Experimental Analysis on Time-Triggered Power Consumption Measurement with DVS-enabled Multiple Power Domain Platform

Songah Chae, Doo-Hyun Kim[1]

Embedded S/W and Sensor (Essens) Lab.
Konkuk University, Seoul, Korea
{sachae, doohyun}@konkuk.ac.kr

Changhee Jung, Duk-Kyun Woo and Chaedeok Lim

Embedded S/W Research Division, ETRI, Taejon, Korea
{chjung, dkwu, cdlim}@etri.re.kr

Abstract. Recently, the battery and low-power H/W technologies for mobile and wearable computing devices have been advanced rapidly. But on the other hand the computation and communication demands of the embedded applications are increasing more rapidly. Therefore, the application developers are still required to develop their codes to utilize the available energy as efficient as possible. The provision of software power measurement with reasonable accuracy, consistency and low overhead is an indispensable factor for software power engineering. In this paper, we present a time-triggered mechanism for providing energy consumption profiles in the level of C functions. The similar mechanisms have already been introduced at the previous researches such as PowerScope and ePRO. Instead, we, in this paper, introduce our efforts to extend these researches to incorporate power domains and DVS(Dynamic Voltage Scaling), then interpret these mechanisms as the view of time-triggered approach for better understanding to the relationships among timer interrupt, context switching, DAQ triggering, multi-channel DAQ delay, and etc. From our experimental results, we could conclude that the time-triggered approach for the function level energy measurement properly worked with low overheads and produced consistent energy consumption profiles on the DVS-applied program codes running upon the platforms supporting multiple power domains.

Keywords: Embedded Software, Power Consumption Measurement, Dynamic Voltage Scaling

[1] Corresponding Author: New Millennium Hall 1203, School of Internet and Multimedia Engineering, Konkuk University, Kwangjin-Gu, Seoul, 143-701, Korea (doohyun@konkuk.ac.kr)

1. Introduction

Power aware computing is becoming fundamentally important with the proliferation of portable, battery operated systems, such as PDA, cellular phones, MP3 player, PMP(Portable Media Player), and etc. In order to support the low power management, there has been variety of research areas in wide levels of spectrum including compiler, OS, system architecture, microprocessor, circuit and other H/W component as well as power consuming applications. Among these areas, the microprocessors have been mainly focused on providing low power functionalities such as DVS(Dynamic Voltage Scaling) and DFS(Dynamic Frequency Scaling). For example, Intel's PXA27x processor family is providing six power modes for enabling more efficient power management than the previous XScale processor family. Although such low-power H/W and battery technologies have been recently advanced rapidly, the computation and communication demands of such embedded applications are increasing more rapidly. Therefore, the application developers are still required to develop their codes to utilize the available energy as efficient as possible; consequently, the provision of software power measurement with reasonable accuracy, consistency and low overhead is an indispensable factor for software power engineering.

In this paper, we present a time-triggered mechanism for providing energy consumption profiles in the level of C functions of the applications using dynamic voltage scaling. The time-triggered mechanism uses periodic timer interrupts for activating DAQ(Data Acquisition) instrument, which is installed outside of the target device, to acquire the power from the concerned power domains such as processor core and memory. In addition, the timer interrupt handler records the instruction counter at the same moment. The stored trail of power and instruction counters is combined by referencing symbol table, generated at the compile time, to produce accumulated energy for each function in the application codes.

The similar mechanisms were already introduced at the previous researches such as PowerScope[7] and ePRO[9]. Instead, we, in this paper, introduce our efforts to extend these researches to incorporate power domains and DVS, then interpret these mechanisms as the view of time-triggered approach for better understanding to the relationships among timer interrupts, context switching, DAQ triggering, multi-channel DAQ delay, and etc. We also present our experimental results as an experimental validation of these time-triggered approaches in the presence of power domain and DVS.

The chapter 2 will provide background for the power estimation of software and introduce related researches and our own efforts. The chapter 3 will be focused on introducing our facilitations for the domain-wise power measurement of DVS-related applications. The section 4 will present our experimental results with our interpretation in the view of consistency and overhead. The section 4 will conclude this paper with suggestions for further research directions such as imposing our efforts into other time-triggered engines like TMO (Time-triggered and Message-triggered Object) [3].

2. Backgrounds and Related Works

2.1 Power Consumption of Software

According to the elaborated research results [1], while the instruction level current consumption in SA-1100 has a variation of about 38%, the variation of the current consumption in programs is much less, that is, 8% in maximum. This means that the current consumption depends on the operating voltage and frequency of the processor rather than a piece of code. But, as mentioned in [1], it might be significant in datapath dominated processors.

Also, the microprocessors have been mainly focused on providing low power functionalities with such as DVS and DFS. There is a quadratic dependency on the voltage with power consumption. This means that halving the voltage reduces the power consumption to one-fourth its original value. Meanwhile, reducing the power to one-fourth its original value only halves the maximum frequency since the maximum frequency is roughly linear in the voltage [2]. Thus, if the codes run at different frequency and different voltage by using DVS capabilities in certain situations, then the same codes lead different power consumptions.

In addition, the Intel PXA27x, for example, provides ten power domains such as VCC_CORE, VCC_MEM, VCC_LCD, and etc, as well as it allows DVM(Dynamic Voltage Management). In conjunction with PMIC(Power Manage Integrated Circuit), it provides six power modes such as IDLE, DEEP IDLE, and STANBY modes of which each mode requires different levels of voltage in each power domain[4-7]. So, the behavior of codes can also significantly affect the usage of power by changing power modes. Accordingly, it is necessary to allow application software developers to analyze the power consuming behavior of his/her program codes in details in the presence of DVS and power domains as well as in the view of the whole behaviors.

2.2 Software Power Measurement Tool

Software power measurement tool is the indispensable factor in leading application software developers to develop their software with the awareness on the power consumption. For this purpose, the tools are required to support several key factors such as the accuracy, consistency, low overhead, usability as well as DVS and power domains. The accuracy is supposed to depend on various aspects such as the usage of instrumental equipments for DAQ, the delay and jitter engaged with the channel multiplexing in DAQ instrument, and timer interrupt regularity.

The consistency is also important factor to the developers in the process of power optimization. The tools have to provide dependable measurements by providing consistent results with reasonable minor ratio of deviations for the same codes and same situations even in case of using time-sharing and multi-tasking platforms. The low overhead requirement is also important since some part of power measurement mechanism potentially intervenes the execution of the concerned application. As a sort of software engineering tools, the power measurement tool has interactions with users, i.e., software developers, hence, is necessarily required to provide decent multilateral views with proper presentation schemes.

2.3 Related Works

There were several research works related to this paper, including PowerScope[8], SES[9], ePRO[10], Arun[11] and Esto[12]. PowerScope[8] uses digital multimeter for DAQ with the support of triggering from OS running in the target system. It does not need to impose extra hardware into the target system. PowerScope maps energy consumption acquired from the external multimeter to program structure to determine the energy consumption of different procedures within a program. SES [9] collects energy consumption data in cycle-by-cycle resolution and maps the data into program structure to provide higher accuracy and resolution. But, SES needs extra acquisition module with measurement circuit, profile controller and acquisition memory, and hence, it may not be applied to ordinary target systems not equipped with the modules.

While it uses similar techniques used in PowerScope and SES, the ePRO [10] does not need extra instrumentation like SES and provides performance profiling as well as energy profiling. As its experimental target system, ePRO uses a commercial toolbox using PAX255 which enables performance profiling by using PMU(Performance Monitoring Uint) provided at architecture level. These previous research works gave many inspirations and technical details to our works. However, those are not dealing with the situations using DVS, nor considering simultaneous acquisitions from separate multiple power domains. The work from Arun Thomas[11] deals with DVS in their measurement platform. But it measures only the power of the overall laptop system with extra external circuits, and does not support function level energy profile.

Esto [12] is a visual IDE(Integrated Development Environment), based on Eclipse 3.0, for the embedded applications running on Qplus Embedded Linux[12]. It supports optimization techniques by transforming loop structure of source code, such as loop distribution, loop interchanging, loop unrolling, and scalarization. This paper is related to enable Esto to provide energy profiling, using Intel Mainstone [4] as a primary target toolbox with DVS and power domains.

3. Time-Triggered Power Measurement

3.1 Architectural Mechanism

As illustrated in Figure 1 and Figure 2, our system consists of target system, host system and DAQ board with a connector block. For the DAQ, we use NI-PCI-6251[13] that can be plugged into PCI slot of the host system. The connector block, SCC-2345[14], enables connections between NI-PCI-6251 and pins from power domains in the target system microprocessor, PXA270A. The NI-PCI-6251 provides simultaneous acquisitions from multiple power domains through channels. For the target system, we use the Intel Mainstone toolbox [4-7] installed with Qplus embedded Linux [12] enhanced with kernel level low power management module, called Harmonia. By using the Harmonia API, application program codes can change the frequency of the processor properly according to the situations of execution. Then, the Harmonia maps the requested frequency to the proper voltage. The host system is just normal desktop Linux computer installed with the device driver for the NI-PCI-6251.

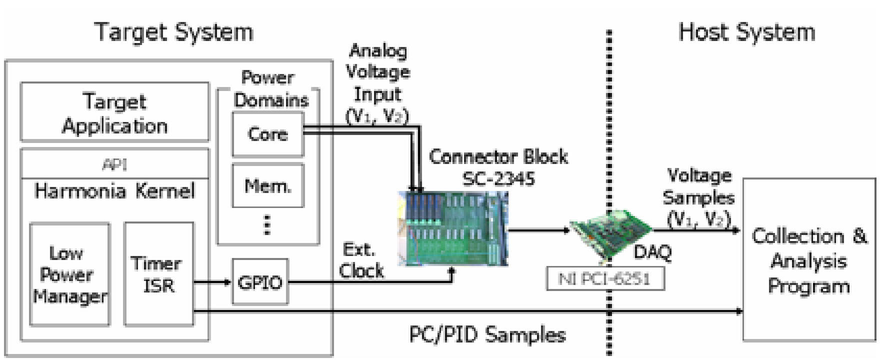

Figure 1. Overall Architecture of Target System and DAQ facilitation

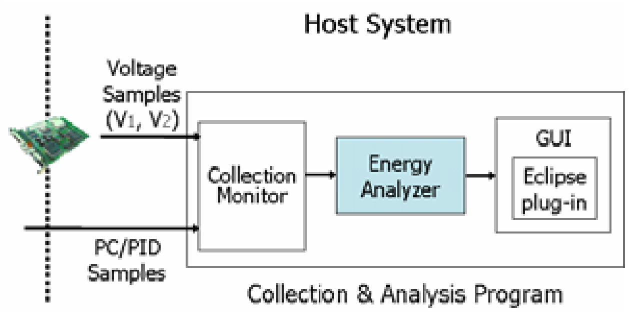

Figure 2. Overall Architecture of Host System

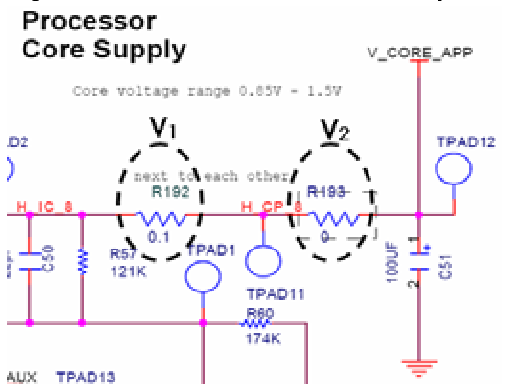

Figure 3. Schematic around header for Core (excerpted from [5])

Simultaneously with the start of target application, the timer starts to generate timer interrupt to activate its corresponding ISR (Interrupt Service Routine). For our experiment, the timer interrupt interval was set-up with 2ms. Whenever the timer ISR is activated, it generates external signal through GPIO for invoking NI-PCI-6251 to acquire voltages from each concerned power domain through the SC-2345 connector block. In addition, the ISR also records program counter (PC) of the application process that has been suspended due to the timer interrupt.

These records are used for deriving function level energy consumption profiles later at the host system.

The Intel Mainstone toolbox provides pairs of two outer pins of each power domain. Among these two pins of a pair, one is to get voltage drops, V1, of the corresponding power domain by measure across the sense resistor, R_{sense}, the other is to get the ground voltage, V2, of the domain (see Figure 3 for schematics). Thus, the power consumption of the corresponding power domain at this very moment is calculated as $P = (V_1 / R_{sense}) \times V_2$. In case of Intel Maintone board, the R_{sense} for PXA270 core domain is 0.1 ohm as shown in Figure 3.

3.2 Software Energy Measurement

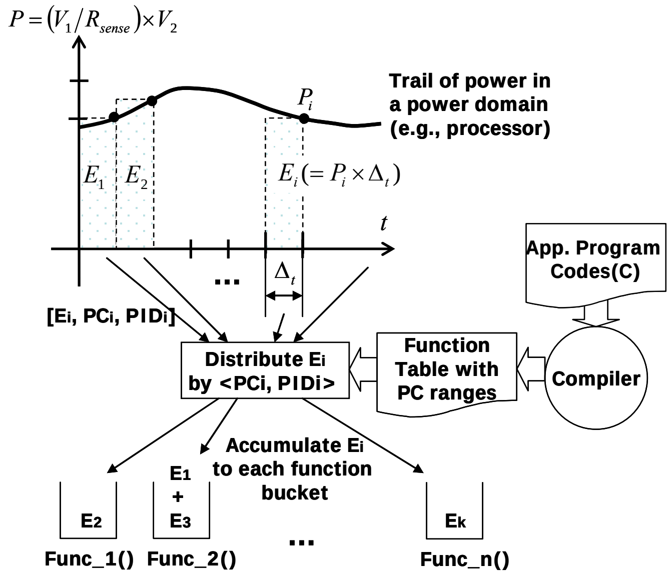
$P = (V_1 / R_{sense}) \times V_2$
P_i
Trail of power in a power domain (e.g., processor)
E_1
E_2
$E_i (= P_i \times \Delta_t)$
t
Δ_t
...
[Ei, PCi, PIDi]
App. Program Codes(C)
Distribute Ei by <PCi, PIDi>
Function Table with PC ranges
Compiler
Accumulate Ei to each function bucket
E2
E1 + E3
Ek
...
Func_1()
Func_2()
Func_n()

Figure 4. Overall Mechanism for Function Level Energy Measurement

As depicted in Figure 4, the function level energy measurement is performed by the interval-based energy approximation. This approximation assumes that the power consumptions at every moment in a period Δ_t are equally P_i which is acquired at a single moment during the period. Thus, the energy consumption E_i during an interval Δ_t is obtained by $E_i = P_i \times \Delta_t$. In fact, the P_i is acquired at the time when the ISR generates the external signal, and the timer interrupt interval forms Δ_t.

Each E_i is also accompanied with the PC at the time when the application process is interrupted. By using both of this PC and the function table generated at compile time, each E_i can be indexed and accumulated to the energy bucket corresponding to the function in the application program codes.

In the course of this indexing process, we have another assumption that the process in attention does not suffer any significant interventions from other processes. This assumption sounds quite unrealistic, but we could get quite consistent experimental data by simply not executing other processes during the measurement.

The above two assumptions necessarily affect the accuracy and consistency of the measurement. At first, the granularity of timer interrupt intervals mainly affects the density of the data acquisitions and eventually does the accuracy of the interval-based approximation. In fact, the regularity of timer interval is related to both of the accuracy and consistency of the measurement. It is true that the drifting of the timer interrupt intervals is rarely avoidable in time-sharing system, and if it goes over a reasonable range, then the accuracy can not be guaranteed. And also, if the irregularity happens, then the results of each measurement become inconsistent and consequently not dependable.

3.3 Timing Analysis

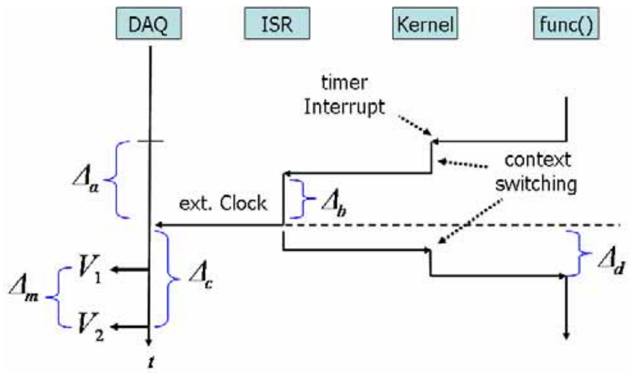

Figure 5. Timing relationship diagram

Another important view point related the accuracy and consistency is the racing situation between the DAQ instrument and the target system. Immediately after the target system releases a external signal through GPIO, the target system continues remained works for finishing ISR and returning to the interrupted process. On the other hand, once the DAQ gets the signal from the target system, the DAQ starts off with scanning the channels. Fundamentally the time needed for a single acquisition should be negligible. But, in case when the DAQ instrument is required to perform multiple acquisitions from multiple channels, then it has to do multiplexing among channels, which takes relatively a while, i.e. Δ_m, compared to the speed of the target system processor. The SC-2345 and NI-DAQ-6251 need around 10 microseconds for acquiring both the V_1 and V_2 by multiplexing [13, 14]. Although these two values are from the different channel, these should be used simultaneously to get the single value of $P = (V_1 / R_{sense}) \times V_2$. Thus, it takes at least the time for multiplexing delay in acquiring a single digital value P at a single moment. Therefore, as shown in Figure

5, there may exist a racing situation between Δ_c and Δ_d. In case of $\Delta_d \geq \Delta_c$, the measurement loses its validation since the DAQ instrument only acquires the power that the ISR uses. But, normally, the situation is $\Delta_d < \Delta_c$ since the Δ_m which holds the most part of Δ_c is usually in the range of microseconds, while the processor takes at most few nanoseconds[2] for Δ_d.

4. Experimental Analysis

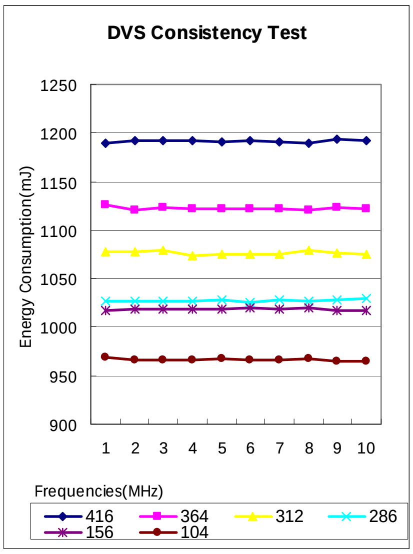

Figure 6. Energy Consumption for same operations with different DVS

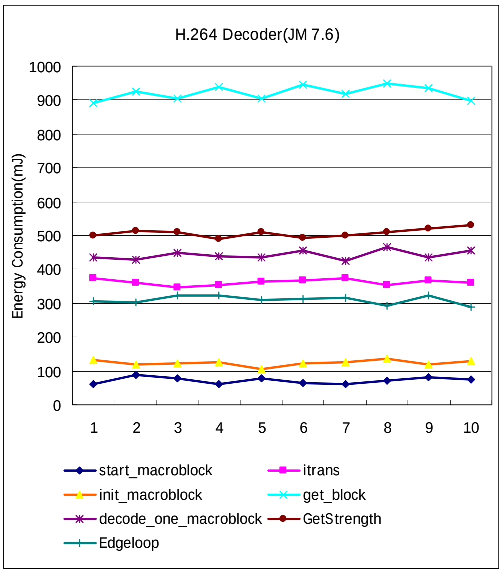

Figure 7. Energy Consumption of major functions in H.264 Decoder (JM 7.6)

Figure 6 shows that the variance among measurements was insignificant when we repeated the same experiment ten times with the same program codes. The standard deviations were 1.03 mJ for 104MHz mode and 1.38mJ for 416MHz mode. This program simply performs a loop of integer multiplications ten million times with different frequencies. In addition, the overhead, $\Delta_a + \Delta_d$, mainly induced by the ISR for timer interrupt was only 0.3% which is relatively insignificant.

[2] The 624 MHz PXA270 is reported to perform 800 MIPS (http://en.wikipedia.org/wiki/Intel_XScale)

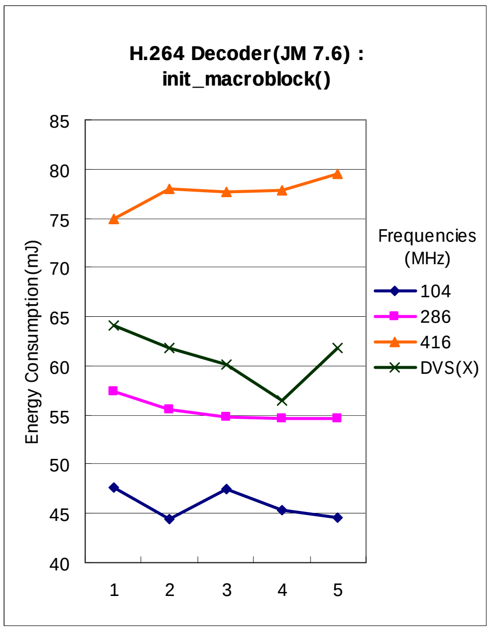

Figure 8. Power Consumption of init_macroblock() with different DVSs

Figure 7 shows our experimental results when we applied our measurements to the JM Ver. 7.4, the open source S/W for H.264 decoder[15, 16]. Seeing the figure, it is consistent and clear that the meaningful functions in the decoder such as get_block(), GetStrength(), and decode_one_block() consume more energy than others like start_macroblock() which only initiates decoding of one block.

Figure 8 shows the results when we applied DVS into the init_macroblock() function so it is executed at 104MHz, 286MHz, and 416MHz modes. It also shows consistently that the same codes can consume more energy when executed with higher frequency, and vice versa. These two experiments on the H.264 decoder imply that it would be effective to elaborate upon the get_block() and change the power modes properly according to the picture types like I, P, B as the previous works [17]. This implication is also meaningful in that the time-triggered approach for the function level energy measurement properly works and gives consistent energy consumption profiles on the DVS-applied program codes running upon the platforms supporting multiple power domains.

5. Conclusions

The provision of software power measurement with reasonable accuracy, consistency and low overhead is indispensable factors for software power engineering. In this paper, we presented the time-triggered mechanism for providing energy consumption profiles in the level of C functions. The similar mechanisms were already introduced at the previous researches such as PowerScope and ePRO[8-11]. Instead, we introduced our efforts to extend these researches to incorporate the platform capabilities such as power domains and DVS, and interpreted these mechanisms as the view of time-triggered approach for better understanding to the relationships among timer interrupts, context switching, DAQ triggering, multi-channel DAQ delay, and etc. From our experiments, we could conclude that the time-triggered approach for the function level energy measurement properly worked with low overheads and produced consistent energy consumption profiles on the DVS-applied program codes running upon the platforms supporting multiple power domains.

Fundamentally, the time-triggered approach depends on the timer interrupts, thus, this approach can be imposed into or integrated with other time-triggered systems such as TMO [3]. The TMO uses clock interrupts for scheduling and dispatching the processor to SpM's and SvM's in TMO objects. If we add our time-triggered energy measurement mechanism into the clock interrupt hander of the TMO engine, then we might expect that the energy consumption behaviors of TMO objects can be analyzed in the level of SpM and SvM. This work is remained as our further research direction.

Acknowledgement: This paper was partially supported by ETRI, ITRC project from MIC, Korea, and Seoul R&BD Program in 2007.

References

1. A. Sinha, N. Ickes, and A.P. Chandrakasan, "Instruction Level and Operating System Profiling for Energy Exposed Software," *IEEE Tr. on VLSI Systems, Vol. 11, No. 6,* Dec. 2003, pp. 1044-1057.
2. T. Mudge, "Power: A First-class Architectural Design Constraint," *Computer, IEEE, Apr. 2001,* pp. 52-58.
3. Kim, K.H., Ishida, M., and Liu, J.Q., "An Efficient Middleware Architecture Supporting Time-Triggered Message-Triggered Objects and an NT-based Implementation", *Proc. ISORC '99,* St. Malo, France, May, 1999, pp.54-63.
4. Intel PXA27x Processor Developer's Kit, User's Guide, Rev. 4.001, Apr. 2004
5. Intel PXA27x Processor Developer's Kit, Schematics, Rev. 4.001, Apr. 2004
6. Intel PXA27x Processor Family Power Requirements, Application Note, 2004
7. Intel PXA27x Processor DVK PMIC(LDO) Card, 2004
8. J. Flinn and M. Satyanarayanan, "Managing Battery Lifetime with Energy-Aware Adaptation," *ACM Tr. on Computer Systems, Vol. 22. No. 2*, May 2004, pp. 137-179.
9. D. Shin, H. Shim, Y. Joo, H.-S. Yun, J. Kim, and N. Chang, "Energy-Monitoring Tool for Low-Power Embedded Programs," *IEEE Design and Test of Computers*, July/August, 2002.
10. W. Baek, Y.-J. Kim, and J. Kim, "ePRO: A Tool for Energy and Performance Profiling for Embedded Applications," in *Proc. of ISOCC'04*, Oct. 2004, pp.372-375.
11. Arun Thomas, "A Measurement Platform for DVS Algorithm Development and Analysis," *TCC 402, Unversity of Virginia,* 2003.04.
12. Qplus, Electronics and Telecommunications Research Institute (http://qplus.or.kr/english/).
13. DAQ M Series User Manual - NI 622x, NI 625x, and NI 628x Devices, National Instruments Corp., 2006. (http://www.ni.com/pdf/manuals/371022g.pdf)
14. SCC-AI Series Isolated Analog Input Modules, User Guide, National Instrument Corp. (http://www.ni.com/pdf/manuals/371066c.pdf).
15. H.264/AVC Reference Software (http://iphome.hhi.de/suehring/tml/).
16. T. Wiegand, G.J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC Video Coding Standard," *IEEE Tr. On Circuits and Systems for Vider Technology, Vol. 13, No. 7,* July 2003, pp. 560-576.
17. K.W. Choi, K. Dantu, W.C. Cheng, and M. Pedram, "Frame-Based Dynamic Voltage and Frequency Scaling for a MPEG Decoder," *Proc. of 2002 IEEE/ACM Int'l conference on Computer-aided design*, San Jose, CA, pp. 732 – 737.