supply voltage (*V*). 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 microprocessor board, while running non-optimized and optimized versions of the code within an infinite loop.

C. Design of the Experiment

One of the fundamental steps in this work was the design of the experiment. This experiment was designed as a two-factor factorial design with one observation per cell, since the current measurements performed do not change from sample to sample. For this reason there was not replicate in each power measure.

There were two factors analyzed in this experiment. The first factor was the high-level language optimization techniques mentioned before *(factor A)* [16]. These optimization techniques were loop restructuring methods used to rearrange iterations and statements within loops. The second factor of the experiment was the optimization phases *(factor B)*. When high-level language optimization techniques are applied, non-optimized and optimized versions of the test code are obtained allowing comparing changes in terms of power consumption. There were eleven optimization techniques implemented, and two optimization phases. Analysis of variance (ANOVA) was used to evaluate the impact of each factor in the response of the experiment. This was done analyzing the *p-value* information given by the ANOVA. Table I shows the results obtained when performing the two-factor factorial experiment.

IV. Target Microprocessor Platform

The target platform for the experiments was an Intel 8051 microprocessor. The 8051 microprocessor has low power consumption features that make it an appropriate choice when designing systems with low power constraints.

V. Results and Analysis

ANOVA is a statistical model widely used for the analysis of the impact of a factor on the response of an experiment. It is analyzed depending on the significance level or *p-value*. The hypothesis we have is that the factors of the experiment do not have any effect on the outcomes. ANOVA will determine if this hypothesis is not true. The significance level is selected according to the type of problem. In this case a *p-value* of 0.05 was selected, hence if the *p-value* is less than 0.05, at least one factor has impact on the experiment, otherwise the evaluated factors do not have any effect on the power consumption. Before performing an ANOVA is important to verify the assumption of equal variances. In this case, fig. 1 shows that this assumption was true, due to their uniform distribution along the zero line. Based on the information given by the ANOVA analysis, the optimization technique but not the optimization phase has a significant effect on power consumption.

However, we can only conclude from these values that from the methods studied under this work, there is not statistical evidence that the method itself produces significant change in power consumption. Further work has to be done in order to understand whether different methods or application of these methods to particular algorithms will have a significant impact on the power consumption. Fig. 2 shows the analysis of variance of the experiment.

TABLE I. Two-Factor Factorial Experiment

Optimization Technique	Optimization Phase	
	P(mW) Non-Optimized	*P(mW) Optimized*
Statement Reordeting	89.37	91.71
Unswitching	90.81	91.26
Loop Peeling	90.63	90.73
Scalar Expansion	93.96	94.23
Loop Fusion	94.05	94.32
Loop Alignment	91.98	90.71
Loop Fission	91.80	94.86
Nested Loops	90.90	90.99
Loop Reversal	93.87	94.95
Loop Interchanging	90.71	90.81
Loop Unrolling	89.28	87.39

VI. Conclusions

The software that runs on a microprocessor is of vital importance in the behavior of an embedded system. One important feature to consider when designing this type of applications is the power consumption, because of constraints of battery lifetime, portability, heat dissipation, size, and costs, among other restrictions.

Design of experiments techniques were the statistical method used to demonstrate the effect that software has on the power consumption of a microprocessor-based system and at this moment we cannot conclude that any of the studied methods have a significant impact on power consumption.

In future work, certain loop transformation methods, such as loop unrolling and loop alignment which showed improvements in terms of power savings, can be used to optimize a set of benchmarks, in order to analyze the effect and relationship between loop optimizations and power consumption applied in a common scenario. Also, another architecture or platform may be used in the experiment with the intention to investigate the effect of different microprocessors on the power consumption when running the same benchmark.

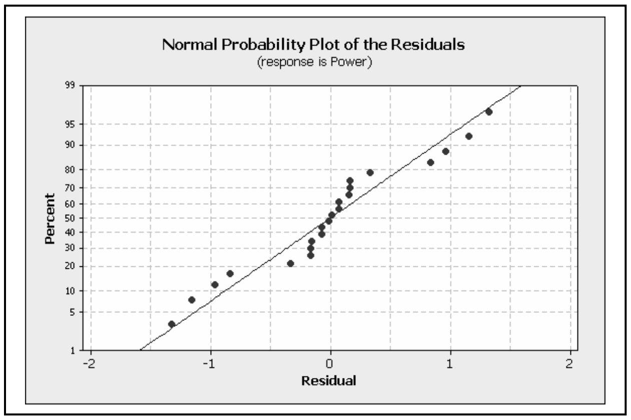
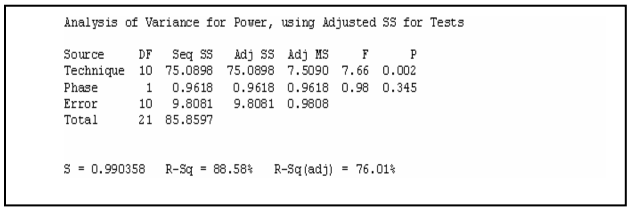


Figure 1. Residual Plot for Power Consumption



```
Analysis of Variance for Power, using Adjusted SS for Tests

Source     DF   Seq SS   Adj SS  Adj MS     F      P
Technique  10  75.0898  75.0898  7.5090  7.66  0.002
Phase       1   0.9618   0.9618  0.9618  0.98  0.345
Error      10   9.8081   9.8081  0.9808
Total      21  85.8597


S = 0.990358   R-Sq = 88.58%   R-Sq(adj) = 76.01%
```

Figure 2. Analysis of Variance for Power Cosumption

ACKNOWLEDGMENT

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, 18:1702 – 1714, Dec. 1999.

[4] V. Venkatachalam and M. Franz. “Power reduction techniques for microprocessor systems,” ACM Computing Surveys (CSUR), 37:195 – 237, Sept. 2005.

[5] F. Gruian. “Low power directed system design,” International Symposium on Low Power Electronics and Design, pages 9 – 14, 2000.

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

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

[12] R. Leupers. “Code generation for embedded processors,” The 13th International Symposium on System Synthesis, pages 173 – 178, Sept. 2000.

[13] T. Simunic, G. de Micheli, L. Benini, and M. Hans. “Source code optimization and profiling of energy consumption in embedded systems,” International Symposium on System Synthesis, pages 193 – 199, Sept. 2000.

[14] 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.

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

[16] M. Wolfe. “High performance compilers for parallel computing,” Adison-Wesley Publishing Company, 1996.