Dynamic Current Modeling at the Instruction Level

Abstract— **Estimation of processor current consumption is important for the design of low power systems. This paper proposes a novel method for estimating the dynamic current consumption of a processor. The method models dynamic current as the output of a linear system excited by a signal comprised of the total current due to each instruction. System identification is performed by cross-correlation of a pseudo-random stimulus with the measured current. The method was applied to the Texas Instruments TMS320VC5510 DSP and was found to provide an average correlation of 93% between estimated and measured dynamic current across a range of benchmarks.**

I. Introduction

Power consumption has become an important issue in embedded system design. As a consequence power aware techniques have been applied at every stage in the design cycle: technology, circuit, logic, architecture, system and software. Traditionally, all efforts were focused on hardware design while software was ignored from the power perspective. While the underlying hardware is the source of power dissipation, the program executing on the architecture needs to be taken into account since the machine is under its control.

The impact of software on power consumption is particularly evident for complex architectures, such as DSP processors. Modern DSP processors offer the programmer a high degree of parallelism. This leads to a high variation in power consumption between instructions which exercise the maximum number of parallel functional units and those which exercise only one. For these processors, power cannot be accuracy predicted from cycle count alone.

Instruction-level power models estimate the current consumption of a processor by analysis of the instructions to be executed. This is considerably more time and cost efficient that using electronic equipment to directly measure power consumption. However, at present, almost all instruction-level models estimate mean power consumption over a long time windows, typically hundreds or thousands of instructions in duration. Although useful for some applications, this is not sufficient for others for which current would ideally be estimated on a cycle-by-cycle basis.

Estimation of dynamic current is important for several applications. Knowledge of the dynamic current consumption of an application can assist programmers in optimizing software for power consumption. Recent work on iterative compilation and compile-space exploration has shown the utility of accurate power estimation within an automated design flow [1]. Battery life can be extended by smoothing the current peaks which occur in mobile devices [2]. It has been shown that encryption systems may be cracked by current analysis attacks. These attacks could be prevented by re-ordering instructions such that the current consumption of the processor is constant [3].

In this paper, a new instruction-level model for dynamic current consumption estimation is presented. A method for identification of the model is described. The model and method are applied to a commercial DSP processor. The results of the estimation technique are presented and compared with meaasurements.

The rest of the paper is organized as follows: section II reviews related work already published in the literature; section III introduces the dynamic current model. Section IV explains the identification method. Section V describes the application of the model and method to the target processor. Results are given in section VI. Conclusions and future work are provided in section VII.

II. Related Work

The power consumption of processors has been modelled at the hardware and instruction levels. Hardware-level power models calculate power and energy consumption from detailed descriptions of the hardware, such as circuit, gate, register transfer and system models. Hardware-level models are slow to simulate and so are impractical for estimation of the power consumption of entire processors. These models often cannot be applied due to the lack of circuit and gate level information [4]. Instruction-level power models deal only with instructions and functional units from the software point of view without detailed knowledge of the underlying circuit architecture. In this context, the term 'instruction' corresponds to the combination of factors such as operation code, addressing mode, operand formats and processor resources which determine execution. Several instruction-level power estimation models have been proposed. These can be classified into two main types - Instrucion Level Power Analysis (ILPA) and Functional Level Power Analysis (FLPA).

The ILPA model, introduced by Tiwari et al. [4][5][6], states that the energy consumption of a program can be computed by summing up the energy cost associated with each instruction plus the incremental inter-instruction cost associated with switching between consecutive pairs of instructions. In order to populate the model, the current consumption of each instruction, and of each instruction combination, must be measured for the target processor. This method has a small margin of error, typically 2 to 4 percent for simple processors. Tiwari's methodology has been succesfully applied to the Motorola DSP56K [7], ARM7 [8], M3DSP [9] , Hitachi SH-4 [10], i960 [11] and Motorola 68HC11 [12], among others.

Measuring instruction and inter-instruction power consumption for the entire instruction set can require considerable effort as the number of measurements is directly related to the complexity of the processor architecture (for example, the DSP 56K requires 1176 measurements [7]). Simplifications to the measurement process have been proposed. The NOP model [7] proposes that the inter-instruction effect is only measured relative to the NOP instruction (proportional to N) rather than for every possible pair of instructions (proportional to N^2). Sinha and Chandrakasan [10] simplify the model to first order, relating current consumption to operating frequency and voltage only. They achieved around 8% error on a StrongARM processor. Russell et al. [11] also used a first order model for the Intel i960 processor achieving an 8% error. It is worth noting that this sort of simplification is only valid for Reduced Instruction Set Computers (RISC) which display little instruction-to-instruction variation in power consumption. This model applied to the TMS320VC5510 DSP, considered herein, could lead to an error of up to $\pm 26.5\%$ and for the Motorola 56K an error of around 20% [8]. A second order model

includes a logical partitioning of the instruction set into "instruction classes". This reduces the estimation error for the StrongARM to 4% [10].

To avoid the large number of measurements and complex modeling of functional unit activity required for ILPA, FLPA models have been proposed. These models rely on sets of parameters describing the underlying activity of the processor. These parameters must be derived for every program analysed. A number of architectures have been modelled using FLPA. In particular, those with large numbers of functional units operating in parallel, such as Very Long Instruction Word (VLIW) processors. Gebotys et al. [13] propose use of six parameters that were found to be power sensitive. They reported achieving an error of less than 2% for a subset of the ISA of an older TMS320C5x processor. Julien et al. [14] used another set of parameters for the Texas Instruments TMS320C6201 obtaining an error of 4%.

ILPA and FLPA models are not intended to model the dynamic variation of the processor current consumption. The models attribute a total current consumption to each instruction or instruction parameter and do not model when the current is actually drawn. As such, ILPA and FLPA are effective in deriving the long term mean current but do not attempt to estimate the cycle-to-cycle current variation.

In [15] the problem of modeling the dynamic current trace is addressed. The model is based on a gamma function. This function is used to approximate the dynamic current consumption due to execution of an individual instruction. Results for the SC140 DSP processor core show an average error of less than 2.2%, and an average correlation coefficient of 98% between estimated and measured current traces. Although good results are reported, the gamma function is not general enought to describe instruction current consumption profiles for all processors. The basic current shape for the SC140 architecture was well described by gamma functions but, as will be shown, other processors have different dynamic current consumption profiles.

III. Dynamic Model

The Tiwari instruction-level model estimates the mean current consumed by a segment of code by summing the current consumption due to each instruction and due to inter-instruction effects [4]. Typically, the Tiwari model estimates the mean current consumption over several thousand clock cycles. The instruction and inter-instruction current consumptions are determined experimentally by putting the instructions in infinite loops and measuring the current. Based on this, the mean energy consumption of a program is expressed in the form:

$$E_p = \sum_i (B_i N_i) + \sum_{i,j} (O_{i,j} N_{i,j}) + \sum_k E_k \qquad (1)$$

where B_i is the base cost of instruction i, N_i is the number of occurrences of instruction i, $O_{i,j}$ is the circuit state overhead for the instruction pair i, j, $N_{i,j}$ accounts for number of successive appearances of instruction i and j and E_k is the energy costs dissipated to other effects.

The Tiwari model does not capture the dynamic current consumption of the processor. This paper proposes a new dynamic model that estimates the current drawn by the processor in every clock cycle.

We model the dynamic current consumption of a processor as the output of a linear system excited by a input signal consisting of the total current consumption due to each instruction including the inter-instruction effect. Thus the estimated instantaneous current $y_e[n]$ can be calculated as the convolution of the discrete input signal, $x_d[n]$, with the system impulse reponse $h_i[n]$:

$$y_e[n] = \sum_{k=0}^{N} h_i[k] x_d[n-k] \qquad (2)$$

The input signal $x_d[n]$ can be derived by applying the conventional static Tiwari model to each individual instruction in the execution trace. As will be shown in the next section, the system impulse response $h_i[n]$ can be determined using system identification techniques. The model is depicted in Fig. 1.

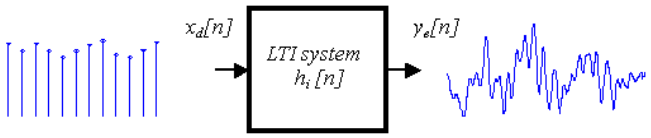

Figure 1: System model.

Our model assumes that:

- The system is Linear-Time Invariant (LTI). For the current and voltage ranges in question, the passive components in the processor power supply system behave in a linear fashion. The only non-linear effect arises from the DC-DC voltage regulator. Voltage regulators have a reference voltage and a feedback loop [16] which introduce some non-linear behaviour. However, as will be shown, this non-linear effect does not significantly reduce the accuracy of the model.
- The dynamic current profile for all instructions is the same. Clearly, different instructions draw different proportions of their total current consumption in different stages of the processor pipeline. For example, some instructions require memory accesses while others do not. Since the impulse response of the system is significantly longer than the depth of the processor pipeline, the error introduced by ignoring pipeline effects is small.
- The static model is accurate. For complex instruction set architectures, such as the one consider herein, it is not feasible to characterize the entire instruction set. In this work, uncharacterized instructions are matched to functionally similar characterized instructions and the total current consumption is assumed to be equal. This is a source of error and is an item for future work. The static model does not consider the data dependency of current consumption. Experiments were conducted across a range of instructions operating on data at varying Hamming distances. For practical DSP routines, the mean data dependency error was found to be 1.5% of total current per instruction. The static model does not cover accesses to different memories banks. Current estimation was conducted on the basis that source code and data is placed in SARAM and DARAM respectively. Different memory configuration may give rise to an error of up to 12% in total current estimation for a single instruction.

The model presented in this paper differs from that used by Muresan

[15] in the basic function employed to describe the instantaneous current consumption due to instruction execution. The Muresan model uses a mathematical function, gamma, to approximate the current trace due to execution of a single instruction. In contrast, our model uses an generalized impulse response. The gamma function is defined as:

$$g(t;n,\lambda) = \frac{\lambda(\lambda t)^n}{n!} e^{-\lambda t} \tag{3}$$

where t is the time variable of the instruction current trace, and parameters λ and n are determined by fitting the gamma function to the measured current trace. Unfortunately, the gamma funtion is not sufficient to accurately represent all possible current traces. For example, the current trace obtained for our target processor is not well modelled by a gamma function. Fig. 2 represents both the fitted gamma function, and the captured current trace resulting from execution of a dual MAC instruction. It can be seen that the gamma function describes just the first part (50.1%) of the measured trace. The remainder of the trace is not modelled (49.1%). This leads to a significant reduction in the accuracy of dynamic current estimation.

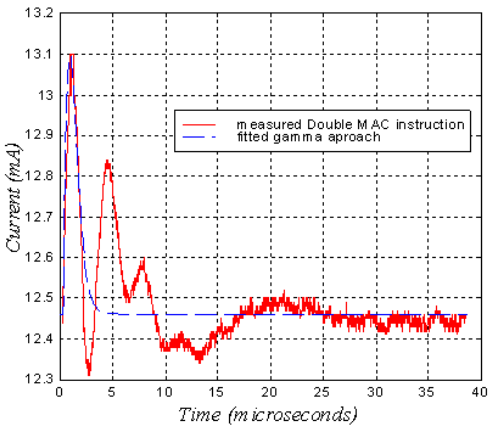

Figure 2: Difference between the gamma model and the actual dual MAC current consumption.

IV. System identification

In order to apply our dynamic model, it is necessary to estimate the impule response of the system. In general, system identification is divided into parametric and non-parametric methods [17]. In parametric methods, a system model is assumed and identification is oriented towards extraction of the model parameters. In non-parametric methods, no assumption is made about the system model and identification is used to directly compute the system response. Non-parametric methods include: correlation analysis, transient-response analysis, and frequency response analysis [17], [18].

In this paper, system identification is carried out by means of cross-correlation analysis. Consider the cross-correlation, $R_{xy}[l]$, of the system input and output signals, $x[n]$ and $y[n]$ respectively:

$$R_{xy}[l] = \sum_{n=-M}^{+M} x[n]y[n-l] = \sum_{n=-M}^{+M} h[n]R_{xx}[l-n]$$

where $R_{xx}[l]$ is the auto-correlation of the input signal.

In the case that the auto-correlation of the input signal tends to a Dirac impulse function $\delta[l]$, then the cross-correlation of the input and output signals tends to the impulse response of the system $h[l]$:

$$R_{xx}[l] = \delta[l] \longrightarrow R_{xy}[l] = h[l] \tag{4}$$

During system identification, the input signal is controlled by modifying the program to be executed. The static current model is utilized to determine the input signal. The corresponding system output is measured using a current probe. The program to be executed is specially designed so that the auto-correlation of its static current is a Dirac impulse. Maximal length sequences are used for this purpose. Pseudo-random binary sequences may be generated using Linear Feedback Shift Registers (LFSRs). A maximal length LFSR sequence (m-sequence) has the property that that its period is equal to $2^m - 1$ where m is the number of bits in the LFSR [17]. Its circular auto-correction $R_{mm}(l)$ is given by:

$$R_{mm}[l] = \begin{cases} 1, & \text{if } l = k(2^{m-1}), \\ \frac{-1}{2^m - 1}, & \text{if } l \neq k(2^{m-1}). \end{cases}$$

As well as having an impulsive circular auto-correlation, m-sequences also have the advantage that they are two valued. Thus the desired input can be generated by executing a stimulus program utilizing just two statically characterized instructions.

A further advantage of this system identification method is that no electrical parameters of the processor chip or printed circuit board need to be determined. This allows dynamic models to be derived by third-parties, not just the chip manufacturer.

V. Method

A. Measurement framework

The target DSP processor used for the study is a Texas Instruments TMS320VC5510, with variable core voltage (0.9-1.6 V) and operating frequency of up to 200 MHz. It incorporates several special architectural features pertinant to the study. Among them, there are low power capabilities, parallel features, on-chip Single Access RAM (SARAM) and Double Access RAM (DARAM), two independent 40 bit MAC units and one 40 bit ALU [19].

The physical measurement methodology was applied using the C5510 Development Software Kit [20], with a 1.6 V core voltage and 24 MHz operating frequency connected to a PC running TI Code Composer Studio (CCS) version 2.56. CCS was used to download and run the test programs. External software routines were used to trigger measurements using a digital storage scope. The current drawn was measured with a non intrusive 0.1 mA resolution current probe. The probe bandwidth is around 50 MHz, enough for these experiments.

In order to avoid noise, each measurement was averaged over 128 instances. The measurements are completely repeatable.

B. Identification procedure

The cross-correlation method requires that the input signal meets certain requirements. Firstly, the inputs must be discrete and have a finite number of levels as they must be generated from instructions. Secondly, the auto-correlation of the input signal must be close to an ideal impulse. Thirdly, the inputs must be zero mean in order for the auto-correlation to be delta shaped. Fourthly inputs must concentrate as much energy as possible in the bandwidth of the system to be identified.

The first three requirements were met by using maximal length LFSRs to generate the stimulus. The binary LFSR outputs were mapped to a low consumption NOP instruction and a high consumption MPY instruction. The mean was subtracted to shape the auto-correlation as an impulse. These signals along with their auto-correlation functions are illustrated in Fig 3.

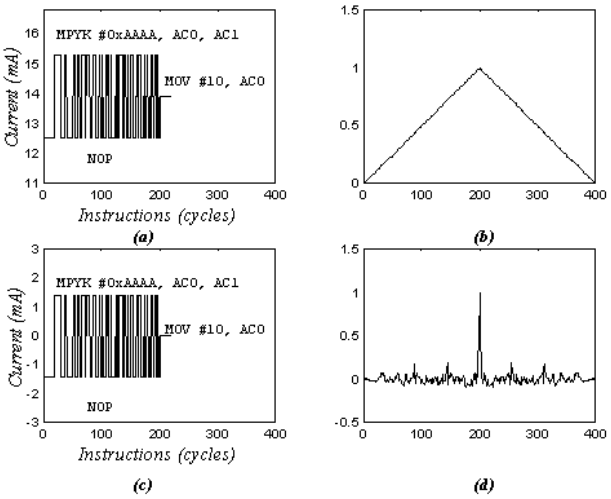

Figure 3: Input Generation: (a) Actual input signal and (b) its auto-correlation. (c) Zero mean input signal and (d) its auto-correlation

The fourth requirement is related to the system bandwidth. Efficient input spectrums for identification have most energy concentrated within the bandwidth of the system. The bandwith of an LFSR derived input signal can be controlled by altering the duration of the LFSR period. In our experiments, this was achieved by adjusting X_c, the number of processor clock cycles per LFSR output. This leads to a input signal length of L cycles where:

$$L = (2^m - 1)X_c \tag{5}$$

For a clock frequency of F_s, this time scaling increases energy in the frequency range from 0 to F_s/X_c.

Several experiments were conducted to estimate the bandwidth of the system. These experiments were performed at a processor clock frequency of 24 MHz. Input signals were applied to the system with X_c factors varying from 1 to 20. For each experiment, both the input and output spectrum were analysed along with the system identification results. 95% of the energy system was found to be within the range 0-2 MHz. X_c equal to 10 increases the energy in the range of 0-2 MHz and was found to provide the most accurate identification.

Fig. 4 (a) shows the input and output magnitude spectrums for X_c=1. The output spectrum magnitude is -20 dB for frequencies greater than 2 MHz. Fig. 4 (d) shows the spectrum of the input signal generated with X_c=10. This signal has more energy within the system bandwidth, providing more accurate system identification. Due to the time scaling process, certain harmonic frequency components are not present in this input signal. Since these components are outside the system bandwith, this does not signifcantly reduce the accuracy of the identification process.

The period of the input signal must be longer than the impulse response of the system and less than the storage capacity of the digital oscilloscope. In our case, X_c and m were chosen to be 10 and 7 respectively, resulting in L=1270. Fig. 4 (b) plots the circular auto-correlations of length 1280 and 1270 input signals, illustrating the low noise property of the latter.

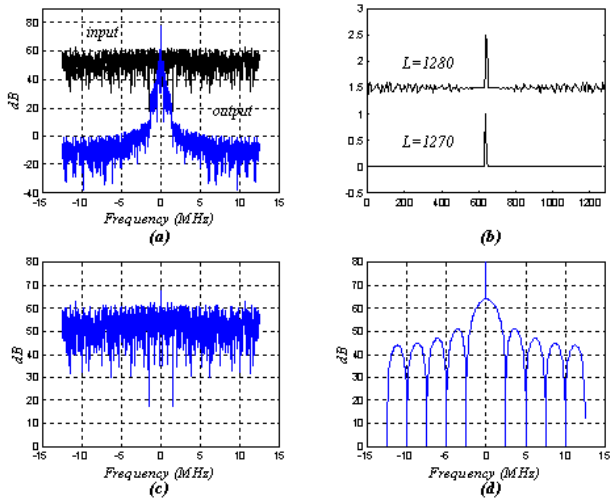

Figure 4: Input Generation: (a) Input and output spectrums with X_c=1; (b) Auto-correlation for LFSR signals of 1280 and 1270 length; (c) Input spectrum with X_c=1; (d) input spectrum with X_c=10 and 1270 length.

VI. Results

The impulse response and transfer function obtained for the TMS320VC5510 are plotted in Fig. 5. The impulse response is truncated to 600 cycles since later samples are close to zero. This also ensures that the filter has unit gain.

The dynamic current consumption of the processor was estimated and measured for ten pseudo-random instruction sequences. The pseudo-random sequences were comprised solely of characterized NOP and MPY instructions. Fig. 6 shows the estimated and measured traces for three of the sequences. The accuracy of the results was assessed by calculation of the square of the correlation coefficient ($CC^2[\%]$), the relative error of the mean current ($RelMError[\%]$) and the root mean square error ($RMSError$) [21] between the estimated and measured vectors. An average CC^2 of 91.11% was obtained while the average $RelMError$ and the average $RMSError$ were 1.2% and 0.338 mA (6.3%) respectively.

The dynamic current consumption was also estimated for five benchmarks from the TI Signal and Image Libraries [22], [23]. These

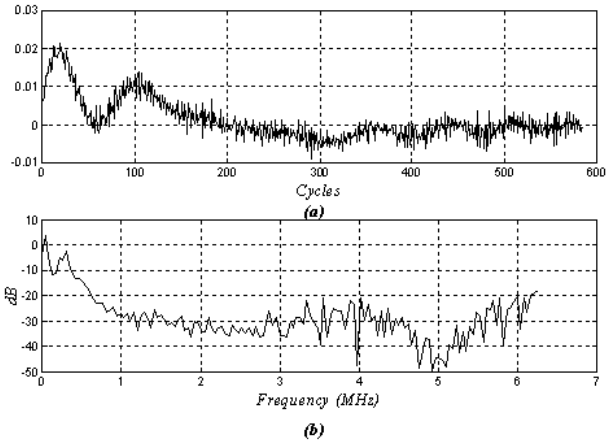

Figure 5: (a) Estimated impulse response along with (b) its spectrum

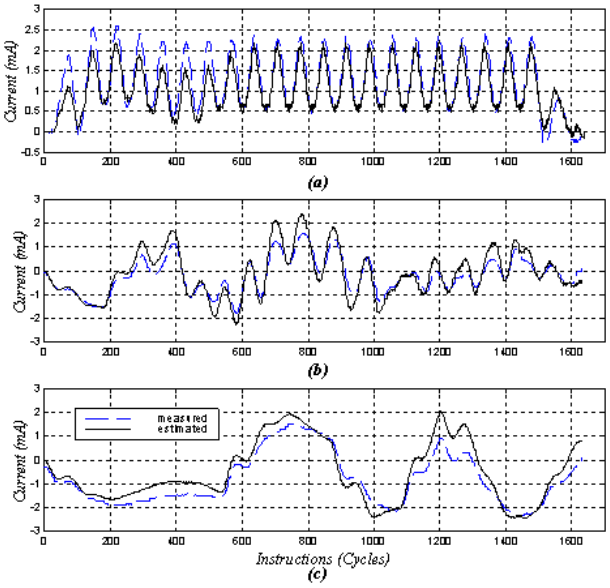

Figure 6: Estimated and measured current consumption for: (a) periodic pseudo-random sequence with X_c=10; (b) aperiodic pseudo-random sequence with X_c=10; (c) aperiodic pseudo-random sequence with X_c=30.

benchmarks cover a range of DSP algorithms such as complex Fast Fourier Transform (FFT), filtering, convolution, correlation and math functions. The benchmarks exercise most the features of the target DSP: software loops, zero-overhead loops, parallel and single instructions. The benchmarks are briefly described in the following list:

- Iir4. Computes a cascaded IIR filter of 16 biquad sections using 32-bit coefficients and 32-bit delay buffers. Each biquad section is implemented using Direct-form II with 4 coefficients.
- Cfft. Computes a complex 512-points IFFT.
- IMG_jpeg_quantize. Performs the quantization step in image/video compression.
- Log10. Computes log base 10 of a vector.
- Dct_idct. Comprises 2-D forward Discrete Cosine Transform (FDCT), format conversion and Inverse Discrete Cosine Transform (IDCT) for a 8x8 block.

For each benchmark, the estimated and measured current traces were obtained and analyzed. Fig. 7 shows the current consumption traces for the benchmarks and Table I gives the corresponding CC^2, $RelMError$ and $RMSError$ figures.

Table I: Benchmarks results.

Benchmark	$RelMError[\%]$	$RMSError[mA]$	$CC^2[\%]$
cfft	0.15	0.034	87.25
iir4	5.06	0.111	97.67
log10	3.07	0.080	94.93
img_jpeg_quantize	0.33	0.044	93.12
Dct_idct	-1.68	0.073	93.31

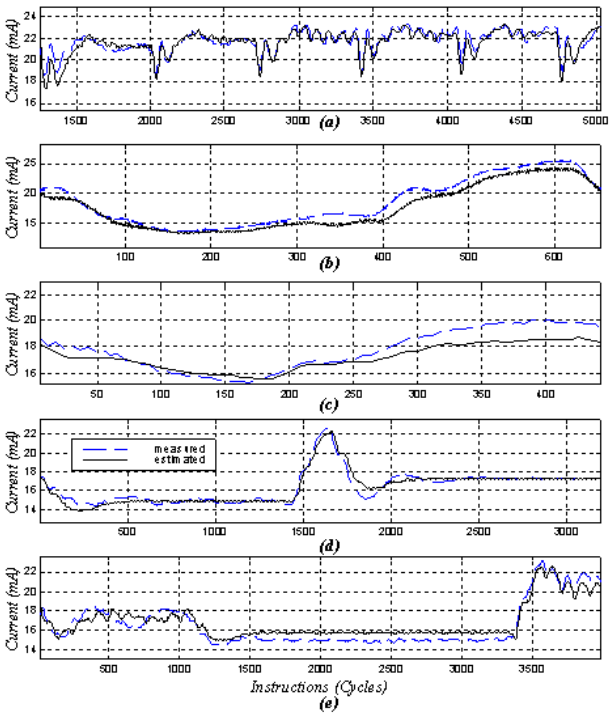

Figure 7: Measured and estimated current consumption for benchmarks: (a) cfft Kernel; (b) iir4; (c) log10; (d) img_jpeg_quantize; (e) Dct_idct

.

Almost all current estimates provide a CC^2 in excess of 93.26%. However, some variations in the current consumption were not well captured. For instance, in the Dct_idct benchmark (Fig. 7 (e)), the estimated trace does not exactly follow the measured trace between cycles 500 and 1000. As was noted in section II, the static model does not contain characterized current values for every instruction of the instruction set. When an instruction is not characterized, errors may be introduced in the signal $x_d[k]$. Other potential sources of errors are discussed in section IV. The $RelMError$ results indicate that the model also accurately estimates the mean current drawn by the processor.

VII. Conclusions and Future Work

This paper presented a novel instruction-level dynamic power model. An identification method was described which allows the model to be applied to third-party processors. The model and identification method were applied to the Texas Instrument TMS320VC5510 DSP.

Results were presented and the accuracy of the model in estimating dynamic current was assessed by comparison with measurements across a range of DSP benchmarks. The results showed that the model describes the dynamic behavior of the processor within an average correlation of 93% and estimates mean current with an average error of 2.05%.

For the processor under investigation, the method is more accurate than previously published methods. The new method is more general and so is applicable to a greater range of processors. The system identification method is also more robust and easier to apply than that described previously.

Future work includes extension of the static model to complex instruction set architectures and modeling of dynamic frequency and voltage scaling.

REFERENCES

[1] G.G. Fursin, M.F.P. O'Boyle, and P.M.W. Knijnenburg, "Evaluating iterative compilation," in *Proc. Languages and Compilers for Parallel Computers (LCPC)*, 2002, pp. 305–315.

[2] D. Rakhmatov, S. Vrudhula, and C. Chakrabarti, "Battery-conscious task sequencing for portable devices including voltage/clock scaling," in *DAC '02: Proc. of the 39th Conf. on Design automation*, 2002, pp. 189–194.

[3] R. Muresan and C. Gebotys, "Current flattening in software and hardware for security applications," in *CODES+ISSS '04: Proc. of the 2nd IEEE/ACM/IFIP Int. Conf. on Hardware/software codesign and system synthesis*, 2004, pp. 218–223.

[4] V. Tiwari, S. Malik, and A. Wolfe., "Power analysis of embedded software: A first step towards software power minimization," *IEEE Transactions on VLSI Systems*, vol. 2, no. 4, pp. 437–445, 1994.

[5] V. Tiwari, M.T.C. Lee, S. Malik, and M. Fujita, "Power analysis and low-power scheduling techniques for embedded DSP software," *Fujitsu Scientific and Technical Journal*, vol. 5, pp. 215–229, 1995.

[6] V. Tiwari, M.T.C. Lee, S. Malik, and M. Fujita, "Power analysis and minimization techniques for embedded DSP software," *IEEE Trans. on VLSI Systems*, vol. 5, no. 1, pp. 1–14, 1997.

[7] B. Klass, D.E. Thomas, H. Schmit, and D.F. Nagle, "Modeling inter-instruction energy effects in a digital signal processor," in *Proc. of the Power-Driven Microarchitecture Workshop*, 1998.

[8] G. Sinevriotis and T. Stouraitis, "Power analysis of the ARM 7 embedded microprocessor," in *9th Int. Workshop on Power And Timing Modeling, Optimization and Simulation (PATMOS)*, 1999.

[9] M. Lorenz, L. Wehmeyer, T. Drager, and R. Leupers, "Energy aware compilation for DSPs with SIMD instructions," in *LCTES/SCOPES '02: Proc. of the joint Conf. on Languages, compilers and tools for embedded systems*, 2002, pp. 94–101.

[10] A. Sinha and A. Chandrakasan, "JouleTrack - a web based tool for software energy profiling," in *DAC '01: Proc. of the 38th Conf. on Design automation*, 2001, pp. 220–225.

[11] J. Russell and M.F. Jacome, "Software power estimation and optimization for high performance, 32-bit embedded processors," in *ICCD '98: Proc. of the Int. Conf. on Computer Design*, 1998, p. 328.

[12] C. Chakrabarti and D. Gaitonde, "Instruction-level power model of microcontrollers," in *Int. Symp. on Circuits and Systems*, 1999, pp. 76–79.

[13] C.H. Gebotys and R.J. Gebotys, "An empirical comparison of algorithmic, instruction and architectural power prediction models for high performance embedded DSP processors," in *IEEE Int. Symp. on Low Power Electronics and Design*, 1998, pp. 121–123.

[14] J. Laurent, N. Julien, E. Senn, and E. Martin, "Power consumption modeling and characterization of the TI C6201," *IEEE Micro*, vol. 23, no. 5, pp. 40–49, 2003.

[15] R. Muresan and C. Gebotys, "Instantaneus current modeling in a complex VLIW procesor core," *ACM Trans. on Embedded Computing Systems (TECS)*, vol. 4, no. 2, pp. 415–451, 2005.

[16] T.J. Pelc L.D. Smith, R.E. Anderson and T. Roy, "Power distribution system design methodology and capacitor selection for modern CMOS technology," *IEEE Trans. on advanced packaging*, vol. 22, no. 3, pp. 284–291, 1999.

[17] Lennart Ljung, *System Identification: theory for the user*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1986.

[18] Matz Lenells Bengt Johansson, "Possibilities of obtaining small-signal models of DC-to-DC power converters by means of system identification," in *Proc. Telecommunicattions Energy Conf*, 2000, pp. 65–75.

[19] Texas Instruments Inc., *TMS320C55x DSP CPU Reference Guide*, 2004.

[20] Spectrum Digital Inc., *TMS320VC5510 DSK Technical Reference*, 2002.

[21] A. G. Bluman, *Elementary Statistics: A Step-by-Step Approach*., Wm. C. Brown Publishers,, Dubuque, Iowa., 1994.

[22] Texas Instruments Inc., *TMS320C55x DSP Library Programmer Reference*, 2004.

[23] Texas Instruments Inc., *TMS320C55x Image/Video Processing Library Programmer Reference Guide*, 2004.