International Journal of Res. In Engg. & Management Vol. 1 No.1. 2016. Pp. 1-3

Full Length Research Paper

Power Analysis of Embedded Software

Rutesh S. Lonkar, Tushar More and V.R.Barwat
Priyadarshini Indira Gandhi College of Engineering, Nagpur, India.

Article history
Received: 12-04-2016
Revised: 06-06-2016
Accepted: 10-06-2016

Corresponding Author:
Rutesh S. Lonkar
Priyadarshini Indira Gandhi College of Engineering, Nagpur, India.

Abstract
Embedded computer systems are characterized by the presence of a dedicated processor and the software that runs on it. Power constraints are increasingly becoming the critical component of the design specification of these systems. At present, however, power analysis tools can only be applied at the lower levels of the design-the circuit or gate level. It is either impractical or impossible to use the lower level tools to estimate the power cost of the software component of the system. This paper describes the first systematic attempt to model this power cost. This technique can be employed to evaluate the power cost of embedded software. This can help in verifying if a design meets its specified power constraints. Further, it can also be used to search the design space in software power optimization. Examples with power reduction of up to 40%, obtained by rewriting code using the information provided by the instruction level power model, illustrate the potential of this idea.

Key words: *Switching Activity, Array Multiplier, and Column Bypass Multiplier.*

Introduction

Embedded computer systems are characterized by the presence of a dedicated processor which executes application specific software. Recent years have seen a large growth of such systems. This growth is driven by several factors. The first is an increase in the number of applications as illustrated by the numerous examples of "smart electronics" around us. A notable example is automobile electronics where embedded processors control each aspect of the efficiency, comfort and safety of the new generation of cars. The second factor leading to their growth is the increasing migration from application specific logic to application specific code running on existing processors. This in turn is driven by two distinct forces. The first is the increasing cost of setting up and maintaining a fabrication line. At over a billion dollars for a new line, the only components that make this affordable are high volume parts such as processors, memories and possibly FPGA's. Application specific logic is getting increasingly expensive to manufacture and is the solution only when speed constraints rule out programmable alternatives. The second force comes from the application houses, which are facing increased pressures to reduce the time to market as well as to have predictable schedules. Both of these can be better met with software programmable solutions made possible by embedded systems. Thus, we are seeing a movement from the logic gate being the basic unit of computation on silicon, to an instruction running on an embedded processor.

A large number of embedded computing applications are power critical, i.e., power constraints form an important part of the design specification. This has led to a significant research effort in power estimation and low power design. However, there is very little available in the form of design tools to help embedded system designers evaluate their designs in terms of the power metric. At present, power measurement tools are available for only the lower levels of the design-at the circuit level and to a limited extent at the logic level. At the least these are very slow and impractical to use to evaluate the power consumption of embedded software, and often cannot even be applied due to lack of availability of circuit and gate level information of the embedded processors.

This paper describes a power analysis technique for embedded software. The goal of this research is to present a methodology for developing and validating an instruction level power model for any given processor.

Experimental method

The power consumption in microprocessors has been a subject of intense study lately. Attempts to model the power consumption in processors often adopt a "bottom-up'' approach. Using detailed physical layouts and sophisticated power analysis tools, isolated power models are built for each of the internal modules of the processor. The total power consumption of the processor is then estimated using these individual models. No systematic attempt, however, has been made to relate the power consumption of the processor to the software that executes on it. Thus, while it is generally recognized that the power consumption of a processor vanes from program to program, there is a complete lack of models and tools to analyze this variation. This is also the reason why the potential for power reduction through modification of software is so far unknown and unexploited. The goal of this work is to overcome these deficiencies by developing a methodology that would provide a means for analyzing the power consumption of a processor as it executes a given program. We want to provide a method that makes it

possible to talk about the"power/energy cost of a given program on a given processor." This would make it possible to very accurately evaluate the power cost of the programmable part of an embedded system.

We propose the following hypothesis that forms the basis for meeting the above goal: By measuring the current drawn by the processor as it repeatedly executes certain instructions or certain short instruction sequences, it is possible to obtain most of the information that is needed to evaluate the power cost of a program for that processor.

In the case of embedded system design, detailed layout information of the CPU is often not available to the designer of the system. Even if it is available, the simulation based tools and techniques are expensive and difficult to apply. A methodology based on laboratory measurements, like the one described below, is inexpensive and practical, and often may be the only option available. Given a setup to measure the current being drawn by the microprocessor, the only other information required can be obtained from the widely available manuals and handbooks specific to that microprocessor.

A. Power and energy

The average power consumed by a microprocessor while running a certain program is given by: $P = I \times V_{CC}$, where P is the average power, I is the average current and V_{CC} is the supply voltage. Since power is the rate at which energy is consumed, the energy consumed by a program is given by: $E = P \times T$, where T is the execution time of the program. This in turn is given by: $T = N \times \tau$, where N is the number of clock cycles taken by the program and τ is the clock period.

Mobile systems run on the limited energy available in a battery. Therefore the energy consumed by the system or by the software running on it determines the length of the battery life. Energy consumption is thus the focus of attention.

B. Current measurement

For this study, the processor used was a 40 MHz Intel 486DX2-S Series CPU. The CPU was part of a mobile personal computer evaluation board with 4 MB of DRAM memory. The reason for the choice of this processor was that its board setup allowed the measurement of the CPU and DRAM subsystem current in isolation from the rest of the system.

The current was measured through a standard off the shelf, dual-slope integrating digital ammeter. Execution time of programs was measured through detection of specific bus states using a logic analyzer. If a program completes execution in a short time, a current reading cannot be obtained visually. To overcome this, the programs being considered were put in infinite loops and current readings were taken. The current consumption in the CPU will vary in time depending on what instructions are being executed. But since the chosen ammeter averages current over a window of time (100 ms), if the execution time of the program is much less than the width of this window, a stable reading will be obtained. Execution time of programs was measured through detection of specific bus states using a logic analyzer. If a program completes execution in a short time, a current reading cannot be obtained visually. To overcome this, the programs being considered were put in infinite loops and current readings were taken. The current consumption in the CPU will vary in time depending on what instructions are being executed. But since the chosen ammeter averages current over a window of time (100 ms), if the execution time of the program is much less than the width of this window, a stable reading will be obtained.

C. Instruction level modeling

An instruction level energy model has been developed and validated for the 486DX2. Under this model each instruction in the instruction set is assigned a fixed energy cost called the base energy cost.

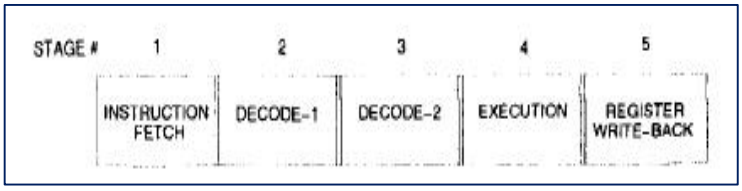

Fig 1. Internal pipelining in the 486DX2

Inter-Instruction Effects

When sequences of instructions are considered, certain inter instruction effects come into play, which are not reflected in the cost computed solely from base costs. These effects are discussed below.

Effect of Circuit State

The switching activity in a circuit is a function of the present inputs and the previous state of the circuit. Thus, it can be expected that the actual energy cost of executing an instruction in a program may be different from the instruction's base cost. This is because the previous instruction in the given program and in the program used for base cost determination may be different. For example, consider a loop of the following pair of instructions:

XOR BX, 1
ADD AX, DX

The base costs of the XOR and ADD instructions are 319.2 mA and 313.6 mA. The expected base cost of the pair, using the individual base costs would be their average, i.e., 316.4 mA, while the actual cost is 323.2 mA. It is greater by 6.8 mA.

Number	Instruction	Current(mA)	Cycles
1	MOV CX,1	309.6	1
2	ADD AX,BX	313.6	1
3	ADD DX,8[BX]	400.2	2
4	SAL AX,1	308.3	3
5	SAL BX,CL	306.5	3

Fig 2. An example of instruction sequence

The reason is that the base costs are determined while executing the same instruction again and again. Thus each instruction executes in what we expect is a context of least change. At least, that is what the observations consistently seem to indicate. When a pair of two different instructions is considered, the context is one of greater change. The cost of a pair of instructions is always greater than the base cost of the pair and the difference is termed as the circuit state overhead. As another example, consider the sequence of instructions shown in Fig 2. The current cost and the number of cycles of each instruction is listed alongside. The measured cost for this sequence is 332.8 mA (avg. current over 10 cycles). Using base costs we get:

(309.6 + 313.6 + 400.2 x 2 + 308.3 x 3 + 306.5 x 3)/10 = 326.8 …….(1)

The circuit state overhead is thus 6.0 mA. It is possible to get a closer estimate if we consider the circuit state overhead between each pair of consecutive instructions.

Conclusion

It is found that, for all input combinations, the Simple Array Multiplier requires the same amount of power i.e. 38.77 Mw. It is found that the power consumption of Column Bypass multiplier (for 0011 x 0010) is 37mW which is approximately 2mW less than the Simple Array Multiplier for same input combination. It is also observed that the power consumption of Column Bypass multiplier depends on number of zeros in the input multiplicand as well as position of zero in the multiplicand. It is also found that the power consumption of Column Bypass multiplier [for (A=0000) x B)] is much less than Simple Array Multiplier i.e. 16mW , but not equal to zero. Total memory usage of Simple Array Multiplier is 139580 kilobytes, whereas total memory usage of Column Bypass multiplier is 138440 kilobytes

Future scope

As an attempt to develop arithmetic algorithm and architecture level optimization techniques for low-power multiplier design, the research presented in this dissertation has achieved good results and demonstrated the efficiency of high level optimization techniques. However, there are limitations in our work and several future research directions are possible. Design can be modified for 8-bit or n- bit operation depending on the requirements. Proposed design is for unsigned multiplication. It can be enhanced for signed operation also. Designed multiplier is for fixed point binary number. It can be modified for floating point binary multiplications

References

[1] M. C. Wen, S. J. Wang and Y. M. Lin, “Low power parallel multiplier with column bypassing,“ IEEE International Symposium on Circuits and Systems, pp.1638-1641, 2005.)

[2] A. P. Chandrakasan and R.W. Brodersen, “Minimizing power consumption in digital CMOS circuits,” Proc. IEEE, vol. 83, no. 4, pp. 498–523,Apr. 1995.

[3] Oscal T.-C. Chen, Sandy Wang, and Yi-Wen Wu, ”Minimization of Switching Activities of Partial Products for Designing Low-Power Multipliers” , Ieee Transactions On Very Large Scale Integration (Vlsi) Systems, Vol. 11, No. 3, June 2003

[4] J. Ohban, V. G. Moshnyaga, and K. Inoue, “Multiplier energy reduction through bypassing of partial products,” IEEE Asia-Pacific Conference on Circuits and Systems, pp.13–17, 2002.

[5] Ying-Tsung Hwang, Jin-Fa Lin, Ming-Hwa Sheu,Chia-Jen Sheu,” Low Power Multiplier Designs Based on Improved Column Bypassing Schemes” , IEEE APCCAS 2006.

[6] Jin-Tai Yan , Zhi-Wei Chen ,” Low-Power Multiplier Design with Row and Column Bypassing “ , IEEE International Conference On SOC, SOCC 2009, Pages 27 - 230

[7] Mangal, S.K.; Deshmukh, R.B.; Badghare, R.M.; Patrikar, R.M. ,” FPGA Implementation of Low Power Parallel Multiplier “ , 20th International Conference On VLSI Design,2007,Pages-115-120.