An Empirical Comparison of Algorithmic, Instruction, and Architectural Power Prediction Models for High Peformance Embedded DSP Processors

Catherine H. Gebotys and Robert J. Gebotysψ
Department of Electrical and Computer Engineering,
University of Waterloo, Wilfrid Laurier Universityψ,Waterloo,Ont,Canada.
cgebotys@optimal.vlsi.uwaterloo.ca

Abstract

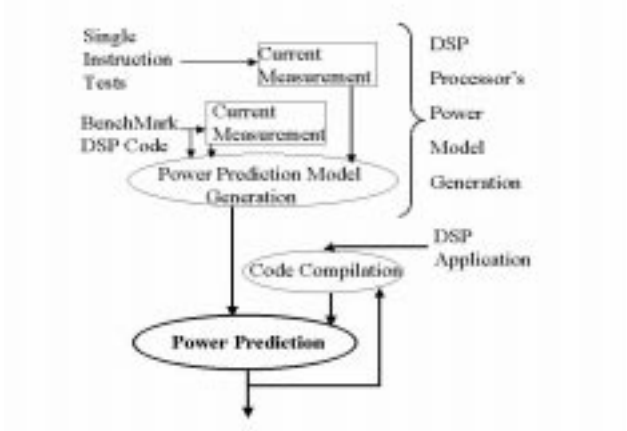
This paper presents a comparison of statistically-derived power prediction models at the algorithmic, instruction, and architectural levels for embedded high performance DSP processors. The approach is general enough to be applied to any embedded DSP processor. Results from 168 power measurements of DSP code show that power can be predicted at instruction and architecture levels with less than 2% error. This result is important for developing a general methodology for power characterization of embedded DSP software since low power is critical to complex DSP applications in many cost sensitive markets.

1 Introduction

As embedded applications rapidly grow more complex, some designers are using fully programmable DSP processors or cores for a lower risk, flexible solution. These systems demand small code size, low power, and high performance. Previous research has mainly concentrated on techniques to improve performance which in turn improve power. However for fixed throughput (or performance) required by many DSP applications, it is not clear how to optimize for power and very few suggested techniques have verified savings in power. Although architectural design and high level application transformations for low power are becoming a well understood area of research, little has been done for low-power software compilation. Previous methods of estimating power at a high level using gate-level or architecture-level simulations were either inaccurate or too time consuming. For embedded markets power measurement and optimization for code is very important, however gate level processor representations are not always available for this purpose. Physical current measurements are accurate and can be obtained extremely fast. Fast, accurate power prediction models for DSP software are needed. The energy dissipation of a processor running a program[1] is the product $P * T = I * Vdd * T = I * N * \tau * Vdd$, where P is the power dissipated, I is the average current, Vdd the supply voltage, and T is equal to $N * \tau$, where N is the number of clock cycles and τ is the clock period. This paper illustrates a general methodology for software power prediction using the TI TMS320C5x processor[3], current measurements on 168 DSP programs, and a statistical linear regression algorithm. The methodology however is general and can be applied to any processor.

The figure below shows the methodology for power prediction of embedded systems DSP design. The power prediction model is generated only once for a DSP processor, from single instruction tests and DSP benchmark code. The single instruction tests involve repeating one instruction several times in a loop. This code is run on the DSP processor hardware using pseudorandom type of input data and the current is measured (see Current Measurement box). The average current (I_i) for each type of instruction (i) is recorded. Several DSP benchmark programs are also run with two samples of voice data and two samples of pseudorandom data and current measurements are recorded. Algorithmic, instruction and architectural level variables are extracted, such as the average bit switching in the instruction register, address busses, etc (available from the code directly) and a special variable, x_p, where $x_p = \frac{(\sum I_i N_i)}{\sum N_i}$ where N_i = *number of instructions of type i in program* (similar to [1] except no overheads are used). The power predictor model generation is based upon a linear regression model with these variables as predictors (using SPSS[4]). The output is a model (or equation y =...etc) which predicts current. The embedded systems designer then generates code for an application and uses it with the power-prediction model to predict current. Code is regenerated (using rescheduling, optimized code gener-

ation[5,6], or rewriting the application) in an attempt to obtain code that meets the performance constraint and minimizes the predicted power dissipation.



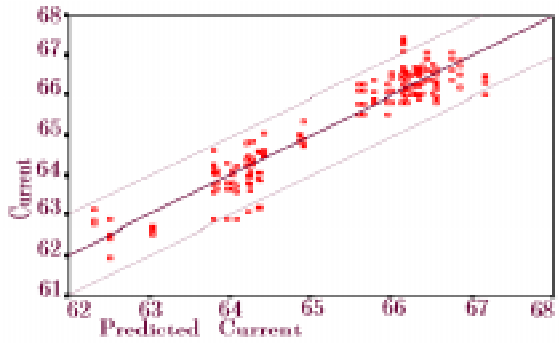
A variety of common DSP applications, such as the fast fourier transform, least means squares, high pass filter, discrete cosine transform, etc were implemented. The programs ranged from 60 to 150 instructions. Different schedules, optimized address generations[5] and optimized codings[6] were used to study power effects. For the DSP benchmark programs, the code sequence was a straight line basic block that was repeated several times and then placed within a loop. Each repeat of the program used a different part of the input data since it is well known that power depends upon not only the program but the signal statistics of the data[2]. For example if a DSP program used 40 words of speech data as input. The DSP program was repeated 100 times in a loop, performing computations on 4000 words of a continuous speech sample. This study was repeated with two different sets of pseudo-random generated data and two different samples of voice data. All experiments were repeatable. After the board was powered up, a warm up period was allowed before any experiments were run. A series of *noop* instructions were executed before and after each DSP program to calibrate any variation in power measurements due to temperature variation, etc. In all cases standard deviations were lower than 0.03mA (using Fluke867B GMM[8]).
Parallel instructions of the TMS320C5x had higher average current readings than the average of the two currents for the equivalent sequential instructions (for example, 14% higher for MACs, $I_{MAC} > \frac{I_{Multiply}+I_{acc}}{2}$). Also pairs of instructions had higher currents than the average of the currents for each single instruction (often 23% higher). The type of addressing also had up to 11 % variation of current for single instructions. An example of some variables obtained directly from analysis of the DSP code are shown in the table below. For example IR refers to the average switching of data stored in the instruction register (available from the DSP code), whereas $DABUS$ refers to the average switching of the data address bus. These variables were recorded for the statistical analysis.

Variable(x's)	Definition
IR	avg switching in instruction register
x_p	instruction-based current variable
$DABUS$	avg switching in data addr bus
$PABUS$	avg switching in program addr bus
MUL	avg # of multiplies
SUB	avg # of subtracts
$LOAD$	avg # of loads
M	avg # of memory accesses

2 Experimental Results

Some interesting observations occured from measuring current on these 168 DSP programs. For example the I did not always increase when more compact code was used, causing the percent improvements in energy to exceed the percent improvements in performance in some cases. Secondly unlike previous research[1], it was observed that the fastest code did not always produce the most energy efficient solution. For example in a FFT program, a minimum energy solution of $I * N = 62.55 * 123 = 7694$ did not utilize the fastest code (whose energy, $I * N = 65.15 * 122 = 7949$, was 3% higher). It was also observed that as much as 4% difference in energy can occur (due to scheduling and code generation) for equivalent type of input data and equivalent performance (N).
Results of the power prediction methodology are shown in the table below, where accuracy is reported for statistically derived linear models based on 168 DSP benchmark programs. For each $Algo$, $Instr$ and $Arch$ model, only algorithmic variables (derived only from C program alone), instruction variables (derived only from assembly code alone, see figure), and architecture variables (switching of internal DSP architecture obtained from custom instruction-level simulator) respectively were input to the linear regression model. The x variables automatically chosen due to their significance by the statistical procedure are listed in order of their importance in predicting current in the table (for example $Instr_1$: $LOAD, PABUS, SUB, IR, DABUS$). The value of R^2 for the models are given (for example $R^2 = 0.89$ means that 89% of the variation in current, y, is accounted for by that model, x's). Additionally the maximum error in current prediction is given (mA) along with the maximum percent error (% , calculated from the maximum error in current divided by the predicted value of current for that case). The equation for model $Instr_1$ is $\hat{y} = 35.99 + (29.74)LOAD + (5.88)PABUS + (14.27)SUB + (1.23)IR + (1.43)DABUS$. The instruction and archi-

tectural level models ($Instr$, $Arch$) are statistically excellent and significant. A good model at the algorithmic level could not be found. The overall fit of model $Instr_1$ can be seen below in the graph of predicted current, $\hat{y}$, versus the actual measured current, I, both in mA's. The middle line graphs $\hat{y} = y$, a perfect prediction line, with lines on each side 1mA apart. The last row in the table, $Arch$, is an architectural level model which uses variables measuring the average switching in internal registers a, t and the average switching in data memory ($dataMemory$). Although this detail is typically not available for embedded system designers, it is presented to illustrate how well the $Instr_1$ model compares in R^2 value. To further independently test out the validity or accuracy of the instruction and architectural models, variables from 84 other DSP programs (that were not used to derive the statistical model) were used in each equation to predict current. Results of this data were compared with actual current measurements. The predicted power or current value had a error less than the maximum error of the actual measured current for each model.



ID	Model Variables	mA	%	R^2
$Algo$	SUB, MUL	2.68	4.14	.18
$Instr_3$	$MUL, PABUS, IR, M$	1.61	2.49	.78
$Instr_2$	$x_p, IR, MUL, PABUS$	1.58	2.44	.82
$Instr_1$	$LOAD, PABUS,$ $SUB, IR, DABUS$	1.26	1.91	.89
$Arch$	$a, dataMemory, t$	2.15	2.91	.91

3 Discussions and Conclusions

A linear model was chosen originally for parsimony, however results indicate that it is applicable to power prediction. This research has identified that accurate power models for software can be obtained from statistical methods using benchmark DSP programs. In cases where designers have good samples of data that their DSP processors will be using, one could use this data for input to the DSP benchmark and single instruction tests for generation of more accurate power prediction models. Results also indicate that prediction can be performed accurately (see $Instr_1$) without detailed switching information from the processors internal registers and busses.

In contrast to previous research we have used statistical procedures that have verified a very accurate power model can be derived for DSP embedded software. Furthermore an independent assessment of the model, using data that did not derive the model, yielded excellent results. Although the results were presented using the TI TMS320C5x DSP processor, the methodology is very general and can be applied to any processor (assuming appropriate variables can be extracted). It has many similar characteristics to other DSP processors, such as the Motorola M56K and TMS320C3x for which address optimization and code generation techniques used in this paper are also applicable.

In summary fast accurate power prediction models ($Instr$) for DSP embedded software were developed. Unlike previous research(with 10% error[1]), one model, $Instr_1$, with only a few variables(not x_p), is very accurate(with 2% error); predicting current using statistical optimization with worst case error to under 1.3mA with an R^2 of 0.89. High accuracy is necessary especially in embedded systems design where throughput constraints are employed and changes in current are typically much less than 10%. This methodology is general (see figure) and can be applied to any processor. The model is derived from random data characterization of DSP benchmark programs which is very useful for many embedded DSP systems where only access to the hardware and a functional simulator is available, specifically detailed simulation models (from which register or bus switching data can be obtained) are typically not possible. This research is important for industry since a methodology for developing power prediction for DSP code is critical as new processors or cores grow in complexity and become integrated into embedded DSP systems which will continue to have stringent power, performance, and cost constraints. More details can be found in [7]. Thanks to Alexander Bond for his excellent work and support in part from NSERC and ITRC.

References

[1] M.Lee, V.Tiwari, S.Malik, M.Fujita, "Power Analysis and Minimization Techniques for Embedded DSP Software", IEEE Trans on VLSI Design, Vol.5,No.1, March 1997, p123-135.

[2] A.Chandrakasan,R.Broderson, Low Power Digital CMOS Design, Kluwer,1995.

[3] TMS320C5x User's Guide, Texas Instruments Inc., 1993.

[4] SPSS User's Guide, Base 7.5 for Windows, SPSS Inc, 1997.

[5] C.Gebotys, "DSP Address Optimization Using A Minimum Cost Circulation Technique", ICCAD,p100-3,1997.

[6] C.Gebotys, "An Efficient Model for DSP Code Generation: Performance, Code Size, Estimated Energy", ISSS,p41-7,1997.

[7] C.Gebotys,R.Gebotys, "Statistically Driven Power Prediction Models for DSP Processors",TechRept,ECE,Univ.Waterloo,1998.

[8] Fluke 867B GMM current meter manual, 1997.