INSTRUCTION LEVEL POWER MODEL OF MICROCONTROLLERS

Chaitali Chakrabarti

Department of Electrical Engineering
Arizona State University
Tempe, AZ 85287-5706
chaitali@asu.edu

Dinesh Gaitonde

Monterey Design Systems
894 Ross Drive, Suite 200
Sunnyvale, CA 94089-1443
gaitonde@mondes.com

ABSTRACT

In the design of low power systems, it is important to analyze and optimize both the hardware and the software component of the system. To evaluate the software component of the system, a good instruction-level energy model is essential. In this paper we present a methodology for instruction level modelling of microcontrollers using gate level power estimation tools. We use the microcontroller, M68HC11, to illustrate this method. We study two different implementations of the microcontroller and show that the energy consumption of each instruction is quite different. Our study reveals that data correlation does not significantly affect the energy consumption of most instructions. Finally, we show the correctness of this model by running some sample programs and showing that the predicted energy estimates are quite close to the actual estimates.

1. INTRODUCTION

In order to design a system for low power and/or embedded computing applications, it is important to analyze and optimize power in all the components of the system. An ever increasing portion of the functionality of today's system is in the form of software. Thus along with the power cost of the hardware component, it is important to estimate the power cost of the software component. In order to systematically analyze the power cost of the software component, the power cost of the individual instructions have to be estimated. Clearly, a good instruction-level energy model is essential to evaluate software in terms of the power metric and also help search the design space for low power software implementations.

There are many advantages in developing an instruction-level power model [1] -[3]. First, it provides a way of assigning a power cost to the software component of a system and helps in verifying if the overall system meets the specified power budget. Secondly, it can be directly used by compilers and code generators to generate code targetted towards low power. Thirdly, it can guide higher level design decisions such as hardware-software partitioning. Finally, it helps in providing a meaningful comparison of the power consumption of different processors (vs a single average power consumption figure).

The instruction level power analysis technique was first developed at Princeton University [1]- [3]. The technique is based on measuring the current drawn by the processor as it repeatedly executes certain instructions. Power models for the Intel 486DX2, the Fujitsu SPARClite 934 and the Fujitsu DSP processor have been developed using this method. The current measurement method was also used in [4] to develop a power predictor model for the TI TMS320C5x processor. The method is very accurate and uses linear regression with instruction and architectural level variables (such as the average bit switching activity in the instruction register, address busses, etc.) as predictors. A variation of the current measurement method that used a digitizing oscilloscope to measure instantaneous power was used in [5] to develop a power model for the JF and HD implementations of the i960 family. The study in [5] showed that the variation in power consumption across assembly instructions is of no statistical significance. This is in disagreement with the results in [1]-[4] which show that while similar instructions consume similar power, the variation across all instructions is significant. Our conclusion is that the result in [5] is specific to the Intel i960 family and cannot be generalized. Complex processors, in general, tend to have less variation in instruction costs compared to smaller DSP processors because of the dominance of the overhead costs as pointed out in [3].

The other related work in high level power analysis techniques assign power costs to architectural modules such as datapath units, control units and memory units [6], [7]. The power cost is the estimated average capacitance that would switch when the given module is activated. Since the activity factors are obtained from functional simulation over

This work was supported by the Center for Low Power Electronics and by UDSL, Motorola Inc.

typical input streams, such a technique takes a long time to evaluate the power consumption of the software component. In contrast, once the instruction level model is developed, evaluating the power consumption of even large programs is very time-efficient.

In this paper we present an instruction level power analysis technique based on gate level power estimation. While this method requires access to the gate level (or at least the RT level) description, it allows us to estimate the energy early on in the design process (vs after the processor has been shipped out), thereby making it possible to study the design space trade-offs for low power implementation. We use a popular Motorola microcontroller, M68HC11, to illustrate our method. We study two different implementations of the microcontroller. We find that the energy consumption of each instruction is quite different for the two implementations. A study of these differences can be used to resynthesize parts of the design for low power applications. We also study the effect of data correlation on the energy consumption of the instructions. Our study shows that data correlation does not significantly affect the energy consumption of most instructions. Finally we use the energy estimates of the instructions to predict the energy consumption of a few sample programs. The predicted values are quite close to the actual values.

2. PROPOSED MODEL

In this paper we present a methodology for instruction level modeling of microcontrollers using gate level power estimation. We use a popular Motorola microcontroller - M68HC11 [8] to investigate the feasibility of using a gate level power estimation tool to characterize the instruction set of this mircrontroller for power. The HC11 microcontroller is used only as an example. The techniques used in this paper can be extended to any microcontroller.

Given the behavioral description of the microcontroller design, high level synthesis tools can be used to transform it into an RT level implementation. The RT level implementation can then be synthesized using a commercial synthesis engine such as Synopsys. A gate level power estimation tool can then be used to estimate the energy consumption of each instruction. In our setup, we used the high level synthesis tool Matisse [1], the synthesis tool Synopsys and the gate level estimator ASPEN [2].

In our model, the base cost of an instruction was modelled in the following way. The base cost of simple instructions, such as the load instruction, was modelled by simply executing such instructions 1000 times and computing the average. The data values used in each instance of the instruction were either random or correlated (where the degree of correlation could be specified). The entire program resided in an on-chip memory (and thus the base cost did not model the cost of an external memory access). Most instructions could not be modelled by themselves and had to be modelled in conjunction with some other instruction such as the load instruction. Consider the case when instruction X could be modelled by itself and instruction Y had to be modelled with instruction X. In both cases, ASPEN was used to generate the average power for 1000 runs. Let P_X be the average power for instruction X and P_{X+Y} be the average power for instruction X followed by Y. Since we know the exact time that it takes to execute a single instruction X, t_X, and the time that it takes to execute instruction X followed by Y, t_{X+Y}, the energy of instruction Y, E_Y, can be calculated in a straight-forward way. The assumption while computing the base cost is that there is no overhead in executing instruction Y after instruction X. Thus $E_Y = t_{X+Y} * P_{X+Y} - t_X * P_X$. The inter-instruction effects have not been modelled explicitly. Experimental results were used to derive an average cost for inter-instruction effects.

[1] Matisse is a high level synthesis tool developed at Motorola.

[2] ASPEN is a gate level simulator developed at Motorola.

3. RESULTS

In this section we describe the results of implementing the experimental setup on two different implementations of the HC11 microcontroller. Since the synthesizable HC11 was generated from a behavioral description of the design, one could generate several candidate synthesizable implementations of the HC11. Both these implementations were optimized for area – one more than the other.

We have categorized the instructions into the following classes: (i) Loads, Stores and Transfers, (ii) Arithmetic Operations, (iii) Multiply and Divide, (iv) Logical Operations (v) Shifts and Rotates, (vi) Stack and Index Register Operations, (vii) Condition Code Register Instructions and (viii) Branches. The details of function of each instruction can be obtained from [8]. We studied the effects of data correlation on the energy consumption values. Two sets of correlated data were generated. Mildly correlated data corresponds to cor(10), while medium correlated data corresponds to cor(50). The results have been tabulated in Table I. We found that the average energy consumption values do not change much when the correlation of the data is increased. This implies that there is not much data dependency on the power consumed by each instruction. Consequently, it is sufficient to work with power consumption values of random data during the implementation evaluation phase. For the branch instructions, the instruction was modelled differently, depending on whether the branch was successful or not. Table 2 gives the energy consumption of a few branch instructions for Implementation 1. The differences in the

Instruction	Normalized energy	
	Successful	Unsuccessful
BNE	2.629	1.286
BEQ	2.696	2.024
BGE	2.773	1.861
BGT	2.924	1.695
JMP	3.075	

TABLE II: Energy consumption values of a set of branch instructions for Implementation 1. The energy values are normalized with respect to the energy consumed by a LDAA instruction

energy of the branch instructions can be utilized to generate code with a low energy cost.

Next we illustrate the difference in the power consumption values for the two architectures. Even though the software (core) of the two implementations is identical, the hardware realizations are significantly different. Consequently, the energy consumption values are also different. An analysis of the differences in the energy consumption values should aid in future implementations of HC11 that would be targetted for low power. Table I lists the the % difference in the energy values for random data of the two implementations. In this sample set of instructions, Implementation 2 has a lower power consumption for the following instructions: LDAA, INCA, DECA, ORAA, ASLA, ASLD, INX.

3.1. Sample Program

The accuracy of the instruction energy estimates was checked by running a few sample programs. In each case, we compared the actual energy consumption values (calculated by ASPEN) with those predicted using the estimates derived using the power model developed in Section 2. The predicted values were always within 12% of the actual values.

Program 1: Computing the MAXIMUM of 5 numbers.
The data is loaded in locations 0000-0004. For input data 73, 36, 24, 82, 49 (where 73 is loaded in location 0000, 36 in 0001, etc), the average energy over 100 runs using ASPEN is 4.066 nJ. The estimated energy using the instruction level power model is 4.017 nJ. The estimated energy is 1.2 % lower than the actual energy.

Program 2: Computing the running sum $y_i = \sum_{j=0}^{2} x_{i+j}$, $0 \leq i \leq 4$.
Random data is loaded in locations 0000-0006. The average energy over 100 runs using ASPEN is 20.0056 nJ. The estimated energy using the proposed instruction level power model is 20.0958 nJ. The estimated value is 0.45% higher than the actual value.

Program 3: Computing the running sum $y_i = \sum_{j=0}^{2} x_{i+j}$, $0 \leq i \leq 4$ – unrolling.
In this example, we simply unrolled Program 2 to generate this program. The average energy over 100 runs using ASPEN is 9.022 nJ. The estimated energy using the proposed model is 8.988 nJ. Thus the estimated value is 3.77% lower than the actual value. An interesting point to note is that the unrolled program (unrolling factor 5) consumes only 45% of the energy consumed by the program with the loop (Program 2). This is because of the large overhead in manipulating loops. Instructions with index register X such as LDAA(ind,X), STAA(ind,X) etc do extra addition operations, thereby increasing the power consumption. Thus for low power applications, loop unrolling should be done as much as possible.

Program 4: Sorting 5 numbers using bubble sort.
The numbers are loaded in locations 0000 through 0004. For input data 19, 23, 35, 57 and 89, where 19 is stored in location 0000, 23 in location 0001, etc., the average energy using ASPEN over 50 runs is 19.757 nJ. The estimated energy using the power model is 22.07 nJ. So, for this data set, the estimated energy is 11.71 % higher than the actual energy. For a different set of input data, the energy values were closer to the actual estimates. If the experiment had been run on a large number of input data sets, we anticipate that the average energy would have been closer to the estimated energy.

4. CONCLUSIONS

In this paper we have described a method based on gate level power estimation to develop an instruction level energy model for microcontrollers. We applied our technique to develop an instruction level energy model for the M68HC11 microcontroller. This model was used to successfully predict the energy consumption of a few sample programs. Our study also showed that (i) data correlation does not affect the energy consumption of most instructions and (ii) the same instruction incurs a different cost for different gate-level implementations (even though the software core is the same). The accuracy of our model can be significantly increased by taking into account the effect of average switching in the data address bus, the program address bus and the instruction register as in [4]. Our next step is to extend this method to develop instruction level models for more complex processors where the effect of pipeline stalls, size of register files, cache misses, I/O accesses etc. would be significant.

ACKNOWLEDGEMENTS

The authors would like to thank Kayhan Kucukcakar for help with the HC11 implementations and many interesting discussions.

Instruction	Energy as a multiple of energy consumed by LDAA instruction						% diff of Impl2 wrt Impl1 for random data
	Implementation1			Implementation2			
	Random	Cor(10)	Cor(50)	Random	Cor(10)	Cor(50)	
LDAA	1	0.994	0. 988	0.998	0.989	0.985	-0.2%
LDAA(dir)	1.833	1.811	1.835	2.006	1.989	2.017	9.4%
STAA	2.864	2.867	2.853	2.867	2.890	2.874	0.1 %
TAB	1.727	1.736	1.733	1.818	1.803	1.801	5.3%
LDD	1.922	1.908	1.880	2.203	2.185	2.155	14.6%
XGDX	3.322	3.340	3.225	3.728	3.750	3.624	12.2%
INCA	2.536	2.554	2.556	2.304	2.335	2.328	-9.1%
DECA	2.631	2.665	2.657	2.273	2.285	2.284	-13.6%
ABA	2.735	2.653	2.667	2.917	2.867	2.849	6.6%
SBA	2.839	2.800	2.839	2.902	2.857	2.927	2.2%
ADDD	4.404	4.365	4.249	5.059	5.089	4.922	14.9%
SUBD	4.401	4.340	4.338	4.436	4.396	4.380	0.8%
CPD	5.832	5.839	5.750	6.057	6.081	6.006	3.8%
MUL	11.054	10.566	10.774	11.849	11.433	11.591	7.2%
ANDA	1.268	1.281	1.279	1.447	1.472	1.473	14.1%
ORAA	1.710	1.666	1.658	1.426	1.422	1.391	-16.6%
ASLA	2.577	2.563	2.549	2.439	2.441	2.427	-5.3%
LSRA	1.988	1.982	1.977	2.079	2.082	2.076	4.6%
ASLD	3.660	3.644	3.616	3.598	3.497	3.492	-1.7%
LDX	1.850	1.827	1.811	1.973	1.949	1.933	6.7%
STX	3.851	3.834	3.826	3.900	3.907	3.901	1.3%
TXS	2.644	2.864	2.867	2.922	2.921	2.924	10.5%
INX	3.030	3.046	3.042	2.802	2.807	2.801	-7.5%

TABLE I: Energy consumption values of a subset of M68HC11 instructions for two different implementations using data with different degrees of correlation

5. REFERENCES

[1] V. Tiwari, S. Malik and A. Wolfe, "Power Analysis of Embedded Software: A First Step Towards Software Power Minimization," *IEEE Trans on VLSI Systems*, Dec 1994, pp. 437-445.

[2] M.T.-C. Lee, V. Tiwari, S. Malik and M. Fujita, "Power Analysis and Minimization Techniques for Embedded DSP Software," *IEEE Trans on VLSI Systems*, Mar 97, pp. 123-135.

[3] V. Tiwari, S. Malik, A. Wolfe and M.T.-C. Lee, "Instruction Level Power Analysis and Optimization of Software," *Journal of VLSI Signal Processing*, 1996, pp. 1-18.

[4] C.H.Gebotys and R.J.Gebotys, "An Empirical Comparison of Algorithmic, Instruction and Architectural Power Prediction Models for High Performance Embedded DSP Processors," *Proc of ISLPED 98*, pp 121-123.

[5] J. Russell and M.F. Jacome, "Software Power Estimation and Optimization for High Performance, 32-bit Embedded Processors," *Proc. of ICCD '98*.

[6] H. Mehta, R.M. Owens and M.J. Irwin, "Instruction Level Power Profiling," *Proc of ICASSP 96*, pp. 3326-3329.

[7] P. Landman and J. Rabaey, "Activity-sensitive Architectural Power Analysis," *IEEE Trans on Computer Aided Design*, June 1996.

[8] M68HC11 Reference Manual, Motorola.