International Journal of Advances in Computer Science and Its Applications– IJCSIA
Volume 4: Issue 2 [ISSN: 2250-3765]
Publication Date : 25 June 2014

Software Power Analysis for Embedded DSP Software

Prashant V.Joshi [1], K.S.Gurumurthy[2]

Abstract— **Power is a major design constraint for low power applications. Its consumption during the execution of the software program is an important issue in designing low power embedded devices. This so called software power can be reduced in many ways. By manipulating the instructions in a code, software related power could be reduced. This work brings about the efficient scheme for instruction level software power analysis for TMS320C6713 DSP processor. This is achieved by measuring the average instantaneous current drawn and hence power dissipated by the processor as it repeatedly executes the set of instructions.**

Introduction

Many applications in special areas such as hand-held computation, tiny robots and guidance system in automated vehicles are powered by batteries of low rating. In order to avoid frequent recharging or replacement of the batteries, there is significant interest in low power VLSI system designs [1], [2]. Most of the efforts in controlling power dissipation have been and continue to be focused on hardware design. However, it would be unwise to ignore the influence of software on power dissipation. In systems based on digital processors or controllers, it is software that directs much of the activity of the hardware. Consequently, the manner in which software uses the hardware can have a substantial impact on the power dissipated by a system [3]. Some system level power saving features have been explored and incorporated into portable devices such as notebook computers [1]. However, there is very little disclosure on power-conscious software designs, which are also very important in minimizing total power consumption in a system. This paper presents instruction level power analysis for TMS320C6713 VLIW. Section 2 presents related work with respect to the instruction level power analysis. Section 3 presents the introduction to target architecture and experimental setup. Section 4 presents the Instruction Level Power Modelling for DSP software and validation of the methodology. Section 5 gives the results and scope for the future work in this direction.

I. Related Work

VLSI 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 [4]. This section summarizes the most recent contributions in instruction level power analysis.

[1] Prashant V.Joshi
REVA Institute of Technology and Management, Bengaluru, INDIA,

[2] K.S.Gurumurthy
REVA Institute of Technology and Management, Bengaluru, INDIA,

An instruction level power analysis for individual processor was first proposed by Tiwari et al [4]. The total power consumption of the processor varies from program to program and hence it is required to develop methodology that provides a means for analyzing the power consumption of the processor as it executes a given program. For a given processor it is possible to very accurately evaluate the power cost of the program part of it. This can be achieved by measuring the current drawn by a processor as it repeatedly executes the instructions. It is also possible to obtain most of the information that is needed to evaluate the power cost of a program for that processor [4]. Tiwari et al. modelled the power consumption of the Intel DX486 processor. This methodology has been developed on the basis of the base cost for each instruction, including the interinstruction overheads that depend on the neighbouring instructions.

C. J. Bleakley et al. proposed software power consumption models and power saving techniques for TMS320VC5510 processor. The experimental framework described includes the features impacting for performance and power/energy consumption in the Texas Instruments TMS320VC5510 Digital Signal Processor [5].

Accurate energy consumption model at the instruction level proposed by sheayun Lee et al. [6] gives a technique to estimate the energy spent by a sequence of instructions. For energy consumption model to be applicable to a program it must have optimization frame work: Accuracy, simplicity, accountability and retargability. The proposed energy model can estimate the energy cost of software using the properties of instructions. It is based on the actual energy consumption data measured from real hardware. The target processor is considered as “Black Box” whose internals are not known and assume that the only accessible information is the responses of this black box for a set of stimuli. The energy consumption of the target processor is measured, when it executes a set of test programs and an energy model is derived by reasoning about the relationship between the instruction sequences and the measured energy. Furthermore this technique also enhances the retargatability of the approach, since the technique is not dependent on the implementation of a specific processor. However this technique does not include the effect of pipeline stalls, load store instructions and the memory devices [6].

Nikolaodis et al. [7] proposed a technique to determine the energy cost of the program by measuring the instantaneous current drawn by the processor during the execution of the instruction. The proposal gives a better way of current measuring technique compared to techniques proposed in [4]. A current mirror is used as current sensing circuit to minimize the supply voltage fluctuations and to increase the resolution of the current measurements [7].

Nikolaodis et al. [8] has proposed a technique to determine the energy models for pipelined processors by monitoring the instantaneous current drawn by a processor at each clock cycle. Knowing the current waveform, the corresponding energy consumed during a clock cycle can be calculated. According to the proposed method the energy costs, base cost and interinstruction are modelled in relation to a reference instruction. The main disadvantage of techniques used in [7], [8] was the complexity involved in measuring the current.

An attempt to modify the original Instruction level power analysis to create an instruction level power model with gate level simulators is carried out by Sama et al. [9]. In this approach, the power cost values were obtained through a power simulator rather than actual measurement and thus modelling is possible at design time. This approach can be part of micro architecture and instruction set architecture exploration. Since the measurements are done at much lower level (gate level), the efforts and time for the instruction level model characterization process is large [9].

II. Introduction to TMS320C6713: Target Architecture

This section briefly describes the target processor's architectural features and experimental setup. The simplified block diagram of TMS320C6713 is shown in Fig.1. The TMS320C6713 onboard the DSK is a floating-point processor based on the VLIW architecture. Internal memory includes a two-level cache architecture with 4 kB of level 1 program cache (L1P), 4 kB of level 1 data cache (L1D), and 256 kB of level 2 memory shared between program and data space. It has a glueless (direct) interface to both synchronous memories (SDRAM and SBSRAM) and asynchronous memories (SRAM and EPROM). Synchronous memory requires clocking but provides a compromise between static SRAM and dynamic DRAM, with SRAM being faster but more expensive than DRAM. The CPU consists of eight independent functional units divided into two data paths, A and B, as shown in Figure 1. Each path has a unit for multiply operations (.M), for logical and arithmetic operations (.L), for branch, bit manipulation, and arithmetic operations (.S), and for loading/storing and arithmetic operations (.D).

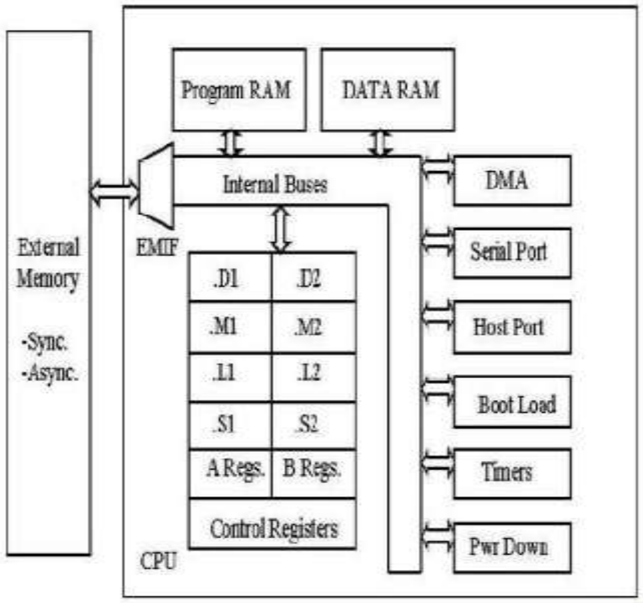

Fig.1: Simplified Block diagram Of TMS320C6713.

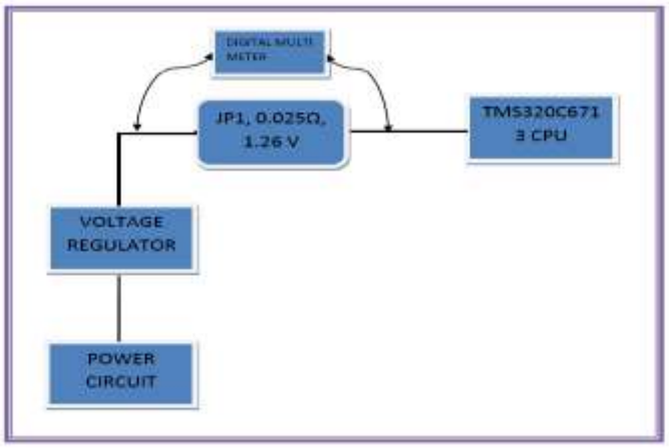

Fig.2: TMS320C6713 DSK

The .S and .L units are for arithmetic, logical, and branch instructions. All data transfers make use of the .D units. The arithmetic operations, such as subtract or add (SUB or ADD), can be performed by all the units, except the .M units (one from each data path). The eight functional units consist of four floating/fixed-point ALUs (two .L and two .S), two fixed-point ALUs (.D units), and two floating/fixed-point multipliers (.M units).Each functional unit can read directly from or write directly to the register file within its own path. Each path includes a set of sixteen 32-bit registers, A0 through A15 and B0 through B15.

The methodology demonstrated in this paper depends on the measurement of the instantaneous current drawn by the TMS320C6713 CPU while executing the instructions. All the current measurements are carried out on the TMS320C6713 DSP Starter Kit (DSK). The DSK operates from a single +5V external power supply connected to the main power input (J5). Internally, the +5V input is converted into +1.26V and +3.3V using separate voltage regulators. The +1.26V supply is used for the DSP core (CPU) while the +3.3V supply is used for the DSP's I/O buffers and all other chips on the board. There are three power test points on the DSK, namely JP1, JP2 and JP4. All I/O current passes through JP2 while all core current passes through JP1. All system current passes through JP4. The operating frequency is 225MHz. The current drawn by DSP Core while executing certain instructions is captured by Agilent U1241B 4 digit digital multimeter. As shown in Fig.2, voltage drop is measured across JP1 with 0.025Ω resistor inserted by the DSK manufacturer in the current path of DSP core. Further current drawn by the processor core in amperes can be determined by:

$$I_{core} = V_{drop} / 0.025\Omega \qquad (1)$$

III. Instruction Level Power Analysis

Instruction level power analysis can be achieved by measuring the current drawn by a processor while executing the instructions. To model the power dissipated for a given program it is necessary to first evaluate the power dissipated by individual instruction. It is carried out by measuring the current drawn by the processor as it repeatedly executes individual instruction and hence it is possible to obtain most of the information that is needed to evaluate power cost of an instruction.Average power dissipated while executing individual instructions is given by:

$$P_{avg} = I_{avg} \times V_{cc} \quad (2)$$

Where

I_{avg} = Average instantaneous Current
V_{cc} = Supply core voltage
P_{avg} = Average power.

Hence, Energy consumed by instruction is:

$$E = P_{avg} \times T \quad (3)$$

Where

P_{avg} = Average power
$T = N \times \Gamma$
Where: N= Number of clock cycles taken
Γ= Clock period in secs

Total instantaneous current drawn by the processor core while executing program is given by:

$$I_{PROG} = I_{BASE\ COST} + I_{INTER\ INSTRUCTION\ EFFECT} \quad (4)$$

The total current drawn by a processor core while executing a program is the sum of the current drawn by individual instruction ($I_{BASE\ COST}$) and the inter instruction effect ($I_{INTER\ INSTRUCTION\ EFFECT}$). $I_{BASE\ COST}$ of instruction is the instantaneous current drawn by a processor core for executing the instruction. Base cost of instruction can be determined by putting an instruction in the loop and then measuring the instantaneous current drawn by processor core. Care should be taken while choosing the size of the loop. The larger sized loop will cause cache misses and smaller size will lead to inaccurate measurement. Hence in this work, we have chosen a loop size. This value is obtained with the help of Code Composer Studio (CCS) profiler. Current drawn due to the inter instruction effect includes- circuit overhead, resource constraints and cache misses.

TABLE.1: Base Cost of Instructions.

Number	Instruction	Current (mA)	Cycles	Energy (nJ)
1	NOP	404	01	2.26
2	MVK	408	01	2.28
3	ADD	412	01	2.30
4	SUB	412	01	2.30
5	MPY	440	02	2.46
6	SHR	412	01	2.30

Table.1 shows the sample values of base cost for some instructions in TMS320C6713 processor. Third column in the table represents observed average instantaneous current values and column-4 shows the total number of clock cycles taken by respective instructions. The overall base energy cost of an instruction which is given in column-5 is the product of the average current drawn by the CPU core, clock cycle taken to execute the instruction , core voltage (1.26 V) and time period- 4.44 nS (since the operating frequency is 225 MHz).

4.1 VALIDATION OF METHODOLOGY

To validate the correctness of the methodology presented in this paper, we have taken a sample program as the case study. Correctness of the methodology is demonstrated and its comparision results are presented in this paper.

We start validation for the program shown in table.2.

TABLE.2 CASE STUDY: Sample Program

PROGRAM	CURRENT (mA)	CYCLES
NOP	404	01
MVK 4,A1	408	01
ADD 4,A1,A2	412	01
SUB 4,A3,A1	412	01

From equation (4), the current drawn by processor I_{base} cost is obtained as

$$I_{base\ cost} = 1236/4 = 309 \text{ mA} \quad (5)$$

Similarly, Current drawn by the processor core by inter instruction effect is given by:

$$I_{inter\ instruction} = I_{nop\text{-}mvk} + I_{mvk\text{-}add} + I_{add\text{-}sub} \quad (6)$$

The inter instruction for obtained is
$I_{nop\text{-}mvk}$ = (408+404)/2 = 406 mA,
but the measured base cost is 416 mA.
Thus inter instruction effect of NOP & MVK is 10mA. Similarly for inter instruction effect for MVK & ADD is obtained as $I_{MVK\text{-}ADD}$= (408+412)/2 =410mA, but measured base cost is 416mA. Thus inter instruction effect of MVK & ADD is 6mA. By following the above said procedure, we have found that the interinstruction effect of 0.5mA. By using equation (4), the overall estimated current drawn by processor core while executing the sample program given in table1 is

Iprog = (409+6+10+0.5) mA
Iprog = 425.5 mA

But measured average instantaneous current is 424.5 mA and hence there is only 1mA difference between the estimated and measured values of currents.

To validate the methodology we have used several sample programs and have observed a close difference between the estimated and measured instantaneous average current values. The main reason for the difference between the estimated and measured values are as follows:

1. For the instructions that require operands, the operand values and the addresses are often not known until the run time.
2. The methodology used in this paper, doesn't take care of the cache penalty.
3. The Digital Multi Meter used in measuring the current values has only 4 digit precision.

IV. Conclusion and Future scope

This paper presents the methodology for software power analysis for embedded DSP applications. This methodology is based on the Instruction Level Power Analysis (ILPA). Since instruction level power analysis deals with power modelling with respect to the individual instruction and hence it provides insight into the power dissipation of processor. Methodology presented in this paper yields the difference of up to 2% between the estimated and measured current values.

Further this work can be extended for the power modelling of all the instructions available in the TMS320CX processors. Base cost of a program also varies, depending on the register numbers, register values and operand values.

References

[1] A. Chatzigeorgiou and G.Stephanides, “Energy issues in software design of embedded systems”, International Conference on Applied Informatics, Rethymnon, Crere, Greece, Jul.2002.

[2] J.T.Russell and M.F.Jacome, “Software power estimation”, IEEE Transactions of VLSI Systmes, Vol, 2, No.4, Dec. 1994.

[3] Mike Tien-Chien Lee, Vivek Tiwari, Sharad Malik, and Masahiro Fujita, “Power Analysis and Minimization Techniques for Embedded DSP Software”, IEEE Transaction on VLSI Systems.Vol. 5, No.1, March 1997.

[4] Vivek Tiwari, Sharad Malik, and Andrew Wolfe, “Power Analysis of Embedded Software: A First Step Towards Software Power Minimization”, IEEE Transaction on VLSI integration Systems, Vol2, No.4, December 1994.

[5] C. J. Bleakley, M. Casas- Sanchez, and J. Rizo-Morente, “Software level power consumption models and power saving techniques for embedded DSP Processors”, Journal of low power electronics, vol.2, pp.281-290, 2006.

[6] Sheayun Lee, Andreas Ermedahl, Sang Lyul Min, and Naehyuck Chang, “An Accurate Instruction-Level Energy Consumption Model for Embedded RISC Processors”, Proceedings of ACM SIGPLAN 1999 Workshop on Languages, Compilers SIGPLAN 1999 Workshop on Languages, Compilers and Tools for Embedded Systems,2001.

[7] S.Nikolaidis, N.Kavvadias, P.Neofotistos, K.Kosmatopoulos, T. Laopoulos and L. Bisdounis, “Instrumentation set-up for instruction level power modelling”, Proceedings of the 12th International Workshop on Power and Timing Modeling, Optimization and Simulation, pp.71-80, Springer, London, UK,2002.

[8] S. Nikolaidis, N.Kavvadias, L.Bisdounis and S. Blionas, “Instruction level energy modelling for pipelined processors”, Journal of Embedded Computing, vol.1, no.3, pp 317-324, 2005.

[9] A.Sama, J.F.M Theeuwen, and M.Balakrishan, “Speeding up power estimation of embedded software”, Proceedings of the International Symposium on Low Power electronics and Design, pp.191-196, ACM, New York, NY, USA, 2000.

[10] Shakeel Sultan, Shahid Masud, “Rapid Software Power Estimation of Embedded Pipelined Processor Through Instruction Level Power Model”, Performance Evaluation of Computer & Telecommunication Systems, 2009. SPECTS 2009.

[11] Manuel Wendt, Matthias Grumer, Christian Steger, Reinhold Weiss, Ulrich Neffe, and Andreas Muehlberger, “Tool for Automated Instruction Set Characterization for Software Power Estimation”, IEEE Transaction on Instrumentation and measurement, Vol. 59, No.1, Jan.2010.

[12] Manuel Wendt, Matthias Grumer, Christian Steger, Reinhold Weiss, Ulrich Neffe, and Andreas Muehlberger, “Tool for Automated Instruction Set Characterization for Software Power Estimation”, IEEE Transactions on Instrumentation and Measurement, Vol. 59, No.1, Jan. 2010

[13] Kenneth Lind, Rogardt, “A practical Approach to size estimation of embedded software components”, IEEE Transaction on Software Engineering, Issue-99, pp-1, 2011.

About Authors:

Mr. Prashant V.Joshi Obtained M.Tech. from Visweswaria Technological University, Belgum in (2009) and. is currently working as research scholar in the Dept. of E&CE REVA Institute of Technology and Management, Bangalore. He is currently pursuing Ph.D. His areas of interest are Low power VLSI, Analog and mixed signal VLSI design, Analog and digital communication and Nanoelectronics.

Dr.K.S.Gurumurthy Obtained Ph.D. from IISc in (1990) and. is currently working as senior professor in the Dept. of E&CE REVA Institute of Technology and Management,Bangalore. He has 33 years of teaching experience. He was awarded with gold medal for his performance in ME, and KHOSLA AWARD for the best Technical Paper published in a journal. He is Member of IMAPS INDIA, ISTE and IEEE. He has published more than 75 research papers. His areas of interest are Low power VLSI, Analog and mixed signal VLSI design, Analog and digital communication and optical fiber sensors.