

Máster en Ingeniería de Sistemas y Servicios para la Sociedad de la Información

Trabajo Fin de Máster		
Título		
Autor		VºBº
Tutor		
Ponente		
Tribunal		
Presidente		
Secretario		
Vocal		
Fecha de lectura		
Calificación		

El Secretario:

Universidad Politécnica de Madrid
Escuela Universitaria
de
Ingeniería Técnica de Telecomunicación



TRABAJO FIN DE MÁSTER

**Máster en Ingeniería de Sistemas y Servicios
para la Sociedad de la Información**

**Energy/Power Consumption Model for an Embedded Processor
Board**

**Rong Ren
Julio de 2012**

“世界不会在意你的自尊，人们看的只是你的成就。

在你没有成就以前，切勿过分强调自尊。”

——比尔·盖茨

The world won't care about your self-esteem.

*The world will expect you to accomplish something BEFORE you feel good
about yourself.*

——Bill Gates

Acknowledgements

I have been in this beautiful city almost two years. This is the first time I have left home such a long time; this is the first time I need to take care of everything by myself; this is the first time I have realized the world is so different. There are countless firsts; however, the most important one is, this is the first time I need to be responsible of my promise.

From the very beginning after I arrived Madrid, I was infected by the enthusiasms of Spanish people, no matter in the work or in the life. I am living here each day to the fullest. I have been enjoying the time here with sunshine, objective and progress. But everyone has roses and thorns in the life.

Last September, I started my research topic here. I feel that I am a little fish who is swimming in the boundless ocean. I often felt perplexed to the thousands paper which have the same key words; I often felt disappointed to my practice results; I often felt doubtful to my decision of continuing study. However, now, the time I am writing this thesis, one stage of my four-year is coming to be finished. Look back to the last two years, my heart is filled with joys and appreciations.

I would like to thank my family. They brought me to this world, and are supporting me on everything, in every minute. No matter how I exactly am, they always give me the confidence. My father is a serious guy. Before, I really disliked that his "heavy" topics such as what I had learned from the activities I participated, I should not only have a long-term goal but also several simple short-term goals, and his experience during his forty-year engineer career. But as I grew up, I have realized that his "lessons" are his expectations on me. He has never directly required me to do what he arranged for me, but he gives me the chance to try, to fight and to decide by myself. His words imperceptibly impact on me during my growth. I am quite lucky to have a sweet home. My mother, my grand-parents, I am so appreciate what you have done for me!

I would like to thank my supervisor, Eduardo Juárez Martínez. His suggestions and advices helped me a lot to overcome the difficulties. I cannot forget his word-by-word corrections on my paper; I cannot forget the nights he stayed with me to improve the paper; I cannot forget the time he celebrated Jianguo's and my first papers. His sincerity, meticulousness and diligence have influenced me a lot.

I would like to thank all the professors in the research group: César Sanz Álvaro, Matías Javier Garrido González, Fernando Pescador del Oso and Pedro José Lobo Perea. They are friendly and facetious, which always let me feel that the lab is a big family.

I would like to thank my lab mates: David, Gonzalo, Ernesto, Juanjo, Enrique, Miguel, Oscar and Arnaud. They are always willing to help me with the problem in work or in life. Moreover, they are the guide of this country. From them, I can often find out a charming and different Spain. Although some of them have left the lab, I hope we can keep in touch forever. I am really hoping that one day we can go to China for a travel together.

I would like to thank all my friends: 魏建国, 黄姗, 美娟, 柴亮, 王宪, 杨振, 土豆, 温馨, 钟如意, either in china or here. I like every time stay with them, no matter face-to-face or through the internet. I am really appreciating that they can bear my bad temper, my carelessness, and give me help without hesitation. A good friend is like a cup of wine which become more and more fragrant over time.

Is this true that a person who has many thank to say is a happy guy? I think so, because now I do am. Anyway, this is just a small step in my life, but the sunflower told me that as long as to endeavor forward the sunshine, every day would become pure and beautiful.

Content

CONTENT	I
LIST OF FIGURES.....	V
LIST OF TABLES	VIII
SUMMARY	IX
RESUMEN	X
1 INTRODUCTION	1
1.1 MOTIVATION	2
1.2 OBJECTIVES.....	5
1.3 OUTLINE	6
2 BACKGROUND: ENERGY/POWER CONSUMPTION ESTIMATION METHODS	7
2.1 INTRODUCTION	8
2.2 LOW-LEVEL POWER ESTIMATION MODELS.....	8
2.2.1 <i>Circuit/Transistor-Level Estimation Models</i>	8
2.2.2 <i>Gate-Level Estimation Models</i>	10
2.2.3 <i>RT-Level Estimation Models</i>	12
2.2.4 <i>Architectural-Level Estimation Models</i>	14
2.3 HIGH-LEVEL POWER ESTIMATION MODELS	16
2.3.1 <i>Instruction-Level Estimation Models</i>	17
2.3.2 <i>Function-Level Estimation Models</i>	19
2.3.3 <i>Component-Level Power Estimation Models</i>	21
2.4 DISCUSSION	25
2.5 CONCLUSION.....	27
3 METHODOLOGY.....	29
3.1 INTRODUCTION	30
3.2 GENERAL IDEA	30
3.2.1 <i>Modeling Flow</i>	30
3.2.2 <i>Performance Monitor Counters</i>	31
3.2.3 <i>Components Classification</i>	32

3.2.3.1	Components Classification	32
3.2.3.2	Energy-related Events	33
3.3	MODELING METHODOLOGY	35
3.3.1	<i>Mathematics Knowledge</i>	35
3.3.1.1	Spearman Rank Correlation Coefficient	35
3.3.1.2	Linear Regression Methods	36
3.3.1.3	Principal Components Analysis	37
3.3.2	<i>Methodology</i>	37
3.3.2.1	PMC-filter	38
3.3.2.2	K-fold Cross-validation	41
3.4	DISCUSSION	42
3.5	CONCLUSION	46
4	IMPLEMENTATION.....	49
4.1	INTRODUCTION	50
4.2	EXPERIMENTS ENVIRONMENT	50
4.2.1	<i>Hardware Platform</i>	50
4.2.1.1	Cortex-A8 CPU	51
4.2.1.2	Storage Hierarchy	51
4.2.2	<i>Platform PMCs</i>	53
4.2.2.1	Platform Available PMCs	53
4.2.2.2	PMCs Interface Implementation	56
4.2.3	<i>Components</i>	69
4.3	MODELING	69
4.3.1	<i>Benchmarks</i>	70
4.3.2	<i>Measurement</i>	70
4.4	CONCLUSION	70
5	VALIDATION AND EVALUATION	73
5.1	INTRODUCTION	74
5.2	PROGRESS OF MODEL	74
5.2.1	<i>PMC Accuracy</i>	75
5.2.2	<i>First Model</i>	77
5.2.3	<i>Second Model</i>	79
5.2.3.1	Random Selection	80
5.2.3.2	L2-Miss-Rate Selection	81

5.2.3.3	IPC Selection.....	82
5.2.4	<i>Third Model</i>	84
5.2.5	<i>Fourth Model</i>	88
5.2.6	<i>Final Model</i>	91
5.3	MODEL LIMITATION AND FUTURE WORK.....	93
5.3.1	<i>Model Limitation</i>	93
5.3.2	<i>Future Work</i>	94
5.4	CONCLUSION.....	95
6	CONCLUSION	97
7	REFERENCE	101

List of Figures

Figure 1-1 Energy Analysis Framework.....	4
Figure 1-2 Simulation-based VS. Traditional Power Profiling Approach ⁶	4
Figure 2-1 A Schemes to Measure the Average Current Drawn	9
Figure 2-2 A Gate-level Example ¹⁵	11
Figure 2-3 An Architecture Power Estimation Methodology ²⁸	15
Figure 2-4 Principle Instruction-level Power Estimation ⁶	17
Figure 2-5 Processor Modeling Methodology ⁴⁹	20
Figure 2-6 Models Based on Datasheet	21
Figure 2-7 General Structure of High-level Modeling Methodology	22
Figure 2-8 Simplest System-Level Power Model	23
Figure 2-9 Power Estimation in Different Design Level	25
Figure 3-1 Modeling Flow	30
Figure 3-2 High-Level Overview of the Embedded System Architecture.....	32
Figure 3-3 General Core Architecture.....	33
Figure 3-4 Shared Variance.....	39
Figure 3-5 Example of the K-fold Cross Validation Method	41
Figure 3-6 Comparison of TLB Instruction Miss	42
Figure 3-7 Comparison of the Cache Hit Rate.....	44
Figure 3-8 Comparison of the Branch Miss Prediction Rate.....	45

Figure 4-1 BeagleBoard with High-Level Block Diagram.....	50
Figure 4-2 OMAP3530 Block Diagram ⁸³	52
Figure 4-3 Recommended CP15 Performance Monitors ⁷²	54
Figure 4-4 PAPI Architecture	57
Figure 4-5 Enable Kernel Support for PMCs	60
Figure 4-6 Enable Debugging Hardware	60
Figure 4-7 Procedure of Using PAPI	61
Figure 4-8 “Switch-case” Pattern to Execute a Benchmark.....	64
Figure 4-9 Identify all the Object Files of Benchmarks	65
Figure 4-10 Add Compile Targets.....	65
Figure 4-11 Indicate Source Code Path	66
Figure 4-12 Compiler Options.....	67
Figure 5-1 PMCs Variation of Do_Queen	76
Figure 5-2 PMCs Variation of Do_READ	76
Figure 5-3 PMCs Variation of MM_MISS.....	76
Figure 5-4 Relationship between Energy and TOT_INC	78
Figure 5-5 Proportional Error of the First Model	78
Figure 5-6 Real Energy Consumption VS. First Model Estimation	79
Figure 5-7 Real Energy Consumption VS. Second Model Estimation(a).....	81
Figure 5-8 Real Energy Consumption VS. Second Model Estimation (b).....	82

Figure 5-9 Real Energy Consumption VS. Second Model Estimation (c).....	83
Figure 5-10 Relationship between Energy and PMC.....	85
Figure 5-11 Real Energy Consumption VS. Third Model Estimation	87
Figure 5-12 Measurement Energy Comparison.....	88
Figure 5-13 Estimation Energy Comparison.....	88
Figure 5-14 Correlation Coefficients in Different Cases	89
Figure 5-15 Model Comparison	90
Figure 5-16 Average Relative Error of G1	92
Figure 5-17 Average Relative Error of G3	93

List of Tables

Table 3-1 Correlation Coefficient Explanation	38
Table 4-1 ARM Core Key Features	51
Table 4-2 Supports of PMC Drivers on ARM Family (Last Update 6 th 2011).....	59
Table 4-3 Events Measured by setting the environment variable PAPI_EVENT	68
Table 5-1 The Correlation Coefficients between PMCs and energy	85
Table 5-2 Relative Errors in Different Cases	91

Summary

This dissertation, whose research has been conducted at the Group of Electronic and Microelectronic Design (GDEM) within the framework of the project Power Consumption Control in Multimedia Terminals (PCCMUTE), focuses on the development of an energy estimation model for the battery-powered embedded processor board.

The main objectives and contributions of the work are summarized as follows:

- A model is proposed to obtain the accurate energy estimation results based on the linear correlation between the performance monitoring counters (PMCs) and energy consumption.
- Considering the uniqueness of the appropriate PMCs for each different system, the modeling methodology is improved to obtain stable accuracies with slight variations among multiple scenarios and to be repeatable in other systems. It includes two steps: the former, the PMC-filter, to identify the most proper set among the available PMCs of a system and the latter, the k-fold cross validation method, to avoid the bias during the model training stage.
- The methodology is implemented on a commercial embedded board running the 2.6.34 Linux kernel and the PAPI, a cross-platform interface to configure and access PMCs. The results show that the methodology is able to keep a good stability in different scenarios and provide robust estimation results with the average relative error being less than 5%.

Resumen

Este trabajo fin de máster, cuya investigación se ha desarrollado en el Grupo de Diseño Electrónico y Microelectrónico (GDEM) en el marco del proyecto PccMuTe, se centra en el desarrollo de un modelo de estimación de energía para un sistema empotrado alimentado por batería.

Los objetivos principales y las contribuciones de esta tesis se resumen como sigue:

- Se propone un modelo para obtener estimaciones precisas del consumo de energía de un sistema empotrado. El modelo se basa en la correlación lineal entre los valores de los contadores de prestaciones y el consumo de energía.
- Considerando la particularidad de los contadores de prestaciones en cada sistema, la metodología de modelado se ha mejorado para obtener precisiones estables, con ligeras variaciones entre escenarios múltiples y para replicar los resultados en diferentes sistemas. La metodología incluye dos etapas: la primera, filtrado-PMC, que consiste en identificar el conjunto más apropiado de contadores de prestaciones de entre los disponibles en un sistema y la segunda, el método de validación cruzada de K iteraciones, cuyo fin es evitar los sesgos durante la fase de entrenamiento.
- La metodología se implementa en un sistema empotrado que ejecuta el kernel 2.6.34 de Linux y PAPI, un interfaz multiplataforma para configurar y acceder a los contadores. Los resultados muestran que esta metodología consigue una buena estabilidad en diferentes escenarios y proporciona unos resultados robustos de estimación con un error medio relativo inferior al 5%.

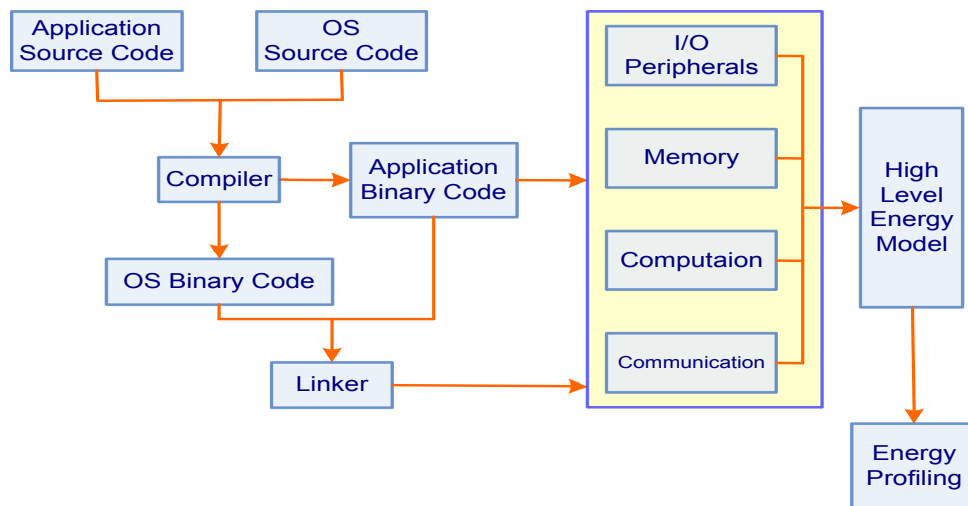
1 Introduction

1.1 Motivation

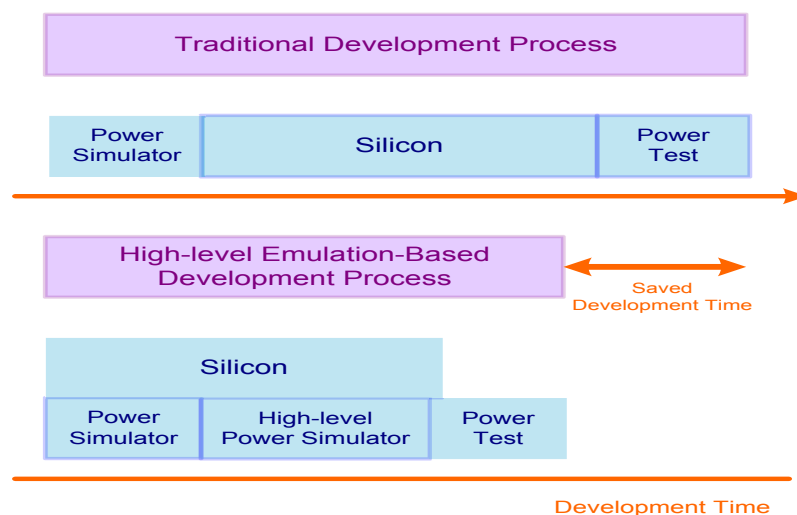
Nowadays, battery-powered consumer electronics devices like smart phones, media players, PDAs and tablets have become more and more indispensable in people's daily life. However, the design of those devices still faces several problems: computing capability, memory constraints and, especially, the limited battery lifetime. Battery lifetime is improved very slowly comparing to the continuously increasing demands for new functionalities such as games, network services and multiplayer. These functionalities usually cause the intensive computations, heavy network transmissions and the always-on display, which are inversely proportional to the battery lifetime. Therefore, low-power design has become a hotspot. It includes several techniques such as the specialized circuit design, the architecture design, the power-aware operating system (OS) scheduler and the power management (PM) policy, all of which work at different levels to address energy/power consumption issues. Low-power design concentrates on increasing the energy efficiency, unfortunately, these techniques alone are not sufficient. High-level strategies, such as energy-aware OS schedulers and PM policies, are becoming increasingly important to maximize battery lifetime. PM policies in mainline OS assume that the energy saving can be achieved by running at the low chip speed. Therefore, one PM policy in work [1] consider the workload completion in low-power mode by scaling chip voltage and frequency. To achieve a balance between energy consumption and performance, PM policies usually, either, run the workload at the maximum performance setting within the longest time on low-power mode, or alternatively, under deadline constraints, try to save more energy by running at the lowest performance setting. However, Snowdon et al. pointed out that such a simple approach leaded to sub-optimal results on actual hardware in [2] and [3]. Furthermore, they presented that the PM policy which considered on energy characteristics of workloads was able to achieve the maximum energy efficiency. Therefore, an accurate estimation model for profiling the on-line energy consumption at a fine granularity is needed. This kind of models, as the foundation of system PM policies, can help PM policies to improve and facilitate the energy efficiency optimization on battery-powered platforms⁴.

Traditional energy profiling is usually based on the direct measurements. Measurement is limited to the entire chip due to chip integration and packaging. Moreover, the final entire chip is not available at early design stages. Therefore, the energy profiling cannot be obtained until the late design stage. In order to avoid this weak point, emulation-based power profiling approaches have been employed to consider energy issue at the beginning of the design stage. These approaches try to extract the physical behavior information during the application executions to make an estimation model based on the

system resources' utilizations and energy consumptions. They can work at the low-level or high-level depending on how they obtain the predictions of power consumption. The low-level approaches exploit the main strategies to simulate the activities from the power-related hardware operation units of the microprocessor architecture. This solution suffers the lack of details on the internal structure of the system and the quite long simulation time. To overcome these problems, high-level abstract models have been proposed. These models concentrate on the events happened on the higher level of the system architecture. For example, to construct an estimation model for the processor, on one hand, they will first measure the average current drawn by the processor during the application execution; on the other hand, they monitor some key energy-related events triggered at this period. The model is constructed by relating these events with the measured energy. The energy-related events can also be divided into different levels such as instruction level, function level or system level. Usually, finding the relationship among events and energy is an off-line procedure, once a fine energy estimation model is constructed, it can be simply applied to the energy-optimizing strategies such as the PM policy and then continuously makes the on-line estimations. A PM unit can be considered as an energy manager. It tries to optimize the energy usage to extend the battery lifetime. Meanwhile, this energy estimation profiling will help it to make a power-aware decision. The Figure 1-1 shows a typical example to attach the software designs together with the hardware simulator platform to figure out an energy profiling. The left part in the figure is the steps involved to use the simulator. The compiler generates the objects codes from the operating system and the applications, then these object codes are linked and work as the stimuli. The right part is the simulation model, which considers the hardware of the system as several components, and then the high-level simulator models each component. Note that the modeling methodology could be various. In this work, they obtained the power models from the data-sheet except for the processor which was modeled by the instruction-level energy models. After a reasonably accurate model was done, the energy consumption profiling of the tasks ran on the system can be expressed as a sum of the energy contribution from each component.

Figure 1-1 Energy Analysis Framework⁵

Based on the energy profiling from the software aspect, high-level energy estimation can help products to decrease their time to market⁶. Energy estimation is needed at different stages in the design process. Ideally, designer would like to estimate the energy of the design very early, such as when only a high-level (behavioral) description of the design is available. In this stage, when the design is still sufficiently flexible, energy information can be delivered to the designers before available silicon by utilizing an FPGA prototyping platform (Figure 1-2) to make energy tests. Therefore, designers do not need to wait until the whole design flow is finished to solve the energy problem, they can make the major changes rather cheaply.

Figure 1-2 Simulation-based VS. Traditional Power Profiling Approach⁶

Although there is much work that has been done to have a good energy profiling, energy estimation is still a very important topic, especially on battery-powered embedded

systems. This thesis concentrates on this topic to provide an accurate energy consumption model of the battery-powered devices. The model is based on the PMCs (Performance Monitoring Counters) which are realized as the hardware registers attached with the processor to measure various programmable events occurring in the processor. It can give a detection of those events which influence the power consumption, and supply the power-aware strategies or individual user the key issues to maximize energy efficiency. Meanwhile, this energy estimation model is able to keep the stability on various cases. For example, if an application has a low performance due to its high cache misses rate and a frequent data transfer between cache and the main memory, the model can detect their bottlenecks and identify its high energy consumption. This model is also generic and portable. It obtains its required information from a high-extracted level to mask the hardware differences. Therefore it can be attached to various platforms with few modifications.

1.2 Objectives

For getting the energy/power consumption profiling, the estimation model should be able to provide a correlation between run-time resource usage and energy consumption to help the energy-related strategies to improve the energy efficiency optimization.

The main goal of this thesis is an exploration of a methodology to build a platform-independent high-level model. It estimates the energy consumption from the analysis of the on-line system energy-consuming behaviors. An ideal estimation model should be suitable for on-line using and should meet several rules described below:

- Non-intrusive and low-overhead: This model should not require too much intrusive hardware adjustments and software overhead when collect the model input parameters. It must give a quick response for real-time optimizations. A long time-taken model, in contrast, will delay the energy-optimizing strategies to make decisions of tasks arrangements or voltage/frequency scaling.
- Easy to develop and use: The model should be simple. This means a model can keep low complexity while provide enough accurate predictions, thus it can be used on different systems without too many modifications and restrict the model's own enable energy consumption in a small limitation.

- Reasonably accurate: The model must be sufficiently accurate to enable energy-efficiency optimization. A certain error range is accepted by considering the model's own overhead.
- Generic and portable: This model should have highly enough abstract level to other systems. It should work for different platforms of various combinations of processor families, memory hierarchies and components within few modifications. Generating a model for a new system, it requires neither exhaustive details nor extensive design exploration.

Considering the requirements of easy usability, good scalability and high speed, an estimation method should be a high-level abstraction to avoid many platform details. In this thesis, a PMC-based approach is focused. The PMC-based model may be built from a specific hardware platform, but the methodology can be used on any PMC available systems.

1.3 Outline

This thesis continues with four chapters:

- Second chapter describes various energy/power estimation methods based on different levels. Their main methodologies are introduced and a comparison to identify the most suitable method for energy optimization strategies is given later.
- Third chapter describes the methodology introduced in this thesis in detail. It includes the basic thought of modeling associated with the PMCs, the needed mathematics knowledge and the improvement of the modeling method to give a good predictability and generalization.
- Forth chapter gives an implementation which uses the third-part existing interface to measure the according set of PMCs of a particular platform. It also introduces the usage of the according functions of the interface to configure the PMCs.
- The fifth chapter describes the progress of the modeling methodology. It begins with the simplest method which has the unacceptable estimation error, and follows with how to derive the methodology to improve the accuracy. The limitation of the methodology and the future work are also discussed in this chapter.

2 Background: Energy/Power Consumption Estimation Methods

2.1 Introduction

Accurate models of energy/power consumption estimation have been a crucial research field of many schemes to improve the energy efficiency. These models focus on the initial design of individual component or the whole system to provide the possibility to identify the key influences of the energy consumption. In this chapter, contributions and a discussion on previously proposed approaches of power modeling in different level are summarized.

An energy profiling is the representation of the system's energy consumption. Because of the general usage, a profiling may highlight some key factors of energy consumption of the system while abstract away some others. Energy profiling can be based on the real measurements or the estimation models. In this dissertation, the modeling method is focused on. Modeling methods of energy estimation involve two important issues during the model constructing: complexity and accuracy, both of which are determined by the abstraction layer on which the energy models are set up. These two issues result in two main categories of the models: low-level and high-level energy estimation models. Low-level models which estimate the energy and power from the detailed design information include circuit level, gate level, register transfer level (RT level) and architectural level. High-level models deal with the instructions, functional units or components to profile the system energy from software point in order to avoid the hardware details.

2.2 Low-Level Power Estimation Models

2.2.1 Circuit/Transistor-Level Estimation Models

A simple and straight method of average power estimation is to simulate the circuit behaviors to obtain the power supply voltage and current waveforms, from which the average power can be computed. Several circuit simulation based approaches have been proposed in work [7] by Kang. In fact, the model in this level is most used for VLSI (Very Large Scale Integrated Circuits) design and the technology choice. The basic scheme of this approach is shown in Figure 2-1. A parallel RC sub-circuit is inserted into a VLSI circuit without any interference of the original circuit. The sub-circuit measures the current drawn from the voltage source and computes the average power as equation 2-1:

$$P(t) = u \cdot i(t) = u \cdot \frac{dq}{dt} = u \cdot C_x \frac{du}{dt} = \frac{C_x V_x(t)}{t} \quad (2-1)$$

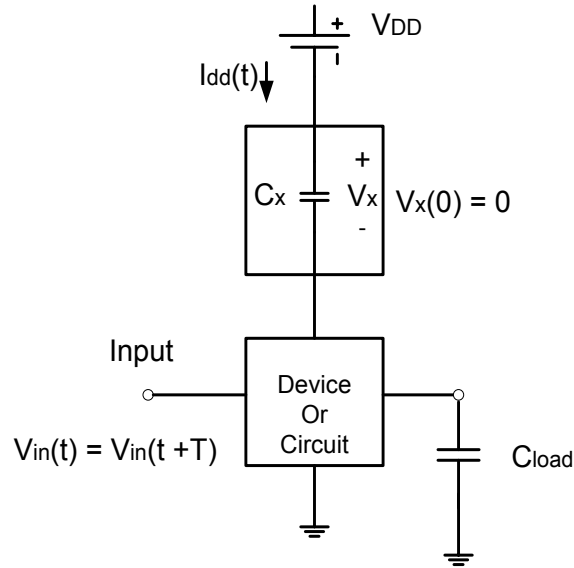


Figure 2-1 A Schemes to Measure the Average Current Drawn

Yacoub et al in [8] proposed a similar circuit simulation technique as the basic one to measure the average current in complementary circuit structures. Their difference was in the complementary circuit structures, where current was not permitted to flow during a steady state of a circuit because there was no path between power and ground in this equilibrium state.

However, these early methods are not suitable to the complex circuit design with higher integration density, smaller device geometry, larger chip size and faster clock frequency caused by the rapid development of the CMOS technology. Subsequent researchers in [9],[10],[11],[12] and [13] proposed the probabilistic approach instead of directly simulating a circuit. Probabilistic approaches compute and propagate the probability for a node to change its logic state. These probability methods usually include two kinds of definition:

- Signal Probability: The average fraction of clock cycles in which the steady state value of node x is logic high;
- Transition Probability: The average fraction of clock cycles in which the value of node x at the end of the cycle is different from its initial value.

Followed with the probabilistic logic, a novel simulator, PowerMil¹⁴, was proposed to build a transistor-level power consumption model by simulating the current and power behavior in modern deep-submicron VLSI circuits. PowerMil provides piecewise linear transistor model to capture transistor characteristics from a table to greatly shorten the

evaluation overhead. However, it is extremely complex to represent a circuit/transistor-level models due to all details of the design flow, layout, routing and parameter extraction. Probabilistic-based models can achieve very good accuracy while take unbearable long time to simulate more than one million transistors, thus, they are not suitable for the real-time demand because of both the high complexity and expense. In the other side, they also require the user to specify complete information about the input patterns.

Although the circuit details can be obtained during the logic synthesis at the early design stage, they may invalid to the transformations or design decisions made at later implementation stages, hence the model in this level is hard to be used in the early design stage. This technique is accurate and general to estimate the power of any circuit with various technology, design style, functionality and architecture. However, the estimation model is driven by the complete and specific input information which causes huge number of input pattern and heavy computation, thus it is impossible to use for the large circuits.

2.2.2 Gate-Level Estimation Models

Gate-level estimation methods aim to describe the different gate circuit behaviors during the system runs. The advantage of such methods is that the simulations are driven by events and take place in the discrete time domain. This means that the model is enable to provide an estimation of the switching activities of the basic logic blocks without actually simulating the circuit with a large number of test patterns. Compared with the circuit-level models, gate-level estimation method is faster, can handle larger circuits, and, the most important difference is to be applied before all the circuits details are available. There are two major types of approaches, dynamic and static, used in gate-level power estimation. Dynamic approaches simulate the circuit based on the input sequence with the system representativeness. Their main shortcomings are their very slow estimation speed and highly dependent results on the simulated sequence. Usually, the required the required number of simulated sequence is high to produce a valid power estimate. To address this problem, Monte Carlo simulation techniques are proposed. These techniques use an input model based on a Markov process to generate the input stream for simulation. The main difficulty is that it is not clear how the input stream can be efficiently generated when the circuit inputs exhibit complex correlations. The static techniques are implemented based on the statistical information abstract from the input sequences to estimate the internal switching activity of the circuit.

Gate-level model assumes that the circuit consists of logic gates and latches, as shown in Figure 2-2. In other words, it consists of latches driven by a common clock and

combinational logic blocks whose inputs (outputs) are latch outputs (inputs). Therefore, the average power consumption of the circuit can be divided into two parts: the power consumed by the latches and that consumed by the combinational logic blocks.

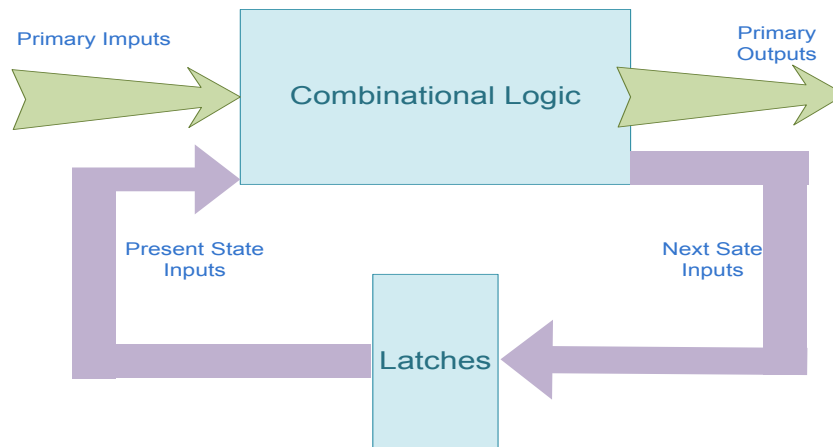


Figure 2-2 A Gate-level Example¹⁵

Gate-level energy estimation needs the gate switching activities, thus pattern independent approaches are well-suited for this kind of power estimation. Pattern independent methods provide an estimate of the average switching activity without actually simulating the circuit with a large number of test patterns¹⁵. Gate-level energy estimation reduce the computational complexity compare with the circuit-level models, meanwhile, it does not loss too much accuracy. Several gate-level energy estimation approaches with good accuracy and high efficiency have been proposed. They are classified into dynamic and static two categories.

The dynamic approaches explicitly simulated the circuit based on the typical input sequences. Their main shortcoming was the very slow simulated speed. Moreover, their results were highly dependent on the simulated vectors. The required number of simulated vectors was usually high to produce meaningful energy estimation. Burch et al. proposed the Monte Carlo approach to solve the vector problem in work [16] by Burch et al. Their approach used the probabilities to compute the power consumption by directly monitoring the total power during the random simulation. The input vector streams for simulation were generated by Markov process. The Monte Carlo approach faced to a main difficulty of clearly showing how the input vectors could be efficiently generated when the circuit inputs exhibited complex correlations.

The static approaches in [17],[18] and [19] relied on statistical information, for instance, the mean activities of the input signals and their correlations. The concept of

probability waveforms, transition density and the enumeration approach based on symbolic simulation were proposed. An example of a gate-level statistical power estimation approach was presented by Chou et al in work [20]. They took the spatial and temporal correlations of logic signals into consideration. Their proposed the states partitions approach to reduce the computation time. Ding et al. in work [21] proposed a similar approach based on tagged (probability) waveforms. The tagged waveforms were obtained by two issues. One is the partition of the logic waveform space of a circuit node, according to the initial and final values of each waveform. The other is to compact all logic waveforms in each partition by a single tagged waveform. Then, the tagged waveform can be used to calculate the switching activity of the circuit node. Note that only tagged waveforms at the circuit inputs were exactly computed, the remaining nodes were computed using a compositional scheme that propagated the tagged waveforms from circuit inputs to circuit outputs.

In practice, the model required input probabilities could be directly provided to eliminate the need for a large set of specific input patterns. The results of the model analysis will depend on the supplied probabilities. Thus, to some extent the process is still pattern-dependent and the user must supply information about the typical behavior at the circuit inputs, in terms of probabilities.

2.2.3 RT-Level Estimation Models

A register transfer level data path is consist with the interconnections of the pre-designed functional blocks such as adders, subtractors, multiplexers, comparators and registers (the control units, buses, memories and clock trees are excluded from this data path category). A register transfer level description captures the application specific integrated circuit (ASIC) behaviors at the physical levels. The simulation approach in RT-level is try to functionally estimate and collect the input sequence including blocks or network on interconnections such as adders, registers, multiplexers and the netlists. The power properties of a block could be traced by the application under the controlled operating conditions of an individual block through its input statistics.

Most RT-level power estimation approaches use the capacitance models with activity profiles of data or control signals, which are signal probability or switching activity. For a node to switch state and consume dynamic power, its current state must differ from its previous one, which meant that if the previous state was zero and the node was now directly set to one. Thus, signal probability (SP) is the fraction of time a signal is logic high. This probability of this occurring was referred to the switching activity (SA), thus a simple dynamic power dissipation model of a gate could be calculated by multiplying the SA, the capacitive

load (C), the clock frequency (f) and the square of the supply voltage for each node as the formula 2-2:

$$P_{dynamic} = \alpha \cdot C_L \cdot V_{dd}^2 \cdot f_{clk} = \alpha \cdot C_L \cdot \Delta V \cdot V_{dd} \cdot f_{clk} \quad (2-2)$$

Here, C_L is the load capacitance, f_{clk} is the clock frequency, V_{dd} is the supply voltage, ΔV is the swing voltage of the node, and α is the node '0 \rightarrow 1' transition activity factor which is defined between 0 and 1.

The switched capacitance and switching probability of each functional module are modeled by formulas that are a function of the module's inputs probabilities. These formulas are computed beforehand for each model using the polynomial simulation scheme, and stored in the model library. The switched capacitance for each instance of a module in the circuit can then be efficiently evaluated for its specific input probabilities. The switching probabilities at the outputs of each model can be computed in a similar manner, thus providing a means of propagating the switching probabilities through the circuit described at the RT level.

Najm gave a good list of several RTL power estimation approaches in work [18]. The main idea is based on the probabilistic and statistical techniques. These techniques are applicable only to combinational circuits. They require the user to specify information on the activity at the latch outputs. Some estimation tools, such as Primepower²² performed power estimation both at the structural RTL and gate levels. Others like HSPICE²² and SPICE²³ can also be used to do the low-level simulate. These tools give very close results compared to the actual power consumptions. However, research studies and these tools show the limitations of RT-level power estimation: long-lasting time and required RTL design details, which are extremely difficult to get. Another tool Hotspot²⁴ although considered the power estimation together with the thermal, it still based the design data from a gate-level netlist and the activity factors from a structural RTL model, thus it cannot avoid those disadvantages mentioned before. Those disadvantages make them not proper for early-stage design explorations. Besides, they cannot be easily updated for the future technology.

RT-level estimation methods cannot totally avoid the pattern-dependence problem in circuit-level or gate-level since some of the inputs provided by the users are typical behaviors. These inputs are usually based on the probability, which is defined as the average fraction of time that a signal stay in high status and the density, the average number of transitions during each second. Comparatively speaking, this information is much more easily obtained by the designers than specific input patterns are. For example, designers can

estimate the average input switching frequencies through the test streams or assume the frequency by the known clock frequency. For the real implementation, the statistical approach can be constructed by the existed simulation tools and libraries, which mainly differs with the dynamic one. In the other word, dynamic modeling approach, which based on the probabilistic technique, requires the specific simulation models.

2.2.4 Architectural-Level Estimation Models

Architects typically make decisions in the planning phase before the design has begun; therefore, those tools such as PowerMill²⁵ and QuickPower²⁵ which operate on the circuit level and need complete HDL design are not helpful for making architectural decisions. Considering with the insufficient usage of the previous estimation methods on energy efficiency optimizing, architectural and software estimation methods, as the addition to the low-level circuit estimation approaches, have become more important. However, this method still suffers from the lack of the efficient simulator tool that analyzes and quantifies the power ramification of various architectures. Such a tool requires to trade-off the low-level details and accuracy against the simulation speed and portability²⁶.

There are three components that define the important contributions to power consumption in CMOS²⁷ technology as formula 2-3:

$$P = ACV^2f + \tau AVI_{short} + VI_{leak} \quad (2-3)$$

The first component is the dynamic power consumption which depends on the capacitive load charging and discharging of each gate, in which factor “A” means the activity of the gates, factor “C” means the total capacitance seen by the gate outputs, factors “V” and “f” are the supply voltage and the system frequency, respectively. The second term stands for the power dissipated on the short-circuit. The factor “ I_{short} ” means the short-circuit current that flows between the supply voltage and ground when the output of a CMOS logic gate switches in the τ period. The third part is the power consumed by the leakage current depends on the number of gates and threshold voltages. Therefore, to estimate by a cycle-accurate simulator is necessary to focus on the activities on the gate level (the first two terms) and the number of the gate of each micro-architecture (the third term). Based on this reason, various architectural power simulators combine with the lower level power consumption models to get the circuit activities and capacitive models of activated components during each cycle²⁸.

Figure 2-3 shows an architecture-level estimation model for power by extending a cycle simulator. Before model constructing, primary technology parameters, such as supply

voltage, threshold voltage, capacitance per area and the sheet resistances of the interconnectors, are needed to well prepare. Furthermore, the micro-architecture specification should also be determined. Usually, the micro-architecture functional blocks are designed as full-custom, thus the design styles influence the power consumption and require the different power models. Therefore, architectural-level model usually divides the chip into several regular functional blocks such as memory, datapath, interconnectors and clock distribution tree, thus the number of the activity patterns is reduced. The model of each block is constructed with the consideration of its specific determination and features only once and kept in the look-up table. Finally, the estimation model profile the energy of each micro-architectural block by combining the power model and the execution statistics from the cycle simulator.

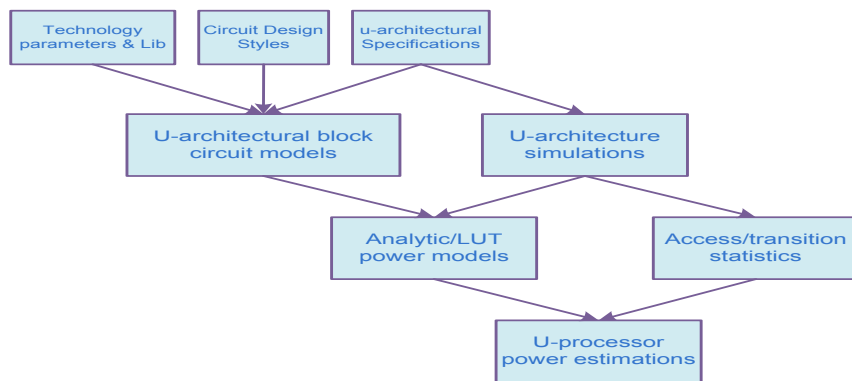


Figure 2-3 An Architecture Power Estimation Methodology²⁸

Architectural-level power models have the same idea with the RTL/gate-level ones, but it uses the instructions and events instead of circuit signals to drive the model, therefore, the number of patterns decreases. The event activity (EA) is defined to determine the probability of the event's occurrence. Most research in architectural-level power estimation were based on empirical methods that constructed the power consumption models based on the measurements from existed implementations in [29],[30],[31],[32],[33],[34]. Measurement-based approaches for estimating the power consumption could be divided into three sub-categories:

The first one is a fixed-activity macro-modeling approach. This approach assumed that the inputs would not affect the switching activities of a hardware block^{35,36,37}. To increase the accuracy, the second approach, an activity-sensitive empirical energy model was developed. These schemes were based on predictable input signal statistics. An example was the method proposed by Landman et al. in work [38]. They introduced a technique to estimate the power of individual architectural blocks by datapath controller and interconnect analysis. Controllers were often the finite state machines (FSM), their stages were provided

by the combinational logic and depended on primary outputs and present state. Controllers guided the sequence of operations to be executed by the datapath, initiate memory accesses and coordinate data transfers over interconnect, and thus the power estimation can be envisioned only by the FSM behavior. Interconnect analysis addresses the problem of estimating the power consumed in charging and discharging interconnected capacitance. The estimation considers the interconnect activity, physical capacitance, wire length and the composite blocks. In case of a microprocessor driven by instructions it is impossible to know prior the typical input patterns to the individual modules. Considering this weak point, the third empirical approach is transition-sensitive energy models which were more concentrated on input transitions rather than input statistics³⁹. This approach does not need any knowledge of module functionality and with the accuracy controlled by the designer.

Models in this abstraction level give a better balance between the estimation accuracy and estimation speed than other low-level models because it provides each functional unit an energy model and used a table to record the power consumed by each input transition. The tables consume substantial time to contain the switch capacitance of each input transition and lookup due to their exponential growth of the size. Therefore, how to design these tables was the challenge. For example, to reduce the size of the table, closely related input transitions and energy patterns could be considered as a cluster. Simulators such as SoftWatt⁴⁰ and Wattch⁴¹ utilized a simple fixed-activity model for the functional unit. These simulators only trace the number of accesses to a specific component and utilized an average capacity value to estimate the power consumption without accommodating the power variance of the access sequences. This approach is not a transition-sensitive approach. Therefore, it allows the designers to aware the power constraint at the early design stage.

2.3 High-Level Power Estimation Models

As the discussion before, low-level estimation models or simulators are suitable for the hardware designers to make a decision on combinational circuit technology choice. Based on the deliberate choices, the greatest benefits are derived by trying to assess early in the design process the merits of the potential implementation. However, low-level estimation methods are not suitable for the software or OS designers for whom it is difficult to obtain the details of the hardware. Thus the high-level estimation models are proposed.

2.3.1 Instruction-Level Estimation Models

The instruction level energy model for individual processors is first proposed by Tiwari et al in work [42]. Its basic idea is try to get the different current drawn by the processor during executes distinct instructions or distinct instruction sequences (Figure 2-4). A base cost of each individual instruction plus the inter-instruction overhead is used as the model of the energy. The huge different combinations of instructions cause to the various inter-instruction effects, which become the disadvantage of this approach. To simplify the combinations complexity, Tiwari et al proposed an experimental approach to empirically determine the base and the inter-instructions overhead cost. In this approach, they used several programs containing an infinite loop consisting of several instances of the given instruction or instruction sequences and then measured the average current drawn by the processor core during the execution of this loop. Tiwari's continued with this topic and his following researches [43],[44] and [45] showed that the complex processor in general have less variation in instruction costs compared to smaller DSP processors because of the dominance of the overhead costs, so the instruction options reduction was able to achieved by ignoring those instructions which are not so important for the power consumptions.

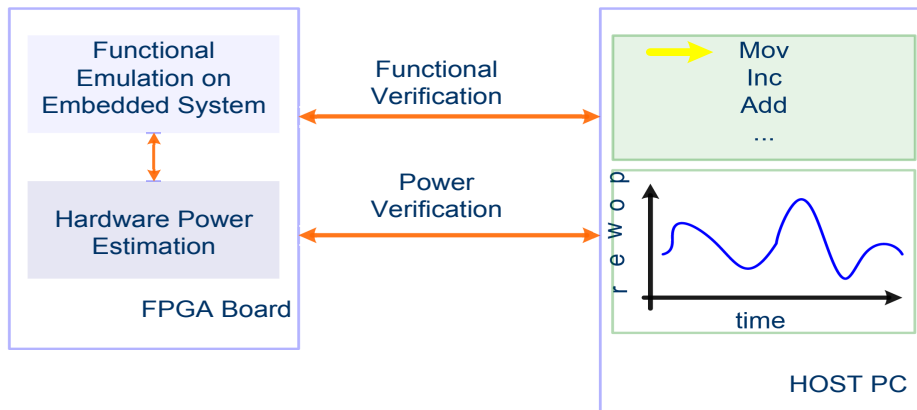


Figure 2-4 Principle Instruction-level Power Estimation⁶

This method uses a table to record the average power of each instruction. An accurate model not only needs the pre-estimated power of each instruction but also the effects of inter-instructions. Unfortunately, such a table requires $O(n^2)$ space which is quite high spatial and temporal expense. A simple solution proposed by Lee et al. in work [46] was to classify instructions into categories based on their functionalities and the addressing mode. In their work, they abstracted six instruction classes including loading immediate data to a register, transferring memory data to registers, moving data between registers and operating in ALU. However, this simply method will meet problem when the instruction set has various addressing modes and high parallelism. Klass et al. in work [47] proposed an approach to

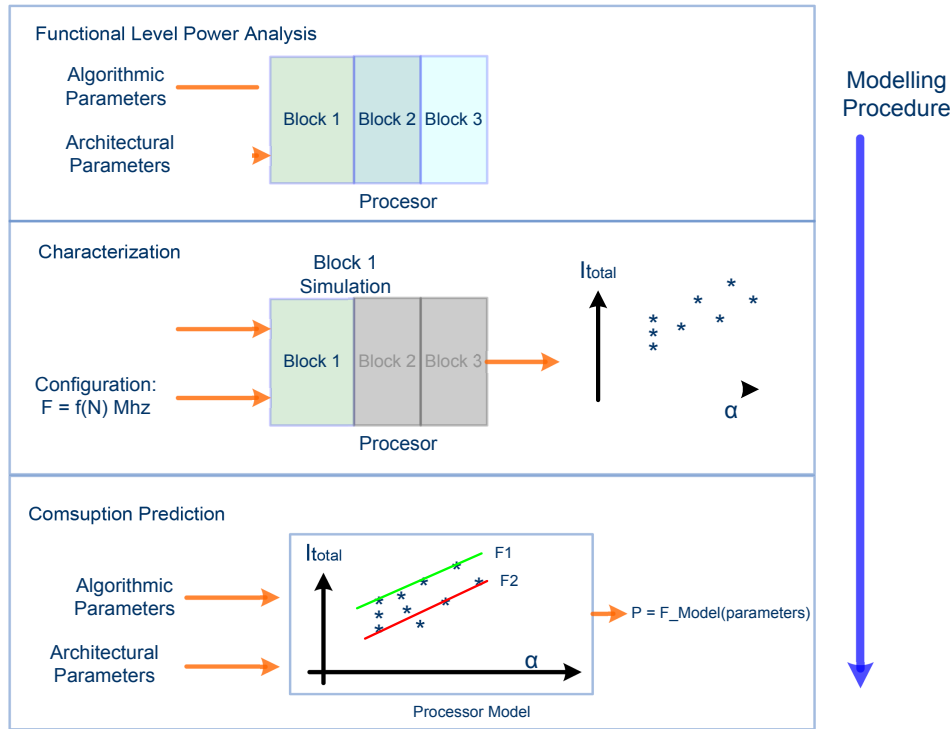
reduce the spatial complexity by observing the inter-instruction effects when a generic instruction is executed after a no-operation instruction. This approach also aims to reduce the difficulties of classifying instructions by assuming the inter-instruction overhead mainly depends on instruction changing. Thus they proposed a NOP model. Its main idea is to insert a NOP instruction before changing any instruction to model the transition overhead. They use a loop to repeat one estimated instruction several times, once alternate to another target instruction, they insert a NOP instruction, and so they do not need to enumerate each pair of instructions to build the instruction power table. The model reduces the size of the table to $O(n)$. In other work, Sama et al. in work [48] attempted to add the gate-level simulator into the instruction-level. Their base energy cost was measured by repeated executions of each instruction individually. Programs have to be made which cause repeated instruction execution and then average power has to be measured during this time. This average power consumed and the execution time of the instruction gives the base energy cost of the instruction. Overhead energy consumption comes from opcodes and control state changes between the continuous instructions, as well as the data passing, which also added into their work. To reduce the complexity of the instruction pairs, this method classifies the instructions based on their functionality and base costs. Therefore, the instruction overheads are only needed to measure for the pairs of intergroup. For those instructions in the same group, this approach assumes the same instruction overhead because the similar functionality and base costs usually indicate similar control state and opcode value. A reduction of the measurement set for characterization process is obtained through the new architecture based model. This is achieved by relating the instruction power dissipation with the processor modules. Since the measurement process is done on a much lower level, it requires a lot of time and efforts. Also for the case of large instruction set processors, this reduction is important. If the link between instruction level and architecture level is established, the effects of small changes in architecture can be propagated to the instruction level without total re-measurements. If the constituent instructions of a program are known, the effect of such changes can be predicted. This can provide us methods for fine tuning the architecture for the application programs that use it.

Instruction-level modeling approach faces three problems. The first is the numbers of current measurements, which has a direct relationship with the size of the instruction set architecture (ISA). The second is the number of parallel instructions in the very long instruction word (VLIW) processor. The last one is the difficulty to draw the whole picture of the full-system power consumption since this approach cannot provide any insight on the isolate other components. The first two problems cause the model is not general for utility, and the previous work discussed before made some trade-off of the estimation accuracy and

the complexity of the individual instruction and the inter-instruction effects. The last one makes this model cannot distinguish the instructions executed on the processor at the specific time related which part on the system, thus the user cannot know which part consume most percentage of the energy.

2.3.2 Function-Level Estimation Models

Function-level power analysis (FLPA) is applicable to all types of processor architectures without the details of the system circuits. Instead of the classical energy characterization abstracted from the instructions, the basic idea of FLPA is to obtain the distinction energy consumption from system activities of different processor functional blocks. Thus, FLPA model first divides the target into several functional units. Then it relates the processor operations to the power activations. Nathalie et al. in work [49] proposed a model for DSP based on FLPA. They divided the DSP processor into four units: instructions management unit (IMU), processing unit (PU), memory management unit (MMU) and control unit (CU). This experimental approach converges at each functional block and discards the blocks which negligible impact on the power-consumption. The processor is divided into different functional blocks; each of them is a cluster of components that are concurrently activated when a code is running. Each functional block consumes energy during the program execution; its energy consumption characterization is figured out during training procedure by the set of relative parameters. The parameters are divided into two parts: One includes the algorithmic parameters which depend on the executed algorithm (typically the cache miss rate), and the other is consist of the architectural parameters which depend on the processor configuration settled by the designer (typically the clock frequency). The first step of the model to characterize the system energy consumption is to select the proper parameters from the two parts. Then the model moves to the second step, on which various scenarios are executed with different parameter configurations. The model takes notes of the energy characterization such as current of each block. Characterization can be performed either by measurements or by simulation. Finally, the relationship among the characterization and the selected parameters are figured out. This approach is shown in the Figure 2-5.


 Figure 2-5 Processor Modeling Methodology⁴⁹

Laurent et al. presented a new approach to characterize energy dissipation on complex DSPs at functional-level in work [50]: Instruction management unit, Processing Unit and Memory management unit. The parameters of the model include parallelism/processing rate, cache miss rate and external data memory access rate by separately simulating each functional unit with small programs written in assembly language. Based on this methodology, the SoftExplor⁵¹, a tool automatically performs power and energy consumption estimations, are widely used by lots of research groups. In this case, the model only requires coarse-grain knowledge on the processor architecture and it achieves a good tradeoff between the estimation accuracy and the model complexity. However, its main disadvantage is the complexity of the components determination, the coverage of all significant influencing parameters and dependency of the corresponding power consumption on the performed instruction. Furthermore, during the determination period of consumption laws, the issue of temperature of the system which has an obvious impact on the static power consumption is not taken into account. The coefficients of the model could be different as the temperature increased.

2.3.3 Component-Level Power Estimation Models

For a better generalization, system/component-level was proposed in a high abstraction layer. It considers main system components (e.g, processor, memory, coprocessor) and leads to more intuitional and feasible models.

Models in high-level abstraction can obtain the static pre-characterized energy consumption from lookup tables such as spreadsheets. These spreadsheets are very useful in the early stage of design process to have the first decision with the power issue⁵². The spreadsheet-based approach facilitates their users, because the energy consumption of each component comes from intellectual property (IP) provider or library cell estimates. Spreadsheet provides a capability to quick estimate on current and power estimation of each block. User can configure the operating frequency, temperature and other parameters to estimate his design's power consumption by using the spreadsheet (Figure 2-6). An example of using spreadsheet is implemented on the BeagleBoard, a commercial prototyping board based on the OMAP processor in work [53] by Conzález et al. However, this approach can only provide the final power information extracted from datasheets, thus it is useful for project planning but may not be able to provide a guidance for block-level hardware power estimation and energy reduction due to its lack of further adjustments regard to the different work modes or workloads. This approach is suitable for blocks with regular activity patterns. However, with the increasing importance of power management techniques, they are limited in the accuracy.

Please refer to "OMAP3530 Power Consumption Summary - Version 1.x" application report for notes and limitations of this tool.

A) HIGH-LEVEL SYSTEM CONFIGURATION

Device	OMAP3530	1
Ambient Temperature (°C)	25°C	
Processors OPP	OPP 5 (ARM 600MHz, IVA 430MHz)	5
Vdd_mpu_iva / Vdd1	1.35V	5
Core OPP	OPP 3 (L3: 166MHz, L4: 83MHz)	3
Vdd_core / Vdd2	1.15V	

B) BASELINE CORE POWER (All clocks propagated, Real usecase would be lower)

PRCM	Include	1.35
DPLL1	Include	1.35
DPLL2+IVA_clk	Include	1.35V
DPLL3	Include (Core_clk = 332MHz)	1.35V
DPLL4	Include	1.35V
DPLL5	Include	1.35V
L3 leakage and clk tree	Include (166MHz)	1.35V
L4 leakage and clk tree	Include (83MHz)	1.35V
Neon clk tree	Include	1.35V
Processors static power	Include	1.35V
TOTAL CORE BASELINE POWER		1.35V

Vdd_mpu_iva / Vdd1			Vdd_core / Vdd2			Vdds_dpll_dli			Vdds_dpll_per		
V (V)	I (mA)	P (mW)	V (V)	I (mA)	P (mW)	V (V)	I (mA)	P (mW)	V (V)	I (mA)	P (mW)
1.350	6.57	8.67	1.15	5.36	6.19	1.8	0.63	1.14	1.8	1.21	2.18
1.350	0.81	1.09	1.15	0.05	0.06	1.8	3.26	5.86	1.8	0.01	0.01
1.350	0.22	0.30	1.15	0.31	0.36	1.8	0.00	0.01	1.8	0.00	0.00
1.350	0.00	0.00	1.15	0.59	0.67	1.8	5.01	9.02	1.8	0.01	0.01
1.350	0.02	0.03	1.15	0.44	0.50	1.8	0.00	0.00	1.8	4.93	8.87
1.350	0.02	0.03	1.15	0.44	0.50	1.8	0.00	0.00	1.8	4.93	8.87
1.350	0.35	0.46	1.15	47.14	54.21	1.8	2.73	4.92	1.8	0.00	0.00
1.350	0.19	0.25	1.15	7.90	9.08	1.8	0.02	0.04	1.8	0.00	0.00
1.350	0.05	0.07	1.15	0.03	0.03	1.8	0.05	0.08	1.8	0.00	0.00
1.350	14.63	19.75	1.15	0.00	0.00	1.8	0.00	0.00	1.8	0.00	0.00
1.350	22.87	30.87	1.15	62.28	71.62	1.8	11.71	21.07	1.8	11.08	19.95

Figure 2-6 Models Based on Datasheet⁵⁴

Later research change the research lane into join together the information from operating system or hardware events monitor and the measurement by connecting a power meter between the system and an AC outlet. (Figure 2-7) The measurements are obtained from the voltage and current drop across different components in the system board or the whole system, thus some previous researchers also divided the system into component level which takes the measurement based on each component. From elementary physics, the current flow is calculable from the voltage drops across the measuring resistor (U_R). Accordingly, the instantaneous power can be calculated as equation 2-4:

(Resistance: R , Voltage: U , electric current: I and Instantaneous electric power: P):

$$P = U \times I = U \times \frac{U_R}{R} \quad (2-4)$$

Hence, integrating will result in the electrical energy consumption as equation 2-5:

$$E = \int P(t)dt \quad (2-5)$$

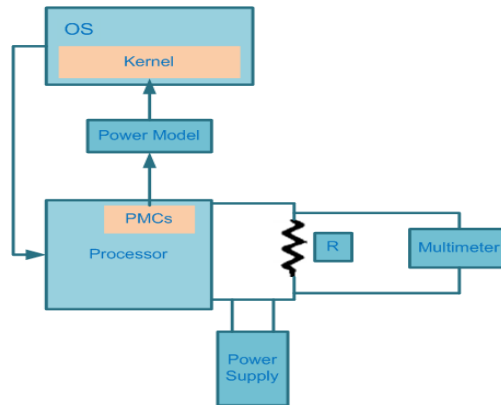


Figure 2-7 General Structure of High-level Modeling Methodology

Many methods based on the component-level energy estimation have been investigated in work [55],[56],[57] and [58]. These methods are based on each component in the design and abstracted as an additive model to get the final energy dissipation of whole system. Once a model of each component is built, the total energy consumption is computed by simply summing the energy of all components. In a broad sense, a component can be an individual functional unit or a block with several similar functional units. The key idea of the system-level power estimation method is to abstract the power behavior of the system. The power behaviors of its components are driven by the some specific events. Figure 2-8 shows a simple diagrammatic sketch of component-level energy estimation model. The hardware events are the key energy influence factors such as cache misses, retired instructions and the memory accesses. These events can be monitored by the system available PMCs.

However, for some devices such as I/O peripheral and network devices cannot be directly monitored by PMCs. Therefore, their requests through the corresponding drivers can be used to estimation their behaviors.

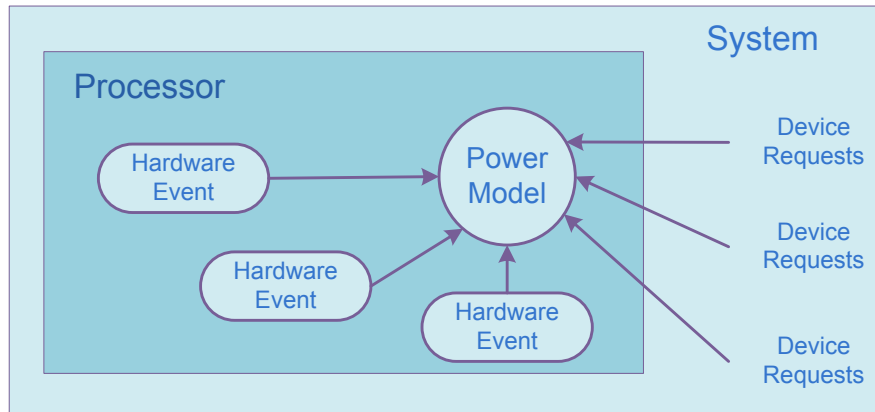


Figure 2-8 Simplest System-Level Power Model

In some complex systems, each component can be described by a simple state machine containing only information relevant to its power behavior. At different execution points, each component is in a specific power state, which consumes a discrete amount of power. To get the total system power usage at a specific time just requires summing up the current power values for all components within the system. Average or peak power dissipation can easily be determined by looking at the power usage over time for a given set of environmental conditions.

However, energy consumption is not only influenced by hardware but also the software. Some previous work pointed out that a good software needs to be designed with power consumption consideration because the processor's power consumption is greatly dependent on its executing workload, which means that the performance characteristic of software at runtime can also be used to change the energy consumption^{59,60}. In modern microprocessors, there is a set of special-purpose registers build in. They are performance monitoring counters (PMCs), which are used to record the number of hardware-related activities occurred in computer systems. Users can supervise and adjust the system performance through the information provided by those counters. Operating systems provides many interfaces to access PMCs. For example, the simple Linux command "Sar" can collect the dynamic data of CPU, disk, network throughput, ect. There are several PMC tools such as Oprofile⁶¹, Perfctr⁶² and PAPI⁶³ that freely provide the details of implementing access to hardware counters on various platforms. PMCs provide the deeper insight of processor's functional units, cache and main memory with low-overhead. In addition, their

wealth of interface can simplify their usage. After Bellosa in work [64] correlated performance monitoring counters (PMCs) with energy consumption to obtain a good estimation, the energy consumption modeling combined with PMCs and linear regression has been widely used in work [59,65,66,67,68,69]. The work such as [65], Li et al. exploited high-correlation between the number of instruction per cycle (IPC) and power consumption to estimate the energy dissipation. This work cannot be simply represented into a new platform with the same accuracy. The reason is that only take account into IPC is not enough to capture system details. A full-featured system usually consists of different devices with various functionalities and energy requirements, thus simple PMC set may be one-sided and easily causes the instabilities. The main challenge of these models is how to choose the best set of PMCs. Most researchers identified the PMCs based on platform architecture analysis [59,66,67,68]. In [66], Lively et al. introduced a description of how to choose the suitable PMCs. They used different PMCs to build models for each application to endure the application trends were correctly represented. Their work aims to achieve an insight to the relationship between the performance characteristics and the PMCs, however, this application-centric method have a high overhead for on-line use if one model is only for one particular application. In [67], Goel et al. proposed a different approach to choose PMCs. Their first step was to identify the candidates by manually separating available PMCs into several categories that impact dynamic power by different issues. Their work can effective reduce the number of PMCs, however, the priori selection may meet the problem due to the limitation of PMCs, especially for the embedded processor with small number of PMCs. On the contrary, Y.Xiao et al. in work [69] built a sub-model of processor without any selection but repeated the same test case for several times with two different PMCs monitored each time to obtain all the information provided by PMCs. This method is quite time-consuming, and it is not suitable to the PM policy since it will cause a long-time delay to get a full estimation of one application.

There is no doubt that PMC-based estimation results are promising, however, it is not easy to repeat one model to other systems because of the uniqueness of appropriate PMCs for each different system. One reason is the types and meanings of hardware counters vary from one kind of architecture to another due to the variation in hardware organizations. Therefore, the exact model based on PMCs will also be different for each platform depending on the availability of native PMCs. Another reason is the number of available hardware counters in a processor is limited while each cpu model might have a lot of different events that a developer might like to measure. Each counter can be programmed with the index of an event type to be monitored, like a L1 cache miss or a branch mispredict. In the other word, although CPUs typically have multiple counters, each can monitor only

one type of event at a time, and some counters can monitor only certain events. Therefore, some CPUs cannot concurrently monitor interesting combinations of events. A challenge of system/component level model is how to select the most suitable power-model for each component by making efficiency and accuracy trade-offs. A good system/component-level can be used by the internal power management system for optimized component usage.

2.4 Discussion

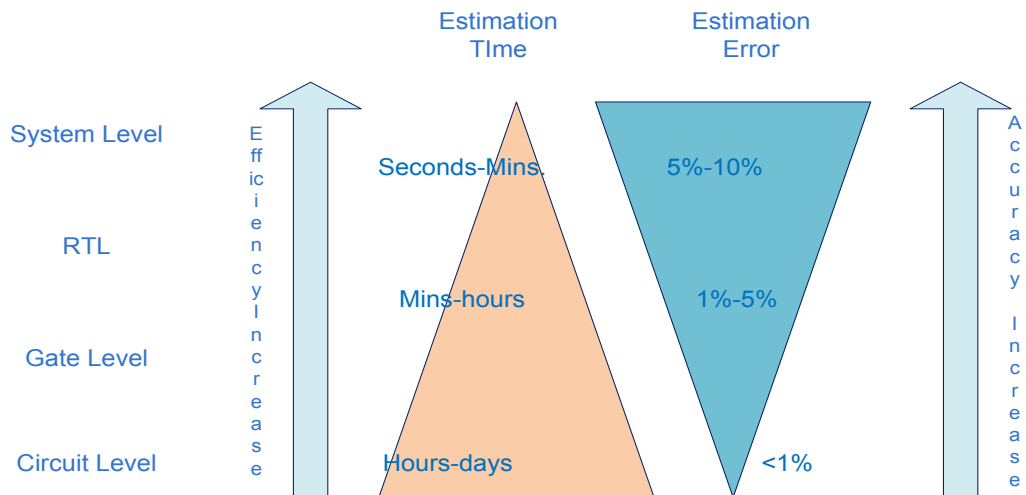


Figure 2-9 Power Estimation in Different Design Level

General speaking, the estimation time and the accuracy are inversely proportional to the abstraction level. As the Figure 2-9 shows, the low-level can reach quite accurate results of those models based on the design details but they are slow and impractical for analyzing the power consumption at the early design stage. In addition, low-level power estimation tools require complete RTL code, and their simulate time range from days to minutes. Circuit-level power estimation tools, though provide excellent accuracy, have the longest simulation time, and require more development efforts. Therefore, low-level models cannot afford to simulate large circuits for long-enough input vector sequences to get meaningful power estimation because of its simulation speed.

The most important parameters of the low-level estimation models are the input patterns or waveforms. This is an attractive feature which makes the power consumption directly relate to the switching activities of the circuits. However, this high pattern-dependent relationship is a serious problem in power estimation because of the difficulty on obtaining the input patterns when the other parts of the chip have not been designed or completely specified.

For the embedded system, a common practiced design concept, intellectual property core (IP), has been proposed in recent 20 years. Reusing IP blocks without too many changes of the design and functionalities provide the possibility of fast prototyping, validating, evaluating and design complexity decreasing. With the increasing IP popularity, hardware low-level design technologies such as load capacitance, supply voltage, frequency and threshold voltage become more mature with some conventionalized design experience, especially for those hard IPs. Therefore, to optimize the energy efficiency from the software aspect becomes more significant. The optimization includes better application program writing skills, OS scheduler and PM policy. Usually, the hardware design is transparent to the software designers, thus to have a better energy estimation for their applications or OS policies, the higher abstract level for estimating is needed. High-level estimate models can avoid the disadvantages of low-level models from the software inspect. Besides the consideration of the accuracy, this energy estimation model is foundation of the energy-aware operating system scheduler, thus it is important to keep the flexibility and the efficiency of the model. Based on these reasons, the high-level model, especially at the component-level is focused. This dissertation gives out the first investigation on this topic. A model is constructed by utilizing a set of most proper hardware PMCs, which indicates the energy behavior of the processor and memory.

Although the PMC-based modeling approach has been proposed by different research groups, most of them are applied to the General Purpose Processor (GPP) and servers. Isci et al. in work [70] gave out an approach to estimate the power consumption of a Pentium 4 processor which has 18 PMCs and can monitor up to 59 events without using circuit-level information. Their model is not applicable to implement on the processor with a small number of available PMCs such as OMAP 3530. Other work such as Li et al. in [65], they exploited high-correlation between IPC and power consumption to estimate the energy dissipation. This work cannot be simply represented into a new platform with the same accuracy. The reason is that there are differences among components, the same PMCs set may easily cause the instabilities, so it is necessary to construct a model based on the different PMC sets of each component. On component-level, the system is firstly divided into several parts and their own sub-models are built, and then the relationship between sub-models and whole models is found out through the observation data. In this way, the accuracy of the model is improved and the power consumption details can also be presented by each sub-model. Another advantage of component-level model is the good portability because it is quite easy to add new sub-model for the extensible system. It should be noted that even though the example presented is for a particular architecture, the methodology can be applicable to other architectures. Particularly, the PMC set of components are needed to

be redefined and redesigned in order to fulfill the requirements of the model accuracy. A third-part software interface PAPI is used to define the relevant available native PMCs on a given platform.

2.5 Conclusion

The power/energy estimation models implemented in the different levels are introduced. The low-level methods can obtain quite accurate estimation but suffer with the hardware details and the expensive time cost, while the high-level methods provide better portability and efficiency but partly decrease the accuracy. Considering the easy usage and quick response, the PMC-based approach in the component-level is decided to construct the estimation model. This approach divides the system into several components based on their functionalities and builds the sub-model for each component with different set of PMCs.

3

Methodology

3.1 Introduction

In this chapter, the introduced methodology will be described in detail. It begins with the general idea of PMC-based approach. This is the basic framework of this component-level methodology, and then the specific modeling knowledge of the mathematics will follow. Finally, the main contribution of this dissertation is focused, which is the two-part modeling methodology which aims to improve the accuracy, efficiency and independence of the energy consumption model.

3.2 General Idea

3.2.1 Modeling Flow

To address the need for fine-grained energy consumption on a system, a component-level model is more suitable with its features such as intuitional, flexible, extensible and efficient. A full component-level model is based on the power measurement and automatic synchronization, and provides correlations between system/application typical activities and system energy consumption.

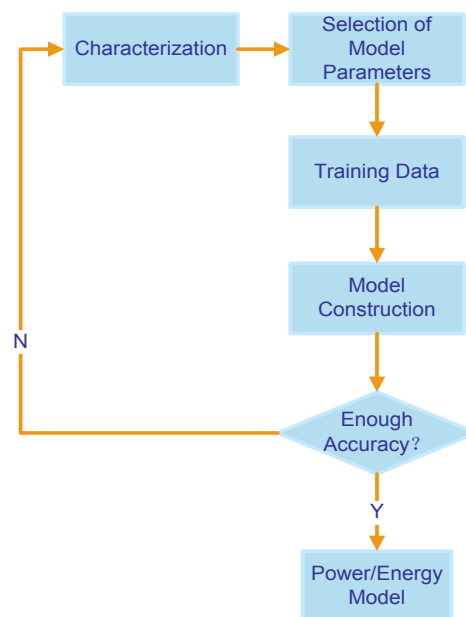


Figure 3-1 Modeling Flow

Generally, a high-level energy estimator has two main steps: parameters determination and model construction. The Figure 3-1 shows the flow of the modeling system. A specific target device is called device under test (DUT). To estimate the energy on

a DUT, the model inputs, which are the samples of PMCs, are obtained during the benchmarks' execution. These model inputs, or model parameters, are selected based on the extracted features of these benchmarks. Then, in order to fit the energy model, the current flows of the whole system or the certain component with the supplied voltage have to be measured. The energy estimation model is constructed by regress method. The model inputs are independent variables and the measurement values are dependent variable. Once the coefficient of each independent variable is set, the model is able to estimate the energy from the model inputs. The regression analysis can be linear or non-linear. Linear regression has already been proved with the enough accuracy to estimate the component energy consumption by previous work. In addition, the real measurements are also used to make an evaluation of the model. Some adjustments would happen if the estimation results have unacceptable errors, which means the estimation of correlations between system typical activities and system energy consumption are not well presented. Once a model is set with the adequate accuracy, this model can be applied to the OS as an independent application which inputs its estimation result to the OS as a parameter needed by the energy-optimizing strategies.

3.2.2 Performance Monitor Counters

PMCs have been briefly introduced in the related work. This sub-section will focus on more details of the PMCs. PMCs count the number of certain types of hardware events such as instructions executed, cache misses or branched mis-predicted. They are used to provide information about how well the operating system or an application, service, or driver is performing. They can help designers to find out system bottlenecks and fine-tune system and application performance. Implementation of PMCs in different processors could differ from the quantity or the monitored types of events. However, there are some common basics:

- A cycle counter: This can be programmed to increment on every cycle;
- Event counters: The concept of event counter and event need to be distinguished. An event counter can be configured to select one specific event among and increments as this event occur. This means that the behavior of hardware PMC can be defined by users according to their individual requirements. Usually, the number of the events is more than the number of PMCs. Moreover, the number of PMCs is more than the number of PMCs that can be used simultaneously. In addition, although some processors provide in the architecture up to large number of PMCs, the actual number is defined by the implementation.

- Controlling counters: There are some counters used to control the according PMC to finish various operations. The operation of counters includes: enable, reset, start, stop, flag overflows, and enable interrupts on counter overflow. Pay attention to the cycle counter, in some platforms, it can be enabled independently of the event counters.

In a broad sense, PMCs consist of these three counters, but in this thesis, we refer PMC as the event counter if there is no particular emphasis.

The most important feature of PMC Application Programming Interface (API) is its extremely low overhead. PMCs can be accessed via special file descriptors. For example, in the windows 2000 operating system and later ones, network and other devices provide counter data consumed by applications to provide users a graphical view of how well the system is performing. The Linux PMC subsystem also provides an abstraction of hardware capabilities such as *perf_event*⁷¹ which is a Linux kernel API and *perfmonX*⁷² which is the hardware-based performance monitoring interface for reading the PMCs from user space. A more detailed explanation of Linux PMCs API will be described in chapter 4.

3.2.3 Components Classification

3.2.3.1 Components Classification

Most embedded systems are single-board computers (SBCs) which are completed computers built on a single circuit board. There are no exact design standards for a complicated embedded system. An embedded system consists of various physical components and can be extended easily. General speaking, those devices of an embedded system can be divided into five main categories: Computation, storage, communication, buses and I/O (Figure 3-2).

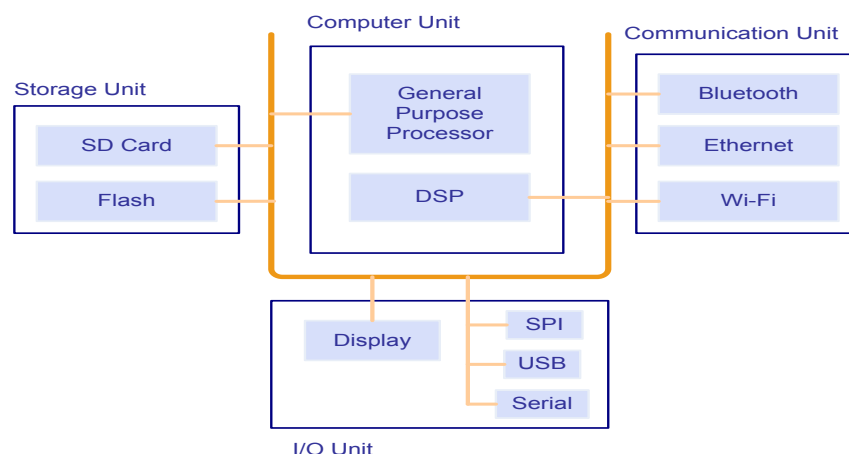


Figure 3-2 High-Level Overview of the Embedded System Architecture

Each of these categories with its own unique objective functionality cannot be replaced by another one; therefore, each category can be considered as an independent component to analyze its energy consumption issues.

3.2.3.2 Energy-related Events

In semiconductor technology, static energy is caused by leakage current and capacitors loading/deloading operations. The leakage current depends on static parameter such as time, voltage and semiconductor properties. Static energy is beyond the scope of this dissertation which focuses on the modeling for the dynamic energy. The dynamic energy consumption depends on the switch frequency of the transistors and the size of the capacitors. One direct method to identify which components have the significant contribution to the total energy consumption is to look at those parts containing most of the capacitors or with higher switching frequencies. However, this method is not suitable for most third-part manufacturer due to the lack of devices design details. PMCs give another possibility to qualitatively measure the energy consumption. PMCs monitor the events occurrence during an application execution. These events are in high abstract level caused by the essential functional units such as datapath, control and memory. Therefore, the energy consumption acknowledgement can be obtained by observing the representative events. The representative events differ from each category in the embedded system; we simply describe the general energy-related events.

(a) Computation

As the most complicated device, there are many details need to be considered. Processors have various hardware architectures, instruction set, pipeline depth, specific accelerate circuits and instruction cycles. These variances have their own contributions to the whole energy consumption. The significant energy-related factors are: execution cycles which are influenced by instruction decoder, pipeline, TLB and cache units.

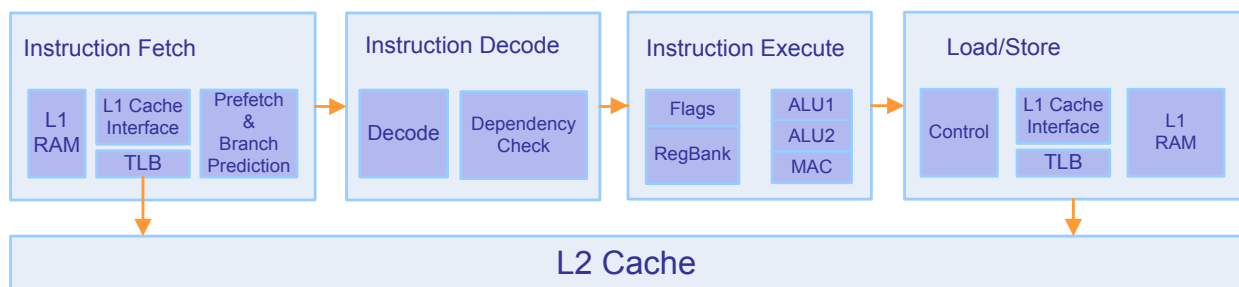


Figure 3-3 General Core Architecture

Figure 3-3 is the general structure of processor Cortex A8⁷³. It is divided into 4 parts which are corresponding to the instruction process. The instruction fetch unit fetches instructions from the L1 instruction cache based on the prediction of the instruction streams, then it places the fetched instructions into a buffer for decode pipeline using. After the instruction decode unit decoding and sequencing instructions, the execute unit starts operating on them. This unit consists of two symmetric Arithmetic Logical Unit (ALU) pipeline, a multiply pipeline, and address generator for loading and storing instructions. It also performs register write back and processes branches and other changes of instruction stream and evaluates instruction condition codes. The load/store unit includes the entire L1 data side memory system and the integer load/store pipeline. The L2 cache unit services L2 cache misses from both the instruction fetch unit and the load/store unit.

Each instruction process consumes baseline energy. In modern processor, additional units such as branch prediction, cache and pipelining are implemented to accelerate the process speed. Therefore, only number of instruction number cannot completely represent the processor's functionality because the accretion units will cause extra influence on the energy. For example, pipeline blocking, cache miss and prediction failure will cause the processor stall and the decrease of the number of issued instruction, but as this time, other units are active to prepare the execute environment. These additional units' events will affect the power consumption. Therefore, more typical events need to be distinguished to each processor to make higher accurate energy estimation.

(b) Communication

There are the basic power consumption of the different device within its specific state, such as transferred and traffic. The total energy can be predicted by counting the traffic time interval and transferred bytes.

(c) Storage

The total energy can be predicted by transferred data, in other words, the direct transferred size or the bandwidth multiply the access times. Usually, a complex storage component may have several states that consumed different energy. A more accurate model also considers the energy consumption during the states transitions.

(d) Buses

Due to the fixed bus frequency of the embedded system, buses' energy can be estimated by the bus activities times and bus width.

(e) I/O devices

Most of the I/O devices have several states which consume different amount of energy. So their energy consumption mainly depends on the number of I/O requests and the according state. I/O devices do not have uniform factors to influence their energy consumption, for example, the energy consumption of display almost depends on the different screen brightness, which does not exist in other device.

To simplify the work, at this moment, only the computation unit and the storage unit are considered. For the storage unit, the PMC-based approach is also implemented to estimate its energy consumption because the transfer data size which is determined by the application itself is not easy to directly obtain. Since the SD card or flash are not as complicated as the hard disc to have different rotation speeds, we can assume that each access of the SD card or flash has the similar energy consumption, thus their energy mainly related with the access times.

3.3 Modeling Methodology

3.3.1 Mathematics Knowledge

3.3.1.1 Spearman Rank Correlation Coefficient

Spearman's rank correlation coefficient (ρ) is a non-parametric statistic parameter proposed by Charles Spearman. It uses a monotonic function to describe the statistical dependence, which is also considered as how strong the relationship between two variables is⁷⁴. One variable is a strictly monotone function of the other if the Spearman correlation coefficient is +1 or -1 when there are no repeated values of the sampling data. These two values, +1 and -1, are called perfect Spearman correlation.

The correlation coefficient ρ can be calculated by equation 3-1 if there is no repeated value in the original data samples. Here d_i is the difference between the ranks of each observation on the two variables. If no, for a sample of size n , the original variables X_i, Y_i are converted to ranks x_i, y_i , and then ρ is need to be calculated by equation 3-2 as the Pearson correlation coefficient.

$$\rho_s = 1 - \frac{6 \sum d_i^2}{n(n^2-1)} \quad (3-1)$$

$$\rho_s = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}} \quad (3-2)$$

3.3.1.2 Linear Regression Methods

In statistics, linear regression is a regression analysis method used to construct the relationship model between one or more independent variables and dependent variable. This function is a linear combination of one or more model parameters which known as the regression coefficients. A linear regression equation with one independent variable represents a straight line.

Given a random sample, $s = (y_i, x_{i1}, \dots, x_{ip} | i = 1, \dots, n)$, a linear regression model allows the imperfect relationship among regress factor y_i and regression variables x_{i1}, \dots, x_{ip} . Usually the model uses an error term ε_i (also a random variable) to capture any other impacts beside for x_{i1}, \dots, x_{ip} . Therefore, a multivariate linear regression model is expressed as the following equation 3-3:

$$y_i = \beta_0 + \beta_1 x_{i2} + \beta_2 x_{i3} + \dots + \beta_p x_{ip} + \varepsilon_i, i = 1, \dots, n \quad (3-3)$$

Linear regression technique is widely used in many modeling scenarios due to its generality and statistical properties. Employing a linear regression model to estimate the energy consumption implies to identify, as independent variables, a set of PMCs with high correlation to the amount of energy consumption. The energy consumption is assumed to be a linear function as that given by equation 3-4:

$$E = \sum_{i=1}^k c_i PMC_i \quad (3-4)$$

where PMC_i are the number of monitored events of type i and c_i is the corresponding coefficient computed by a linear regression method.

There are inevitable some outliers or high leverage points in the observations. An outlier is an observation which is distant from the prediction value based on the regression equation or markedly far from other sample data⁷⁵. Outlier points may indicate faulty data, erroneous procedures or an invalid theory for a specific situation. However, a small number of outliers is acceptable.

The tendency of the straight line after the linear fitting depends on the spatial distribution of the samples. Intuitively speaking, the outlier points have a great influence of the fitting results. Their effects are called leverage effect. Leverage is a term used to identify those outliers which are far away corresponding average prediction values. Leverage points are those observations with extreme values of the independent variables which lack of neighboring sample data to impel the fitted regression model pass as close as possible to

them. Outliers and high leverage points bring difficulties to the least-squares regression which is the simplest linear regression method because there is no sufficient reason to remove those unusual data unless they can be proved to be a recorded error. The classic principle least-square regression is not provided with robustness, thus the outliers and high leverage points will have a devastating impact on the estimation of the coefficients of regression equations. Robust regression ensures fewer effects of errors on model coefficients estimation to increase the reliability⁷⁶. Therefore, we employ the robust regression methods in this dissertation.

3.3.1.3 Principal Components Analysis

In the statistical analysis, principal component analysis (PCA) is an algorithm to analysis and simplify of data collection. The PCA is often used to reduce the dimension of the data set, while maintain the data sets which keeps the most contribution to the variance. This is done by retaining the low-level principle component and ignoring the higher order principle components. PCA was first proposed by Carl Pearson in 1901 was used to analyze data and build mathematical models⁷⁴. This approach is to obtain the main components data (the eigenvector) and their weights (intrinsic value) by eigen decomposition of the covariance matrix. PCA is the simplest multivariate statistical method based on eigenvector analysis. However, there is a serious robustness problem in ordinary statistical PCA based on eigenvalue decomposition, which will greatly affect the computation accuracy of the PCA.

3.3.2 Methodology

In short, our methodology to produce component sub-models follows the common modeling steps:

- First, we define the model inputs. In our case, there are the occurrences of different energy-related events recorded in the corresponding PMCs. Since the PMCs can be configured to monitor different events, in rest of the dissertation, PMC and its corresponding event have the same meaning. In addition, the appropriate events are unique to each individual system. Thus we define a methodology, PMC-filter, to automatically identify the best PMCs set from all the platform-available PMCs without any prior analysis.
- Second, we train the model by using the individual benchmark groups according to each component and collect the PMCs' samples. To avoid the interference among components, each benchmark group addresses the characteristics of each component. During the procedure with the modeling

research, we find that the accuracy of the estimation is highly related with the training benchmarks. If the group of benchmarks cannot cover enough characteristics of an application, the fitting curve will lead to the wrong prediction. Thus we use the k-fold cross validation to train the model.

- Third, the final system model is built by simply combining the individual sub-model of each component. Since the components have the different contribution to the total energy consumption, a method to identify their energy weights is also needed.

3.3.2.1 PMC-filter

In statistical, the correlation coefficient presents the relationship between two variables. In other words, if two variables are perfectly correlated, they share all the features⁷⁵. Many researchers have made proposals to judge the variable dependence with the correlation coefficient. However, these proposals are mainly based on the experience^{77,78}. Table 3-1 lists the explanation of the correlation coefficients.

Value of Correlation Coefficients	Explanation
0.8~1.0	Very Strong Related
0.6~0.8	Strong Related
0.4~0.6	Moderately Related
0.2~0.4	Weak Related
0.0~0.2	Very Weak or No Related

Table 3-1 Correlation Coefficient Explanation

This look-up table evaluates the relationship between two variables, but this rule is mainly dependent on the subjective judgment. A more accurate method to interpret the correlation coefficient is to calculate their coefficient of determination (R^2). R^2 reflects the percentage of the variance of one variable can be explained by the variance of another one. For example, if the correlation coefficient between variable X and variable Y is 0.7, namely $\rho_{YX} = 0.7$, then the $R_{YX}^2 = 0.49$ which means that 49% of the variance of variable Y can be explained by the variance of X. The stronger the correlation, the more variance can be interpreted. Since the variance presents the distribution characteristics of a series of data or the statistical population, thus the more variance can be explained, the more features shared

by the two variables, also, the more information can be represented by another variable. However, $R_{YX}^2 = 0.49$ means that 49% of the variance can be explained, it also means that 51% of the information cannot be replaced. This is because that even these two variables have a strong correlation ($\rho_{YX} = 0.7$), there is still unexplained reasons cause the differences between them. The idea of the shared variance can be vividly shown in the Figure 3-4. The gray area stands for the shared variances of two variables. The larger it is, the stronger correlations the two variances have.



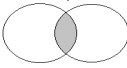

Correlation Coefficient	Coefficient Determination	Variable X	Variable Y
$\rho_{YX} = 0$	$R_{YX}^2 = 0$		Share 0% 
$\rho_{YX} = 0.5$	$R_{YX}^2 = 0.25$		Share 25%
$\rho_{YX} = 0.9$	$R_{YX}^2 = 0.81$		Share 81%

Figure 3-4 Shared Variance

In the first case, there is no overlap of the two circles because there is no relationship between them. In the second case, the two circles begin to overlap because they share the 25% information. In the third case, one circle almost perfectly covers the other one due to the quite high correlation of them.

It should be noted that the correlation is independent on the cause. In other words, the variables X and Y increasing (or decrease) together do not mean that the change of one variable is the reason to cause another variable change. During the model fitting, we use the correlation information among the PMCs aim to reflect more energy variation features within fewer PMCs. Let us use an example to better explain this idea:

Consider an accurate energy estimation model with two independent variables Y and X, namely $E = aY + bX$. Assume that 75% of the features in Y can be explained by variable X, thus the variable Y can be estimate by variable X like equation 3-5:

$$Y = cX + \varepsilon \quad (3-5)$$

where the ε is the estimation error due to the 25% unexplained features of Y . Thus the energy model can be rewrite as equation 3-6:

$$E = acX + bX + a\varepsilon = (ac + b)X + a\varepsilon \quad (3-6)$$

If the relationship between X and Y is stronger, the value of item $a\varepsilon$ is smaller which means the error caused by representing Y is smaller.

In our case, if an energy estimation model includes all the available non-derived PMCs in the platform, most of the details of the application are covered. But a large number of PMCs increases the complexity of on-line modeling and sampling time. Thus, it is necessary to have a methodology to reduce the number of PMCs without losing too many of the application behavior features. Therefore, identifying the PMCs which are intimately related to energy consumption is the first requirement of the modeling process.

Correlation coefficient is used to decide the relationship between two elements as the discussion before. Therefore, correlation coefficients, ρ_{S_i} , between each PMC and energy consumption are first computed. In this dissertation, the Spearman's rank correlation has been employed considering that the overall distribution of the sample data is unknown.

After the first step, a threshold, α , of ρ_{S_i} is set to identify the PMCs with the largest energy correlation and to eliminate, from the initial set of PMCs, those whose coefficients are below α . It is worth noting that α may vary from system to system because of the different PMCs availability.

Again, if two PMCs have high cross-correlation, the information kept by one PMC is also able to be reflected by the other. Therefore, to further reduce PMC redundancy, correlations between each pair of PMCs, $\rho_{(i,j)}$ are computed to identify the PMC relationship. The purpose is to lessen the outlier influence while maintaining the captured application features. The previous PMC selection set is iteratively refined as follows. First, starting from a PMC_a with the largest correlation, ρ_{S_a} , those PMCs whose correlation, $\rho_{(a,j)}$, exceed certain threshold, β , are eliminated. Then, the process continues with PMC_b , the second largest PMC with correlation value, ρ_{S_b} , and, again, eliminating the PMCs whose $\rho_{(b,j)}$ exceed β . This process is repeated until there is no more PMC to eliminate. At last, the remaining PMCs, which form the set named, P_e , are the most important ones, as far as the energy correlation concerns.

Considering the correlation analysis and the energy model introduced by Lively et al. in work [66], in this dissertation, the threshold α is set to 0.5 and the threshold β is set to

0.90. Threshold β is set quite high in order to ensure that the retained PMCs record most of the application behavior information to maintain the accuracy of the estimation.

3.3.2.2 K-fold Cross-validation

To make an estimation model with good predictability and generalization, a common employed technique is the k-fold cross-validation, which randomly partitions the original samples into k subsamples, then repeats the model training process k times (the folds). Each time one of the subsample sets is used to test the model and the other k-1 ones are used as training data. At last, a single estimation result is obtained by averaging the k sub-models. Figure 3-5 is an example which can better explain how this method works.

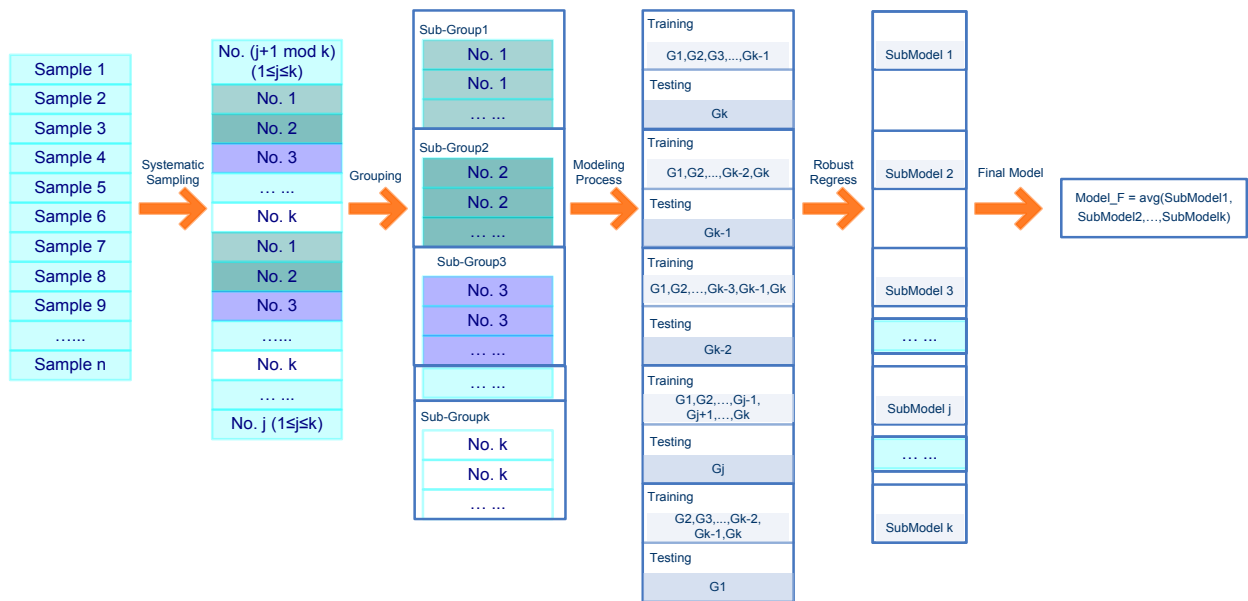


Figure 3-5 Example of the K-fold Cross Validation Method

This method starts to divide the whole population into several sub-groups. The partition procedure is simply implemented by systematic sampling. All samples are numbered cyclically from 1 to k and are selected into the same group with the same number. For example, samples with number i ($1 \leq i \leq k$) are selected into group i . Afterwards, the robust regression method is employed to identify the fitting coefficients. Then, these coefficients are integrated with the corresponding PMCs to generate, for the current training group, a linear estimation model, named as M_{P_i} . This procedure is repeated k times and, at last, the final model is obtained by averaging all the M_{P_i} models.

The fold number k is usually set experimentally. Kohavi in his theoretical work [79] recommended setting k equals to 10 in order to provide fewer bias during the model training.

However, Resende et al. in work [80] have consciousness that the model prediction ability relates independently to the different values of k . Considering the rotation overhead and the sampling time for an on-line modeling, the value of k has been set as the cardinal of the set, P_e , which is always less than 10. K-fold cross validation method uses every sample point to test the model exactly once, and to train the model $k-1$ times. Thus it can better reduce the bias of training the model.

3.4 Discussion

Since the PMC-based model is constructed in the high abstract level, a challenge is if the PMCs can reflect the different processor architecture features. To confirm this query, the same benchmarks are executed on two processors: the ARM Cortex A8 and the Intel Xeon X3450 that have the different instruction set. The Cortex A8 is a 32-bit reduced instruction set computer (RISC) which is widely used in many embedded system design. X3450 is a complex instruction set computer (CISC). In CISC, about 20% of the instructions will be repeatedly used which accounted for 80% of the entire program code. This feature can be represented by the PAPI_TLB_IM, which monitors the instruction TLB (Translation Lookaside Buffer) misses. As the Figure 3-6 shows, in most cases, the Cortex A8 processor has much more misses than the X3450 processor. Note that the TLB misses of different applications ranges a large scope, so the Y axis which represents the number of TLB miss occurrence uses the logarithmic scale based on 10.

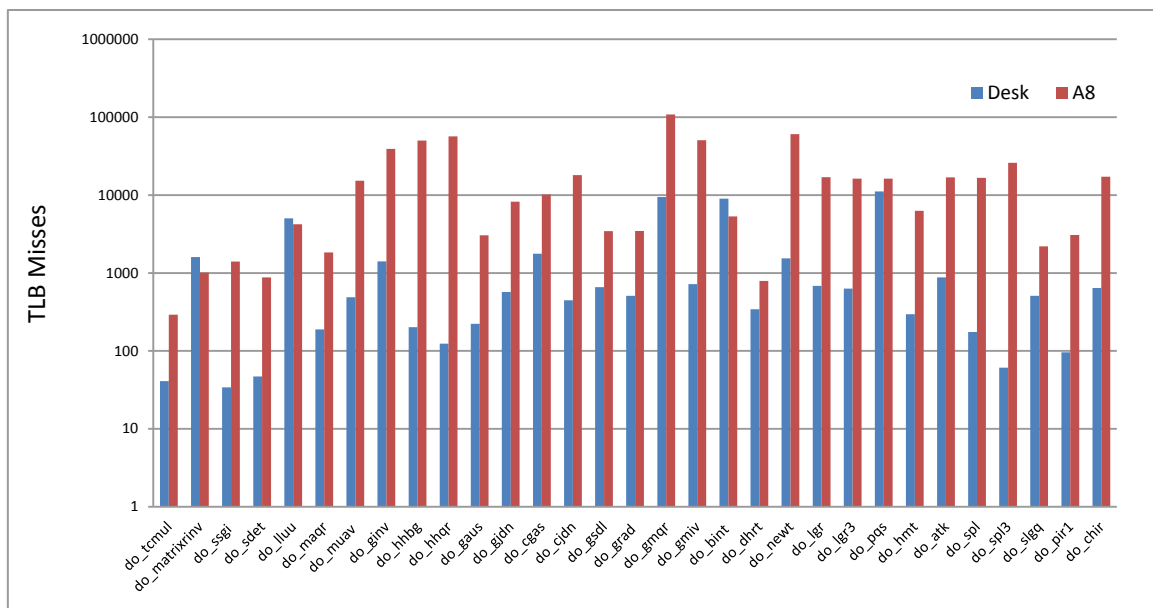


Figure 3-6 Comparison of TLB Instruction Miss

In computer architecture, there is a term called CPI (clocks/cycles per instruction) can used to describe a processor's performance. If the number of cycles is less during an instruction execution, the more efficient the processor behaves at this period. Assume that the execution time can be calculated as equation 3-7:

$$EXE_{Time} = CPI * INS * CYC_{Time} = CPI * \frac{INS}{Freq} \quad (3-7)$$

Where the EXE_{Time} is the execution time, INS stands for the number of the total instruction and the CYC_{Time} means the clock time. So the two processors can be compared like the equation 3-8:

$$Speed = \frac{EXE_{Time_X}}{EXE_{Time_A}} = \frac{CPI_X}{CPI_A} * \frac{INS_X}{Freq_X} * \frac{Freq_A}{INS_A} = \frac{CPI_X}{CPI_A} * \frac{Freq_A}{Freq_X} * \frac{INS_X}{INS_A} \quad (3-8)$$

So the comparison of the CPI can be calculated as equation 3-9:

$$\frac{CPI_X}{CPI_A} = \frac{EXE_{Time_X}}{EXE_{Time_A}} * \frac{INS_A}{INS_X} * \frac{Freq_X}{Freq_A} \quad (3-9)$$

Substitute the average value of the observations in our tests, then

$$\frac{CPI_A}{CPI_X} = \frac{EXE_{Time_A}}{EXE_{Time_X}} * \frac{INS_X}{INS_A} * \frac{Freq_A}{Freq_X} = \frac{79.74s}{3.72s} * \frac{43264740017}{8814186320} * \frac{600MHz}{2.67GHz} = 23.09 \quad (3-10)$$

The result shows that the X3450 processor is 23 times faster than the Cortex A8 processor. However, considering it is a 4-core processor with the frequency 2.67 GHz which is 4.54 times of the frequency of the Cortex A8, the CPI of X3450 processor is not too much higher than Cortex A8's as anticipation. One reason for this observation is that RISC architecture extensively uses the registers. Its data processing instructions only operate with registers, and only load/store instructions can access memory in order to improve the efficiency of instruction execution. On the contrary, CISC architecture needs more Absolute/Direct addressing mode, while RISC architecture has few instructions in this mode, thus CISC CPU needs more clock cycles to calculate the effective address. Meanwhile, the CISC instructions vary in lengths which cause the unified execution cycles. Some instructions with too many stages lead to rises of waiting time of other idle units.

However, CPI is a very basic and primary indicator to evaluate the performance, there are many other things need to be consider as the same time. For example, the cache hit rate and the branch prediction miss rate. Figure 3-7 and Figure 3-8 show the

comparisons of these two factors. Note that the cache hit rate here is the sum of the hit rate of each level as equation 3-11:

$$R_{cache_{hit}} = R_{cache_L1_hit} + R_{cache_L1_miss} * R_{cache_L2_hit} + \dots + R_{cache_L1_miss} * R_{cache_L2_miss} * \dots * R_{cache_L-n-1_miss} * R_{cache_Ln_hit} \quad (3-11)$$

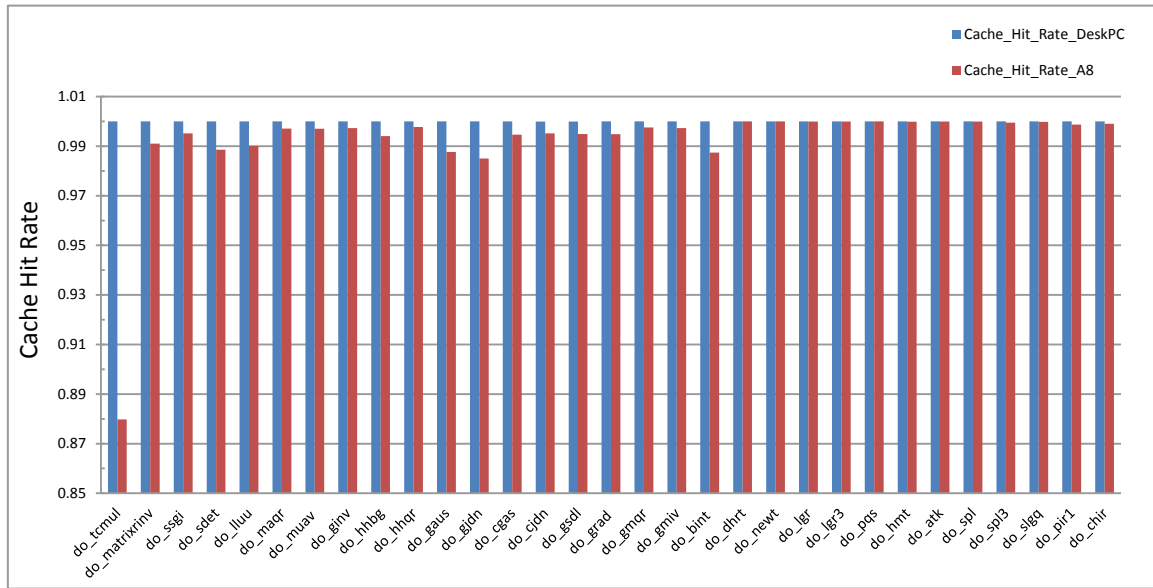


Figure 3-7 Comparison of the Cache Hit Rate

In the first case, both processors present quite good cache hit capabilities: the X3450 processor nearly 100% hit and Cortex A8 achieves an over 97% hit ratio only except one with 87%. As the new generation processor with high performance, two processors use many new techniques to improve their performances. In theory, the size of cache should be sufficiently large in order to real improve the speed of the processor, so the ARM Cortex A8 increases its level 1 cache to 32KB(data cache)+32KB(instruction cache) and add a 265KB-size level 2 cache to reduce the frequently delay by accessing RAM. The X3450 processor provides unified third-level cache shared by all cores in the physical. It is 8MB, 16-way associated and writeback. The L3 is designed to use the inclusive nature to minimize snoop traffic between processor cores.

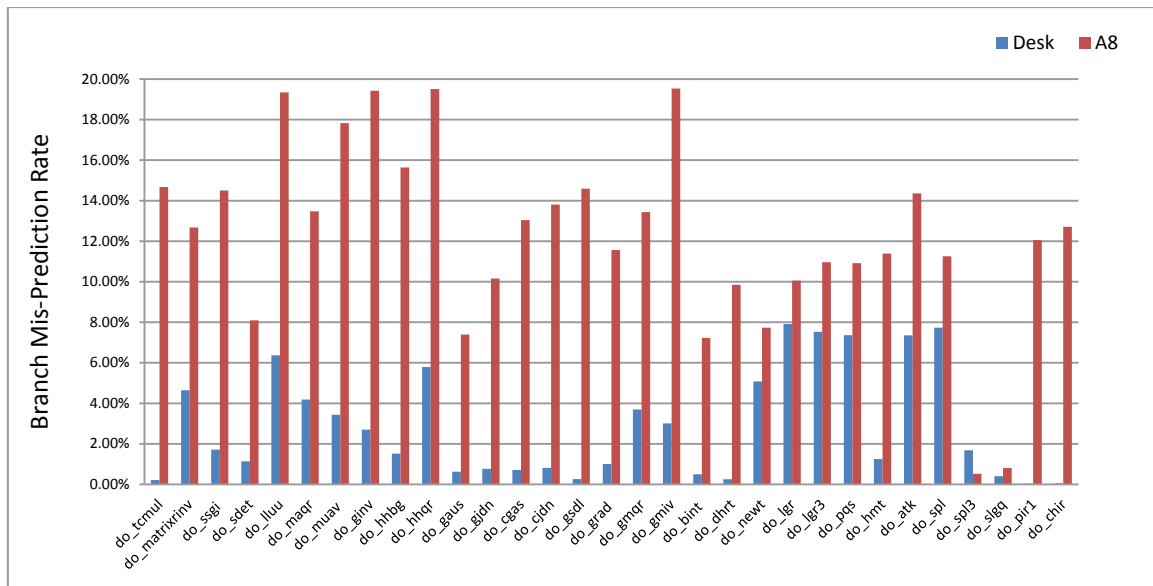


Figure 3-8 Comparison of the Branch Miss Prediction Rate

In the second case the branch prediction ability is considered. The modern embedded processors use pipeline to exploit parallelism and improve performance. Conditional branches in the instruction stream degrade performance by causing pipeline flushes. Branch prediction mechanisms can overcome this limitation by predicting the outcome of the branch before its condition is resolved. As a result, instruction fetch is not interrupted as often and the window of instructions over which ILP (Instruction Level Parallelism) can be exposed increase. Improving branch prediction accuracy is important because the new generation of embedded processors have deeper pipelines, which result in larger mis-prediction penalties/latencies. For example, Cortex A8 is a symmetric, superscalar pipeline for full dual-issue high performance processor with the 13-depth pipeline while the X3450 processor, which is based the new technology of Intel's Nehalem architecture, its total length of pipeline measured by branch mis-prediction delay is 16 cycles⁸¹. In this dissertation, the details of the branch predictor implementation are beyond the scope, only the prediction result and its relationship with the energy consumption are focused. When comparing with the branch prediction capacities, these two processors present a big difference. The prediction accuracy of Cortex A8 ranges from 80% to 99.5% or a mis-prediction rate of 0.5% to 20% while the mis-prediction rate of X3450 ranges from 0.04% to 8%. Several new schemes are proposed to save and restore the predictor state on context switches in order to improve prediction accuracy⁸². For example, the Cortex A8 processor implements a two-level history predictor: the Branch Target Buffer (BTB) and the Global History Buffer (GHB) which are accessed in parallel with instruction fetches. The BTB indicates whether or not the current fetch address will return a branch instruction and its branch target address. On a hit in the BTB a branch is predicted and the GHB is accessed.

The GHB keeps the direction information of branches. X3450 uses a new second-level branch target buffer (BTB) to improve branch predictions in applications by predicting the path of the branch and caching information used by the branch. The new renamed return stack buffers (RSBs) implemented on this architecture store forward and return pointers associated with call and return instructions help to avoid many common return instruction mis-predictions⁸³. The loss in the accuracy can be significant and depends on the predictor type and size. Considering the more critical on-chip resources in ARM Cortex A8 than that in the X3450, the later one behaves better in the accuracy than the former one which is the same as the test results. However, this is a trade-off between the accuracy of the branch predator and its penalty/latency. In the Cortex A8, it is 13 cycles while in the X3450 has a long latency of 35-40 cycles. The minimal latency is measured if the frequency ratio between core and un-core is unity.

All the features mentioned above lead to the different behaviors of energy consumption. Since the high-level models conceal the details of low-level hardware design, their features must be represented by the PMCs. As the analysis before, it shows that the statistics in the PMC can truly present the features of the processor, which means that it is reasonable to use PMC to construct the relationship between the hardware events and the final energy issues.

It is worth mentioning that energy consumption estimation model based on PMCs is also able to apply to software performance examination. Hardware is a cattier of software; software function is ultimately reflected by the hardware activities. Low-power software design advocate increase the instructions-level parallelism (LLP) and decrease the processor idle cycles. Temporal locality and spatial locality are important to good program performance, thus it is worthy concerning the temporal-spatial-friendly algorithms with the awareness of memory access, address translation and control transfer which can be represented by the PMCs.

3.5 Conclusion

It has been realized that the model accuracy is related to the degree of coverage of the application behavior features by PMCs and training samples. If the PMC set cannot monitor all the high energy-related activities, the estimation will be out of control. Similarly, if the training samples do not cover enough application characteristics, the fitting result will be leaded to a devious tendency. PMC-filter finally identifies a set of proper PMCs, which means that, on one hand, those PMCs have the strong relations with the energy consumption to abstract enough features of the applications to construct an accurate model,

and on the other hand, they have as little redundancy as possible to reduce the difficulties of modeling but maintain the accuracy of the estimate result. In addition, the k-fold cross validation method is used to avoid the bias of sampling to further keep the prediction and the generalization.

4 Implementation

4.1 Introduction

The introduced methodology is implemented on a commercial embedded board. In this chapter, the details of the experiment environment are first introduced. The environment includes the hardware platform, the PMC interface implementation, the platform available PMCs and the components of the platform. Then the model procedure is described with the benchmarks and the measurement.

4.2 Experiments Environment

4.2.1 Hardware Platform

Our experiments are carried out on the Beagleboard which is a low-cost, fan-less single computer with laptop-like performance and no-expense expandability⁸⁴. Here, a general description of the design of the BeagleBoard and its overall architecture is given (Figure 4-1).

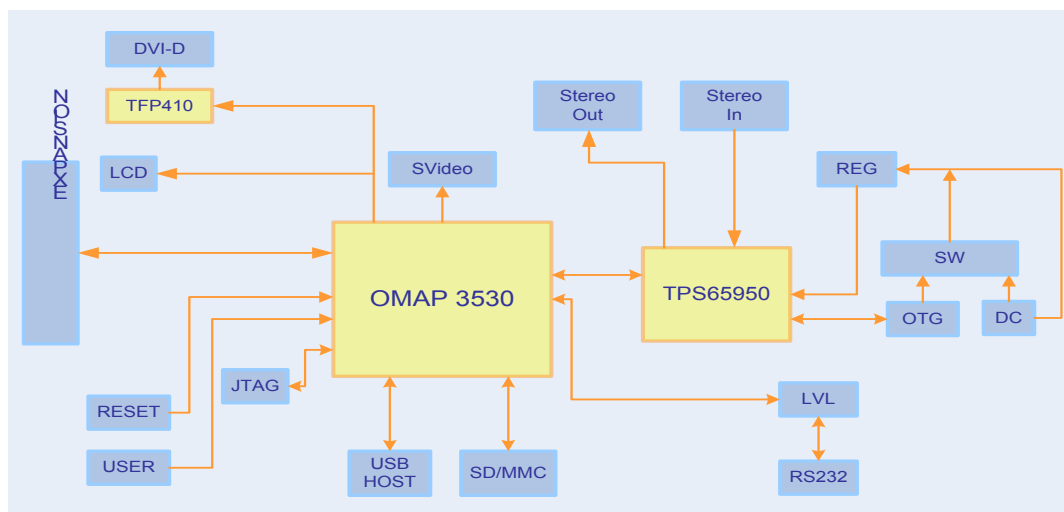


Figure 4-1 BeagleBoard with High-Level Block Diagram

As Figure 4-1 shown, the BeagleBoard has all the functionality of a basic computer. The OMAP3530 includes an ARM Cortex-A8 CPU and a TMS320C64x+DSP. Most operating systems such as Windows CE, Linux, Risc OS and Android have been ported to this ARM CPU. The DSP core is used for accelerating video and audio encoding/decoding. Video out is provided through separate S-Video connection and High-Definition Multimedia Interface (HDMI). A single SD/MMC (Multi Media Card) card slot supports Secure Digital Input Output (SDIO). The I/O device including a USB On-The-Go (OTG), a RS-232 serial connection, a joint test action group (JTAG) connection and two stereo 3.5 mm jacks for

audio in/out are provided. Built-in storage and memory are provided through a Package-on-Package (PoP) chip that includes 256 MB of NAND flash memory and 256 MB of RAM.

The full Beagleboard specification can be easily divided into four categories: computation, storage, I/O plus communication and buses. Note that here the communication unit and the I/O unit are combined together. This is because that there are no physical interfaces of the communication devices on this board, but the network interconnection can be implemented through the USB port. There are several adapters including USB to Ethernet, USB to WiFi and USB to Bluetooth on the market. These devices can easily add Ethernet, WiFi and Bluetooth connectivity to BeagleBoard by using the USB OTG port in the host mode. However, in this dissertation, we are trying to find out a methodology to relate the PMCs to the energy consumption with better generalization. To simplify the work, we will only focus on the Cortex-A8 CPU and the memory system.

4.2.1.1 Cortex-A8 CPU

ARM architecture is widely used in many embedded system design. Due to the energy-efficient characteristics, ARM processor is very applicable in the field of mobile communications, consistent with its primary design goal of the low-cost, high-performance and low power consumption characteristics. The ARM Cortex™-A8 processor has the key features list in the following Table 4-1⁷³:

Feature	Comment
ARM version 7 ISA	Based on the ARMv7 architecture with the standard ARM instruction set + Thumb-2, Jazelle RCT accelerator and media extensions.
L1 Icache and Dcache	16KB, 4way, 64-byte cache line and 128-bit interface.
L2 Cache	The L2 cache and cache controller are embedded in the ARM core.
TLB	Fully associative and separate ITLB with 32 entries and DTLB with 32 entries.
Branch target address cache	512 entries
Enhanced Memory Management Unit	Mapping sizes are 4KB, 64KB, 1MB and 16MB. ARM MMU adds extended physical address ranges.

Table 4-1 ARM Core Key Features

4.2.1.2 Storage Hierarchy

The Micron PoP (Package on Package) memory is used on the Rev C4 BeagleBoard and located on top of the processor. It provides a 256MB 16-bit NAND and a 256 MB 32-bit

MDDR SDRAM. There is no other memory devices on the BeagleBoard but additional memory can be added by SD/MMC slot, USB Thumb driver or hard driver if there is relevant drivers are supported in the OS. Beagleboard has a memory hierarchy with several layers. Each layer is a different type of memory with its own speeds, sizes and usages. Some layers can be physically integrated on the processor such as the registers and the level 1 cache. Other types of memory can be located outside of the processor such as ROM, lever 2 cache and the main memory. The secondary/tertiary memory is the memory that connected to the board such as floppy drivers or MMC.

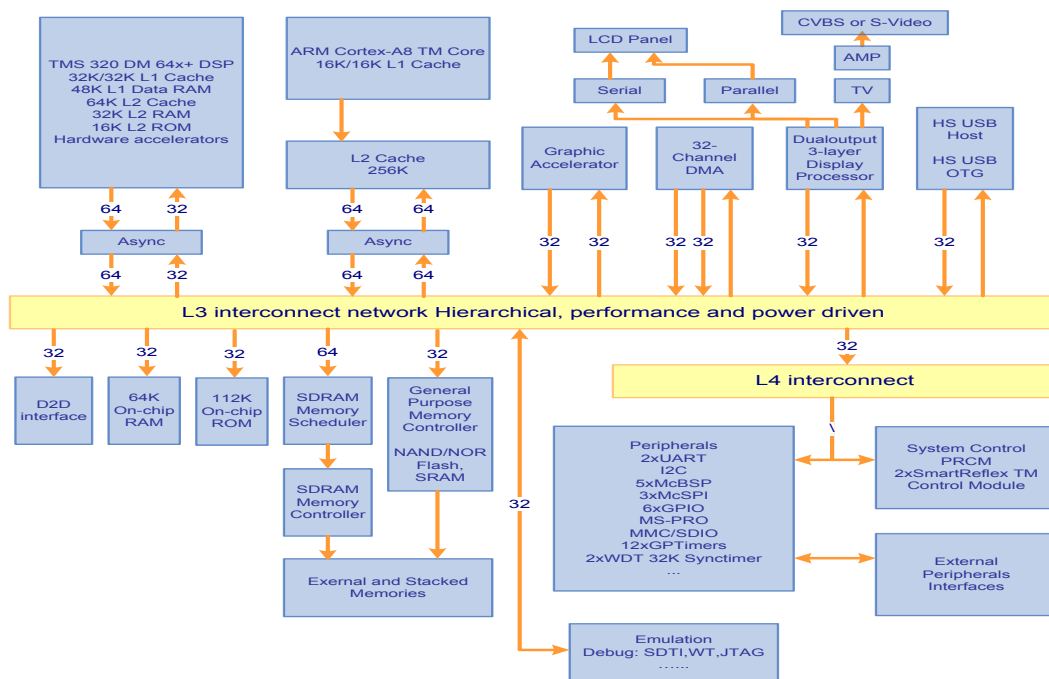


Figure 4-2 OMAP3530 Block Diagram⁸⁴

From Figure 4-2 we can see the memory hierarchy of OMAP3530 divided into four layers. First layer is the level 1 cache which includes data cache and instruction cache. In this level, data and instruction are separated. It is internal to the CPUs. It concerns data exchange with the internal Level 1 cache memory subsystem, and it is the closest memory to the microprocessor unit (MPU) core and the IVA2.2 core. Second layer is the level 2 cache which is shared by both data and instruction. Level 2 Cache can be accessed by both DSP subsystem and the MPU subsystem. The third layer includes on-chip RAM, on-chip ROM, SDRAM and NAND flash while the forth layer is the external memories such as SD card. The third and the forth layers also include the according interconnections and the controllers. These two layers enable communication among the modules and subsystems in the device. Layer 3 handles many types of data transfers, especially exchanges with system-on-chip/external memories. It transfers data with a maximum width of 64 bits from the

initiator to the target. Layer 4 is composed with the different peripheral interconnects and handles data transfers to peripherals. In addition, it can send an acknowledge signal to change the peripherals into an idle state.

4.2.2 Platform PMCs

4.2.2.1 Platform Available PMCs

There are four PMCs in the Cortex A8 processor. These PMCs can be accessed in system control coprocessor (CP15) space. The purpose of CP15 is to control and provide status information for the functions implemented in the processor. Its main functions of the system include the overall system control and configuration, the cache configuration and management, the memory management unit configuration and management, the preloading engine for L2 cache and the system performance monitoring, which is one function we are interested in.

In the dissertation, PMCs are configured to monitor and count system events such as cache misses, TLB misses, pipeline stalls and other related features to enable system developers to profile the energy-related behaviors of the processor when it executes different applications. There are many situations where PMCs integrated into the core are valuable for applications and for application development. Moreover, there is also the possibility to enable interaction with external monitoring. An implementation might consider additional enhancements such as:

- Provision of a set of events which can be exported onto the system buses. However, for very high frequency operation, this might cause an unacceptable timing requirement. In addition, the different clock frequency between the core and the buses (or other subsystems) may become a problem. A suitable approach might be to edge-detect changes in the signals and to use those changes to increase a counter.
- Provision of the memory-mapped access to the PMCs to monitor and detect the memory performance, therefore, there is a more effective and accurate method to know how good the spatial locality is.
- Provision of implementation specific events. Processor architecture usually defines a set of events to be used, however, there is always a large space reserved for implementation defined events. There is no requirement to implement full set of events.

In the Cortex-A8 architecture, the PMC registers are mapped into part of the CP15 register. Figure 4-3 shows the recommended performance monitor registers encodings of the register C9, which is a register in the system used to control coprocessor CP15 and the reserved encodings for implementation defined performance monitors.

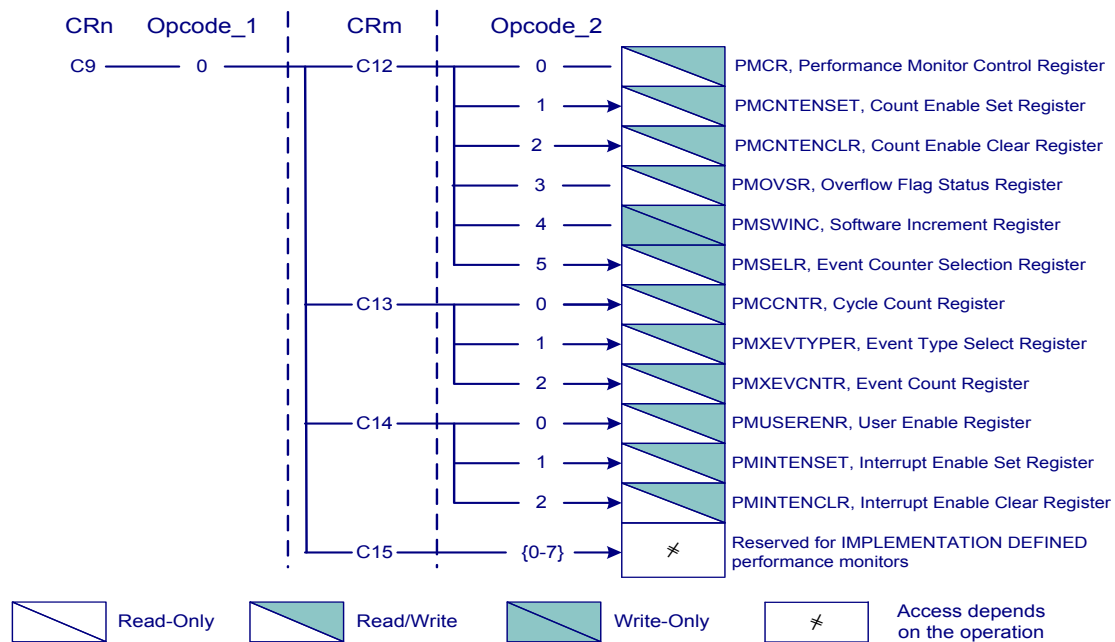


Figure 4-3 Recommended CP15 Performance Monitors⁷³

Here, the CR_n is the destination coprocessor register, $Opcode_1$ is a coprocessor-specific opcode, CR_m is an additional destination coprocessor register, and the $Opcode_2$ is a coprocessor-specific opcode. If omitted, $Opcode_2$ is assumed to be 0.

The purpose of the PMNC (Performance Monitor Control) Register is to control the operation of the four PMCs and the cycle counter register. Enabling or disabling any of the PMCs, the CNTENSET/CNTENCLR register are needed. The enable-bit in both registers that reads as 0 indicates the counter is disabled while reads as 1 indicates the enabled counter. When writing this register, the enable-bit written into CNTENSET with the value of 0 is ignored while written with the value of 1 indicates to enable the counter. Similarly, written into CNTENCLR with the value of 0 cannot update the counter state while written with the value of 1 clears the enable-bit to 0 to disable the counter. The purpose of the PMC PMOVSr (Overflow flag Status Register) is to enable or disable any of the PMCs to produce the overflow. When read this PMC, any overflow flag of value 0 indicates the counter has not overflowed while a value of 1 indicates the counter has overflowed. When write into this PMC, any overflow flag written with a value of 0 is ignored to keep the current state while the value of 1 clears the counter overflow to 0. The SWINCR (Software Increment) register is

used to increment the count of a corresponding PMC. When write into the specific bit of this PMC, the value of 1 increments the specified counter while the value of 0 does nothing. To select one PMC, writing into the last five bits of the PMNXSEL (Performance Counter Selection) register with the corresponding number. The CCNT (Cycle Count) register is to count the number of clock cycles since the PMC was reset. The Cortex A-8 processor has four registers (PMCNT0 – PMCNT3) to count instances of an event selected by the PMC EVTSEL. Each PMCNT monitor an event which is selected by writing into the EVTSEL (Event Selection) register with the according value. Note that accessing to the PMCs in user space need to enable the user mode of the PMCs. The USEREN (User Enable) register is used to control this configuration. The purpose of the PMC INTENS/ INTENC (Interrupt Enable Set/ Interrupt Enable Clear) is to determine if any of the PMCs, PMCNT0-PMCNT3 and CCNT, generates an interrupt on overflow. When reading this PMC, if the overflow-enable bit is read as 0, it indicates the interrupt overflow flag is disabled. On the contrary, the bit reading as 1 indicates the interrupt overflow flag is enabled. When write into this PMC, any interrupt overflow enable bit written with a value of 0 is ignored while any interrupt overflow enable bit written with a value of 1 sets/clears the interrupt overflow enable bit.

These PMCs can be accessed by reading or writing CP15 with the MRC and MCR instructions, respectively. For example, to access the PMNC register, read or write CP15 with:

```
MRC p15, 0, <Rd>, c9, c12, 0; and
```

```
MCR p15, 0, <Rd>, c9, c12, 0;
```

“MRC” instruction transfers a co-processor register to an ARM register with the format:

```
MRC <co-pro>, <op>, <ARM reg>, <co-pro reg>, <co-pro reg2>, <op2>
```

Here <ARM reg> performances as the destination register, < co-pro reg > and <co-pro reg2> are two source registers. The < co-pro reg> register is written to <ARM reg> by using operation <op> while < co-pro reg2 > register is written by using operation2.

“MCR” instruction has the same format but it use to transfers an ARM register to a co-processor register.

The basic use of the PMCs is like this:

The PMC PMNC controls the operation of the PMCs, with one register used to set up each counter. They specify the events to be counted, how they should be counted and the privilege levels at which counting should take place. The four PMCNTx contain the event counts for the selected events being counted. The MRC instruction can be used by programs or procedure running at any privilege level to read these counters. One PMC is started by writing valid setup information in the PMC CNTENS and PMC EVTSEL. The counters can be stopped by clearing the enable counters flag or by clearing all the bits in the CNTENC. The Cortex A8 processor provides the option of generating a local APIC interrupt when a PMC overflows. This mechanism is enabled by setting the interrupt enable flag in PMC INTENS. The primary use of this option is for statistical performance sampling. An event monitor application utility or another application program can read the information collected for analysis of the performance of the profiled application.

4.2.2.2 PMCs Interface Implementation

Although PMCs have been widely implemented in most modern processors, the direct accesses are limited to the privileged modes, which always need a specific driver or interface for user space accessing. PMCs can be accessed from the user space by using several mature high-level interfaces. This means that users can obtain the fine grain through their own applications without too much interrupt of the operating system. Moreover, many applications written in a high-level language can run on multiple platforms. Therefore, the interface implementations of these proprietary counters also need to be portable. However, there are only a few APIs that allow access to these counters, and most of them are poorly documented, unstable, or unavailable. In addition, performance metrics may have different definitions and different programming interfaces on different platforms. Multiplatform interface such as PAPI can mask some of the differences among platforms. PAPI is the abbreviation of Performance Application Programming Interface, which is a specification of a cross-platform interface to hardware PMC on modern microprocessors^[63]. This dissertation uses PAPI as the interface of PMCs.

(a) PAPI's Characteristic

PAPI can provide a set of API's for accessing PMCs from applications. A considerable characteristic of PAPI is that it includes both high-level and low-level sets of interfaces for accessing PMCs. The high-level interface includes start, stop and read sets of PMCs, and other simple operations which can obtain accurate measurement of applications. The fully programmable low-level interface provides the possibility to control the counters. PAPI has been implemented on a number of Linux platforms. The latest release now builds

for using the libpfm4 interface by default and provides new supports for AMD Bobcat, Intel SandyBridge and ARM Cortex A9 and A8 which is the MCU in our embedded system.

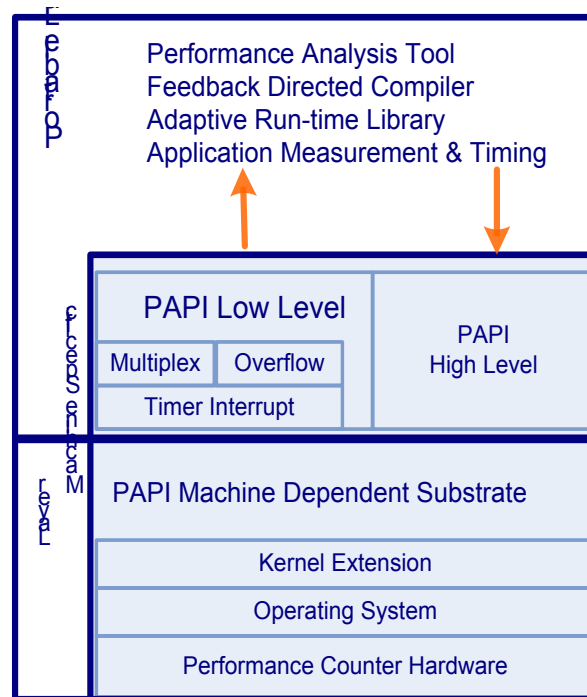


Figure 4-4 PAPI Architecture

The PAPI architecture uses a layered approach including portable and machine-dependent layers, as shown in Figure 4-4⁶³. The highest layer is a portable one consists of the high and low level PAPI interfaces. This layer is completely machine independent and requires little porting effort. It contains all of the API functions as well as numerous utility functions such as perform state handling, memory management, data structure manipulation and thread safety guarantee. In addition, this layer provides advanced functionalities such as event profiling and overflows handling which is not always provided by the operating system. The highest layer calls the substrate, which is the internal PAPI layer. It handles the machine-dependent specifics of accessing the counters. The substrate uses appropriate methods to facilitate counter access. The machine specific layer defines and exports a machine independent interface to machine- dependent functions and data structures. These functions are defined in the substrate layer, which uses kernel extensions, OS calls, or assembly language to access the PMCs. PAPI chooses the most efficient and flexible one of the three methods if they are all available. Moreover, most modern microprocessors have a very limited number of events that can be counted simultaneously, thus only a few events can be measured at once. This limitation severely restricts the amount of performance information that the user can gather during a single run. It is not a good solution to repeat large applications which can run days or weeks several times to gather enough information

for energy profiling. This limitation can be overcome by multiplexing the counter hardware. Multiplexing usage provides the users with the view that many more hardware events are countable simultaneously. Some platforms may support multiplexing of a counter by subdividing the usage of the PMCs over time, which gives the user the illusion of a larger number of registers. On the ARM platforms, PAPI implements the multiplexing by swapping events in and out of the counters based on a timer interrupt. It provides the possibility of multiplexing implementation through the interval timer if the operating system or kernel-level PMC interface does not support multiplexing. It is unavoidable to cause a small amount of overhead and adverse effects of the accuracy; however, the multiplexing method has been proven useful in commercial kernel level PMC interfaces.

One note is that there are two kinds of events PAPI supports. One are “preset” events that try to abstract away all hardware differences. Things like “PAPI_TOT_CYC” which should give you total cycles no matter where the user runs it: Intel, AMD, ARM, etc. Then PAPI has another idea of “native events” which are the underlying events. These will differ from machine to machine. So a native event such as “RETIRED_INSTRUCTIONS” on an AMD machine will not work on Intel or ARM, even though the preset event “PAPI_TOT_INS” will. The PAPI library names a number of pre-defined or preset events. This set is a collection of events typically found in many CPUs that provide performance counters. Usually, a PAPI preset event name is mapped onto one of the countable native events on each hardware platform, but in rare cases, to maintain the generality, a preset event is calculated by more than one native event.

(b) PAPI’s Implementation of PMC Driver

PAPI can be considered as a black-box that masks the details of the variance of different processor architectures. PAPI tries to provide a uniform environment across platform. It implements some features that the hardware does not support in software, therefore the interface of PAPI remains constant, but how it is implemented can vary. So the usage of the PMCs through PAPI is portable and machine-dependent. PAPI supports two PMC drivers, “perfctr” and “perf_event”, with the same event names on each, so it does not matter what underlying driver used.

These PMC drivers work as a kernel patch on Linux, but one thing need to pay attention is that one PMC driver may not support all the processors. For the processors in the ARM family, the following Table 4-2 lists the supports of PMC driver in detail.

Name	Family	Perf_event	Perfctr	Libpfm3	Libpfm4	PAPI
XScale 1	ARMv5	2.6.38	Yes	No	No	No
XScale 2	ARMv5	2.6.38	Yes	No	No	No
N/A*	ARMv6	2.6.34	No	No	No	No
Cortex A5	ARMv7	3.1	No	No	No	No
Cortex A8	ARMv7	2.6.34	No	No	Yes	Yes
Cortex A9	ARMv7	2.6.34	No	No	Yes	Yes
Cortex A15	ARMv7	3.1	No	No	Soon	Soon

Table 4-2 Supports of PMC Drivers on ARM Family (Last Update 6th 2011)⁸⁵

The “perf_event” column tells which Linux kernel first supported the chip using the perf_event subsystem. The “perfctr” column says whether the current “perfctr” patch supports this chip. The “libpfm3” and “libpfm4” columns say whether those respective libraries support the events for the chip. The “PAPI” column says whether the current version of PAPI supports a processor. PAPI has the possibility to choose which PMC driver to use. Although “perfctr” is the one most commonly used, unfortunately, it does not support the Cortex A8, so for this processor, PAPI is implemented with the perf_event driver with Linux kernel 2.6.34 or later version.

The PMC “perf_events” driver was first merged into the Linux kernel in version 2.6.31. The driver accessed PMCs through special file descriptors. Each virtual counter has its own file descriptor and is enabled or disabled through ioctl or prctl. When a PMC is disabled, it does not count but maintain its previous count value. PMCs work in two ways: counting and sampling. A “counting” PMC is one that is used for counting the number of a specific event that occurs. The application needs to read these counters to obtain their values. A read operation on a PMC returns the current value of the counter as an unsigned 64-bit integer. “Perf_event” was specifically designed to hide which events end up in which counters. In general, if the multiplexing is not used, perf_event should be almost consistent about which PMC gets which event. Sampling PMCs periodically interrupt the processor to collect the execution information. In this dissertation, we use PMCs as the counting way.

* The name is not provided in the reference.

Note that the “libpfm4” is not a driver. It is a library for taking event names and converting them to values used by the various drivers. It is used by PAPI to translate the event names for use by the perf_event driver. This means that “libpfm4” only provides some assistant methods for PMC driver using. Part of the addition is that there is another driver, “perfmon2”, which PAPI also supports as an external patch that needed to be applied for using. It was written by the same person who wrote libpfm3 and libpfm4, however, libpfm4 can be used independent without this driver⁷².

In our case, we choose to use the Linux kernel 2.6.34. In order to use the PMCs, we need to do some specific configuration when kernel configures (Figure 4-5).

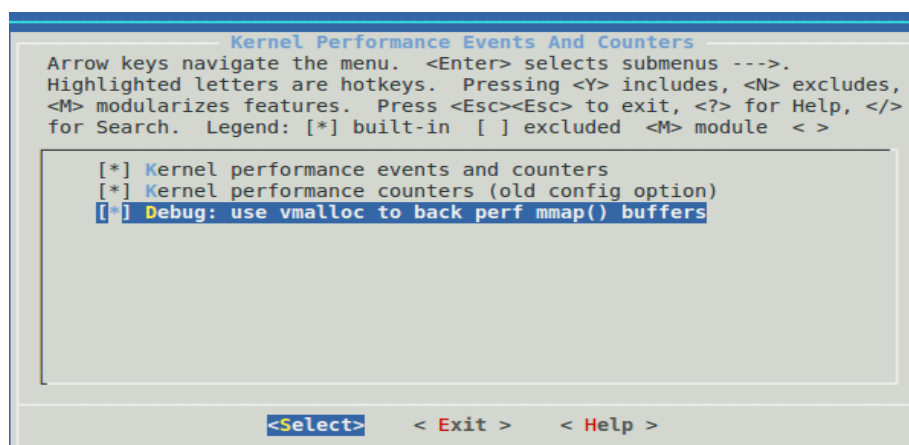


Figure 4-5 Enable Kernel Support for PMCs

The Linux Performance events sub-system provides an abstraction of these software and hardware events by using interface tools. The first step is to set the "CONFIG_PERF_EVENTS" during the kernel configuration to enable the kernel supports for various performance events.

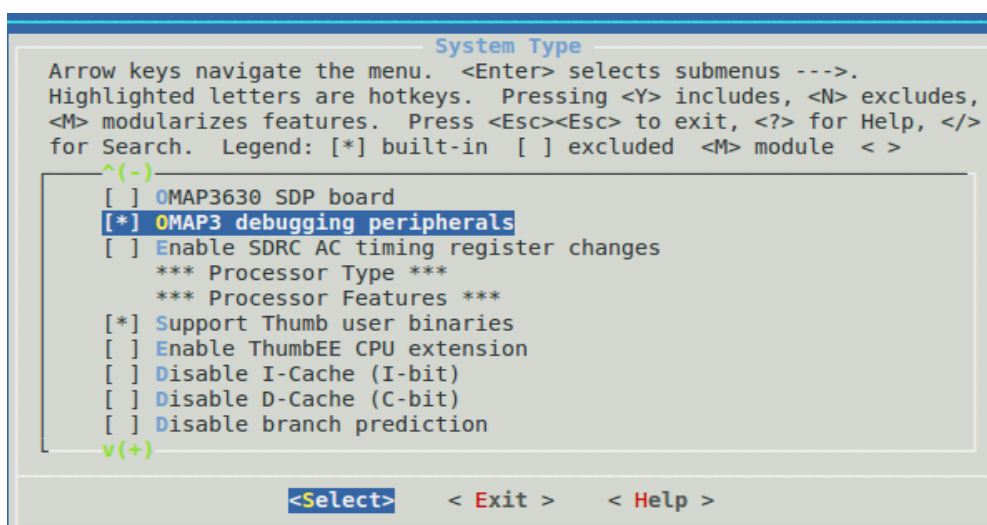


Figure 4-6 Enable Debugging Hardware

The second step (Figure 4-6) is to include the OMAP3 debugging peripherals to enable the according hardware on OMAP3530.

(c) PAPI's Usage

There are two ways to use the functions provided by PAPI. The main and simplest approach of PAPI to event counting is the caliper mode which reads the PMCs before and after a performance-critical region of code. Another approach is the performance counter sampling. In this method, one or several specific are configured to have their sampling thresholds. One a PMC's count value exceed its threshold, it causes an interrupt to record all the PMCs' values and reset all the PMCs. One application could be interrupted several times. Since we will use some home-made functions won't run in a long time, we can simply insert the PMC start function before execute the application and the PMC stop as well as the PMC read functions once the application finishes. Figure 4-7 shows the producer to use PAPI.

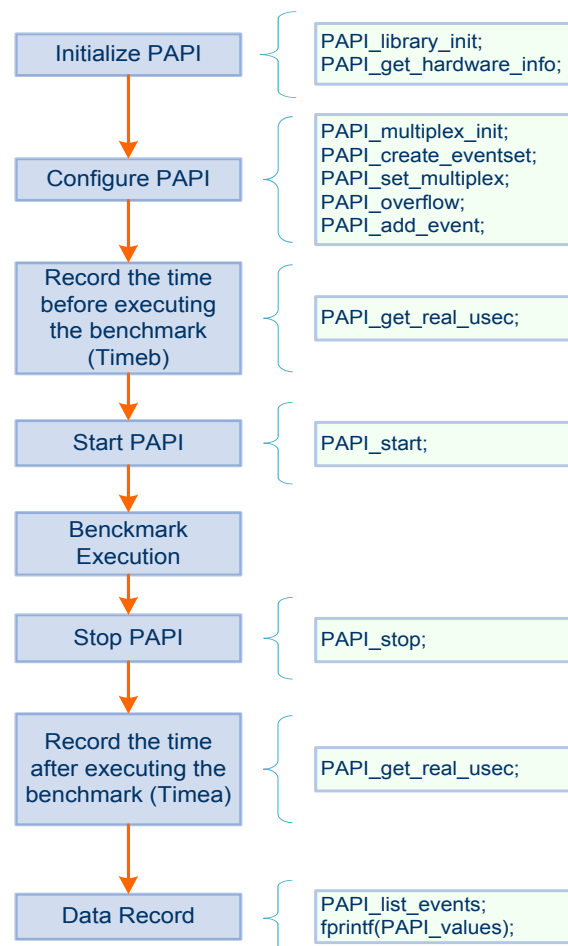


Figure 4-7 Procedure of Using PAPI

The main PAPI functions we use are:

- `PAPI_library_init(int version)`: Initialize the PAPI library;
- `PAPI_get_hardware_info(void)`: To get information about the system hardware. Such as the number of CPUs in the entire system, cupid family, PAPI memory hierarchy description and cycle time of this CPU.
- `PAPI_multiplex_init(void)`; To initialize multiplex support in the PAPI library.
- `PAPI_create_eventset(int *EventSet)`: To create a new empty PAPI event set.
- `PAPI_set_multiplex (int EventSet)`: Convert the already existed event set to a multiplexed event set;
- `PAPI_get_real_usec(void)`: Return the total number of microseconds since some arbitrary starting point. This function is inserted before and after one benchmark to get its the execution time;
- `PAPI_overflow(int EventSet, int EventCode, int threshold, int flags, PAPI_overflow_handler_t handler)`: Set up an event to begin registering overflows;
- Note that currently, PAPI only supports thread level monitoring. That means that PAPI will not inherit the counting information or values from the parent threads. This feature is helpful to distinguish individual thread, it does not confuse the parent thread with the child thread.
- `PAPI_start (int EventSet)`: Start counting hardware events in an event set;
- `PAPI_stop (int EventSet, long long *values)`: Stop counting hardware events in an event set and return current events;

The accuracy of the model depends on the features of the application abstract by the energy-related events. Thus there is no reason to delete any of the events at the beginning. The event set is filled up with all the available non-derived events in the experiment platform. The following functions are used:

- `PAPI_get_event_info (int EventCode, PAPI_event_info_t *info)`: Get the name and description for a given preset or native event code;

- `PAPI_add_event(int EventSet, int Event)`: Add single PAPI preset or native hardware event to an event set. Not that the native/preset events may be the derived events, but we only add those non-derived events;
- `PAPI_enum_event (int * EventCode, int modifier)`: Return the event code for the next available present or native event.

Here two functions are used to label the data get from PAPI:

- `PAPI_list_events(int EventSet, int * Events, int * number)`: Decomposes an event set into the hardware events it contains;
- `PAPI_event_code_to_name(int EventCode, char * EventName)`: To translate a 32-bit integer PAPI event code into an ASCII PAPI event name. Either preset event codes or native event codes can be passed to this routine. Native event codes and names differ from platform to platform.

The simple “switch-case” pattern is used to execute a benchmark each time as shown in the Figure 4-8. The name of the benchmark is passed through the command line arguments.

```

switch(funcN)
{
    case(Dir_oper_Int_test):    do_flops_M(Dir_oper_Int_test,atoi(argv[3]));    break;
    case(Dir_oper_Flo_test):    do_flops_M(Dir_oper_Flo_test, atoi(argv[3]));    break;
    case(do_conv):              for( i=0; i<atoi(argv[3]); i++)    do_conver ();    break;
    case(Dir_oper_Flo):         do_flops_M(Dir_oper_Flo,atoi(argv[3]));    break;
    case(Dir_oper_Int):         do_flops_M(Dir_oper_Int, atoi(argv[3]));    break;
    case(do_sort):              sort(atoi(argv[3]),argv[4]);    break;
    case(MM_Miss_oper):         do_flops_M(MM_Miss_oper,atoi(argv[3]));    break;
    case(do_misse):             do_misses(atoi(argv[3]),atoi(argv[4]));    break;
    case(do_Slgq):              do_SLGQ(atoi(argv[3]),(double)atof(argv[4]),
                                (double)atof(argv[5]));    break;
    case(do_Slq3):              do_SLQ3(atoi(argv[3]),atoi(argv[4]),
                                (double)atof(argv[5]),(double)atof(argv[6]));    break;
    case(do_sqrt):              do_usqrt (atoi(argv[3]));    break;
    case(do_Greedy):            do_greedy(atoi(argv[3]),(double)atof(argv[4]));    break;
    case(do_Rank):              do_RANK(atoi(argv[3]),atoi(argv[4]));    break;
    case(do_Dhrt):              do_DHRT((double)atof(argv[3]),(double)atof(argv[4]),
                                (double)atof(argv[5]),atoi(argv[6]));    break;
    case(do_Queen):             do_queen(atoi(argv[3]));    break;
    case(do_Lgr):               do_LGR(atoi(argv[3]),atoi(argv[4]),
                                (double)atof(argv[5]));    break;
    case(do_Pqs):               do_PQS(atoi(argv[3]),atoi(argv[4]),
                                (double)atof(argv[5]),(double)atof(argv[6]));    break;
    case(do_Lgr3):              do_LGR3(atoi(argv[3]),atoi(argv[4]),
                                (double)atof(argv[5]),(double)atof(argv[6]));    break;
    case(do_Spl):               do_SPL(atoi(argv[3]),atoi(argv[4]),atoi(argv[5]),
                                (double)atof(argv[6]),(double)atof(argv[7]));    break;
    case(do_Hmt):               do_HMT(atoi(argv[3]),atoi(argv[4]),
                                (double)atof(argv[5]),(double)atof(argv[6]));    break;
    case(do_Pir1):              do_PIR1(atoi(argv[3]),atoi(argv[4]));    break;
    case(do_solcubic):           do_solcecubic((double)atof(argv[3]),(double)atof(argv[4]),
                                (double)atof(argv[5]),(double)atof(argv[6]));    break;
    case(do_Chir):              do_CHIR(atoi(argv[3]),atoi(argv[4]));    break;
    case(do_lis):               do_LIS(atoi(argv[3]));    break;
    case(do_Atk):               do_ATK(atoi(argv[3]),atoi(argv[4]),
                                (double)atof(argv[5]),(double)atof(argv[6]));    break;
    case(MM_Miss_oper_test):     do_flops_M(MM_Miss_oper_test, atoi(argv[3]));    break;
    case(do_LIUu):              do_LLUIU(atoi(argv[3]));    break;
    case(MM_oper):              do_flops_M(MM_oper, atoi(argv[3]));    break;
    case(MM_Miss_oper2):         do_flops_M(MM_Miss_oper2, atoi(argv[3]));    break;
    case(MM_oper_test):         do_flops_M(MM_oper_test,atoi(argv[3]));    break;
    default:                    do_flops_M (Dir_oper_Int, atoi(argv[3]));
}

```

Figure 4-8 “Switch-case” Pattern to Execute a Benchmark

The other modifications have been done in the makefile. In software development under the Linux, Make is a tool that automatically builds and maintains the programs and libraries from source code by reading files called makefiles which specify how to derive the target program. Compiling the source code files can be tedious, especially when you want to include several source files and have to type the compiling command every time you want to do it. Makefiles are special format files the together with the make utility will help you to automatically build and manage your projects. How to write makefile is beyond the scope of this thesis, here we only introduce some modifications. The “multiplex2.c” is the main function to run one benchmark every execution by calling those benchmarks as the sub-functions. So we need to identify all the object files of these benchmarks (Figure 4-9), add them as the target when you want to compile the multiplex2.c file (Figure 4-10), and indicate the path to find the source code of these benchmarks (Figure 4-11).

```
NUTILIBS= ./testlib/do_test.o ./testlib/test_utils.o ./testlib/dummy.o ./testlib/do_sorts.o ./testlib/do_arrayMulti.o ./testlib/do_FFT.o ./testlib/do_greedy.o ./testlib/do_Huffman.o ./testlib/do_LIS.o ./testlib/do_queen.o ./testlib/do_root.o ./testlib/basicmath.o ./testlib/do_dijkstra.o ./testlib/do_patricia.o ./testlib/do_TcMul.o ./testlib/do_MatrixRInv.o ./testlib/do_SSgi.o ./testlib/do_SDet.o ./testlib/do_RANK.o ./testlib/do_LL UU.o ./testlib/do_MAQR.o ./testlib/do_MUAV.o ./testlib/do_GINV.o ./testlib/do_GSDL.o ./testlib/do_RAD.o ./testlib/do_GMQR.o ./testlib/do_GMIV.o ./testlib/do_GAUS.o ./testlib/do_BINT.o ./testlib/do_GJDN.o ./testlib/do_CGAS.o ./testlib/do_CJDN.o ./testlib/do_DHRT.o ./testlib/do_NEWT.o ./testlib/do_ATKN.o ./testlib/do_PQRT.o ./testlib/do_HHBG.o ./testlib/do_HHQR.o ./testlib/do_QRRT.o ./testlib/do_SRRRT.o ./testlib/do_LGR.o ./testlib/do_LGR3.o ./testlib/do_PQS.o ./testlib/do_HMT.o ./testlib/do_ATK.o ./testlib/do_SPL.o ./testlib/do_SPL3.o ./testlib/do_SLQ3.o ./testlib/do_SLGQ.o ./testlib/do_PIR1.o ./testlib/do_CHIR.o ./testlib/inputOutput.o

#./testlib/do_dhry1.o ./testlib/cpuid.o ./testlib/cpuidc.o
```

Figure 4-9 Identify all the Object Files of Benchmarks

```
multiplex2: multiplex2.c $(NUTILIBS) $(PAPILIB)

$(CC) $(INCLUDE) $(CFLAGS) $(TOPTFLAGS) multiplex2.c $(NUTILIBS) $(PAPILIB) $(LDFLAGS) -lm -lpthread -o $@
```

Figure 4-10 Add Compile Targets

The makefile defines the path to the include file and lib directories, and places the object files in an obj subdirectory within the source directory. Here the items, which are in form of “\$(*)”, are the defined variables which are in the name of compiler parameters. The other items begin with “-l” stand for the local library, whose name are given by the words followed “-l”. “-lm” is the math library to support some complex math functions such as pow(), sqrt(), floor(), ect. “-lpthread” is a library that support the multi-threads. Note that there is an item named “\$(PAPILIB)”, this is the path of the library of PAPI. Since it is not a local library,

it is required to identify the path manually; otherwise the compiler does not know where to find this library.

```

../testlib/test_utils.o: ../testlib/test_utils.c ../testlib/papi_test.h ../testlib/test_utils.h
cd ../testlib && $(MAKE)
../testlib/do_loops.o: ../testlib/do_loops.c ../testlib/papi_test.h ../testlib/test_utils.h
cd ../testlib && $(MAKE)
../testlib/do_test.o: ../testlib/do_test.c ../testlib/papi_test.h ../testlib/test_utils.h
cd ../testlib && $(MAKE)
../testlib/dummy.o: ../testlib/dummy.c
cd ../testlib && $(MAKE)
../testlib/basicmath.o: ../testlib/basicmath.c ../testlib/papi_test.h ../testlib/snipmath.h ../testlib/sniptype.h ../testlib/round.h ../testlib/pi.h
cd ../testlib && $(MAKE)
../testlib/do_RANK.o: ../testlib/do_RANK.c ../testlib/papi_test.h
cd ../testlib && $(MAKE)
../testlib/do_LLUI.o: ../testlib/do_LLUI.c ../testlib/papi_test.h
cd ../testlib && $(MAKE)
../testlib/do_ATKN.o: ../testlib/do_ATKN.c ../testlib/papi_test.h
cd ../testlib && $(MAKE)
../testlib/do_PQRT.o: ../testlib/do_PQRT.c ../testlib/papi_test.h
cd ../testlib && $(MAKE)
../testlib/do_LGR.o: ../testlib/do_LGR.c ../testlib/papi_test.h
cd ../testlib && $(MAKE)
../testlib/do_PQS.o: ../testlib/do_PQS.c ../testlib/papi_test.h
cd ../testlib && $(MAKE)
../testlib/do_HMT.o: ../testlib/do_HMT.c ../testlib/papi_test.h
cd ../testlib && $(MAKE)
../testlib/do_ATK.o: ../testlib/do_ATK.c ../testlib/papi_test.h
cd ../testlib && $(MAKE)
../testlib/do_SPL.o: ../testlib/do_SPL.c ../testlib/papi_test.h
cd ../testlib && $(MAKE)
../testlib/do_SPL3.o: ../testlib/do_SPL3.c ../testlib/papi_test.h
cd ../testlib && $(MAKE)
../testlib/do_SLQ3.o: ../testlib/do_SLQ3.c ../testlib/papi_test.h
cd ../testlib && $(MAKE)
../testlib/do_PIR1.o: ../testlib/do_PIR1.c ../testlib/papi_test.h
cd ../testlib && $(MAKE)
../testlib/do_CHIR.o: ../testlib/do_CHIR.c ../testlib/papi_test.h
cd ../testlib && $(MAKE)
../testlib/do_sorts.o: ../testlib/do_sorts.c ../testlib/papi_test.h ../testlib/test_utils.h
cd ../testlib && $(MAKE)
../testlib/do_greedy.o: ../testlib/do_greedy.c ../testlib/papi_test.h ../testlib/test_utils.h
cd ../testlib && $(MAKE)
../testlib/do_LIS.o: ../testlib/do_LIS.c ../testlib/papi_test.h ../testlib/test_utils.h
cd ../testlib && $(MAKE)
../testlib/do_queen.o: ../testlib/do_queen.c ../testlib/papi_test.h ../testlib/test_utils.h
cd ../testlib && $(MAKE)

```

Figure 4-11 Indicate Source Code Path

Similar, all the object files of the benchmarks are generated and managed by another makefile which manage all the benchmarks. Due to the various functions of benchmarks, each benchmark may need different compiler or compiler options (Figure 4-12).

```
do_LLUI.o: do_LLUI.c papi_test.h test_utils.h
    $(CC) $(INCLUDE) $(CFLAGS) $(TOPTFLAGS) -c do_LLUI.c -lm
do_ATKN.o: do_ATKN.c papi_test.h test_utils.h
    $(CC) $(INCLUDE) $(CFLAGS) $(TOPTFLAGS) -c do_ATKN.c -lm
do_PQRT.o: do_PQRT.c papi_test.h test_utils.h
    $(CC) $(INCLUDE) $(CFLAGS) $(TOPTFLAGS) -c do_PQRT.c -lm
do_LGR.o: do_LGR.c papi_test.h test_utils.h
    $(CC) $(INCLUDE) $(CFLAGS) $(TOPTFLAGS) -c do_LGR.c -lm
do_LGR3.o: do_LGR3.c papi_test.h test_utils.h
    $(CC) $(INCLUDE) $(CFLAGS) $(TOPTFLAGS) -c do_LGR3.c -lm
do_PQS.o: do_PQS.c papi_test.h test_utils.h
    $(CC) $(INCLUDE) $(CFLAGS) $(TOPTFLAGS) -c do_PQS.c -lm
do_HMT.o: do_HMT.c papi_test.h test_utils.h
    $(CC) $(INCLUDE) $(CFLAGS) $(TOPTFLAGS) -c do_HMT.c -lm
do_ATK.o: do_ATK.c papi_test.h test_utils.h
    $(CC) $(INCLUDE) $(CFLAGS) $(TOPTFLAGS) -c do_ATK.c -lm
do_SPL.o: do_SPL.c papi_test.h test_utils.h
    $(CC) $(INCLUDE) $(CFLAGS) $(TOPTFLAGS) -c do_SPL.c -lm
do_SPL3.o: do_SPL3.c papi_test.h test_utils.h
    $(CC) $(INCLUDE) $(CFLAGS) $(TOPTFLAGS) -c do_SPL3.c -lm
do_SLQ3.o: do_SLQ3.c papi_test.h test_utils.h
    $(CC) $(INCLUDE) $(CFLAGS) $(TOPTFLAGS) -c do_SLQ3.c -lm
do_SLGQ.o: do_SLGQ.c papi_test.h test_utils.h
    $(CC) $(INCLUDE) $(CFLAGS) $(TOPTFLAGS) -c do_SLGQ.c -lm
do_PIR1.o: do_PIR1.c papi_test.h test_utils.h
    $(CC) $(INCLUDE) $(CFLAGS) $(TOPTFLAGS) -c do_PIR1.c -lm
do_CHIR.o: do_CHIR.c papi_test.h test_utils.h
    $(CC) $(INCLUDE) $(CFLAGS) $(TOPTFLAGS) -c do_CHIR.c -lm
do_test.o: do_test.c papi_test.h test_utils.h
    $(CC) $(INCLUDE) $(CFLAGS) $(TOPTFLAGS) -c do_test.c -lm
do_sorts.o: do_sorts.c papi_test.h test_utils.h
    $(CC) $(INCLUDE) $(CFLAGS) $(TOPTFLAGS) -c do_sorts.c -lm
do_arrayMulti.o: do_arrayMulti.c papi_test.h test_utils.h
    $(CC) $(INCLUDE) $(CFLAGS) $(TOPTFLAGS) -c do_arrayMulti.c -lm
do_LIS.o: do_LIS.c papi_test.h test_utils.h
    $(CC) $(INCLUDE) $(CFLAGS) $(TOPTFLAGS) -c do_LIS.c -lm
do_queen.o: do_queen.c papi_test.h test_utils.h
    $(CC) $(INCLUDE) $(CFLAGS) $(TOPTFLAGS) -c do_queen.c -lm
do_root.o: do_root.c papi_test.h test_utils.h
    $(CC) $(INCLUDE) $(CFLAGS) $(TOPTFLAGS) -c do_root.c -lm
basicmath.o: basicmath.c papi_test.h snipmath.h sniptype.h round.h pi.h
    $(CC) $(INCLUDE) $(CFLAGS) $(TOPTFLAGS) -c basicmath.c -lm
```

Figure 4-12 Compiler Options

PAPI interfaces deal with a group of PMCs which are configured to monitor the according system events. These events are called EventSets. Combine more than one

events can reflect the system behavior or the application features. For example, relating the level 1 cache misses to the accesses can indicate locality performance of memory management. In addition, PAPI can automatically fill the event set with as many non-derived events as possible.

(d) Platform Available PMCs

The following Table 4-3 lists the events that can be counted with the PMCs and read with the RDPMC instruction for the Cortex A8 processor. All of these performance events are model specific for this processor and may not available in other processors.

PAPI_EVENT	EVENT Measured
PAPI_L1_DCM	Level 1 data cache misses
PAPI_L1_ICM	Level 1 instruction cache misses
PAPI_L2_TCM	Level 2 total cache misses
PAPI_TLB_DM	Data translation lookaside buffer misses
PAPI_TLB_IM	Instruction translation lookaside buffer misses
PAPI_STL_ICY	Cycles with no instruction Issue
PAPI_BR_TKN	Conditional branch instructions taken
PAPI_BR_MSP	Conditional branch instructions mispredicted
PAPI_TOT_INS	Total instructions executed
PAPI_LD_INS	Load instructions executed
PAPI_SR_INS	Store instructions executed
PAPI_BR_INS	Total branch instructions executed
PAPI_TOT_CYC	Total cycles
PAPI_L1_DCA	Level 1 data cache access
PAPI_L1_ICA	Level 1 instruction cache accesses
PAPI_L2_TCA	Level 2 total cache accesses

Table 4-3 Events Measured by setting the environment variable PAPI_EVENT

Note that the events listed here are native events which can be obtained directly from a single event. Derived events that use more than one event at same time could intensify the limitation of the simultaneous PMCs number. Considering with the overhead of multiplex and complexity, we only use the native events.

4.2.3 Components

Based on the Beagleborad architecture which has been discussed before, to simplify the work, the energy estimation in this dissertation focuses on two components: on one hand, the processor system and, on the other hand, the outer-MPU hierarchical memory system. The rationale for the component division is that they both are two indispensable parts in any system consumes large proportion of the whole energy and encompasses the highest complexity. Note that the processor model also consists of the cache memories. This partition mainly because that cache is a MPU exclusive device which has high relationship with MUP activities.

A simple method to determine the weights of the two model components is expressed in equation 4-1. The method considers the memory system weight as a percentage of the STORE and LOAD instructions compute with the bottom level cache memory miss rate, while the processor system weight is its complementary. The STORE and LOAD instructions are access operations that occur in any level of the storage hierarchy, since there is no direct PMC to monitor and count the storage components except the cache memory, thus, bottom level cache miss rate is used to presume the possibility of the main memory or MMC card access.

$$\omega_s = P_{SR+LD} \times L2_MISS_RATE = \frac{SR_INS+LD_INS}{TOT_INS} \times \frac{L2_MISS}{L2_ACC}, \omega_p = 1 - \omega_s \quad (4-1)$$

4.3 Modeling

The modeling process involves several steps:

- To measure subsystem-level power using subset of workloads.
- To automatically choice PMCs from all the PMCs which are supplied by a specific platform. A proper PMC set can avoid the insufficient accuracy caused by a pool of candidate PMCs;
- To confirm the coefficients of PMCs. The remained PMCs are the most relevant ones to the subsystem energy consumption; however, the training process also affects the model accuracy. Training the model only once may not achieve the sufficient accuracy; an appropriate iteration is needed to avoid the bias. The k-fold cross validation method is employed to tune the final model.

- To compare modeled versus measured energy of each benchmark by the related error to assess the accuracy.

In this sub-section, two points, the benchmarks and the measurement are discussed.

4.3.1 Benchmarks

Several C-written simple applications are used as the benchmarks. They includes numerical computations such as matrix inversion, rank, decomposition, eigenvalue or eigenvector calculation, linear/nonlinear algebraic equation solving as well as interpolation and approximation methods, along with some classical algorithms, sorting, eight queen problem and greedy algorithm, for instance. These programs are basic algorithms in science and engineering computing or software design. Although they are not as long-time-last or computation-intensive as the real implementations, they can reflect the relationship between PMCs and application features. Some test programs to address one event such as cache miss and MMC load/store operation are also involved in order to better understand event's contribution and influence to the whole energy consumption.

4.3.2 Measurement

To find out the correlation of PMCs and energy consumption and to evaluate the accuracy of the estimation model, direct measurements of current and voltage are needed while the decoder executes. To avoid waiting for the battery recharge, a model of the platform battery has been developed and a battery emulator has been implemented using a power supply. The power supply has a digital voltmeter to provide the measurements of the voltage drop across the internal resistance to simulate the behavior of the real battery based on the pre-defined battery model.

Due to the sampling frequency limitation and the difficulties of the synchronism between the device under test (DUT) and the battery emulator, the average power during the decoder execution is used to provide the energy consumption as the product between the average power and the execution time. It is needed to point out that this average power can hardly reflect the instantaneous power variations and, therefore, action sequences are periodically summed up as individual PMC samples.

4.4 Conclusion

In this chapter, the implementations of the estimation model are particularly introduced. The platform runs a Linux operating system with the kernel 2.6.34, which is the

first Linux kernel version that supports the PAPI release for PMC measurement purposes⁸⁵. PAPI, known as the performance application programming interface, is a cross-platform interface to PMCs on MPUs. There are several profiling tools to measure and analyze the performance and behavior of application programs. PAPI is finally selected as the profiling tool because of its friendly user interfaces and its PMCs multiplexing implementation. For the sake of the simplicity, the model focuses on two components: the processor and the outer-MPU hierarchical memory system. A basic metric to combine the two components, the benchmarks and the measurement are also introduced.

5 Validation and Evaluation

5.1 Introduction

The energy consumption model needs an offline calibration phase. In this phase, the purpose is to relate the real energy measured by the multimeter with the modeling metrics represented by the events of interests. Once the relationship is found out, the continuous values from the corresponding PMCs can be used to estimate the energy. The calibration phase is run only once for each system. In this chapter, the progress of the model during the research will be described. The results of each model are analyzed to know how the improvements of the accuracy of the estimation are achieved. In addition, the model limitations and the future work are also introduced.

5.2 Progress of Model

During the modeling study, we found that the accuracy of the model is quite sensitive with the collected statistics of each PMC. It means that an accurate model should use the information kept by the PMCs as much as possible. In this subsection, we will show the progress of the second and the third steps of modeling method described above to see how the two aspects, proper set of PMCs and the Linear Regress Method, influence the accuracy of the model.

In order to have a better explanation of the model results, we have some definition in this dissertation:

$$T = \{t_1, t_2, \dots, t_n\} \quad (5-1)$$

$$TM_k = \{t_i, \dots, t_j | 1 \leq i \leq n, 1 \leq j \leq n\}, 1 \leq k \leq 5 \quad (5-2)$$

$$NTM_k = \{t_p, \dots, t_q | 1 \leq p \leq n, 1 \leq q \leq n\}, 1 \leq k \leq 5 \quad (5-3)$$

$$A_{TM_k} = \{\alpha(t_i) | t_i \in TM_k\}, 1 \leq k \leq 5 \quad (5-4)$$

$$A_{NTM_k} = \{\alpha(t_i) | t_i \in NTM_k\}, 1 \leq k \leq 5 \quad (5-5)$$

where the T is the set of all the benchmarks, TM_k is the set of each training group and NTM_k presents the Non-training group. The $TM_k \cup NTM_k = T$ and $TM_k \cap NTM_k = \emptyset$, $\alpha(t_i)$ are set as the metrics to evaluate the model accuracy of each individual test application, in this dissertation, we consider the relative errors. The simplest metric to compare two models is their average errors calculated as equations 5-6 and 5-7.

$$E_{TM_k} = \frac{\sum_{t_i \in TM_k} \alpha(t_i)}{Card(TM_k)} \quad (5-6)$$

$$E_{NTM_k} = \frac{\sum_{t_j \in NTM_k} \alpha(t_j)}{Card(NTM_k)} \quad (5-7)$$

5.2.1 PMC Accuracy

The platform-supported PMCs provide approximately accurate performance information. To keep the implementation and validation cost low, a reasonable degree of inaccuracy in the counts is acceptable. There is no exact definition of reasonable degree of inaccuracy, but there are some recommended guidelines:

- Under normal operating conditions, the counters must present an accurate value of the count;
- In exceptional circumstances, such as changes in security state or other boundary conditions, it is acceptable for the count to be inaccurate.
- Under very unusual non-repeating pathological cases counts can be inaccurate. These cases are likely occurring as a result of asynchronous exceptions, such as interrupts.

As the results of benchmark shows, the values of the PMCs are sensitive to some other issues such as the interrupts, OS system calls and the threads scheduling. Thus even for the same application, the values of PMCs are not exactly equal. The following three tables, Figure 5-1, Figure 5-2, and Figure 5-3 show the rate of variation of the same benchmark when it is repeated several times. The rate of the PMC_i is calculated as the equation 5-8:

$$R_{i,j} = \frac{|PMC_{i,j} - avg(PMC_i)|}{avg(PMC_i)} \quad (5-8)$$

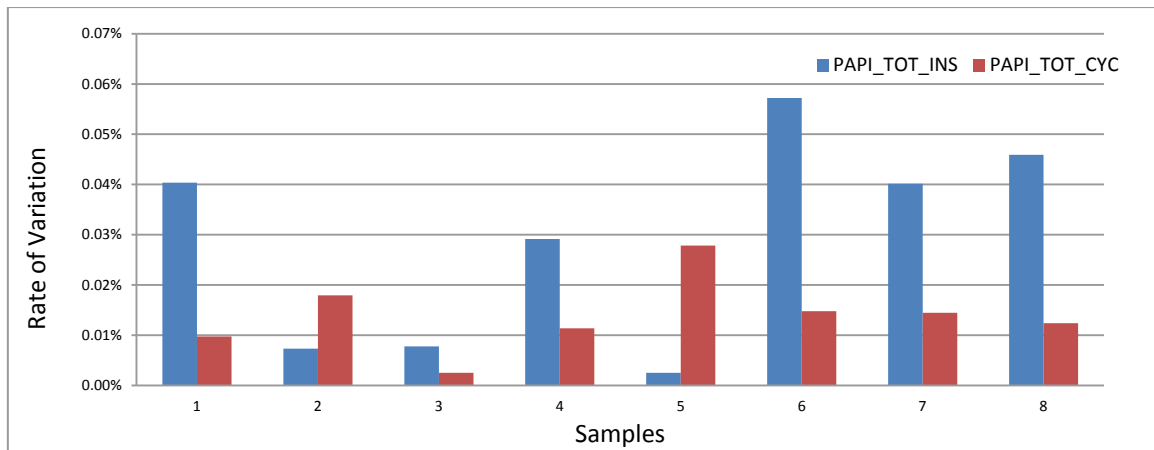


Figure 5-1 PMCs Variation of Do_Queen

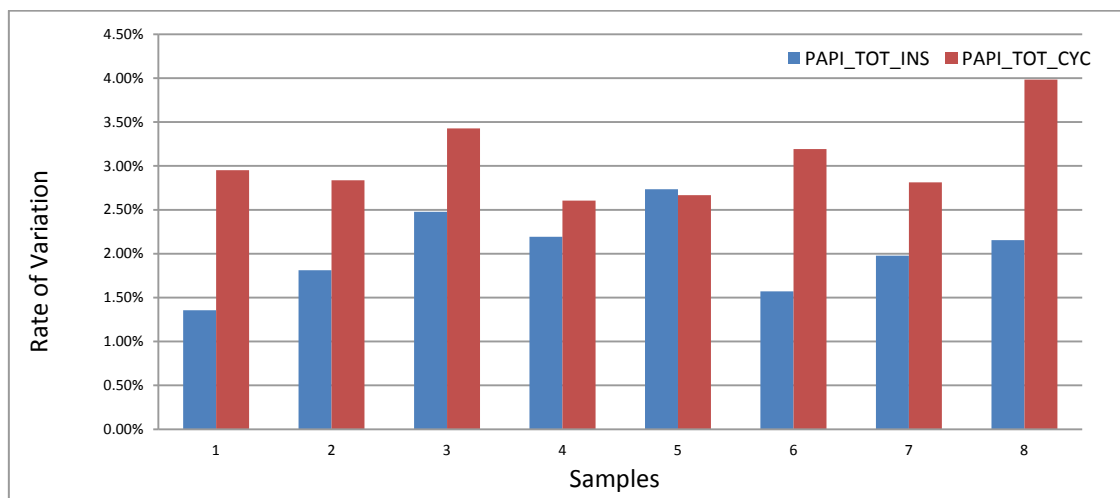


Figure 5-2 PMCs Variation of Do_READ

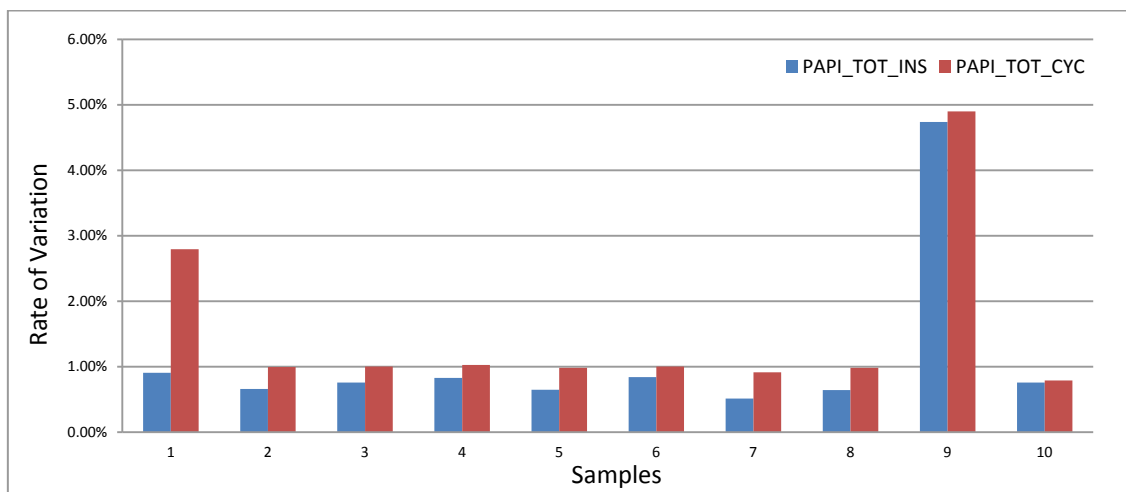


Figure 5-3 PMCs Variation of MM_MISS

From the results, the PMCs' statistics all remain in the acceptable range of the variations. However, the variations also reflect the different features of the three applications.

Do_queen is a program to solve the eight queen problem. It is computation-intensive, and it does not need too many operating data from other resource, it may keep the data in registers or have more push-pop instruction because of its main recursion algorithm, so the number of instruction and cycles are more stable. Do_READ is a program mainly read the data from the MMC card, it reflects a slight jolt because the address translation operation. The MM_MISS is program intentionally generate many cache misses, from the Figure 5-3 there are two obvious variations occur in number 1 and 9. No. 1 is first to execute this program after finishing several other benchmarks and NO. 9 are to run it as the first application at the first time after the OS booting. Because the cache-filling and data-preparation, it is easily understanding such a distinct variation.

From the information shown in figure Figure 5-1, Figure 5-2 and Figure 5-3, it implies two advantages:

- PMCs have the ability to capture the applications' features. Once the operations or functionality of the benchmarks changes, the PMCs also have the according variation with reasonable explanation;
- It is important to choose the PMCs to model. It means that we need to indentify the essential PMCs which have the most closed relationship with the energy consumption to maintain the accuracy with the PMCs variations.

5.2.2 First Model

It is easy to understand that the total instruction number reflects the tendency of energy consumption.

Figure 5-4 shows a very good relationship between the total instructions and the energy consumption with the coefficient of determination (R^2) nearly to 1. R^2 is used in the context of statistical models to predict the future outcomes on the basis of other related information. It is the proportion of variability in a data set that is accounted for by the statistical model. It provides a mechanism to judge how good the predictive ability of the model is. In addition, instruction number is the commonest event which is implemented on most modern processors. The first model is built based on the total instruction number performance counter.

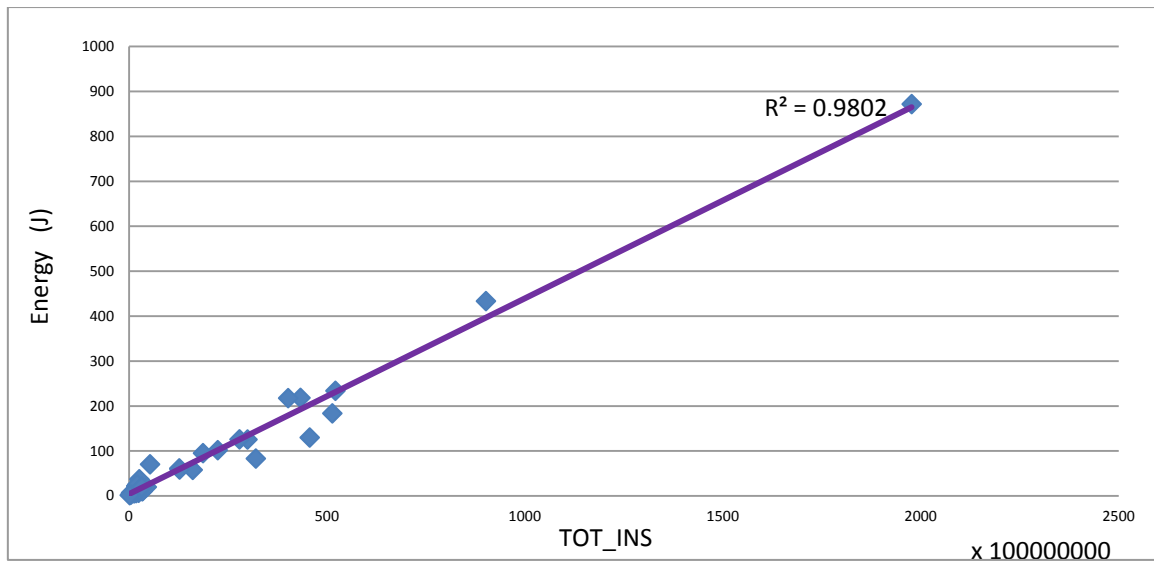


Figure 5-4 Relationship between Energy and TOT_INC

From Figure 5-5 , it is can be seen that the results of this model. Among the 36 benchmarks, 38.9% of them have the error larger than 50% and several ones even exceed 100%. This result is unacceptable.

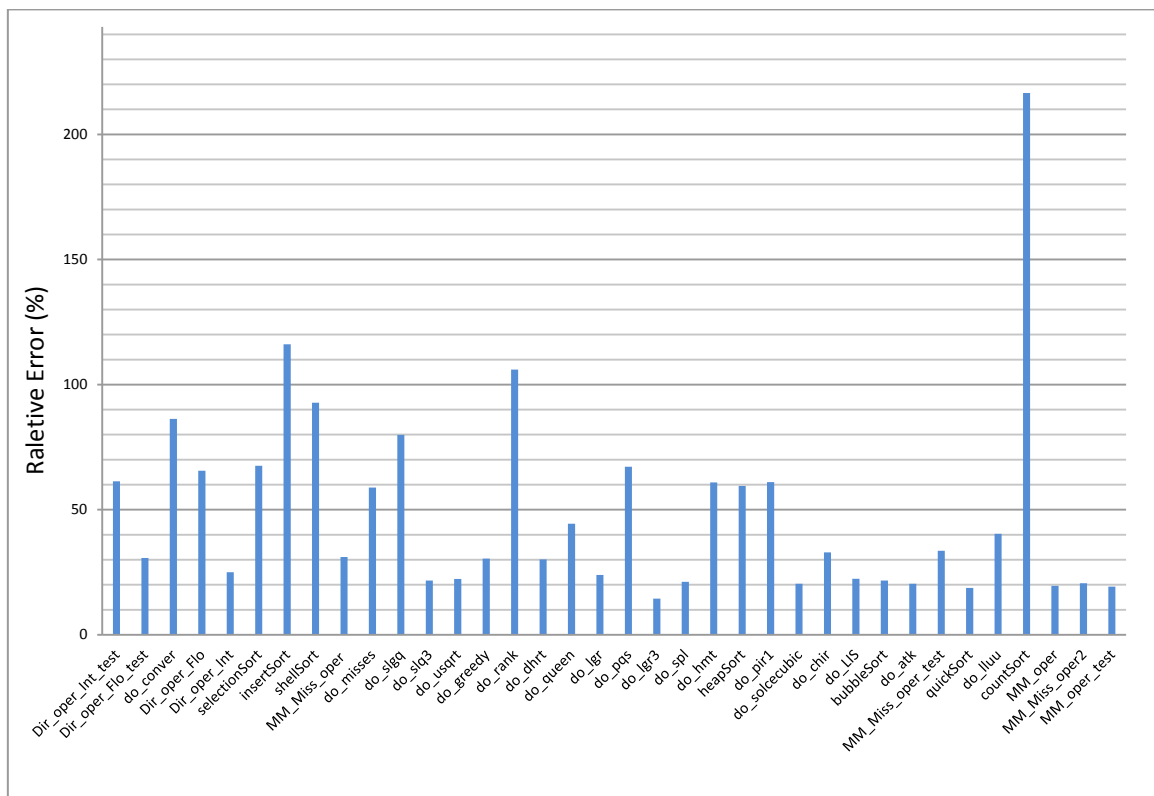


Figure 5-5 Proportional Error of the First Model

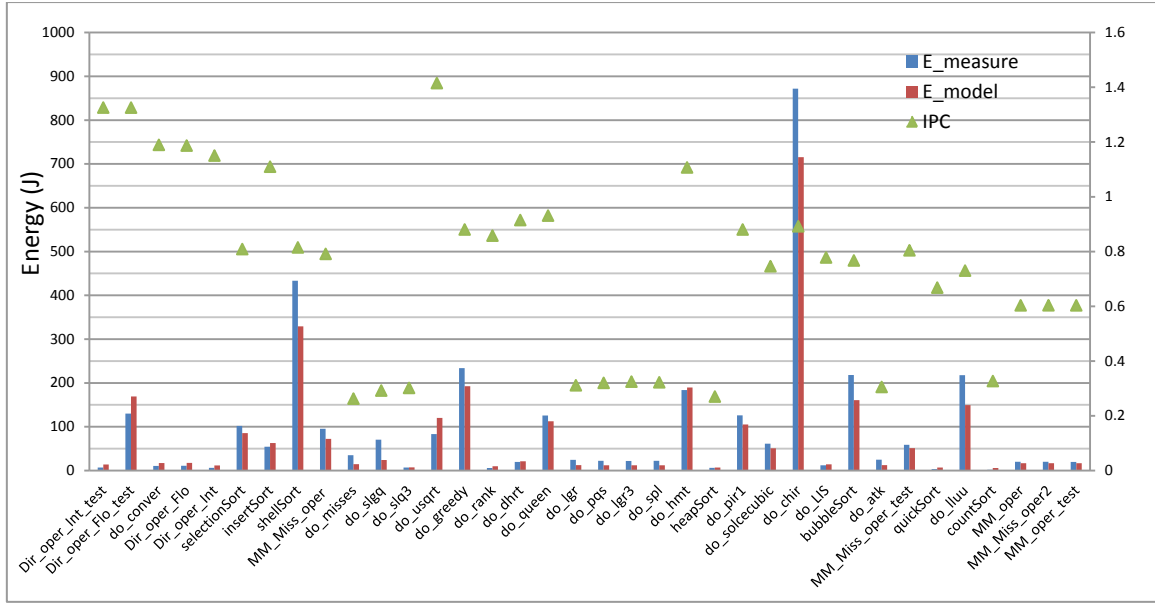


Figure 5-6 Real Energy Consumption VS. First Model Estimation

This result shows that the model only based on the total instruction number is very inaccurate. In Figure 5-6, the intuitive observation is shown. The energy consumption refers to the principal axis in the left side and the IPC (instruction per cycle) refers to the secondary axis in the right side. This figure shows that the IPC is a primary factor influencing the deviation. When the IPC is larger than 1, in most case the model estimations are higher than actual measurements. This is because the model mistakenly considers the system is busy with less percentage of the idle time. On the contrary, when the IPC is smaller than 1, the model estimations are lower. In this situation, the model again improperly states the system in a less-usage case. This is caused by the different microinstructions (uops) of each instruction. For example, in a RISC instruction set, a single ADD instruction may implement within different sources and destinations. A source may fetch a value from memory, a register or an immediate. Therefore it is composed of different uops. This model only uses the IPC means that it assumes all the instruction consume the same amount of energy, so it simply misses other operation units' effects during the instruction execution. From the previous work [66], this model could be a simple and accurate one for estimating the processor's energy consumption, but it is incompetent to profile the energy behaviors of other components because this dominating model only obtains limited information from one PMC

5.2.3 Second Model

Taking into account the defects of the first model, we consider to construct the model to obtain as much information as possible from the whole set of PMCs. The excessive

number of PMCs will increase the needless complexity of linear regression fitting without any screening. For a multiple linear regression model, it is difficult to avoid the dependence of the variables, thus the model may become not precise enough if the number of the independent variables is large⁷⁴. The main idea of the second model is to reduce the number of the impendence variables. As we discussed in section 3.2.1.3, PCA is a commonly used method to reduce the elements meanwhile retain most of the data information. So we will use PCA to re-associate the information recorded by all the PMCs. In addition, we also want to observe the relationship between the model accuracy and the training samples, so we use three ways to select the training samples.

5.2.3.1 Random Selection

There is no specific rule to choose the samples, so all the samples have an equal probability of selection. This minimizes selection bias and simplifies analysis of results. It is not necessary to partition all the samples, because there is no relationship among each application, so we list the PMCs values as their execution order and select elements at regular intervals through that ordered list. In our case, the components are the 16 PMCs. The population is the 36 benchmarks. It is needed to randomly select half of samples from the population to produce the principal components because that the PCA requires the number of samples larger than the number of the components. Thus we do the sample with a skip of 2, the set of selected samples is {1,3,5,7...}. Figure 5-7 shows the results of the second model with the random-selected samples. The energy consumption refers to the principal axis in the left side and the relative error refers to the secondary axis in the right side. Since the variation of the energy consumption of the benchmarks, the principal axis is a 2-base logarithmic scale. The result shows a great improvement. In most cases, the errors are less than 50%, nearly quarter of the benchmarks have the error less than 4% and 80% less than 30%.

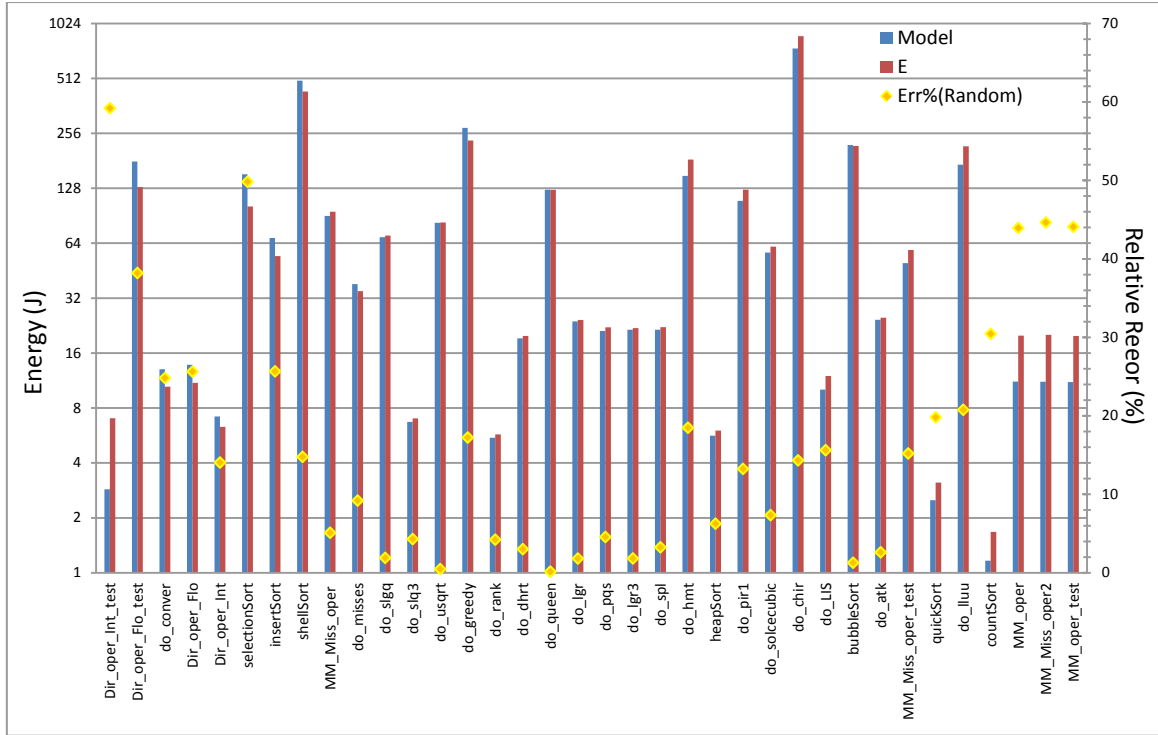


Figure 5-7 Real Energy Consumption VS. Second Model Estimation(a)

However, the random selection method can be vulnerable to sampling errors because the randomness of the selection may result in a sample that does not reflect the features of the population. For example, suppose we wish to sample the applications with different execution time. A simple random selection could easily end up with too many applications with short execution time and too few ones with long execution time (or vice versa). This situation will lead to an unrepresentative sample. An attempt to overcome this problem is to use the information about the population to choose a more representative sample. Two trials discussed below are used to test the results.

5.2.3.2 L2-Miss-Rate Selection

The level 2 cache miss is supposed to cause the CPU stall and the memory accesses, both of which effect a lot the system energy consumption. Therefore, we hope to evenly sample applications with the different level 2 cache miss rate. So in this procedure, we sort all the sample refer to the level 2 miss rate from lowest to the highest. This means that whole population is spread evenly along the level 2 miss rate and the training samples can cover the different degree of this rate. The level 2 miss rate is calculated from PAPI_L2_TCM and PAPI_L2_TCA, and the sampling interval also equals to 2.

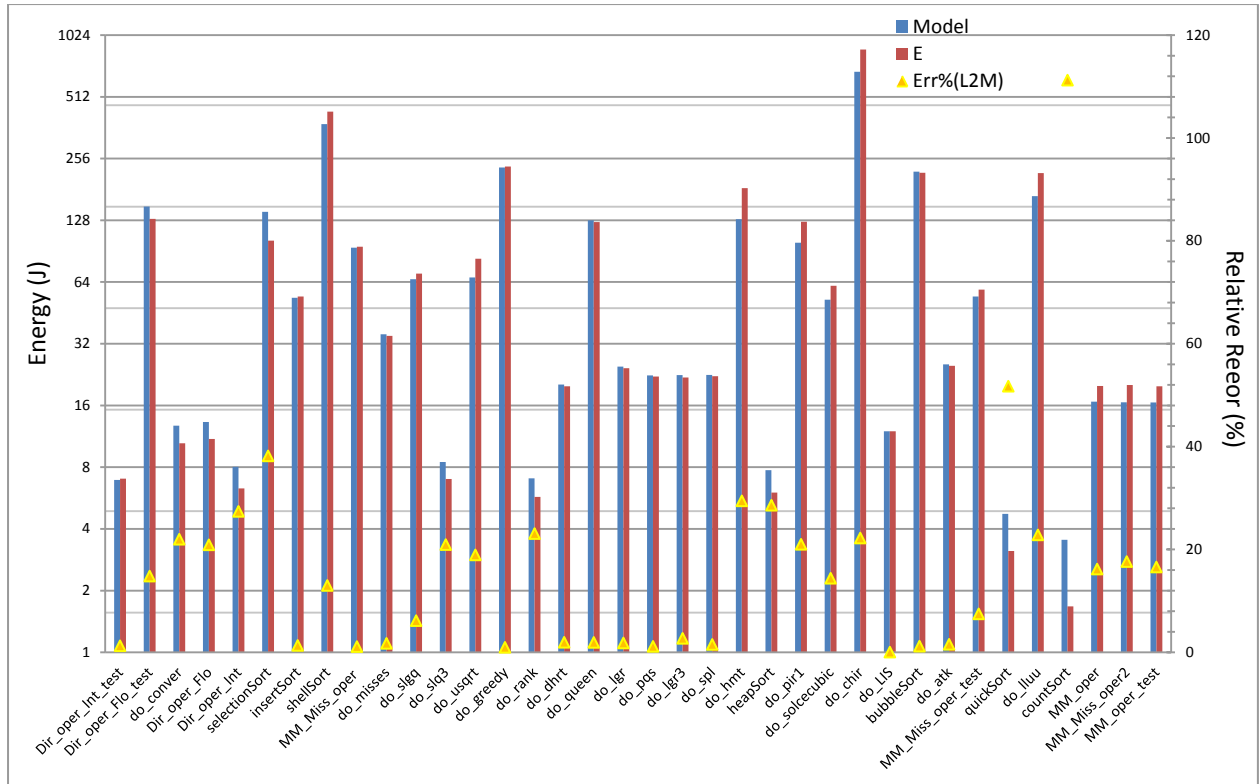


Figure 5-8 Real Energy Consumption VS. Second Model Estimation (b)

Figure 5-8 shows the results of the second model with the samples based on their level 2 cache miss rate. The same, energy consumption refers to the principal axis which is a 2-base logarithmic scale in the left side and the relative error refers to the secondary axis in the right side. It is not simple to say this is better than the previous one, the model based on the random selection because in rare cases the error exceeds 100% which mean the model totally deviate from the expectation. However, in this case, 89% errors are less than 30% and 39% errors are less than 4%. General speaking, in most cases, the model's behavior is better than the previous one.

5.2.3.3 IPC Selection

Similar to the procedure described in the sub-section (b), we sort the samples by IPC derived from PAPI_TOT_INS and PAPI_TOT_CYC.

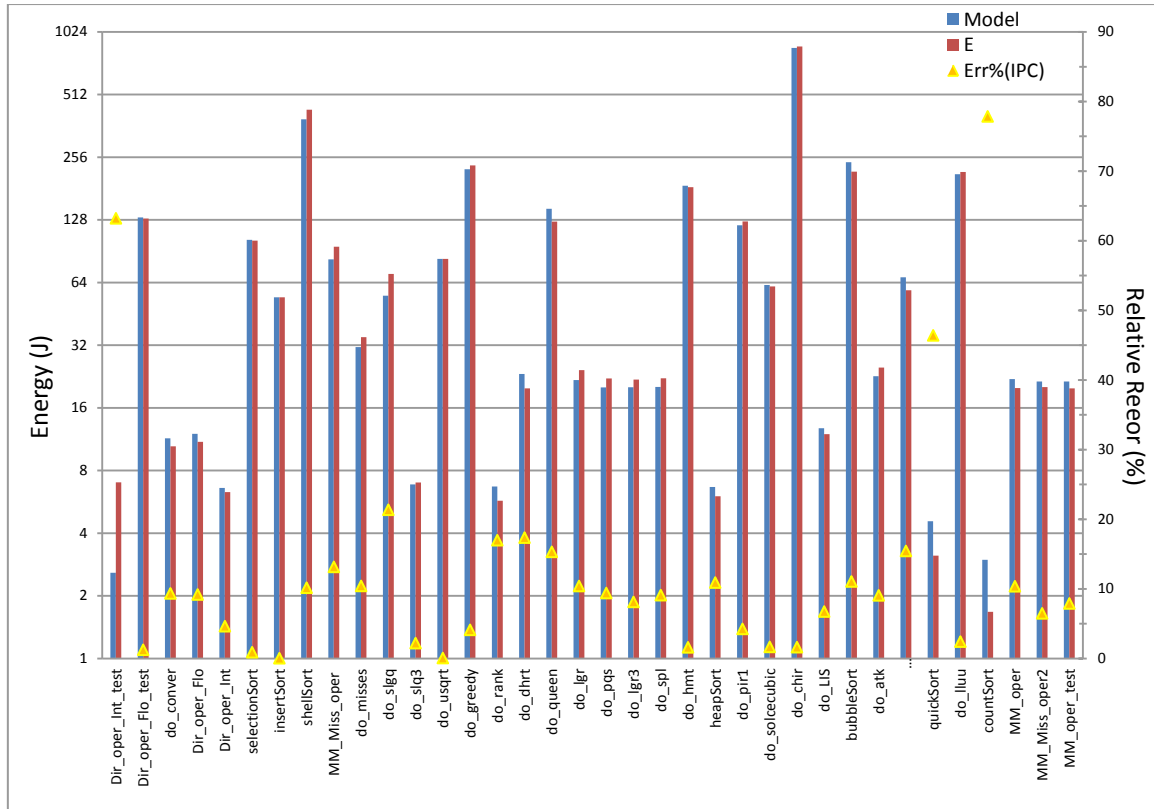


Figure 5-9 Real Energy Consumption VS. Second Model Estimation (c)

Figure 5-9 shows the results of the second model with the samples based on their IPC with the same axis's configuration: the left 2-base-logarithmic-scale principal axis to represent the energy consumption and the secondary axis in the right side to show the relative error. It is not quite different to the two models in sub-section (a) and (b). The percentage of the error less than 4% is 25% which is less than that of model(a) and model(b), and the maximal error is 77.87% which is larger than model(a), however, in this case the error less than 30% is 91.7% which is the largest one among the three models. General speaking, this model is better than model (b) in capturing the tendency of energy consumption and slightly ahead than model (a).

Compare with the three models, it is a little difficult to choose a better one from the model (a) and model (c), but they both behave better than model (b). However, we can know that the accuracy of a model is influenced by the representativeness of the samples from the population. In another word, this simply selection method is vulnerable, it may get a better accuracy by chance and maybe it is hard to repeat the same accuracy with other applications.

The second model is promising when we increase the information recorded by the PMCs. However, when we added some computation-intensive applications with short

execution time (e.g, countSort), we receive significant relative error, it means that there had to be some "flaw" in our methodology. One reason could be caused by the PCA method. There are two main aspects of the PCA robustness considerations:

- Consideration of the independence between each principle components. From the theory of probability, each principle component is independent if and only if the input x subject to zero mean and the covariance matrix is an n -dimensional Gaussian distribution. When the input dose not obey the Gaussian distribution, the traditional PCA algorithm only consider the second-order characteristic of covariance matrix, therefore the obtained principle components are only satisfied to be uncorrelated with each other but not independent.
- Consideration of the outliers in the sample data. It is worthy to considering how to remove or weaken the impact of outliers in the limited training sample to follow the accurate main direction. Outliers make cause a large error in PCA calculations. In addition, since the number of samples is limited, even all the samples are generated from the same data distribution. Several "outliers" may caused by the inadequate number of samples. Outliers are a major aspect influence the PCA robustness.

Since the PMCs which configured to monitor several events are considered as the components, it is difficult to avoid the dependency among the PMCs, moreover and some outliers, moreover, the benchmarks are various with their behaviors and functionalities, which will bring difficulties to figure out the common principal components for all the applications. As the results shown, when we used this model to estimate the applications which were not used G3 for PCA analysis, the estimation errors are much larger. We believed that the principal components decided by the 36 benchmarks were on longer suitable for others. So we probably did not get enough information to reflect their energy behaviors.

5.2.4 Third Model

Because the real world applications have various behaviors and the values obtained by the PMCs are impressionable by many other factors, static principal components with the fixed eigenvector may cause the larger estimation errors. So a methodology used to decide the proper PMCs based on their relationships with the energy consumption was tried this time. In this model, the PMC filter which is introduced in the section 3.3.2.1 to select the

PMCs is implemented. Table 5-1 lists the correlation coefficients of each pair of one PMC and the energy consumption.

L1_DCM	L1_ICM	L2_TCM	TLB_DM	TLB_IM	STL_ICY	BR_TKN	BR_MSP
0.56	0.92	0.57	0.53	0.63	0.71	0.83	0.52
TOT_INS	LD_INS	SR_INS	BR_INS	TOT_CYC	L1_DCA	L1_ICA	L2_TCA
0.90	0.88	0.72	0.83	1.00	0.88	0.91	0.71

Table 5-1 The Correlation Coefficients between PMCs and energy

The scatter diagram can be used to intuitively observe the relationship between one PMC and the energy. Figure 5-10 show the plots with the relationship strength from the highest to the lowest. In case (a), which has the correlation coefficient nearly to be 1, the energy consumption can be reflected by the total cycles as the straight line. With the decrease the correlation coefficients, the relationship lines are not as straight as case in case (a), however, we still can find out the increased tendency of the energy with the raised occurrence of the according PMCs.

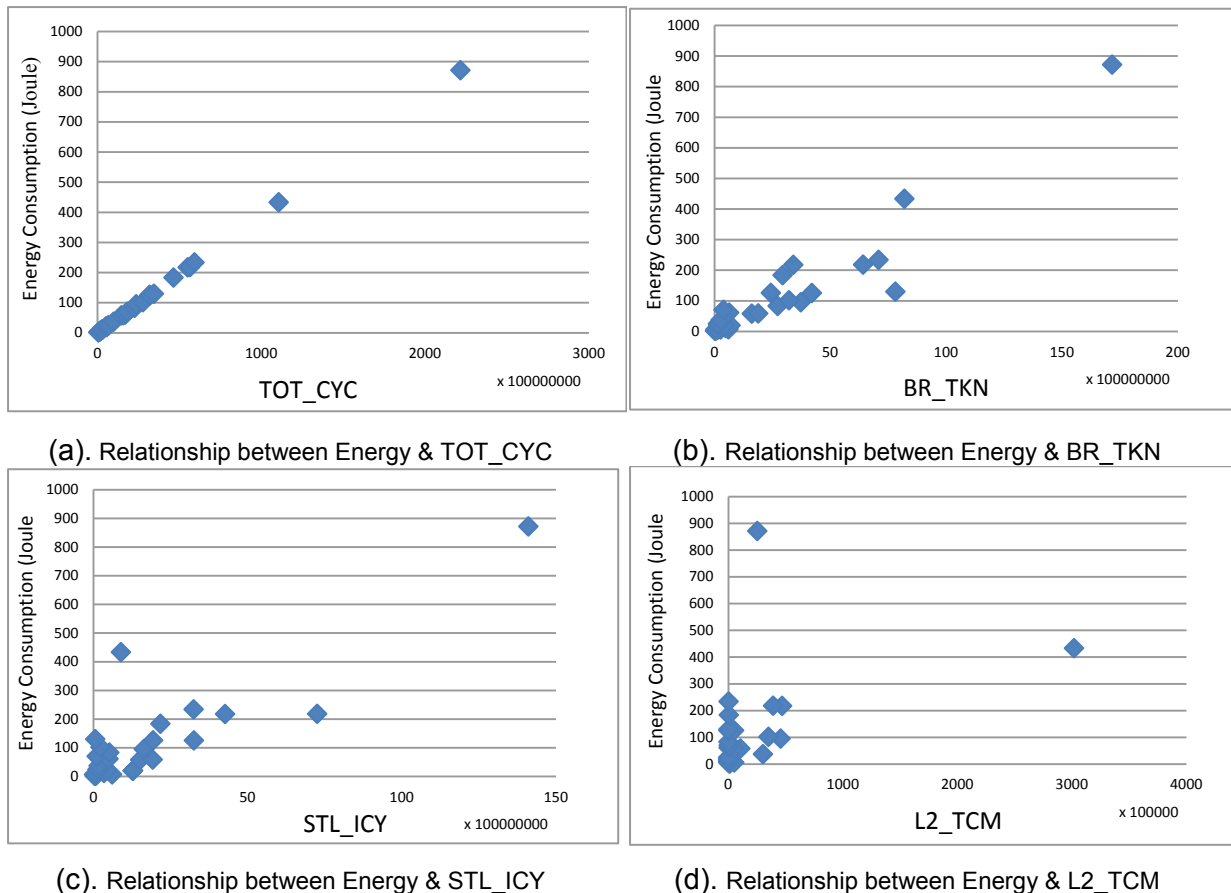


Figure 5-10 Relationship between Energy and PMC

Under the Beagleborad environment, five PMCs, PAPI_L2_TCM, PAPI_TLB_IM, PAPI_BR_TKN, PAPI_SR_INS and PAPI_TOT_CYC are finally decided to use. These selected PMCs monitored different events, which reflect the different heavily energy-hungry activities:

- PAPI_TOT_CYC: the number of cycles always shows positive and high correlation with the energy cost. It presents a basic principle that the application energy tendency is depend on its execution time. However, this prediction is quite coarse.
- PAPI_L2_TCM: The total cycle of an application also includes the stall cycles which are the cycles without instruction issue. A significant percentage of stall cycles might be attributed to cache missed. L2 cache misses, which means that the necessary data will be obtained from the memory, will cause more energy cost, thus PAPI_L2_TCM is a key energy-related issues.
- PAPI_TLB_IM: Level 2 cache reflects effects of both instruction and data misses, however, TLB misses by themselves, have greater influences on energy consumption since the processor needs to handle memory page table. In this case, the instruction TLB misses are needed to be taken into account.
- PAPI_BR_TKN: Branch prediction is other key issue of CPU stalls. If the branch prediction fails, the pipeline will stop to wait for the new instructions filling. This effect of energy consumption is also significant.
- PAPI_SR_INS: The store instructions will give a direct observation to data-relate operation include the write operation in any layer. Since the write operation are more complex than read operation such as how to keep the data consistency, the store instructions will have further impact on energy consumption. The Robust Regression method (Figure 5-11) is employed to build the model.

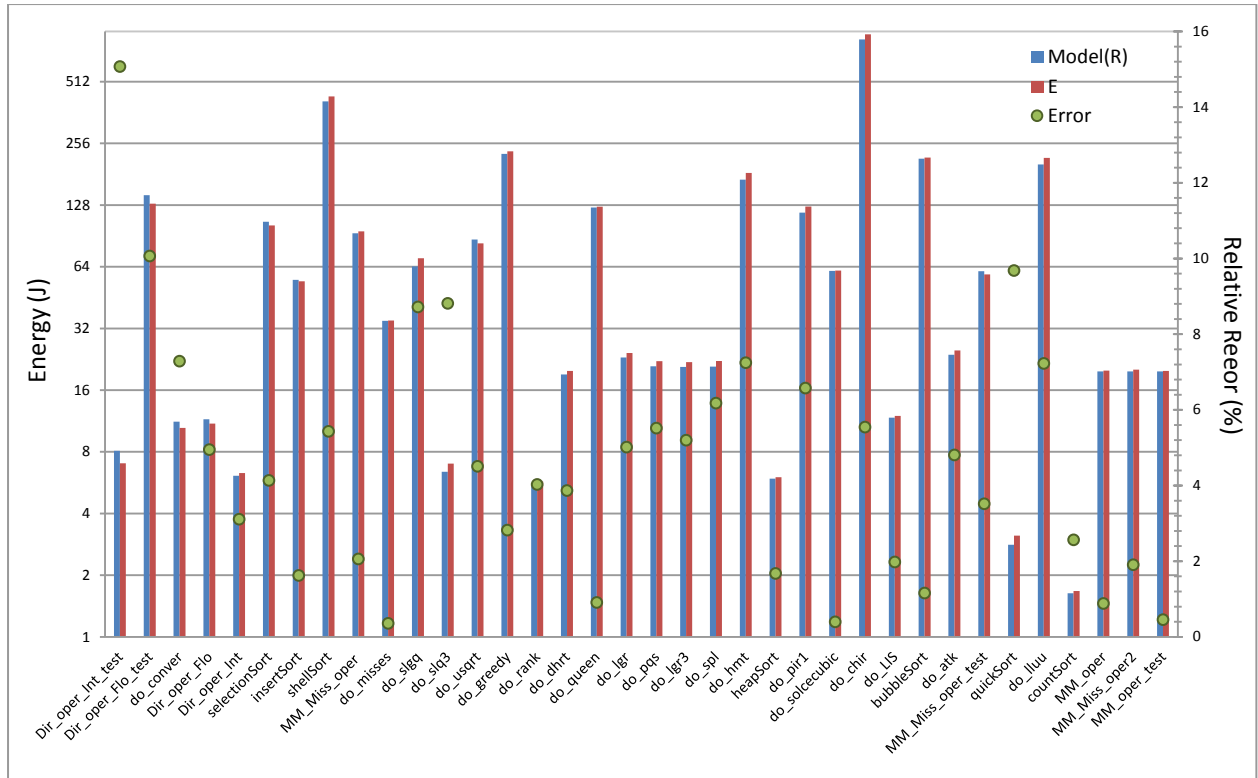


Figure 5-11 Real Energy Consumption VS. Third Model Estimation

Figure 5-11 shows the results of the third model with the Robust Regression method. In this figure, the energy consumption refers to the 2-base-logarithmic-scale principal axis in the left side and the relative error refers to the secondary axis in the right side. The percentage of the benchmarks with the error less than 4% is 44.4%, error less than 5% is 58.3%. A good improvement is that the largest error fall down to 15.1%. The third model with the robust linear regression obtain not bed results, it means that:

- Even limited number of PMCs with enough information from the entire PMCs can built an accuracy model. It implies that it is an important step to filter the PMCs.
- The correlation of the elements pairs will affect the accuracy of the model. Stepwise method is superior the Robust regression because that the former method further removed the elements that not significantly contribute to the energy consumption which make the model is more concise with smaller disruption.

5.2.5 Fourth Model

Briefly, the third model obtained a fairly good result, but this model considers the processor and the outer-chip memory as the same target to construct a general model. Two groups of the benchmarks are use to comparison the energy consumption. The benchmarks from the two groups with the same name have the same functionalities. The only differences are how to get the source data and how to hand with the results. The data used in the "WITH_MMC" group is read from a file saved in the SD card and the results are written back to the SD card. However, the data used in the "WITHOUT_MMC" group is the randomly generated by the processor and the results are not saved.

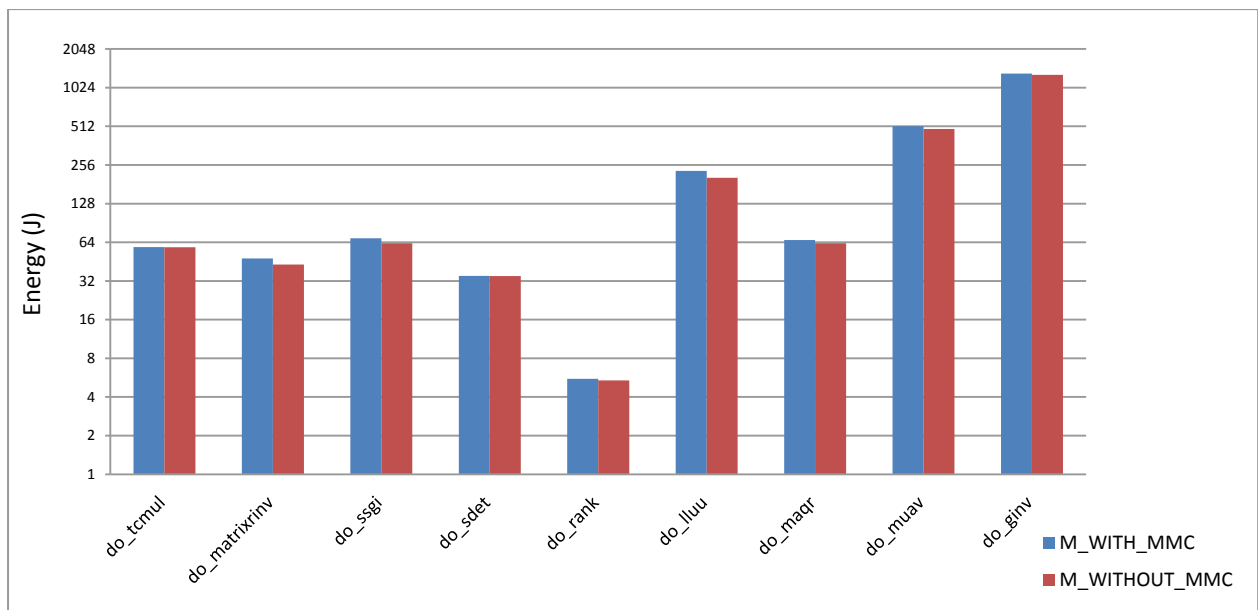


Figure 5-12 Measurement Energy Comparison

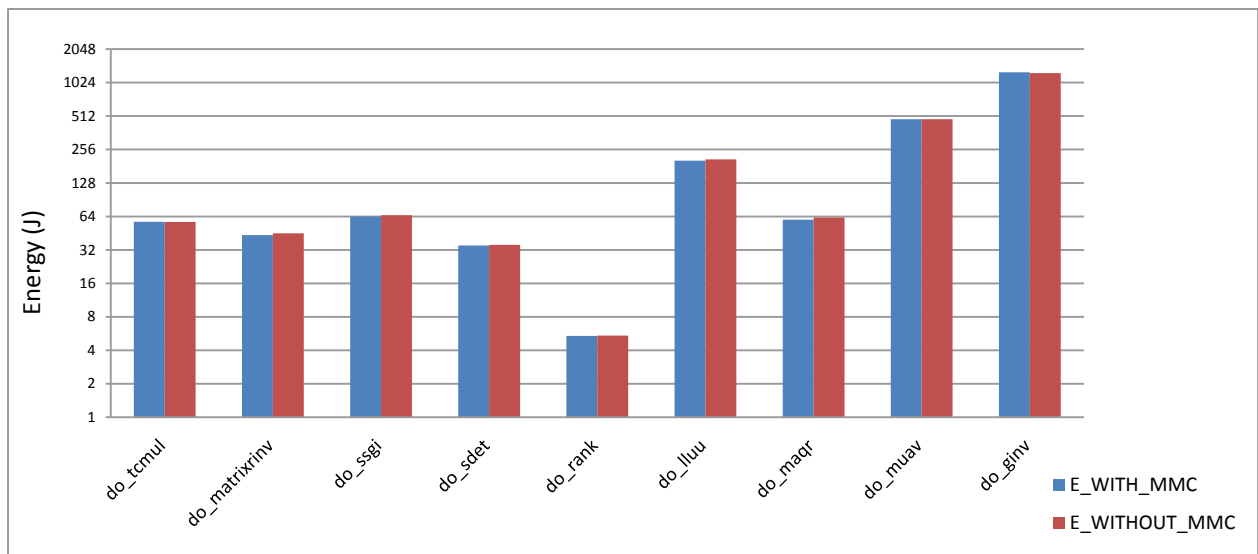


Figure 5-13 Estimation Energy Comparison

Figure 5-12 shows the comparison of the directly measurements of the two groups. It is significantly shown the more energy required by the applications which operated with the SD card. However, Figure 5-13 shows the energy estimation of the two groups with tiny difference. This result implies that a model with more accurate results should dividedly consider each component.

In addition, the memory access behaviors also change the relationship of each pair of the energy consumption and the PMC. Figure 5-14 shows the Spearman Rank Correlation in different cases are also changed. For example, when the application is computation-intensive, the coefficients of the level 1 cache data cache miss and level 2 cache miss are much lower than the memory-intensive applications and the applications with high overload of both computation and memory access. It is reasonable because that the operating data is not frequently translated between processor and memory. This means that different component will have different factors which significantly influence the energy consumption. This may be one reason to cause the result of the third model is in-and-out.

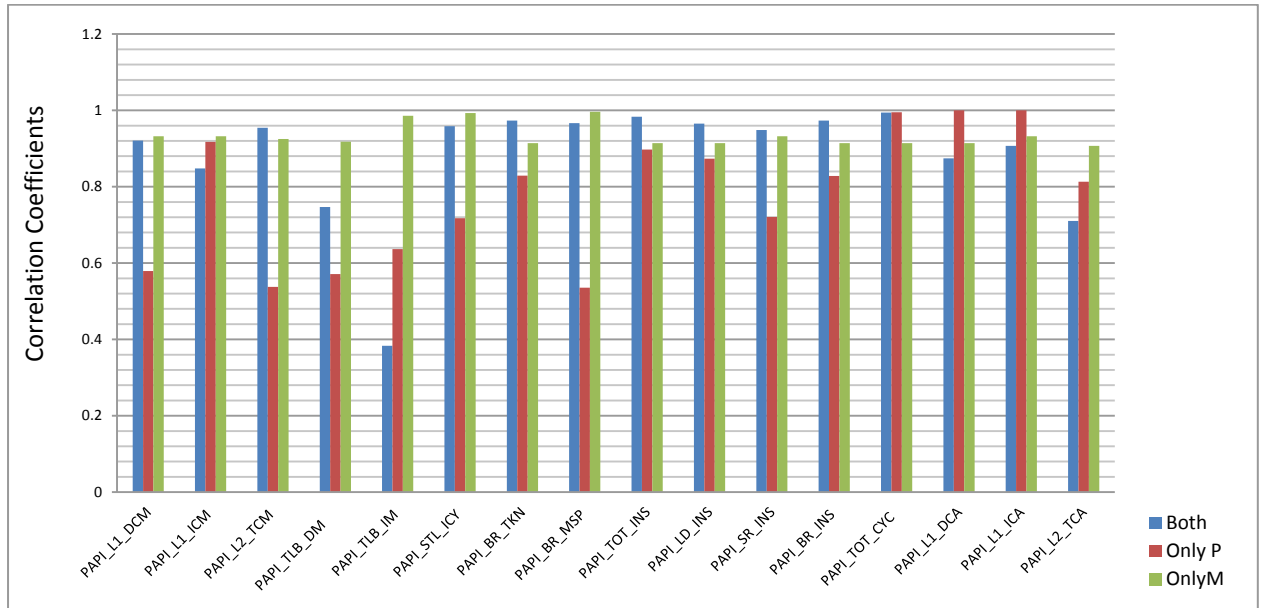


Figure 5-14 Correlation Coefficients in Different Cases

In order to improve the results, we will build sub-models for each component, processor and outer-chip memory, as the method introduced in section 4.2.3.

As the same way, the PMC-filter is used choose the PMCs as the third model. Since this estimation model focuses on processor and outer-chip memory, to address the two system components, the benchmarks are also divided into three categories that depend on their characteristics: computation intensive (G1), data-transfer intensive (G2) and both (G3).

In addition, G1 and G2 are further divided into training and non-training groups. The training groups are used to build the processor model and memory model, respectively.

Although the PMC-filter works as a black-box to produce PMCs selection, the result should be reasonable to represent the realistic scenarios. The set of PMCs shows that PMC-filter presents more concentrate on the memory related PMCs. This observation corresponds to the conclusion from Jimenez et al. in work [86] that the memory-intensive application causes the system to consume more power. However, it is also noted the difference between the final PMC set of processor and memory components. The PMC-filter only retains PAPI_BR_TKN and PAPI_TOT_CYC to build the memory estimation model. One possible reason is the benchmarks. In order to address the memory access, the benchmarks focus on reading or writing data into different files with only limited computation operations like loop control. In this case, all the benchmarks have the similar features, in other words, issues including PAPI_L2_TCM, PAPI_TLB_IM and PAPI_SR_INS will be proportional with the total cycles. Based on the PMC-filter method, information kept by these three PMCs can be represented by the total number of cycles.

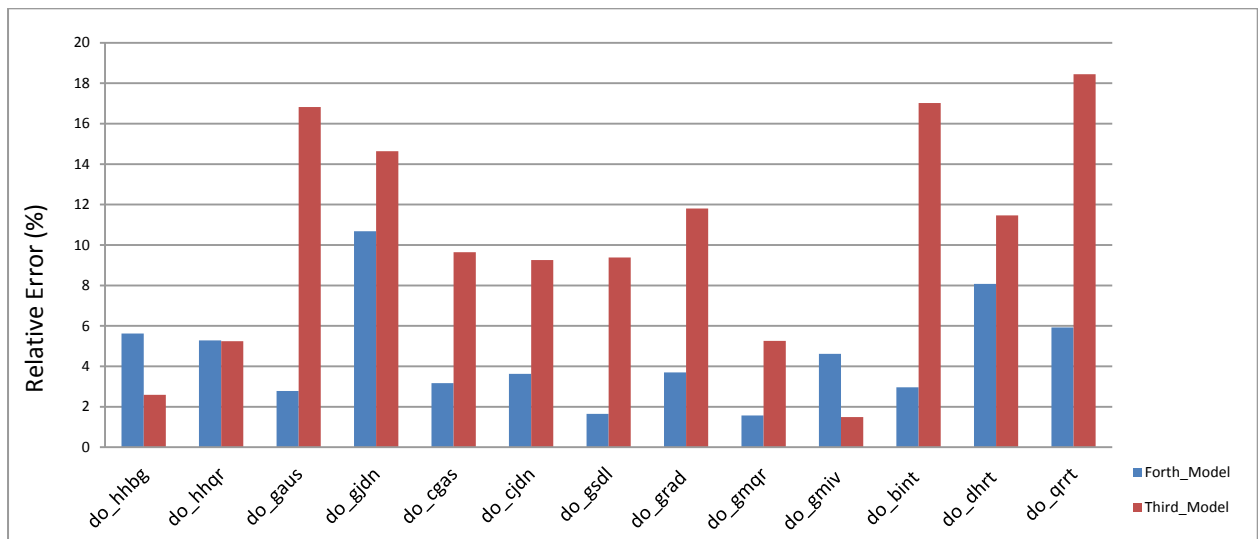


Figure 5-15 Model Comparison

Figure 5-15 shows the comparison of the fourth model and the third model. Here the benchmarks with the memory access behavior are listed. Generally, the forth model improve the average accuracy. In most cases, the forth model decrease the error to acceptable values, however, there are rarely situation that the forth model increase the predication accuracy but the error still keep in the acceptable area.

5.2.6 Final Model

Compare the previous four models, there is a great improvement from first one the forth one. It is observed that the key of keeping the accuracy and stability of the model is to choose the proper PMCs before curve fitting. However, since the sampling method is vulnerable to the benchmarks used to training the model, to make an estimation model with good predictability and generalization, we will use the k-fold cross-validation introduced in the section 3.3.2.2 to construct the final model. Table 5-2 lists the G1 and G3 relative percentage errors of estimations and measurements in different cases. To balance model efficiency and accuracy, the k-fold selector is only applied to the processor component because of its complexity.

	PM ₁	PM ₂	PM ₃	PM ₄	PM ₅	M
G1_Tr	2.188	3.696	4.185	3.575	1.79	-
G1_nT	3.78	4.184	3.249	6.243	4.334	-
G1_Av	3.382	3.968	3.483	6.125	4.209	4.233
G3_Av	2.528	1.937	5.012	7.89	6.655	4.804

Table 5-2 Relative Errors in Different Cases

The k-fold selector is applied five times to obtain five sub-models, from PM1 to PM5. Model M gives the averaged estimation. The results of the memory model provide a similar accuracy, with an average relative error of 1.384% for the training group and 3.402% for the non-training group. Since benchmarks in G3 are computation and data-transfer intensive, they are employed to test together the processor and memory model.

The results of rows G1_Tr and G1_nT show that there are no significant variations between training and non-training groups and, in addition, the errors are limited in 6.5%. Therefore, the model is able to keep a good stability. Rows G1_Av and G3_Av give out the average relative errors of each PM and the final model. There are two conclusions: First, the errors of final model of G3 and G1 are in the same grade, this shows that weight-based combination can capture both the processor and memory activities to keep the model accuracy. Second, comparing errors in G1_Av and G3_Av of the same PM_i, it is noted that the model with the highest accurate prediction of group G1 may not be the most suitable one for group G3. This observation reflects the different prediction abilities of each PM_i. Therefore, averaging all the PM_i results generated with the k-fold selector produces a credible result.

Figure 5-16 and Figure 5-17 show the relative errors of benchmarks G1 and G3, respectively. In Figure 5-16, the largest error is 6.8% and for most benchmarks (65%) there is a good matching (<5%). In Figure 5-17, most of the benchmarks with the estimation error around 6% include the extra MMC card accesses operations (first six ones in Figure 5-17, do_spl3 is the interpolation, differential coefficient and integration of the cubic spline function, arrayMulti is a function to deal with the matrix multiplication and do_patricia is a function use the data structure Patricia tries to represent routing tables in network applications. All the inputs of data for these benchmarks are read from the files recorded in the MMC card and their results are written back to according files). This observation shows that energy estimation for external memory using PMCs is not as precise as the estimation for the processor. The main reason is the limitation of the PMCs, which suffer the lack of performance events to trace the memory transactions executed on the outer-MPU memory layers. However, the methodology introduced in the dissertation is proven to keep the model accuracy in an acceptable area (<7%).

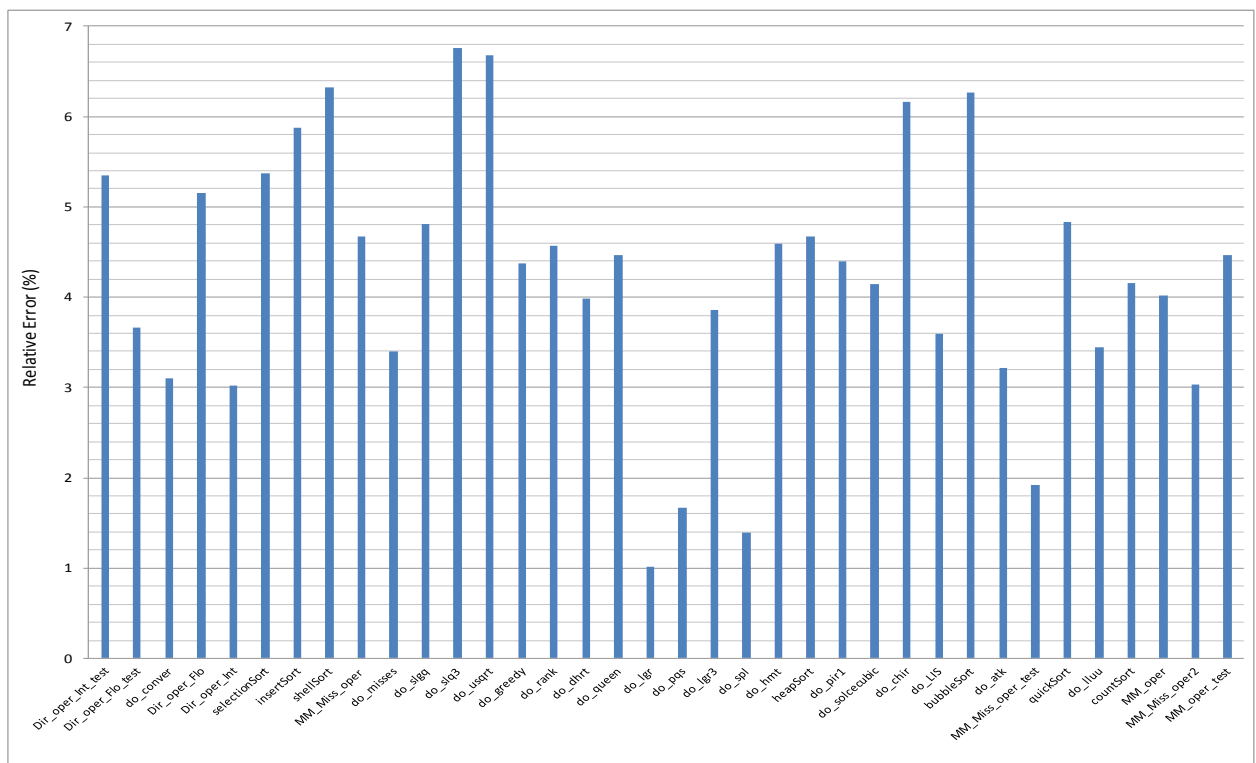


Figure 5-16 Average Relative Error of G1

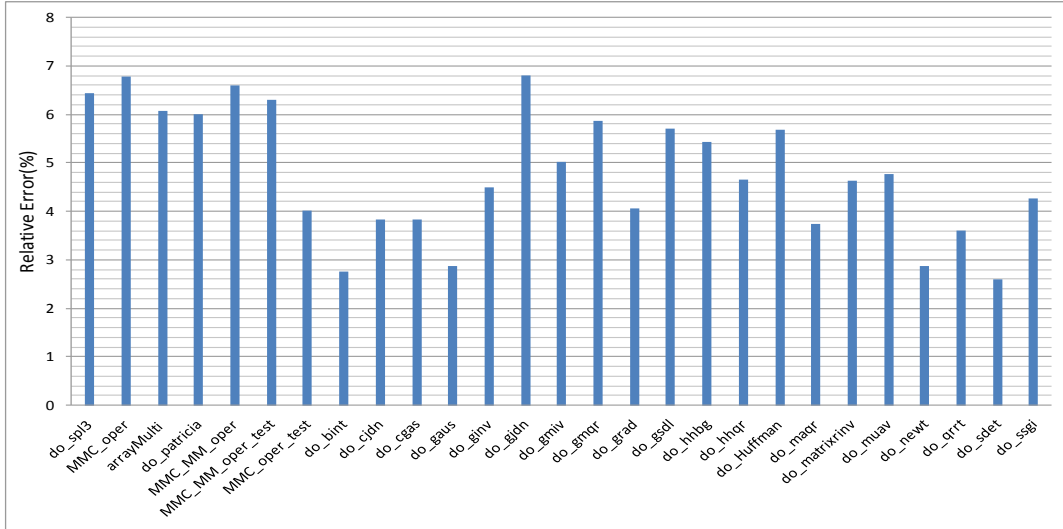


Figure 5-17 Average Relative Error of G3

5.3 Model Limitation and Future Work

5.3.1 Model Limitation

In the last sub-section, we show the improvements of the energy consumption estimation model in detail. The final model we obtained is promising to relate the PMC statics with the energy estimation without previous knowledge of the details of the hardware. However, this model has met several limitations:

- First, this model only involves two main parts, the processor sub-system and the memory sub-system. As the prosperous developments of the battery-powered devices, their functionalities require more ancillary equipments, which consume non-neglected energy. A more helpful energy analysis tool should be extended to the whole system including other components such as communication, I/O peripherals, buses and even multi-cores, thus the information from PMC measurement should be available not only on the processor but also the chip and system.
- Second, a static model is limited. Although the k-fold cross validation method is used to enlarge the coverage of the application features, the estimation error of a static model still has the possibility to be divergent due to the always-changeable operations of users. Thus it is necessary to add some feedback into the model in order to adjust the estimation direction during the users' behaviors.

- Furthermore, the model makes the estimation based on the history of the applications. It lacks of the metrics to predict the future behaviors of the applications which are important for the PM policy. As we know, the basic idea of the commonly used PM policy tries to scale the voltage or frequency of the components based on their idle period. It meets a problem called over-prediction/under-prediction which means the predicted idle period is longer/shorter than the actual one. The over-prediction cause the high overhead of the state transition, usually from the sleep state to the run state, while the under-prediction limits the possibility of energy saving⁸⁷. Therefore, if the model can provide the PM unit information about the future workload, the PM policy would be more efficiency.

5.3.2 Future Work

In the future, we will try to overcome the weak points mentioned above gradually.

For the first extension point, a possibility is by means of the interface. PAPI recent trends toward massively parallel multi-core systems with often heterogeneous architectures present new challenges for the measurement of hardware performance information. Good news are that PAPI is recently extend into Component PAPI, or PAPI-C in which multiple sources of performance data from I/O systems, memory or disks can be measured simultaneously via a common software interface⁸⁸. This work is necessarily somewhat dependent on the underlying hardware. It is more practicable to have an easier method to estimate the energy consumption of I/O peripherals or network devices through their corresponding drivers. Some heuristic assumptions can also be used to model the peripheral equipment requirements. For example, low memory activities and IPC could cause by frequent I/O interrupts. So, combine the extended PAPI and some necessary assumption may indicate a new direction to obtain the energy estimation model for the whole platform.

Aim to the second point, a simple and accurate feedback usually comes from the on-chip thermal sensors or internal ADCs to detect the current or voltage. For example, the hardware platform we are using now, Beagleborad, has an according block, TPS65950, to supply several key functions on the power and reset functions. In the boards there is a resistor which can be used to measure the Beagleboard's current. This resistor is connected to a 10-bit ADC and the measurement result can be passed to the processor via an I²C bus⁸⁴. However, a feedback methodology for the platform in which no ADC is available should also be considered.

The last point is the most important issues in the whole project. It will be the heart of the future research stage to focus on how to provide the estimation result to the PM unit to have a more efficient power management strategy.

5.4 Conclusion

In this chapter, the validation and evaluation of the estimation model is discussed. The results are evaluated based, on one hand, the relative errors by comparing the model results with the real measurements, and on the other hand, the scalability of the model, which denotes that the model should be accurate in various scenarios. The progresses of the model improvements are introduced to figure out the most important issues for the accuracy: a proper set of PMC to gather useful information of the system and the training method to avoid the bias of sampling. Both of the two issues aim to extract good representativeness. Our final model shows a good stability in different scenarios and a robust estimation result with an average relative error less than 5%

6 Conclusion

With the increasing gap between the complexity of the battery-powered devices and the battery capacity, the effort of energy optimization plays more important role in system designs. As a premise of the energy-optimizing strategies, this dissertation aims to provide a platform-independent methodology to estimate the energy.

This methodology relies on performance monitoring counters (PMCs) to gather relevant energy-related information from the system. PMC indicates the performance monitoring counter, which is widely implemented on most of the modern processors as a hardware register. PMCs can be configured to monitor the key energy-influencing events such as the total executed instructions and cache misses with the simple implementation and low overhead via the PMC interfaces. Moreover, PMCs can provide a future insight of the power bottleneck of the system. Considering the uniqueness of the PMC set of each platform and the limitation of the PMC that can be used simultaneously, a two-part methodology is proposed to build the estimation model. The first part is a PMC-filter, which identifies the most appropriate PMCs. It has two steps. First, it identifies the strong energy-related PMCs by figuring out the relationship between PMCs and energy. Second, it reduces the PMC redundancy to decrease the complicity of the robust linear regress method by considering the relationship between each pair of the PMCs. Both of the relationships are determined by the Spearman's correlation coefficients. The second part of the model is to improve the predictability and the generalization. It is a k-fold cross validation method to ensure each benchmark has the chance to be used to training the model.

For the sake of the simplicity, the implementation is based on two main components: the processor system and the outer-MPU hierarchical memory system, both of which are indispensable in any system and encompass the highest complicity. The two components are combined by their weights which are simply defined as their utilization ratios during the execution. The Linux 2.6.34 operating system is employed because it is the first Linux kernel version that supports the PAPI release. PAPI means the Performance Application Programming Interface. It is a cross-platform interface providing fully programmable interface to control the PMCs. The real measurement is taken to evaluate the accuracy of the model. It has been noted that the accuracy is highly related to the coverage of the system features abstracted by the selected PMCs and the training data. The final results show that the model can keep a good stability in different scenarios and provides a robust estimation result with an average relative error less than 5%.

This methodology works as a black-box which is able to automatically expose the energy estimation issue on any PMC-available battery-powered system. For the future work,

this methodology will be extended to the full system and applied to the power management unit to optimize the energy efficiency.

7 Reference

-
- [1] M. Weiser, B. Welch, A. J. Demers and S. Shenker, "Scheduling for reduced CPU energy", Proceeding of the 1st USENIX conference on Operating Systems Design and Implementation, pp. 13-23, 1994.
- [2] D. C. Snowdon, E. L. Sueur, S. M. Petters and G. Heiser, "Koala: A Platform for OS-Level Power Management", 4th EuroSys Conference, Nuremberg, Germany, pp.289-302, Apr.2009.
- [3] D. Grunwald, P. Levis, K. I. Farkas, C. B. Morrey III and M. Neufels, "Policies for dynamic clock Scheduling", In 4th OSDI, pp. 73-86, Oct. 2000.
- [4] D. C. Snowdon, S. Ruocco, and G. Heiser, "Power management and dynamic voltage scaling: Myths and facts", In 2005 WS Power Aware Real-time Computer, Sep 2005.
- [5] T.K. Tan, A. Raghunathan and N.K. Jha, "EMSIM: An Energy Simulation Framework for an Embedded Operating System", In Proc. ISCAS 2002, May 2002.
- [6] A. Genser, C. Bachmann, J. Haid, C. Steger and R. Weiss, "An Emulation-Based Real-Time Power Profiling Unit for Embedded Software", Proceedings of the 9th international conference on Systems, architectures, modeling and simulation, pp.67-73, 2009.
- [7] S. M. Kang, "simulation of power dissipation in VLSI circuits", IEEE J. Solid-State Circuits, vol. SC-21, pp. 889-891, Oct. 1986.
- [8] G. Y. Yacoub and W. H. Ku, "An accurate simulation technique for short-circuit power dissipation based on current component isolation", IEEE Int. Symp, on Circuits and Systems, pp. 1157-1161, 1989.
- [9] R. Burch, F. Najm, P. Yang, and D. Hocesvar, "Pattern-independent current estimation for reliability analysis of CMOS circuits", In Proceedings of the 25th ACM/IEEE Design Automation Conference, 1988.
- [10] F. Najm, "Transition density, a stochastic measure of activity in digital circuits," In Proceeding of the 28th ACM/IEEE Design Automation Conference, pp.644-649, Jun. 1991.
- [11] R. Marculescu, D. Marculescu, and M. Pedram, "Logic level power estimation considering spatiotemporal correlations," In Proceeding of the IEEE International Conference on Computer Aided Design, 1994.
- [12] J. Monteiro, S. Devads, B. Lin, C-Y Tsui, M. Pedram, "Exact and approximate methods of switching activity estimation in sequential logic circuits," In Proceedings of the 1994 International workshop on low-power design, pp.117-122, Apr.1994.
- [13] B. J. George, G. Yeap, M. G. Wloka, S. C. Tyler, and D. Gossain , "Power analysis for semi-custom design", In Proceedings of the IEEE Custom Integrated Circuits Conference, 1994.
- [14] C. X. Huang, B. Zhang, A. Deng, and B. Swirski, "Design and implementation of PowerMill", in Proceedings of the International Symposium on Low Power Design (ISLPED '95), pp. 105–109, Apr. 1995.
- [15] F. Najm, "A survey of power estimation techniques in VLSI circuits," IEEE Trans. VLSI Syst., vol. 2, pp. 446–455, Dec. 1994.
- [16] R. Burch, F. N. Najm, P. Yang and T. Trick, "A Monte Carlo approach for power estimation", IEEE Trans. VLSI syst., vol.1, pp. 63-71, Mar. 1993.
- [17] F. N. Najm, R. Burch, P. Yang, and I. N. Hajj, "Probabilistic simulation for reliability analysis of CMOS circuits," IEEE Trans. Computer-Aided Design, vol. 9, pp. 439–450, Apr. 1990.

-
- [18] F. N. Najm, "Transition density: A new measure of activity in digital circuits," *IEEE Trans. Computer-Aided Design*, vol. 12, pp. 310–323, Feb. 1993.
 - [19] A. A. Ghosh, S. Devadas, K. Keutzer, and J. White, "Estimation of average switching activity in combinational and sequential circuits," in *Proc. 29th Design Automation Conf.*, pp. 253–259, June 1992.
 - [20] T. Chou and K. Roy, "Accurate power estimation of CMOS sequential circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 4, no. 3, pp. 369–380, 1996.
 - [21] C. Ding, C. Tsui, and M. Pedram, "Gate-Level Power Estimation Using Tagged Probabilistic Simulation", *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, No. 11, Nov. 1998.
 - [22] Synopsys, Inc., "Synopsys products".
 - [23] EECS Department of the University of California at Berkeley, "Spice".
 - [24] K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan, "Temperature-aware micro architecture: Modeling and implementation," *ACM Transactions on Architecture and Code Optimization*, vol. 1, no. 1, pp. 94–125, Mar. 2004.
 - [25] J. Frenkil, "Tools and Methodology for low power design", *Proceedings of the 34th annual Design Automation Conference*, pp.76-81, June 1997.
 - [26] D. Brooks, and V. Triwari, "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations", *Proceedings of the 27th annual international symposium on Computer architecture*, vol.28, no.2, May 2000.
 - [27] T. Mudge, "Power: A First-Class Architectural Design Constraint", *Published in Journal computer*, vol.34, no.4, Apr.2001.
 - [28] N. S. Kim, T. Austin, T. Mudge and D. Grunwald, "Challenges for architectural level power modeling", *Published in book power aware computing*, pp.317-337.
 - [29] S.Wilton, and N.Jouppi, "An Enhanced Access and Cycle Time Model for On-Chip Caches", 1994.
 - [30] G. Jochens, L.Kruse, E.Schmidt, and W.Nebel, "A New Paramiterizable Power Macro-Model for Datapath Components", *Proceedings of the conference on Design, automation and test in Europe*, 1999.
 - [31] M. B. Kamble, and K. Ghose, "Analytical Energy Dissipation Model for Low-Power Caches", *Proceedings of the 1997 international symposium on low power electronics and design*, pp.143-148, Aug. 1997.
 - [32] A. Bogliolo, L. Benini, and G. D. Micheli, "Regression-based RTL Power Modeling", *Published in Journal ACM transactions on design automation of electronic systems (TODAES)*, vol.5, no.3, pp.337-372, July 2000.
 - [33] M. M. Khellah, and M. I. Elmasry, "Effective Capacitance Macro-Modeling for Architectural-Level Power Estimation", *Proceedings of the Great Lakes Symposium on VLSI*, pp.414, 1998.
 - [34] C. Z. ping, K. Roy, and E. K. Chong, "Estimation of Power Dissipation Using a Novel Power Macromodeling technique", 2000.
 - [35] S. Powell and P. Chau, "Estimating power dissipation of VLSI signal processing chips: the FA technique", in *proceedings IEEE workshop on VLSI Signal Processing*, vol.4, pp.250-259, 1990.
 - [36] N. Kumar, S. Katkoori, L. Rader, and R. Vemuri, "Profile-driven behavioral synthesis for low-power VLSI systems", *Published in Journal IEEE Design & Test*, vol.12, no.3, May 2010.
 - [37] D. Liu and C. Svensson, "Power consumption estimation in CMOS VLSI chips", 1994.

-
- [38] P. E. Landman and J. M. Rabaey, "Activity-sensitive architectural power analysis for the control path", proceedings of the 1995 international symposium on low power design, pp.93-98, Apr.1995.
- [39] H. Mehta, R. M. Owens, and M. J. Irwin, "Energy characterization based on clustering", proceedings of the 33rd annual design automation conference, pp.701-707, Jun. 1996.
- [40] S. Gurumurthi, A. Sivasubramaniam, M. J. Irwin, N. Vijaykrishnan, M. Kandemir, T. Li and et al, "Using complete machine simulation for software power estimation: the softWatt approach," in Proceedings of the 8th International Symposium on High-Performance Computer Architecture, pp. 141–151, Feb. 2002.
- [41] D. Brooks, V. Tiwari, and M.Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations," SIGARCH Computer Architecture News, vol. 28, no. 2, pp. 83– 94, 2000.
- [42] V.Tiwari, S.Malik and A.Wolfe, "Power Analysis of Embedded Software: A First Step Towards Software Power Minimization", IEEE Trans on VLSI Systems, pp. 437-445, Dec. 1994.
- [43] M. T. C. Lee, V. Tiwari, S. Malik and M. Fujita, "Power Analysis and Minimization Techniques for Embedded DSP Software", IEEE Transaction on VLSI Systems, pp. 123-135, Mar.1997.
- [44] V. Tiwari, S. Malik, A. Wolfe and M. T. C. Lee, "Instruction Level Power Analysis and Optimization of Software," Journal of VLSI Signal Processing, pp. 1-18, 1996.
- [45] C. H. Gebotys and R. J. Gebotys, "An Empirical Comparison of Algorithmic, Instruction and Architectural Power Prediction Models for High Performance Embedded DSP Processors," Proceedings of the 1998 international symposium on Low power electronics and design, pp 121-123, Aug. 1998.
- [46] M. T. C. Lee, M. Fujita, V. Tiwari, and S. Malik, "Power Analysis and Minimization Techniques for Embedded DSP Software", IEEE transactions on VLSI systems, vol.5, no.1, pp.123-135, Mar 1997.
- [47] B. Klass, D. E. Thomas, H. Schmit, and D. F. Nagle, "Modeling inter-instruction energy effects in a digital signal processor," in Proceedings of the Power Driven Microarchitecture Workshop in Conjunction with International Symposium Computer Architecture, June 1998.
- [48] A. Sama, J. F. M. Theeuwens, and M. Balakrishnan, "Speeding up power estimation of embedded software," in Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED '00), pp. 191–196, 2000.
- [49] N. Julien, J. Laurent, E. Senn and E. Martin, " Power consumption modeling and characterization of the TI C6201", IEEE Micro, vol.23, no.5, pp.40-49, Sept-Oct, 2003.
- [50] J.Laurent, N.Julien, E. Senn, E. Martion, "Functional level power analysis: An efficient approach for modeling the power consumption of complex processor", proceedings of the conference on Design, automation and test in Europe, vol.1, pp.10666, Feb.2004.
- [51] J.Laurent, "SoftExplorer: Estimation, characterization and optimization of the Power and Energy Consumption at the Algorithmic Level", IEEE power and timing modeling, optimization and simulation, 2004.
- [52] W. Wolf, "Household hints for embedded systems designers," IEEE Computer Society Press, 2002.
- [53] G. González, E. Juárez, J. J. Castro and C. Sanz, "Energy Consumption Estimation of an OMAP-Based Android Operating System", VLSI Circuits and systems conference, pp.18-20, Apr.2011.
- [54] OMAP3530 Power Estimation Spreadsheet,
http://processors.wiki.ti.com/index.php/OMAP3530_Power_Estimation_Spreadsheet

- [55] L. Benini, R. Hodgson, P. Siegel, "System-level Power Estimation And Optimization", Proceedings of the 1998 international symposium on low power electronics and design, pp.173-178, Aug. 1998.
- [56] L. Benini, and G. Micheli, "System-Level Power Optimization: Techniques and Tools", ACM transactions on design automation of electronic systems, vol.5, no.2, Apr.2000.
- [57] D. Sunwoo, H. Al-Sukhni, J. Holt and D. Chiou, "Early Models for System-level Power Estimation", 8th International Workshop on Microprocessor Test and Verification, 2007.
- [58] Y. Cho, Y. Kim, S. Park and N. Chang, "System-Level Power Estimation using an on-chip Bus Performance Monitoring Unit ", Proceedings of the 2008 IEEE/ACM international conference on computer-aided design, pp.149-154, 2008.
- [59] G. Contreras, M. Martionosi, "Power Prediction for Intel XScale Processor Using Performance Monitor Unit Events", In proceedings of the International symposium on low power electronics and design, pp. 221-226, 2005.
- [60] A. S. Dhodapkar and J. E. Smith, "Managing Multi-Configuration Hardware via Dynamic Working Set Analysis", Proceedings of the 29th annual international symposium on computer architecture, vol.30, no.2, May 2002.
- [61] Oprofile, <http://oprofile.sourceforge.net/about/>.
- [62] Perfctr, http://www.aie.csce.kyushu-u.ac.jp/~satoshi/how_to_use_perfctr.htm.
- [63] PAPI, <http://icl.cs.utk.edu/papi/>.
- [64] F. Bellosa, "The benefits of event-driven energy accounting in power-sensitive systems", In proceedings of the 9th ACM SIGOPS European Workshop, pp.37-42, Sept. 2000.
- [65] T. Li, and L. K. John, "Run- time Modeling and Estimation of Operating System Power Consumption", In Proceedings of the ACM SIGMETRICS international conference on Measurement and modeling of computer systems, pp160-171, vol.31, no.1, June 2003.
- [66] C. Lively, X. F. Wu, V. Taylor, S. Moore, H. Chang, and C. Su et al, "Power-Aware Predictive Models of Hybrid (MPI/OpenMP) Scientific Applications on Multicore Systems", International Conference on Energy-Aware High Performance Computing, Sept. 2011.
- [67] B. Goel, S. McKee, R. Gioiosa, K. Singh, M. Bhadauria, and M. Cesati, "Portable, scalable, per-core power estimation for intelligent resource management", In Proceedings of the 2010 International conference on Green Computing, pp135 –146, Ago. 2010.
- [68] V. Jimenez, F.J. Cazorla, R. Gioiosa, M. Valero, C. Boneti, E. Kursun, et al, "Characterizing Power and Temperature Behavior of POWER6-Based System", Published on Emerging and Selected Topics in Circuits and Systems, pp.228-24, Sept. 2011.
- [69] X. Yu, R. Bhaumik, Z.Y. Yang, M. Siekkinen, P. Savolainen, and A. Ylä-Jääski, "A System-level Model for Runtime Power Estimation on Model Devices", IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing, pp.27-34, Dec.2010.
- [70] C. Isci and M. Martonosi, "Runtime power monitoring in high-end processors: methodology and empirical data," in MICRO-36: Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture, IEEE Press, pp. 93–104, Dec.2003.
- [71] Vince Weaver, "perf_event programming guide", <http://web.eecs.utk.edu/~vweaver1/projects/perf-events/programming.html>.
- [72] perfmon2, http://perfmon2.sourceforge.net/docs_v4.html.
- [73] Cortex A8 Technical Reference Manual, Revision r2p2, 2008.

- [74] N. J. Salkind, "Statics for People Who (Think They) Hate Statistics", SAGE Publications, Inc.
- [75] F. E. Grubbs, " Procedures for detecting outlying observations in samples", *Technometrics* 11, pp.1–21, 1969.
- [76] R. Gnanadesikan, and J. R Kettenring, " Robust Estimates, Residuals, and Outlier Detection with Multiresponse Data", Published on International Biometric Society, pp. 81-124, Mar.1972.
- [77] J. Cohen, "Statistical power analysis for the behavioral sciences (2nd ed.)," Hillsdale, NJ: Lawrence Erlbaum, 1988.
- [78] A New View of Statistics, Hopkins, Will G., electronic edition:
<http://www.sportsci.org/resource/stats/index.html>.
- [79] R. Kohavi, "A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection," 14th international joint conference on Artificial intelligence, pp.1137-1143, Aug. 1995.
- [80] M. F. R. Resende, J. P. Munoz, M. D. V. Resende, D. J. Garrick, R. L. Fernando and J.M Davis et al, "Accuracy of Genomic Selection Methods in a Standard Data Set of Loblolly Pine", *GENETICS*, Apr.2012.
- [81] The Architecture of the Nehalem Processor and Nehalem-EP SMP Platforms
- [82] S. Pasricha, A. Veidenbaum, "Improving Branch Prediction Accuracy in Embedded Processor in the Presence of Context Switches", 2003.
- [83] White Paper, Intel Next Generation Microarchitecture (Nehalem)
- [84] BeagleBoard System Reference Manual Rev C4", Dec.2009.
- [85] <http://web.eecs.utk.edu/~vweaver1/projects/perf-events/support.html>
- [86] V.Jimenez, F.J.Cazorla, R. Gioiosa, M. Valero, C. Boneti, E. Kursun and et.al, "Characterizing Power and Temperature Behavior of POWER6-Based System", *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol.1, no.3, pp. 228-241, 2011.
- [87] L. Benini, A. Bogliolo, G. Micheli, "Survey of Design Techniques for System-Level Dynamic Power Management", *IEEE transactions on VLSI systems*, vol.8, no.3, pp.299-316, 2000.
- [88] D. Terpstra, H. Jagode, H. You, and J. Dongarra, "Collecting Performance Data with PAPI-C", Published in *Tools for High Performance Computing*, pp.157-173, 2010.