A Survey of Software Optimization Techniques for Low-Power Consumption

Oscar Acevedo
Advisor: Dr. Manuel Jimenez

Electrical and Computer Engineering Department
University of Puerto Rico, Mayagüez Campus
Mayagüez, Puerto Rico 00680
oscara@ece.uprm.edu

Abstract

Power consumption in embedded applications can be reduced either by hardware, software or both. Each instruction of a given program activates specific parts of the microprocessor. Therefore, the election of the correct instruction can generate a reduction of the power consumed by the processor. This paper presents a survey of software power reduction techniques from several authors and the future work that will be realized for the Power PC 603E microprocessor.

1. Introduction

Low power is a key feature for portable electronics systems. In the last years, the research has been mainly focused on the hardware component. Now, the software component is receiving more attention to obtain the lower power system [Tiwari94] [Mehta96].

The generation of a software optimizer requires the construction of a profile of power consumption of each microprocessor instruction and the adequate selection of compiling techniques that can reduce the energy consumed by a program. Because the architecture of each microprocessor is different, the modeling and instruction profiling are an individual processes. Thus, the results from one processor might not be applicable to any other. This makes the election of a microprocessor for targeting low power software an important step in this work.

This paper presents a survey of techniques reported on this subject. Section 2 presents methods used to determine the power consumption of the microprocessor instruction set. Section 3 presents compiler techniques, which have been fully applied to reduce the power consumption in a program execution. Section 4 presents the requirement for the software optimizer and section 5 presents the future work on the Power PC 603E microprocessor.

2. Instruction Profiling

Profiling is the first step to obtain a tool that can measure the power dissipated by a given program. The generation of the instruction profile requires the use of a system that can measure the power consumption of each instruction of the microprocessor, and capable of determining the power cost of special cases that can occur in the execution of a program.

Two main methods have been reported to achieve these objectives. The first method, proposed by Tiwari et al., measures the current drawn by the processor as it repeatedly executes certain instructions [Tiwari94]. It is possible to obtain the information needed to evaluate the power cost of a program for that processor.

The sum of the power cost of each executed instruction is enhanced by the power cost of additional effects and the power cost of the program can be estimated [Tiwari95][Russel98].

The current drawn by the processor can be measured using an oscilloscope with a shunt resistor connected in series with the supply voltage pin of the microprocessor [Russel98].

Other way is a direct measure with an ammeter. It is important that the used ammeter can measure high frequency signal in order to obtain a stable measure of the current drawn by the microprocessor [Tiwari95].

The hypothesis presented by Tiwari et al. is empirically validated in [Tiwari94] using two commercial microprocessors and making a measurement of the current drawn by them. The
average power is then calculated as \( P = V_{cc} \cdot I \), where \( V_{cc} \) is the supply voltage and \( I \) is the current drawn by the microprocessor.

The energy also can be calculated as \( E = P \cdot N \cdot T \), where \( N \) is the number of clock cycles taken by the program and \( T \) is the clock period.

This method is applicable to pipelined processors, but there are effects whose energy contribution are not accounted in the power calculation. As an example, the energy cost of an instruction may be different from the calculated, depending of the last state of the microprocessor. The solution to this problem can be achieved by calculating the power cost of a pair of instructions, which gives a better estimate of the power cost of a given program [Tiwari95].

Another effect is the cache miss ratio. It requires additional execution time or extra cycles that lead to an energy penalty [Tiwari95].

The second method is based on a simulation of the microprocessor and the effect of the instruction set in the microprocessor model [Mehta97][Mehta96][Dalal01]. Mehta et al. suggest that lower level simulation can provide an estimate of the current drawn to calculate the power consumption of each instruction.

Chakrabarti et al. built a model of each basic module of the microcontroller HC11. The model was done by hardware description languages [Chakrabarti99]. Nevertheless, black box models [Dalal01] or other kind of model where is possible to make a current or power measurement can be used.

Once the modules activated by each instruction have been determined, the power consumption can be calculated with the sum all the energy of the active modules in a given instruction [Mehta96].

A drawback of this method is that detailed information of the CPU must be available to do the simulation. The power estimated by this technique is within 1% to 10% of the real value [Chakrabarti99].

3. Compilation Techniques

The next step is the application of some sort of technique to achieve the power reduction of the program. Knowledge of the architecture is important in this step, because not every technique can be used in every processor. Moreover, new techniques can be derived from those studied and from the special features of the microprocessor under test.

The first technique is based on Instruction Reordering to Reduce Switching [Pedram01][Tiwari94]. The energy consumed during the execution of an instruction will vary depending on what the previous instruction was. Thus, an appropriate reordering of instructions in a program can result in a lower energy. The application of this technique in the Intel 486 processor reveals that a reduction in switching activity can be achieved by this technique does not translate into very significant overall energy reduction. The results showed a variation of only 2% in the energy cost. Nevertheless, in certain DSP processor, this technique can achieve a reduction in the power consumption between 30% to 65% [Wiratunga00][Sarta99].

Another technique using Code Generation through Pattern Matching has been proposed by Tiwari et al. [Tiwari94]. This technique modifies the cost function in the compiler (normally the number of execution cycles), to obtain a code generator that targets energy reduction. The results show that the resulting code was similar to the code generated when targeting cycles. The reason for this is that the energy cost of an instruction pattern is obtained as the average power times the number of clock cycles.

A loop unrolling technique was successfully applied to DSP processors by Vishal et al. [Wen-Tsong01]. In this approach the main sources of power consumption were the arithmetic/logic circuits and the memory circuits [Wen-Tsong01]. The objective was to reduce the total number of comparisons in a given program. The result reported a low use of the ALU (20% performance) with the drawback of increasing code size (>10%) [Wen-Tsong01].

A Memory Operand Reduction technique was proposed by Tiwari et al. [Tiwari94]. This approach was based on the hypothesis that instructions with memory operands have very high-energy cost compared to instructions with register operands. Thus, reduction in the number of memory operands can lead to large energy savings. The best way to achieve power reduction is an efficient register management. This entails optimal register allocation
of temporaries and global register allocation for the most frequently used variables.

Some programs have been used to prove these techniques [Tiwari94]. Hand tuning of the code for shorter running time leads to a 13.5% reduction in energy. So far only temporary variables have been allocated to registers and the appropriate memory operands are replaced by register operands. Even though redundant instructions are not removed, there is a 5% reduction in current and a 7% reduction in running time. Finally, more variables are allocated to registers and all redundant instructions are removed. Compared to the original program it had a 40.6% lower energy consumption.

4. Software Optimizer

Once realized the previous steps, it is necessary to create a tool that allows the direct application of the compilation techniques in a real program.

The challenge in this step is the development of a unique tool that can realize the software optimization for any program and microprocessor. Unfortunately, the work done is not enough to generate a general formula for each type of microprocessor; nevertheless, the actual information is a start way to achieve this objective. The use of the software optimizer requires a continuous feedback with the results obtained, to detect all possible sources of error in both the calculation of the power consumption and the application of the compilation technique.

5. Future Work

The information presented in this paper is the first step in the development of a final product that can be used in real systems.

The election and study of a specific microprocessor will be the next step. In this regard, has been preliminary chosen the Power PC microprocessor 603E from Motorola.

The next step is the creation of the tools required to generate the profile of the instruction set of this microprocessor. The methods exposed in this paper will be considered, but other techniques might arise after a close study of the processor architecture.

The application of compilation techniques will be the next step. Here again, previously reported approaches will be considered, but the specific technique will depend on the study of the processor architecture and the compiler structures it might embed. The result of this step is the software optimizer.

6. Conclusion

The reduction in the power consumption is an important issue in modern systems; therefore, any method that provides a way to reduce this consumption must be studied, evaluated and applied to the devices in development. The techniques presented in this paper promise an interesting way to achieve this issue.

References


