DEPARTAMENTO DE INGENIERÍA TELEMÁTICA Y ELECTRÓNICA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
Y SISTEMAS DE TELECOMUNICACIÓN

Energy Management and Optimization for Video Decoders based on Functional-oriented Reconfiguration

TESIS DOCTORAL

Rong Ren

Master Universitario en Ingeniería de Sistemas y Servicios para la Sociedad de la Información

DIRECTOR

Eduardo Juárez Martínez

Docteur ès Sciences Techniques por el École Polytechnique Fédéral de Lausanne

César Sanz Álvaro

Doctor Ing.de Telecomunicación por la Universidad Politécnica de Madrid

2015

“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

Time flies!

Countless fleeting images during the past four years are shining in my mind. There were joys, there were sorrows; there were tears, there were sweats. During the four years, 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, my four-year is coming to be finished, my heart is filled with appreciations. Ups and downs were not the pursuit of life, but must be full of life.

Foremost, 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 am especially grateful to thank my supervisor, Eduardo Juárez Martínez for providing me with an opportunity to carry out my doctorate thesis project. I thank him for his support and encouragement. He guides me throughout the project and cleared all my doubts and questions patiently. 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 for their help and support all the way through my study. They have been very inspiring in my research and thoughtful in every aspects of my life.

I would like to thank all my colleagues at the research group for their valuable suggestions throughout the project.

I would like to thank all my friends: 魏建国, 黄姗, 美娟, 柴亮, 王宪, 汤琼, 土豆, 温馨, 钟如意, either in china or here, for their love and concern. 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

Figure List

Table List

Resumen

En los últimos años, la sofisticación creciente de los sistemas empotrados y las tecnologías de comunicación inalámbrica ha promovido la utilización cada vez más importante de las aplicaciones de vídeo *streaming*. Tal y como se ha publicado en el año 2013, la generación de jóvenes con edades comprendidas entre 13 y 24 años emplea, aproximadamente, 16.7 horas a la semana viendo vídeos en línea a través de las redes sociales, sitios web de negocio o de vídeo *streaming*. Se puede decir, por tanto, que el vídeo forma parte ya de la vida de las personas. Hasta ahora, la investigación en estos asuntos se ha centrado en la mejora del rendimiento, es decir, el incremento de la tasa binaria y la reducción del tiempo de respuesta. Sin embargo, la mayoría de dispositivos móviles están alimentados por baterías. Esta tecnología, es bien sabido, que avanza a una menor velocidad que los desarrollos multimedia o hardware. Debido a que la investigación en baterías no satisface la creciente demanda de energía de los dispositivos móviles, la investigación en aplicaciones de vídeo se centra más y más en la eficiencia energética. Cómo utilizar eficientemente el presupuesto escaso de energía disponible se ha convertido en uno de los principales retos de la investigación. Además, los estándares de vídeo de última generación tienden hacia la diversificación y personalización. Por tanto, es también deseable disponer de mecanismos para optimizar la energía con mayor flexibilidad y escalabilidad.

En este contexto, el objetivo principal de esta tesis es encontrar un mecanismo de gestión y optimización que reduzca el consumo de energía de los descodificadores de vídeo aplicando la idea de reconfiguración funcional. El tiempo de uso de la batería del sistema se extiende como resultado de un compromiso entre energía consumida y calidad de vídeo. La reconfiguración funcional aprovecha las similitudes entre estándares para construir descodificadores de vídeo mediante la interconexión de unidades funcionales existentes. En el caso de que se disponga de un canal de retorno entre el descodificador y el codificador, el primero puede señalar al segundo cambios en los parámetros de codificación o en los algoritmos para adaptarse con el fin de ahorrar energía.

El mecanismo propuesto de optimización y gestión de energía se materializa en el descodificador. Este mecanismo está formado por un gestor de reconfiguración basado en criterios energéticos, implementado como bloque adicional del motor genérico de reconfiguración, un estimador del consumo de energía, incorporado al descodificador, y, si está disponible, un canal de retorno conectado al codificador. El gestor de reconfiguración verifica el nivel de la batería, selecciona la descripción del nuevo descodificador e informa al motor de reconfiguración de la recomposición de un nuevo descodificador. Nótese que el análisis del consumo de energía es fundamental para el

funcionamiento correcto del mecanismo de gestión y optimización de energía. En esta tesis se propone un método de estimación de energía basado en la observación de eventos del sistema. También se propone un filtro de estos eventos para automatizar la selección de los más relacionados con el consumo de energía. Por último, se incluye un estudio detallado de la influencia de las secuencias de aprendizaje en la precisión del modelo.

La metodología de modelado del estimador de energía se ha evaluado en diferentes plataformas, mono- y multinúcleo con bancos de prueba de características diferentes. Los resultados confirman que la precisión del modelo es buena y su carga computacional baja. Las modificaciones realizadas en el motor de reconfiguración para implementar el gestor basado en criterios energéticos se han verificado en diversos escenarios. Los resultados indican la posibilidad de alargar el tiempo de vida de la batería del sistema en dos casos de uso diferentes.

Abstract

In recent years, the increasing sophistication of embedded multimedia systems and wireless communication technologies has promoted a widespread utilization of video streaming applications. It has been reported in 2013 that youngsters, aged between 13 and 24, spend around 16.7 hours a week watching online video through social media, business websites, and video streaming sites. Video applications have already been blended into people daily life. Traditionally, video streaming research has focused on performance improvement, namely throughput increase and response time reduction. However, most mobile devices are battery-powered, a technology that grows at a much slower pace than either multimedia or hardware developments. Since battery developments cannot satisfy expanding power demand of mobile devices, research interests on video applications technology has attracted more attention to achieve energy-efficient designs. How to efficiently use the limited battery energy budget becomes a major research challenge. In addition, next generation video standards impel to diversification and personalization. Therefore, it is desirable to have mechanisms to implement energy optimizations with greater flexibility and scalability.

In this context, the main goal of this dissertation is to find an energy management and optimization mechanism to reduce the energy consumption of video decoders based on the idea of functional-oriented reconfiguration. System battery life is prolonged as the result of a trade-off between energy consumption and video quality. Functional-oriented reconfiguration takes advantage of the similarities among standards to build video decoders reconnecting existing functional units. If a feedback channel from the decoder to the encoder is available, the former can signal the latter changes in either the encoding parameters or the encoding algorithms for energy-saving adaption.

The proposed energy optimization and management mechanism is carried out at the decoder end. This mechanism consists of an energy-aware manager, implemented as an additional block of the reconfiguration engine, an energy estimator, integrated into the decoder, and, if available, a feedback channel connected to the encoder end. The energy-aware manager checks the battery level, selects the new decoder description and signals to build a new decoder to the reconfiguration engine. It is worth noting that the analysis of the energy consumption is fundamental for the success of the energy management and optimization mechanism. In this thesis, an energy estimation method driven by platform event monitoring is proposed. In addition, an event filter is suggested to automate the selection of the most appropriate events that affect the energy consumption. At last, a detailed study on the influence of the training data on the model accuracy is presented.

The modeling methodology of the energy estimator has been evaluated on different underlying platforms, single-core and multi-core, with different characteristics of workload. All the results show a good accuracy and low on-line computation overhead. The required modifications on the reconfiguration engine to implement the energy-aware manager have been assessed under different scenarios. The results indicate a possibility to lengthen the battery lifetime of the system in two different use-cases.

PART A

Chapter 1: Introduction

1. Introduction

When using computer equipment to process data, it is always desirable to employ natural methods to obtain more intuitive results. The word "natural" means that the processing method can obtain the results directly through sensorial organs rather than using the brain to reprocess or recalculate them. Multimedia information which includes audio, images, video, and text is exactly in the line with this demand and thus is increasingly being favored. Currently, the computing capabilities of mobile devices have been rapidly improved and lead to a boost development of multimedia applications. However, most of the mobile devices are powered by batteries, which, unfortunately, are experiencing a relatively slow development. The battery lifetime of many mobile devices, especially Smartphones and tablets, easily fails to guarantee the user-desired lifetime. Energy constraint has become the major limitation on the developments of computation intensive applications. This chapter will present an introduction of the battery-limited problem. It is organized as follows: section 1.1 discusses energy constraint challenges, experienced in multimedia applications, especially, video coding applications; section 1.2 presents the research motivations of low-power design; section 1.3 introduces the objectives and methodologies of the thesis work. Briefly speaking, this thesis proposes a mechanism of energy optimization and management for video coding, especially for video decoding, based on functional-oriented reconfiguration. The goal is to extend the battery lifetime of mobile devices through the energy consumption control of video decoding; finally in section 1.4, the contents of this dissertation are outlined.

1.1. Challenges

Vision is the most direct approach for understanding this world. Approximate 70% of the outside accepted information comes from the visual sense. This information presents a colorful world in the form of images. With the social progress and technology development, people demand of image information is gradually increasing. The demand promotes flourish improvements on the related technologies. Forming as dynamic images, video information is featured by a huge quantity of data which introduce great difficulties on expression, organization, storage, and transmission. It is fundamental to compress the original video data for practical demands. Video coding has thus gradually become a hot topic in research communities.

Since 1980s, video coding standardization has been progressing. A series of international standards of digital video coding were established for different network bandwidth and quality requirements. At the same time, the implementation of digital video coding technologies has been astoundingly advancing. One trend of video applications is the real-time data processing accompanied

by a new computing load, named as streaming computing. The so-called streaming is an uninterrupted, continuous, and moving data queue. The application which can organize data into a video streaming and operate on it is called streaming application. The embedded multimedia systems which support streaming applications are increasingly widely used in communication, networking, consumer electronics, and other fields. Streaming applications represented by video and audio serve as an important role in multimedia. Millions of Internet users enjoy video streaming every single day. For instance, video streaming has constituted the largest part of the multimedia applications in Facebook, Google+ and other social media [1]. Whitepapers from comScore [3] [4] reported that Smartphone users in Europe had passed 50% in December 2012 and Spanish users showed the highest adoption (66%. 75%) of Smartphones in five European countries (EU5: France, Germany, Italy, Spain and UK). In addition, over 23 million people in the EU5 countries had both Smartphones and tablets which had increased 94% than 2011. The total number of online video viewers had increased to 162% compared to the past year. In USA, the online video market had attracted 75 million audiences every day and nearly 40 billion videos per month in average. Smartphone market penetrated 50% at the end of third quarter in 2012 and 25% Smartphone owners also had a tablet which had emerged as a critical piece of mobile device landscape. Half of the tablet owners had reported that they used their tablets to watch video and TV programming. Both in Europe and USA, nearly 33% digital media minutes were spent on Smartphones and tablets. In addition, the Cisco Visual Networking Index forecast report also indicated that all forms of video content would continue to occupy approximately 90% of the global consumer internet traffic by 2015 [2]. Besides the daily normal utilizations, video coding and streaming are envisioned in numerous using areas including such as battlefield intelligence and reconnaissance, public security and surveillance, emergency response and disaster rescue, and telemedicine.

Digital video has rapidly migrated across platforms and application contexts with the increase in user requests and demands. The demand for high-definition video was predicted to surpass standard definition format by the end of the year 2011 [2]. Higher quality video requires higher bit rates and thus requires greater performance and energy consumption. In recent years, with the increasing sophistication of the manufacture processing, the size of integrated circuits has been scaling down. Then, hardware technology prompts users to have greater demands of video quality. Meanwhile, the coding algorithms have also been improved. For example, the video coding standard H.264/AVC is able to provide half or even less the bit rate of previous standards while maintain the video quality [5]. The next generation video standard puts forward requirements of codec performance, compression efficiency, and other key technologies, towards a development on diversification and personalization.

However, energy consumption of mobile devices such as Smartphones and tablets is becoming increasingly serious with the development of video coding technologies. Mobile device itself is facing several energy challenges; video application makes the energy problem particularly obvious. In video application, video coding is considered as the most computation and energy intensive part. In particular, the stream computing is also bandwidth and delay intensive. These features lead to a great demand on system capacities for communication and computing, and consequently, a very high demand on energy support. Unfortunately, most mobile devices are designed as battery-powered and the development of battery technology falls far behind than that of either video applications or hardware, which doubles the processing power in every two years through Moore's law; Batteries do not event offer capacities twice larger over the last decade. Currently, most mobile devices are powered by lithium-ion batteries which offer more energy than other types of batteries [7]. Even this, it is not sufficient to increase the amount of energy created by chemical reactions. Some research groups have analyzed the nonlinear characteristics of different types of battery to achieve battery utilization. Other researchers try to exploit user movements to recharge batteries, but this is an initial research field [11]. Relying solely on battery performance improvement is difficult to solve the problem fundamentally. The only way to produce more powerful batteries seems to increase their size. However, this goes against to the lighter-and-thinner design trend of mobile devices which would offer more space for additional components rather than battery. The slow development of battery technology cannot provide sufficient energy, which makes critical difficulties to provide an as-long-as-possible battery life with a limited energy budget. The battery depletion becomes the major drawback of the electronic field to constraint video coding developments. As a consequence, more researches shift into energy-efficient designs.

1.2. Motivations

As discussed above, excessive energy demand and consumption are critical limitations on the evolution of mobile hardware and services due to the quite moderate battery capacity developments [7]. The operational time within one charging cycle is limited to the fixed amount of energy stored in batteries. As the operational time is one of the most important factor for user satisfaction assessment on mobile consumer electronic devices [8][9], a failure to guarantee the user-expected lifetime will significantly decrease user desirableness. Therefore, the consumer electronics industry is motivated to find solutions for operation time extension.

Research results indicated that processor intensive applications such as video playback can consume over 60% of the energy budget [6]. Thus, in this context, energy management and optimization on video application has become an attractive research topic to extend the battery

lifetime. How to build a framework which is able to adaptively optimize the energy consumption catches many research interests. In general, two kinds of method have engaged researches on this issue:

- The first method provides the energy efficiency optimization achieved by improvements on new techniques of micro-electronics. As known that the power consumption is proportional to the load capacitance, circuit switching activities, and operating voltage, current and frequencies, these technologies aim to reduce the power-impact factors by improving circuit layouts, circuit logic designs, register transfer level designs, and advanced architecture designs with lower operating voltage supply. In this area, there have been many researchers and large teams working for a long time and it is difficult to make new contributions.
- The second direction investigates how applications can deal with power management. As the temporal efficiency and spatial resource conflict in computer science, it is also difficult to simultaneously achieve low energy consumption and high QoS (Quality of Service) in video applications. Various approaches in this direction conducted adaption based on workloads. For instance, dynamic voltage and frequency scaling (DVFS) is able to dynamically adjust the supply voltage and frequency according to the workload, which, in the video application field, is the amount of computational complexity related to the quality of the rendered video images and data compression ratio. Although DVFS has achieved positive results of management and reduction of energy consumption, this approach does not change the workload (computational complexity); if there would be a method which could tradeoff power consumption and the workload by adapting the computational complexity, it would potentially provide further battery life extension.

This thesis will follow the second research line. It proposes an energy management and optimization mechanism which can wisely switch among decoders with different computational complexities to adapt the energy consumption of the decoding process based on energy awareness.

1.3. Objectives

1.3.1. Brief Description of the Proposal

Figure 1-1 depicts a scenario in which users with different remaining battery over diverse wireless networks accesses connect to a video server. In this context, two fundamental issues need to be considered during the playback of a video stream. The former one is how to deliver the same video content over different access networks. The latter one is how to deal with the video playback time over devices with different remaining battery capacities. In other words, the problem could be restated asking how to deliver the same video content to various mobile devices under the network bandwidth

and battery energy constraints, while satisfying a user-desired operational time and video quality. Mathematically, the problem can be expressed as equation 1-1:

$$Max\ T, \qquad B \le B_0, E \le E_0, \ and\ Q \ge Q_0 \tag{1-1}$$

where T is the operational time, Q_0 is the dissatisfactory video quality, B_0 and E_0 are sustainable requirements of network bandwidth and remaining battery capacity, respectively. Note that the value of Q_0 could be different to each single user.

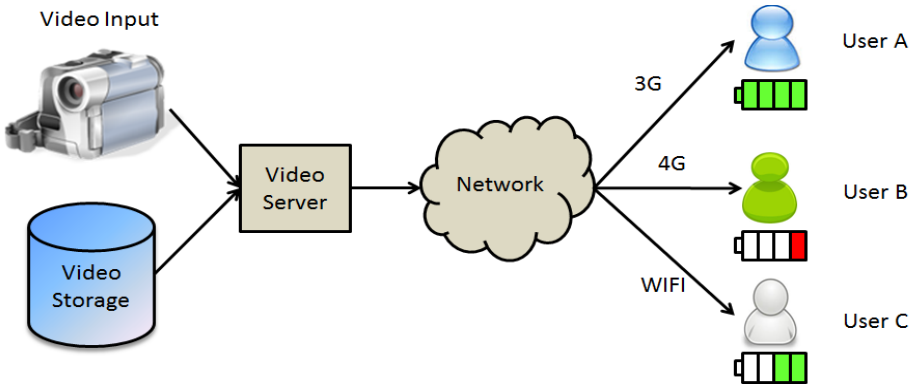


Figure 1-1 Example of Different Users with Different Remaining Battery over Wireless Networks

This problem statement encompasses three different issues:

- To deliver the same video content with different performance and capacity via different networks;
- To ensure a quality level above user expectations;
- To prolong the battery life of devices.

The main goal of this thesis focuses on the third issue mentioned above, i.e. to prolong the battery life as a trade-off between energy consumption and video quality. An example of this trade-off is shown in Figure 1-2. A video sequence is presented with different qualities according to different battery levels. Figure 1-2 (a) shows the scenario in which a device with a full battery presents a high image quality sequence. On the other hand, Figure 1-2 (b) illustrates a scenario in which the same device with a lower battery level presents a lower video quality sequence. Video quality can be adjusted by either configuring encoding parameters, such as bit-rate, or changing the encoding algorithms, such as the entropy coding.

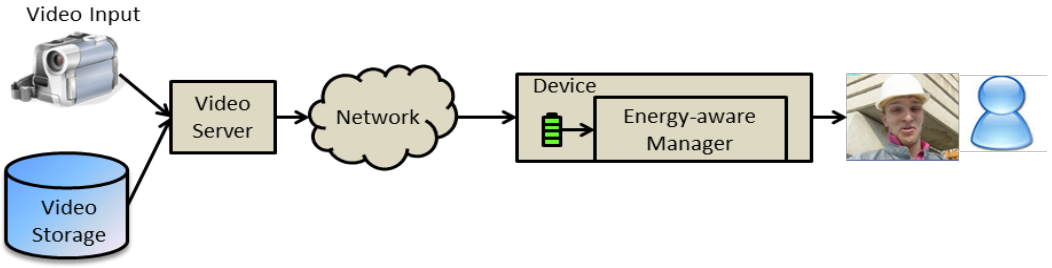


Figure 1-2 (a) Full Battery Case

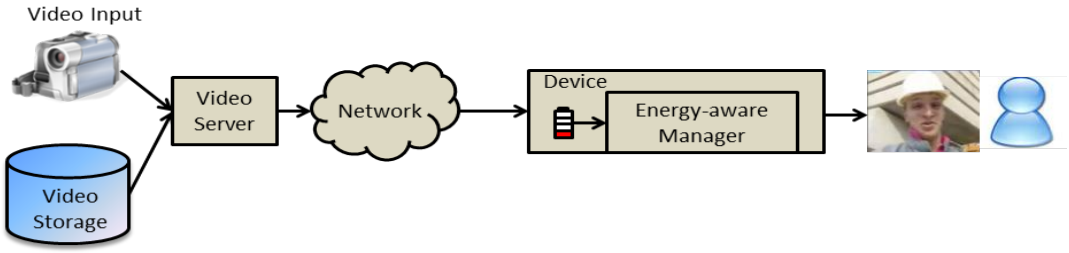


Figure 1-2 (b) Low Battery Case

Figure 1-2 Example of Battery Capacity Level and Video Image Quality

In video coding, decoding operations are the inverse of the encoding operations. Thus, decoders are restricted by the encoders. Video coding standards ensure the compliant compressed video data can be recognized and decoded. Thus, to keep the compatibility, a change in any encoding tool employed in the encoder side implies a change in the decoding procedure. Along with video coding technologies, user demand for video application exhibits an increasing diversification, i.e. higher quality requirements in addition of flexibility and scalability preferences [10]. Therefore, to fulfill these requirements, a variety of video standards have been developed. However, among the plethora of standards, some coding tools, namely functional unit (FU), are reutilized from one to the other.

In this thesis, a technique known as functional-oriented reconfiguration is employed. The functional-oriented reconfiguration takes advantage of the shared functionalities among standards to build a decoder by selecting and connecting FUs from an FU pool. To direct this process, a decoder description, i.e. a list of FUs and their interconnection, can be conveyed into the encoded bitstream. In case different decoder descriptions are received at the decoder end, depending on its current battery level, the decoder might dynamically adapt the quality of the decoded images to extend the battery lifetime. This technique might offer a large potential for energy savings.

1.3.2. Objectives

The main goal of this thesis is to find an energy management and optimization mechanism to reduce the energy consumption of video decoders based on the idea of functional-oriented reconfiguration. The framework that describes this mechanism is outlined in Figure 1-3.

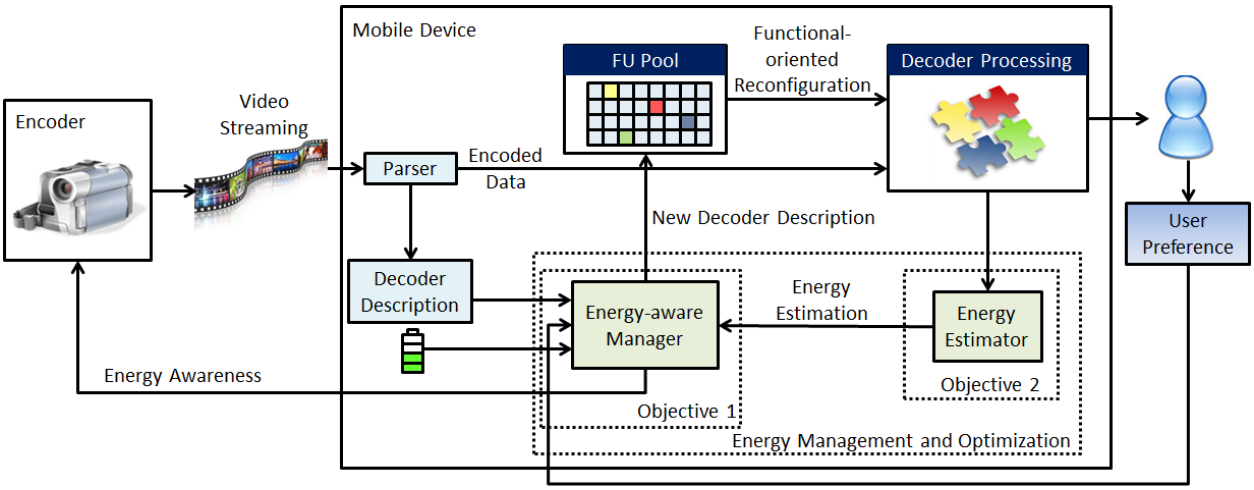


Figure 1-3 Outline of the Framework that Serves the Proposed Energy Management and Optimization Mechanism

The previous objective can be decomposed into the following two objectives:

- Objective1: to propose and implement an energy-aware manager to drive the functional-oriented reconfiguration;
- Objective 2: to propose and implement an energy estimator for a video decoder.

As can be seen in Figure 1-3, to optimize and manage the energy consumption of a video decoder, an energy-aware manager needs to be implemented as an additional unit of the original engine. The energy-aware manager checks the battery level and once the remnant energy is not enough to maintain the user-defined lifetime (a user preference), the energy-aware manager selects a new decoder description with different decoding features based on a set of criteria. Afterwards, the energy-aware manager informs the engine to reconfigure the decoder to build a new one. In is worth noting that the manager should ensure that the new selected decoder description is compatible with the received encoded video stream. In case it is required, the energy-aware manager might also inform the encoder to adapt its encoding algorithms based on the actual decoder reconfiguration.

The analysis of the energy consumed by a decoder is fundamental for the success of the energy management and optimization mechanism. In this thesis, en estimation method driven by the platform performance monitoring counters (PMCs) which record several performance events (e.g., number of executed instructions) that reflect the energy consumption in real-time is proposed.

1.4. Outline

The thesis is organized as following:

PART A (Chapter 1). This introduction has stated the energy challenges in multimedia applications, especially in the field of video decoding. To extend the battery life time, this thesis proposes a new energy optimization and management mechanism for video decoding based on the ideal of functional-oriented reconfiguration, which can provide flexible energy saving solutions.

PART B (Chapter 2 and 3) mainly discusses about energy estimation. Wherein, Chapter 2 introduces the state-of-the-art of energy models, from general to coding-focused models, and discusses the merits and demerits of each type. An estimation methodology is introduced next in Chapter 3 including its basic mechanism, fitting methods, and model accuracy. Chapter 3 proposes a modeling method which fits into the reconfigurable framework. In particular, the method is designed to improve the model generalization and accuracy.

PART C (Chapter 4 and 5) discusses energy optimization. Specifically, Chapter 4 first introduces different optimization techniques and then analyzes the main features of the current reconfiguration technologies. Before introducing the functional-oriented reconfiguration technique and its framework, an introduction is followed to discuss the computational characteristics of video decoding tasks, i.e. the fundamental factors for implementing the reconfiguration of video coding. Chapter 5 conducts a research on the energy optimization and management mechanism based on the functional-oriented reconfiguration. It proposes the energy-aware manager and a management and optimization metric for video decoding, giving a design example at the end.

PART D (Chapter 6 and 7) introduces the implementations of the proposed management and optimization mechanism. Chapter 6 first introduces the infrastructure of the implementation. Development environments, reconfiguration engine, hardware platforms, bench marks and modeling assistant tools are presented. Chapter 7 introduces the integration of the energy estimation model into the reconfiguration framework and extend the reconfiguration engine with the energy-aware manager to achieve the implementation of the energy optimization and management mechanism.

PART E (Chapter 8) draws the experimental results. Firstly, it gives the validation and evaluation of the models, including their estimation results, a guideline to choose the training sequences to improve model accuracy, extensions of the application range of the model, and the computation overhead. Second, the functionality of the energy-aware manager is tested and verified and, at last, the energy optimization potentialities are illustrated.

PART F (Chapter 9) summarizes the thesis and describes some problems needed to explore in the future.

PART B

Chapter 2: Energy Estimation: Research and Problems

Chapter 3: Generalization and Accuracy Improvements of the Energy Estimation Model

2. Energy Estimation: Research and Problems

With the gradually increased gap between battery capacity and energy demand of multimedia applications, efficient utilization of the limited energy has become one of the most attractive research topics in mobile multimedia systems design. Many dramatic optimization techniques for battery life extension have been proposed by research communities. In general, the optimization/reduction decisions are determined based on the energy awareness, which is usually obtained from estimation. This chapter starts with a brief introduction of energy estimation models in two kinds: general models and coding-focused models. After a discussion about advantages and disadvantages of each kind, the related techniques of the most suitable model for energy estimation on video streaming, especially for reconfigurable coding, are introduced, and finally, problems of this candidate model are presented.

2.1. Energy Estimation Models

An energy estimation model is the basis of adjustment of power consumption. A detailed survey of different energy modeling techniques will be described in following.

2.1.1. General Models

Although video coding is a class of application with its own characteristics independent to operating systems, it is bound to have common features as all computer applications, i.e., it is a collection of instructions and parameters, and the set of instructions controls computer running based on the established logics. Therefore, the general estimation models could be applied to video coding.

General models can be abstracted into low and high levels. Low-level models estimate energy consumption by extracting information from circuits, gates, register transfers, and architectures. Avoiding hardware details, high-level models process with instructions, functional units, and device components to profile system energy consumption from a software point of view. Usually, low-level models provide more accurate results but involve more complex design details and require long time estimation. These physical-level models are more proper used for power analysis during the design stage rather than power estimation during runtime. Considering the quick response and easy usage, it is more suitable to employ the high-level estimation models into the coding field, which has more strict time constrains. A brief overview of different high-level models is introduced below.

2.1.1.1. Instruction-Level Estimation Models

The power model based on instruction level for individual processors was firstly proposed by Tiwari et al [12]. Figure 2-1 shows how this model works. The power consumption of each instruction

is measured when a sequence of instructions is executed, for example, on a processor instantiated FPGA board. For each individual instruction, its power consumption is defined to include a base cost and an additional overhead. Then, the model is drawn from the measurements. At last, the model is employed to estimate the power consumption of a different piece of code. Tiwari et al continuously proposed experimental approaches to empirically determine the base cost and the inter-instructions overhead cost [13]. Their subsequent researches showed that for both the complex General Purpose Processors (GPPs) and Digital Signal Processors (DSPs) the base cost could be reduced to as an averaged constant because of the dominance of the overhead costs.

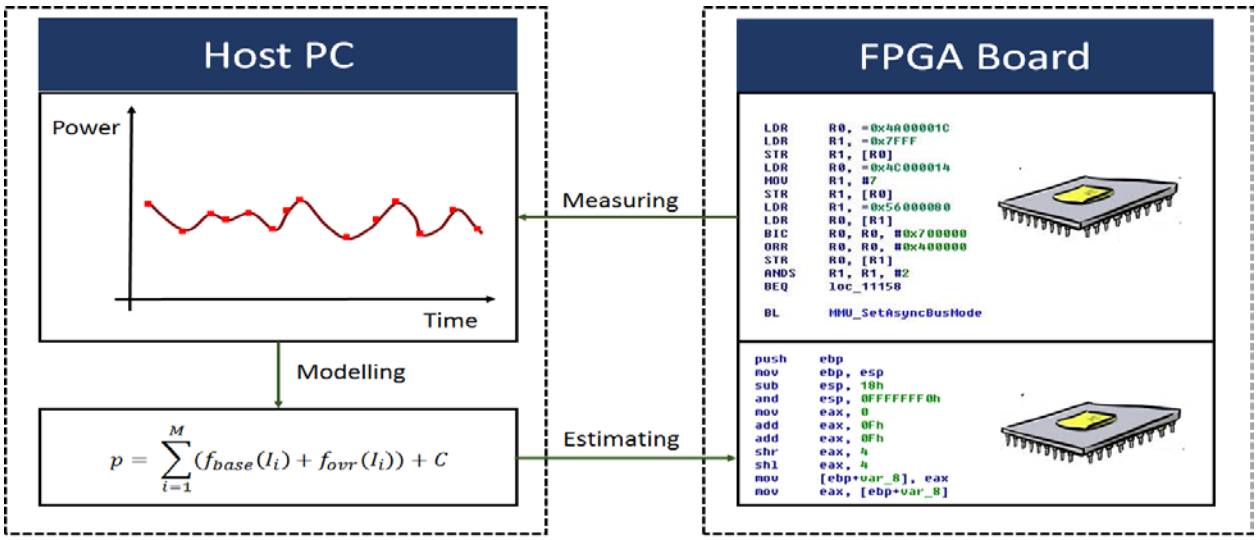


Figure 2-1 Principle Instruction-level Power Estimation

The different combinations of instructions cause a vast number of inter-instruction effects, which is the main disadvantage of this approach. A simple solution proposed by Lee et al. [14] was to classify instructions into four categories based on their functionalities and addressing modes: loading immediate data to a register, transferring memory data to registers, moving data between registers, and operating in ALU. However, this simply method encountered difficulties when the instruction set had various addressing modes and high parallelism. Klass et al. [15] proposed an approach to reduce the complexity by observing the inter-instruction effects when a generic instruction was executed after a no-operation instruction. They assumed that the inter-instruction overhead mainly depended on instruction changes. Thus they inserted an NOP instruction before changing any instruction to quantify the transition overhead. As a result, they did not need to enumerate each pair of instructions to build the instruction model. In other work, Sama et al. [16] attempted to provide substantial improvements based on Tiwari's work [13]. The base energy cost was measured by individually repeating executing each instruction and the overhead energy consumption came from the changes of opcodes and control states between the subsequent instruction, and the data passing was also added into their model. To reduce the complexity of instruction pairs, this method classified instructions on

the basis of the functionalities and base costs. Therefore, the instruction overheads were only needed to be measured for the intergroup pairs. For those instructions in the same group, this approach assumed the same instruction overhead because the similar functionality and base costs usually indicated similar control states and opcode values.

2.1.1.2. Function-Level Estimation Models

Function-level power analysis (FLPA) is applicable to all types of processor architecture without taking into account the details of the system circuits. Instead of classical energy characterization abstracted from the instructions, the basic idea of FLPA is to obtain the distinct energy consumption from system activities of different processor functional units. Thus, a FLPA model usually divides the target device into several functional units and relates the processor operations to the power activities. Each functional unit is a cluster of components which are concurrently activated when a task is running. The FLPA modeling procedure is abstracted in Figure 2-2.

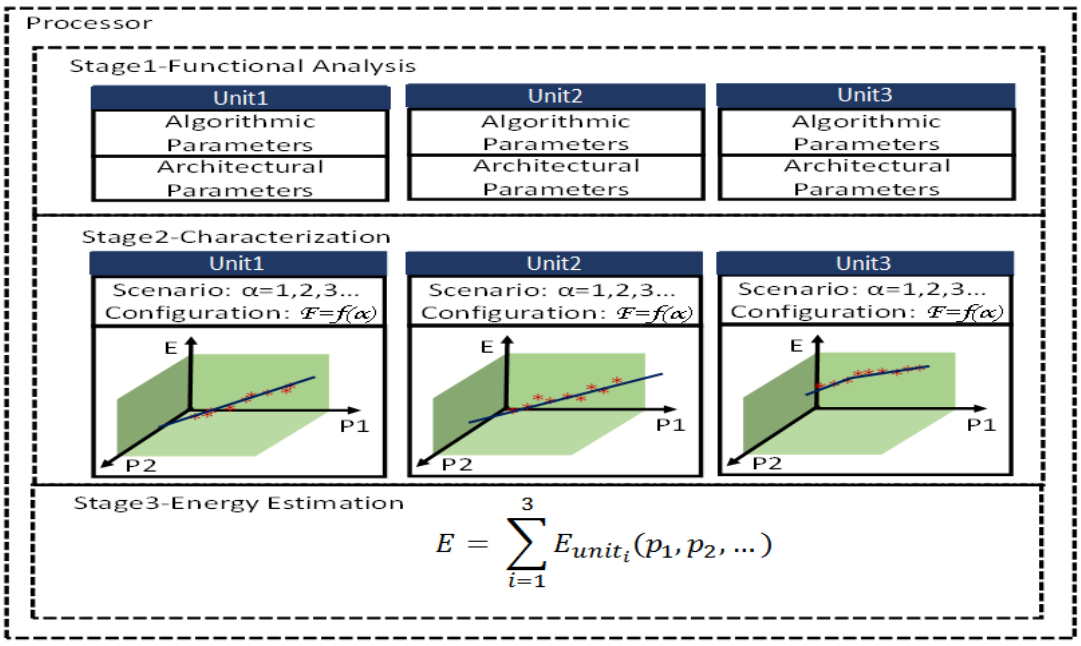


Figure 2-2 Processor Modeling Methodology

In this figure, the target processor is divided into 3 units. The first step of modeling is to characterize the system energy consumption of each unit. The proper parameters are selected from the executed algorithms (typically, the cache miss rate) and the processor configuration (typically, the clock frequency). For example, Nathalie et al. [17] divided a DSP processor into four units: instructions management unit (IMU), processing unit (PU), memory management unit (MMU), and control unit (CU). Similarly, Laurent et al. [18] abstracted complex DSPs into IMU, PU, and MMU units including parallelism/processing rate, cache miss rate and external data memory access rate as modeling parameters obtained by simulating each functional unit with small programs written in

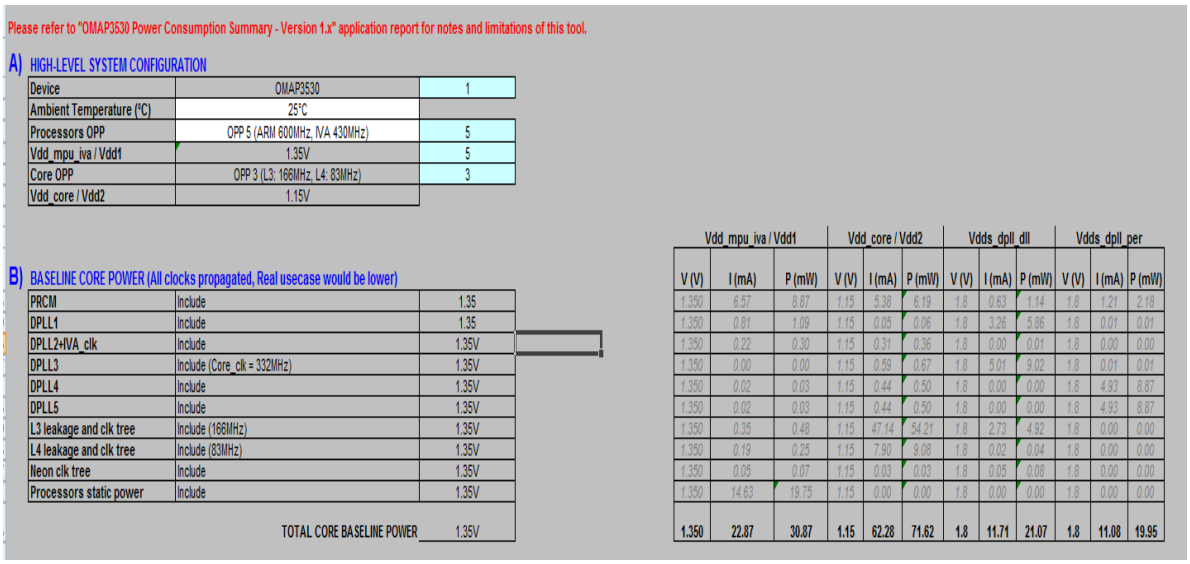
assembly language. Then, in the second step, various scenarios, α_i, are executed. Each scenario has different system configuration $F = f(\alpha_i)$. The consumed energy (e) and the values of modeling parameters (P_1, P_2) of each unit are recorded. Finally, the relationship between the consumed energy and the selected parameters is decided and modeled for each unit. The whole energy consumption of this processor is the summation of the energy consumption of each unit.

Based on FLPA methodology, the SoftExplor [19], a tool which automatically performs power and energy consumption estimations, has been widely used by lots of researches. SoftExplor facilitates the modeling process. It only requires coarse-grain knowledge on processor architectures and it achieves a good tradeoff between estimation accuracy and model complexity. More specifically, E. Senn et al. in their work Open-PEOPLE [20] presented a platform dedicated to provide a set of power analysis tools as a library of power models to develop power modeling methodologies with the considerations from entire embedded system including applications, hardware components, operating system (OS), and the associated services.

2.1.1.3. Component-Level Power Estimation Models

For better generalization, models abstracted in higher component-level have been proposed. Component-level models consider main system components (e.g., processor, memory, and coprocessor) and lead to more intuitive and feasible models.

High-level abstraction models can obtain the static pre-characterized energy consumption from spreadsheets provided by manufacturers. These spreadsheets are very useful in the early stage of the design process to achieve initial decisions with power issues [21]. Spreadsheets provide a capability to quickly estimate the current and power consumption of each intellectual property (IP) core or library cell. Developers can configure the operating frequency, temperature, and other parameters to estimate the power consumption of his design by using spreadsheets (Figure 2-3). An example of using a spreadsheet to estimate the power consumption of processes and multimedia applications was developed for the BeagleBoard, a commercial prototyping board based on the OMAP processor, by González et al. [22]. However, this approach is valid only when the hardware exhibits regular activity patterns. It might not be able to provide guidance for block-level hardware power estimation due to its lack of flexibility when the hardware has different work modes workloads. With the increasing importance of power management techniques, they are limited in 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_dll			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	8.87	1.15	5.38	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	0.48	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 2-3 Models Based on Datasheet

Several more general component-level energy estimation models were introduced in works [23]-[26]. In a broad sense, a component can be an individual functional unit or a block with several similar functional units. The key idea of this method is to profile the energy insights of each component. Generally speaking, those devices of an embedded system can be divided into four main categories: computation, storage, communication, and I/Os. Each of these categories has its own unique objective functionality and thus cannot be replaced by another one. The energy consumption issues of each category can be independently analyzed and thus a component-level model can be easily extended by adding new components.

Power behavior of components is driven by specific events. Devices requests and occurrences of hardware events such as cache misses, retired instructions, and memory accesses can be considered as influence factors of energy consumption. In some complex systems, each component can be described by a simple state machine containing information relevant to its power behavior. At different execution moments, each component is in a specific power state, which consumes a discrete amount of power. Average or peak power dissipation can be easily determined by looking at the power usage over time for a given set of environmental conditions.

A new methodology is to relate energy consumption to software behaviors. In essence, all the hardware activities are driven by a series of software operations, i.e., a sequence of instructions. Any instruction in its execution stage will activate some modules in the processors and contribute to the energy consumption. In modern microprocessors, a set of special-purpose registers, named as performance monitoring counters (PMCs), is employed to record the number of hardware-related activities occurring during program execution. The original purpose of PMC design is to provide a practical method for developers to supervise and adjust system performance through the information

provided by those counters. Since PMCs provide a deeper insight into functional units of processor, cache, main memory, as well as some peripherals with low-overhead to represent the performance characteristics of applications at their runtime, a new direction for energy estimation has been introduced by PMCs. Figure 2-4 shows the general structure of a PMC-based modeling methodology. This structure is usually divided into software, middleware and hardware levels. Applications and estimation models run at the software level while the operating system runs at middleware level. The latter controls the PMCs configuration and provides an interface for applications to set the configuration. The components under-test components and the PMCs are implemented at the hardware level. The energy estimation is profiled by mathematical fitting based on real measurements of current and voltage, either from the entire platform or for each component, and PMC samples.

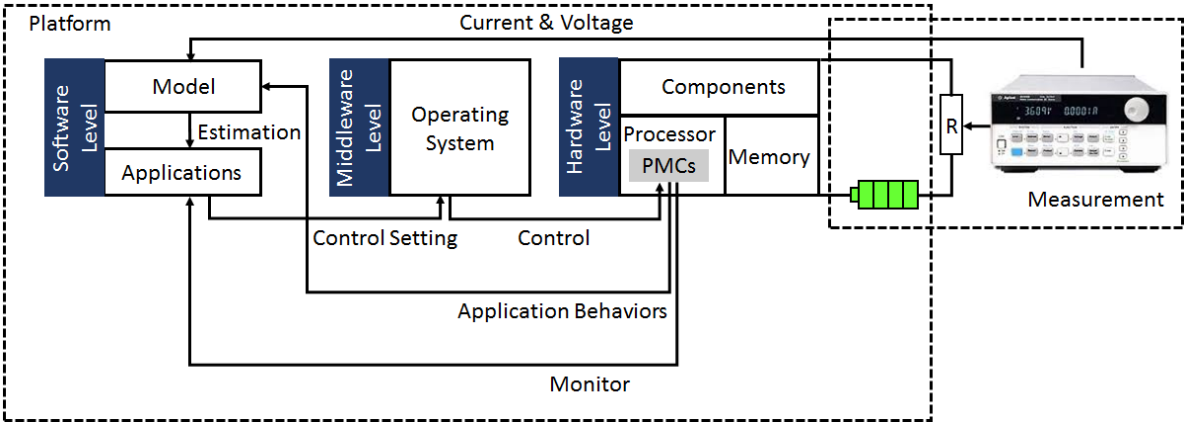


Figure 2-4 A General Structure of PMC-based Modeling Methodology

After Bellosa in work [27] correlated PMC-monitored events (PMC events) with energy consumption to obtain a good estimation, many studies have been continued in this direction [28]-[33]. The work of Li et al [29] exploited the high-correlation between the number of instructions per cycle (IPC) and power consumption to estimate the energy dissipation. The main challenge of this approach is how to choose the best set of PMC events. Most researchers identified PMC events on the basis of the platform architecture analysis [28]-[33]. For example, Goel et al. [31] proposed an approach for choosing PMC events. Their first step was to identify the candidates by manually separating available PMC events into several categories that impacted dynamic power through different issues. Their work could effectively reduce the number of PMC events. However, a priori selection might encounter difficulties due to the limitation of events that could be monitored by PMCs and the limitation of the number of PMCs that could be used simultaneously. More specifically, Lively et al. [32] introduced how to choose suitable PMC events. They used different PMC events to build models for each application to ensure that the energy behavior trends were correctly represented. In contrary, X. Yu et al. [34] built a sub-model of a processor without any selection but repeated the same test case several times with two different PMC events each time to obtain all the information provided by the entire set of PMC events. This method was quite time-consuming, and it was not

suitable for the power management policy since it would entail a long delay to get a full estimation of one application.

PMC-based methods for energy estimation are getting promising results in different fields, and thus the energy consumption modeling combined with PMC events and mathematical fitting are been widely used.

2.1.2. Video-Coding-focused Models

Video streaming includes images, audio and data. Large amounts of data are required to accurately represent video information. Video coding is significantly more complex than other applications due to the high complexity of the video compression algorithms. As an individual application, there have been researches on energy estimation models specifically focus on video coding fields.

X. Lu et al [35] proposed a model for an H.263 encoder based on the bit rate and parameters from functional units such as DCT computation, quantization, and motion estimation. A similar model for an MPEG-4 simple profile encoder, based on the quantization parameters and the INTRA fresh rate, was proposed in work [36]. X. Li et al [37] proposed a model for an H.264/AVC decoder in which the consumption was estimated with the product of multiplying the video spatial resolution, temporal resolution, and quantization. Yahia Benmoussa et al [38] introduced an energy model of H.264/AVC decoder based on a set of hardware and video stream parameters such as bit-rate, clock frequency, and quantization parameters (QP). The entire energy model included four sub-models: QP-rate model, dynamic power model, static power model, and time model. The coefficients of those parameters were obtained by consumption measurements and regression analysis. This methodology was featured to achieve a good tradeoff between prediction property and lower-level model details. In work from Z. Ma et al. [39], they did not directly implement an energy/power model but proposed a model for an H.264/AVC decoder complexity estimation. The infrastructure of this complexity model was the complexity unit (CU) which was the fundamental operations of each decoding module (DM) over a time interval, such as one frame. The complexity of one DM was the product of the average complexity of one CU and the required number of CUs. Among several possible ways of defining the CU for a DM, they determined the final choice by considering if it was fairly constant for a given decoder and if it was able to be accurately predicted by a simple linear function. The entire decoder could be decomposed as several DMs, and then the decoder complexity would be easily obtained by summing up all the complexities of each DM. Since the decoder complexity is a direct representation of the energy consumption, then its exploration could become a tool to predict energy consumption.

2.1.3. Discussion

As introduced above, each method has its advantages and disadvantages. There is no perfect one but with the comprehensive consideration of their features, some of them could be more proper for video coding modeling.

Modeling approach at instruction-level mainly faces three problems:

- The number of measured instructions needs to be quantified. This number has a direct relationship with the size of the instruction set architecture (ISA).
- Modern architectures of processors have been implemented to use pipelines, which allow the execution of several instructions at the same time to improve the processing speed. Thus, the number of parallel instructions needs to be taken into consideration.
- Drawing the whole picture of the full-system power consumption is difficult because that this approach cannot provide insight on the other isolated components, especially peripherals.

The first two problems cause the model to be non-generalizable. They are needed to deeply understand design details of hardware, especially the supported ISA because the base cost of instruction may vary within the number of operands and the accessing methods. The last one leads to the difficulty for a model, at a specific time, to distinguish out what the issued part/component on the system in relation to the currently executed instructions is. Thus the developers cannot know which part/component consumes the greatest percentage of energy.

The mandatory requirement of FLPA methodology is to decompose the whole system into several functional blocks with the consideration of their impacts on the power consumption. The point is to balance between accuracy, estimation cost, and decomposition granularity. The main disadvantages of FLPA are the complexity of the components determination, the coverage of all significantly influencing parameters, and the dependency among power consumption and performed instructions.

Energy estimation models give a possibility to understand and analyze power behavior on real systems. Instruction-level or function-level models are usually drawn by detailed analysis on system architecture, i.e, the instruction set or the functional blocks, which limits the quick adaption of these models for various platforms. More specific models are those designed for video coding. They are abstracted from coding algorithm without hardware information during model building. This liberates video coding designers from their unfamiliar fields. However, it is not possible to detect the distribution of power consumption of different components from those video-coding-focused models. As a consequence, it is difficult to determine the bottlenecks of the design. In addition, the complexity

parameters may need to be redesigned for each codec standard, which is also a limitation on generalization.

A more suitable energy estimation model for video coding should take consideration on both model abstract level and information provided by platform behavior observation. Modeling in a high abstract level means that most of the hardware details are hidden and models can be employed to model the energy consumption on different platforms with little modification. In addition, the modeling parameters can be easily determined. For platform behavior observation, the model should represent how the power is consumed within components. For example, for a processor model, the model could demonstrate the power distribution on ALU unit, memory unit or other accelerator modules such as branch prediction or pipeline. A PMC-based mechanism can better fit the two requirements mentioned above. PMCs are widely available in most of the modern processors and they can be accessed by the same pattern of interfaces provided by high level tools. In addition, PMCs translate hardware details to occurrences of different events. Different components can use different events as their way to represent energy. Therefore, the PMC-based estimation can be a good candidate for energy modeling of video coding.

2.2. Introduction of PMC-Based Methodology

2.2.1. PMC Introduction

PMCs have been briefly introduced in the component-level estimation models. In modern processors, they are provided within a Performance Monitoring Unit (PMU) to gather statistics on the operation of the processor and memory system. Each PMC can count any of the available events in the target processor.

Implementation of PMCs in different processors could differ from the quantity and the types of monitored events. Each processor has its specific events for monitoring. In order to achieve a better generalization, platform specific events should not be included into estimation models as candidates. In a broad sense, PMCs consist of the following three kinds of counters:

- A cycle counter: This counter can be programmed to increase every main system clock cycle. It is only used to count the cycle numbers. Attention must be paid to the cycle counter. In some platforms, it may need to be enabled independently to the event counters.
- Event counters: The concept of event counter and event need to be distinguished. An event is a special occurrence caused by computing operations. An event counter can be configured to select one specific event among all the platform-available events and increase its value once this event occurs. Thus, the behavior of an event counter can be defined by users

according to their individual requirements. Usually, the number of the available events is much more than the number of PMCs. For example, the PMU of cortex-A9 processor provides 6 PMCs and each one can count any of the 58 available events. Also note that the number of PMCs may be greater than the number of PMCs that can be used simultaneously. In this dissertation, PMC is referred as the event counter if there is no particular emphasis.

- Controlling counters: There are some counters used to control other PMCs to complete various operations. The operation of these counters includes: enable, reset, start, stop, overflow flag set, and interrupts enable.

There are many methods to access PMCs. For example, PMCs can be accessed via special file descriptors. In windows 2000 operating system and later ones, users are provided with graphical view of how well the system performs by counting the data consumed by applications. The Linux PMC subsystem also provides an abstraction of the hardware capabilities, such as *perf*_event [40] which is an application programming interface (API) of the Linux kernel and *perfmonX* [41] which is a hardware-based performance monitoring interface for reading the PMCs from user space.

2.2.2. Correlation Coefficient

The basic principle of PMC-based energy estimation model is to use different system events to represent energy consumption. The model accuracy highly depends on the selection of PMC events. Whether or not an event should be used for modeling is determined by its relatedness to energy consumption. Usually, the correlation coefficient (r) is used to express the strength of relatedness between two variables. The value of r is between -1 and +1. r greater than zero indicates a positive correlation, i.e., if the value of one variable increases, the value of the other one will also increases. Similarly, r lower than zero shows that two variables are negatively correlated. The larger the absolute value of r, the stronger they correlate to each other. Please note that the correlation coefficient does not reflect a causal relationship, i.e., one variable is not the incentive of another one. For example, in summer, the beer sales and the ice-cream sales rates both increase, which somehow indicates a kind of “common increase” relationship, and they both reflect the phenomenon of temperature increase. But selling a beer is substantially independent to an ice-cream sale; the real cause of increments of these two rates is the high temperature.

Correlation coefficient was firstly proposed by the statistician Karl Pearson [42]. It is also named as Pearson correlation. A Pearson correlation coefficient can be applied to the following cases:

- Two variables are continuous and linearly related to each other.
- The overall population of two variables is normally distributed, or is closed to a unimodal distribution.

- Observations of two variables are paired; each pair of observations is independent.

Pearson correlation coefficient has strict preconditions for model establishment. If the previous conditions are not satisfied, it is possible to use Spearman's rank correlation as a substitution. Spearman's rank correlation coefficient is a non-parameter rank statistic, which was proposed by Charles Spearman in 1904 [43]. The requirements on sample data to use Spearman's rank correlation coefficient are less strict than those required to use Pearson correlation coefficient. As long as the ranks of observed values are paired. Spearman's rank correlation coefficient can be employed regardless of the overall distribution and sample size. One variable is a strictly monotone function of the other if the Spearman's 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 spearman's rank correlation coefficient is defined as the Pearson correlation coefficient between the ranked variables. In the actual computation, the original variables X_i and Y_i are converted to ranks x_i, y_i, i.e., the positions of original variables after sorting. If there is no repeated ranks, Spearman's rank correlation coefficient uses the monotonic function (equation 2-1) to describe the statistical dependence, where d_i is the difference between the ranks of each observation of the two variables and n, the simple size. If there are repeated values, r is need to be calculated by equation 2-2 as the Pearson correlation coefficient of ranks.

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

$$r_\rho = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 (y_i - \bar{y})^2}} \quad 2\text{-}2$$

2.2.3. Fitting Methods

There are many fitting methods can be used for model building. Linear regression is the most commonly used one. However, in some complex cases, in different intervals, power consumption and events may variously related, thus, in these cases, piecewise fitting methods like Multivariate Adaptive Regression Spline (MARS), are recommended to obtain better accuracy in complex situations.

2.2.3.1. Linear Regression Methods

Linear regression is an important branch of mathematical statistics [44]. Multiple linear regression is the study to research if there is a linear relationship among a number of independent variables (or predictors) and a dependent variable (or a response), and to use a multiple linear regression equation to express this relationship. It can also be used to quantitatively characterize the linear relationship among a dependent variable and several independent variables. Multiple linear

regression modeling is an effective tool for model predicting. For a practical problem, the linear regression model of the obtained *n* sets of data $(x_{i1}, x_{i2}, \ldots, x_{in}; y_i | \ i = 1,2,\ldots,n)$ can be expressed as equation 2-3:

$$\begin{gathered} y_1 = \beta_0 + \beta_1 x_{11} + \beta_2 x_{12} + \cdots + \beta_p x_{1p} + \varepsilon_1 \\ y_2 = \beta_0 + \beta_1 x_{21} + \beta_2 x_{22} + \cdots + \beta_p x_{2p} + \varepsilon_2 \\ \ldots\ldots \\ y_n = \beta_0 + \beta_1 x_{n1} + \beta_2 x_{n2} + \cdots + \beta_p x_{np} + \varepsilon_n \end{gathered} \quad 2\text{-}3$$

and can be written in matrix form as equation 2-4:

$$y = X\beta + \varepsilon \quad 2\text{-}4$$

where,

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, X = \begin{bmatrix} 1 & x_{11} & x_{12} & \ldots & x_{1p} \\ 1 & x_{21} & x_{22} & \ldots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \ldots & x_{np} \end{bmatrix}, \beta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ y_n \end{bmatrix}, \quad 2\text{-}5$$

$$\varepsilon = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

According to the least squares method, the obtained regression coefficients are cast as equation 2-6:

$$\hat{\beta} = (X'X)^{-1}X'y \quad 2\text{-}6$$

then, vector $\hat{y} = X\beta = (\widehat{y_1}, \widehat{y_2}, \ldots, \widehat{y_n})'$ is the return value of the dependent variable vector $y = (y_1, y_2, \ldots, y_n)'$. For the model parameter estimation, regression equation 2-3 has the following basic assumptions [44]:

- Independent variables $x_1, x_2, \ldots, x_p$ are deterministic variables, not random ones, and it is required that $RANK(X) = P + 1 < n$. This is to say, there is no perfect correlation among independent variables and the sample size should be larger than the number of explanatory variables.
- The random error term has homoscedasticity which indicates that error term and independent variables are independent.
- The random error term is normally distributed.

2.2.3.2. MARS

Table 2-1 shows the correlation coefficients between PMC events and energy consumption for one of the experiments described more in detailed in chapter 8. Correlation coefficients lower than 0.7

indicate that no absolute linear relationship exists between the concerned events and the energy. In this case, the trend line to fit PMC events to energy consumption is likely to exhibit various slopes depending on the range of the independent variables. Therefore, a piecewise method (which is proposed for building a model with different slopes in each interval), MARS, is recommended to achieve more accurate results.

Table 2-1 Correlation Coefficients Between PMC Events and Energy

Data Cache Access	Instruction Cache Misses	Data Translation Lookaside Buffer Misses	Instruction Translation Lookaside Buffer Misses	Hardware Interrupts	Conditional Branch Instructions Mispredicted	Instructions Issued
0.51	0.68	0.52	0.38	0.74	0.63	0.79

Multivariate Adaptive Regression Spline (MARS) was proposed by Jerome H. Friedman in 1991 [45] and has been widely used in many complex fitting problems. “Multivariate” shows its ability to generate models in high dimensional problems. “Adaptive” refers to its flexibility and adaptability to adjust the model. “Regression” indicates its functionality to estimate the relationship among independent and dependent variables. And “Spline” is a special function that is piecewise-defined by polynomial functions. Spline interpolation is widely used due to its simplicity of construction, accuracy of evaluation, and capacity to approximate complex curve fitting and interactive curve design. MARS defines the tensor product of spline functions as the basis functions. The process of the generation of basis functions has strong adaptability, which can be completed without manual operations. In a multidimensional situation, due to the expansion of the sample space, how to divide the space has become a critical issue. MARS method does not require a space-disjointed partition as long as all sub-spaces can cover the entire sampling space. Each divided space corresponds to a coefficient and an input variable x_i. MARS model obtains its prediction value by combining all basis functions.

For a system, the output set, which is the dependent variables set $y = (y_1, \dots, y_q)$, and the input set, which is the independent variables set $x = ((x_{11}, \dots, x_{1p}), \dots, (x_{q1}, \dots, x_{qp})$, have the relationship shown in equation 2-7:

$$y_j = f_j(x_{j1}, \dots, x_{jp}) + \varepsilon_j \qquad 2\text{-}7$$

where q is the number of observations and p is the number of independent variables, $\{f_j\}$ is a set of deterministic functions, and $\{\varepsilon_j\}$ is a set of random variables, which reflect the random disturbance of the system. By convention, the expectations of ε_i are set to zero, i.e., $E(\varepsilon_i) = 0$. The objective of MARS is to obtain an approximated function $\hat{f}_j$ instead of f_j to analyze and calculate the system

response through a series of training data $\{y_{i1}, \ldots, y_{iq}; (x_{11}, \ldots, x_{1p}), \ldots, (x_{q1}, \ldots, x_{qp})\}_{i=1}^{q}$. A MARS model can be represented by summing up a set of basis functions as indicated in equation 2-8:

$$\hat{f}(x_1, \ldots, x_p) = \sum c_m B_m(x_{im,\ldots,}x_{jm}) \quad 2\text{-}8$$

where c_m is the coefficient of each basis function $B_m(x_{im,\ldots,}x_{jm})$. The forms of basis functions in MARS method are expressed in equation 2-9:

$$B_m(x_{im,\ldots,}x_{jm}) = \prod_{k=1}^{K_m} b_{km}(x_{v(k,m)}|P_{(k,m)}) \quad 2\text{-}9$$

$B_m(x_{im,\ldots,}x_{jm})$ is obtained by multiplying K_m (at least one) basis functions b_{km}, which are specified by the input variables $x_{im,\ldots,}x_{jm}$, a subset of independent variables denoted as $x_{v(k,m)}$, and a set of functional parameters$P_{(k,ij)}$; b_{km} is a constant or a hinge function expressed in equation 2-10:

$$b_{km}(x|s,t) = [s(x-t)]_+ \qquad (-\infty \le t \le +\infty) \quad 2\text{-}10$$

The subscript "+" of equation 2-10 indicates a positive part, i.e.,

$$[z]_+ = \begin{cases} z & z > 0 \\ 0 & otherwise \end{cases} \quad 2\text{-}11$$

In the equation 2-10, the parameters s and t are the truncated direction ($s = \pm 1$) and the knot position of the basis function, respectively, i.e., the item $P_{(k,m)} = (s_{km}, t_{km})$ of equation 2-9.

The procedure of MARS algorithm is to obtain a set of basis functions $c_m B_m(x_{v(k,m)})$ through a forward iterative process and a backward iterative pass to make the objective function $\hat{f}$ approximate the expected accuracy. Forward pass iteratively divides the training data and fits the estimation models. It will produce a large number of basis functions. The backward pass will selectively remove some basis functions with the premise to ensure the highest goodness of fit of the final model.

During the forward pass, appropriate knot selection at each iteration is crucial for the model accuracy. To maximize estimation accuracy and to save computing time, there is no need to test each point to determine if it is appropriate for a new basis function. A minimum step size *L* for variable selection is introduced, which results in less selection from a large amount of data for knot calculation. The step *L* is calculated as equation 2-12:

$$L(a) = \frac{-log_2[-\frac{1}{pN} ln(1-a)]}{2.5} \quad 2\text{-}12$$

where a locates in a closed interval $[0.01, 0.05]$, which is a reasonable range of narrowing the selection of candidate nodes. The quantity N is the number of observations and p is the number of predictors or input variables. With this selection step, the estimation accuracy is almost unchanged, while the approach speed is significantly improved. The entire iterative process will continue until the number of basis functions reaches the user-defined maximum number or the minimal lack of fit (LoF), which is the different between the real function f and the model function $\hat{f}$, is achieved.

The forward pass usually leads to the over-fitting because MARS algorithm only allows building new basis functions based on previously generated basis functions. Thus, forward iteration will construct a large number of basis functions. The originally generated functions may have little or no contribution to the final model. Their function is to produce subsequent basis functions. To improve the generalization ability, the model will be pruned by MARS backward pass. It deletes the least effective basis function at each loop until it finds the best sub-model. The estimation performance is evaluated with new data rather than with training data. However, new data are always not available at the time of modeling building. Thus, the sub-model performances are evaluated by using the generalized cross validation (GCV) criterion. GVC is computed as in equation 2-13; the lower the value, the better the performance. It takes the trade-off between goodness-of-fit and model complexity.

$$GCV(M) = \frac{RSS}{N*(1-\frac{C(M)}{N})^2} = \frac{\frac{1}{N}\sum_{i=1}^{N}[y_i-\hat{f}_M(x_i)]^2}{[1-\frac{C(M)}{N}]^2} \quad 2\text{-}13$$

where RSS is the residual sum-of-squares on the training data and N is the number of observations; $C(M)$ is the effective number of parameters, which is defined as $trace(B(B^TB)^{-1}B^T) + 1 + d * M$; $trace(B(B^TB)^{-1}B^T) + 1$ is the number of MARS terms, i.e., the number of included basis functions; d is the penalty factor whose value is between 2 and 4, and M is the number of hinge-function knots. RSS always decreases as MARS terms increases. This is to say, the backward pass will always choose the model with largest terms if and only if the RSS is used to evaluate the model performance. Including too many items typically causes a model to have low generalization. Therefore, the GCV criterion takes into account the model generalization to adjust the training RSS and penalizes the addition of knots.

The final model obtained by MARS algorithm is expressed in equation 2-14:

$$\hat{f}(x) = c_0 + \sum_{m=1} c_m \prod_{k=1}^{K_m} [s_{km}(x_{v(k,m)} - t_{km})]_+ \quad 2\text{-}14$$

where c_0 is a constant basis function, each addend term is a basis function $c_m B_m$, and $s_{km} = \pm 1$. When MARS algorithm is used in this dissertation, to simplify the model, K_m is set to 1 and only the basis function with one variable is employed, i.e, $c_m B_m = c_m [s_m (x_i - t_m)]_+$.

As explained before, the hinge function is defined as $b(x|s,t) = [s(x-t)]_+$. To make the model be continuous and have continuous first derivative, the hinge function can be replaced by its corresponding cubic truncated form as equation 2-15 to 2-20:

$$C(x|s=+1, t_-, t, t_+) = \begin{cases} 0 & x \le t_-, \\ P_+(x-t_-)^2 + r_+(x-t_-)^3 & t_- < x < t_+, \\ x - t & x \ge t_+ \end{cases} \quad 2\text{-}15$$

$$C(x|s=-1, t_-, t, t_+) = \begin{cases} -(x-t) & x \le t_-, \\ P_-(x-t_+)^2 + r_-(x-t_+)^3 & t_- < x < t_+, \\ 0 & x \ge t_+ \end{cases} \quad 2\text{-}16$$

$$P_+ = \frac{(2t_+ + t_- - 3t)}{(t_+ - t_-)^2} \quad 2\text{-}17$$

$$r_+ = \frac{(2t - t_+ - t_-)}{(t_+ - t_-)^3} \quad 2\text{-}18$$

$$P_- = \frac{(3t - 2t_- - t_+)}{(t_- - t_+)^2} \quad 2\text{-}19$$

$$r_- = \frac{(t_- + t_+ - 2t)}{(t_- - t_+)^3} \quad 2\text{-}20$$

$C(x|s, t_-, t, t_+)$ is first order differentiable, but its second derivative is not continues at $x = t_\pm$. Each knot *t* can define a linear truncated function, while a cubic function needs three knots: t , t_+ , t_-. Figure 2-5 shows an example of linear and cubic hinge functions.

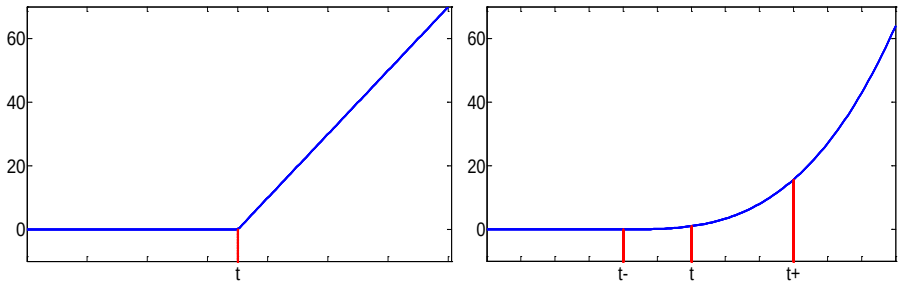


Figure 2-5 (a) Linear and Cubic Basis Function when S=1

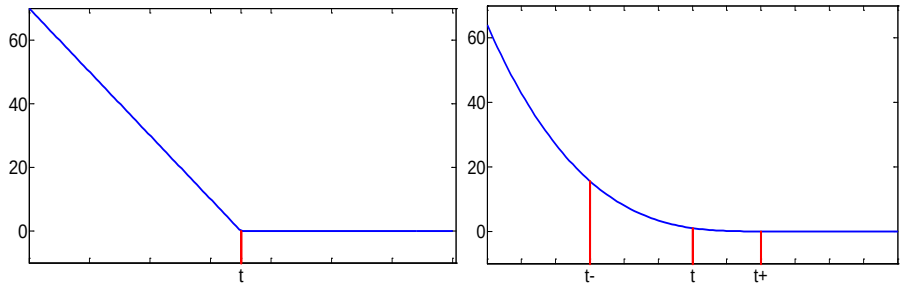


Figure 2-5 (b) Linear and Cubic Basis Function when S=-1

Figure 2-5 Linear and Cubic Basis Function

This post pass ensures that the model is everywhere differentiable in the variable space. Therefore, the model is smoother and the fitting accuracy is improved without introducing heavy computable complexities.

2.2.4. Discussion

There is no doubt that a PMC-based model combined with linear regression or MARS method can achieve very accurate predictions. However, this method has a weakness when a model obtained for one system wants to be applied to other systems with the condition that the appropriate PMC events for each system could be unique. There are reasons of this uniqueness. One reason is that the type of observable events and the number of hardware counters vary from one kind of architecture to another due to the variation in the hardware organization. Therefore, the exact model will be different to each platform depending on the availability of native PMC events. Another reason is that the number of available hardware counters in a processor is limited while each model might have a lot of different PMC events that a developer might like to measure. Each counter can be programmed with the indexes to monitor various types of PMC events. In other words, although CPUs typically have multiple counters, each of them can only monitor one PMC event at one time, and some counters can only monitor specific PMC events, such as the PMC used to count clock cycles. Therefore, architectures cannot concurrently monitor in general the interesting combinations of PMC events. A challenge on the development of PMC-driven component-level models is how to select, using efficiency and accuracy trade-offs, the most suitable PMC events for each component.

Besides selection on PMC events, in practical problems, regression analysts usually tend to thoughtfully select the relevant indicators to avoid missing important system characteristics. However, these indicators are often highly correlated, which is the multi-collinearity phenomenon of multi-variable system [46]. Multi-collinearity occurs when two or more predictors in the model are correlated and provide redundant information about the system response. Thus, it becomes difficult or impossible to distinguish individual effects on the dependent variables. Essentially, collinearity makes

the same variable enter into a model twice, which results in an extreme case of confounding. In general, it is not necessary to have “perfect” collinearity to cause problems; as long as two variables are highly correlated, they will cause a “near-collinearity” problem. In multiple linear regression analysis, the multi-collinearity often seriously affects the parameter estimation, enlarges the standard modeling error, introduces the model distortion and decreases the model robustness. Because of its serious harm and widespread presence, it is needed to eliminate its adverse effects.

2.3. Conclusion

In this chapter, general and video-coding-focused energy estimation modeling methodologies are introduced. Comparing their different features, PMC-based estimation leads a promising mean to balance modeling generalization and accuracy. Thus, it is recommended for energy estimation on video coding. In order to use this mechanism, knowledge of PMC is introduced. The key point to obtain an accuracy model is how to define a proper set of PMC events for energy prediction. Correlation coefficient is a common method to evaluate the degree of relationship of two variables. Therefore, it can be employed to choose the energy-correlated PMC events. The correlation coefficients that can be used depend on the characteristics of sampling data. Pearson correlation coefficients and Spearman’s rank correlation coefficients are two of the most widely used approaches. With the selected PMC event set, fitting methods are employed next to obtain the final energy functions. Linear regression is a common method for data fitting and has shown its accuracy on PMC-based models. With the consideration of nonlinear factors, it could be more optimistically to use a piecewise method, MARS, to obtain higher accuracy. Furthermore, with a discussion about PMC-based methods, two problems have emerged: difficulties on PMC events selection and harms from multi-collinearity phenomenon. Next chapter proposes a more general methodology to solve these two problems.

3. Generalization and Accuracy Improvements of the Energy Estimation Model

To achieve the support on multiple hardware platforms and various coding standards, the estimation model should meet two requirements: one is to involve not too many details of the hardware platforms, and the other one is to avoid including knowledge from specific coding standards. That is to say, the model should have a high generalization and can be employed to any platform and coding standard with little modification. In this context, a PMC mechanism, which has been widely used due to its simplicity and high efficiency, is suggested combining into the optimization module. Although a PMC-based model is not a new idea, model generalization and multi-collinearity of model predictor (independent variables) are still two issues with impact on the accuracy of models and are worthy of improvement. In this chapter, section 3.1 will first state the two problems and then the solutions are proposed in section 3.2. The conclusion will be drawn in the final section 3.3.

3.1. Problem Statement

3.1.1. Generalization Problem

Before going into any further step, it is worth noting that this thesis will focus on modeling the energy consumption of computing processor and memory units. Other peripheral activities are estimated by the number of interrupts and each interrupt is assumed to consume the same energy. For these two focused components, models usually consider the number of executed instructions, or simply, the number of instructions, as the most intuitive renderer of energy consumption. However, in modern processors, additional units responsible for branch predictions and cache memory and techniques such as pipelining are implemented to accelerate the processing speed. These elements are factors that greatly impact on the energy consumption.

Figure 3-1 is a general block diagram of the architecture of a processor including the memory. The processor is divided into 4 parts corresponding to the 4 stages of an instruction process. The instruction fetch unit fetches instructions from the L1 instruction cache memory based on a prediction over the instruction streams. Then, it places the fetched instructions into the input buffer of the pipelined instruction decoder. After the instruction decode unit decodes and sequences instructions, the execution unit starts to execute the decoded instructions. This unit may consist of several identical pipelined Arithmetic Logical Units (ALUs), pipelined multipliers and an address generator for loading and storing instructions. It also performs register write-back operations, processes branch estimations, and other changes on the instruction stream such as instruction condition code evaluations. The

load/store unit includes the entire L1 data cache 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. With this complex structure of processor, only the number of executed instructions cannot completely represent the processor functionality. Note that several processor units will introduce additional energy consumption. For example, pipeline blocking, cache miss, and prediction failure will cause the processor to stall and decrease the number of issued instructions, but at the same time, other units will be active to prepare actions such as pipeline discard (due to wrong branch predictions) or memory access (due to cache misses). The activity of these additional units impacts on the total energy consumption and thus brings difficulties for estimating energy using the number of executed instruction, i.e. only by relating energy consumption to arithmetic unit activities. The unsatisfactory results from the only-instruction-based model will be shown in the results chapter.

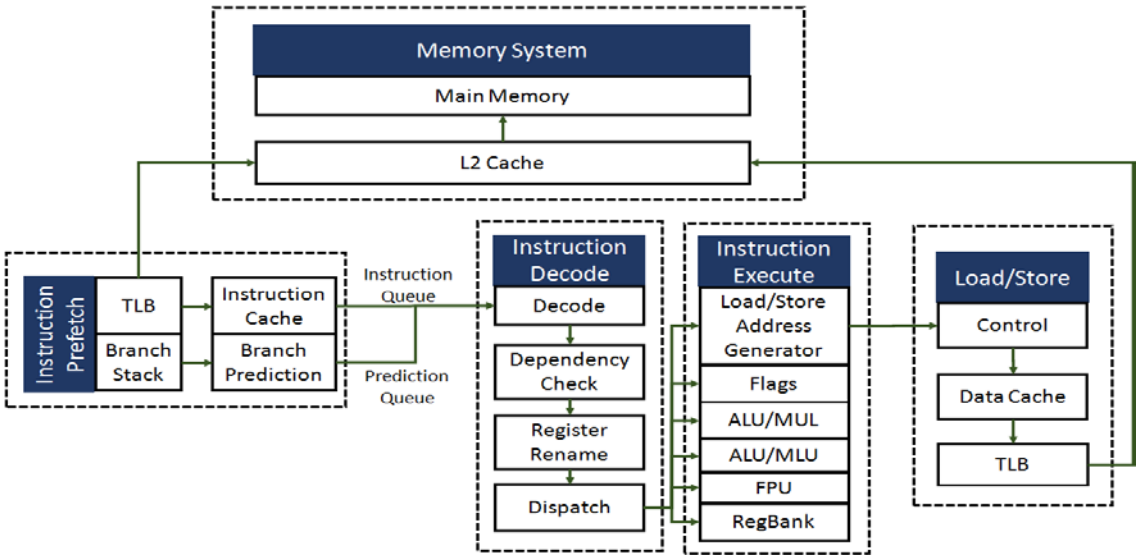


Figure 3-1 General Processor and Memory Architecture

To improve the estimation accuracy of PMC-based models, the principle is to increase the information about energy-related activities obtained from predictors. Briefly speaking, the idea is to avoid biases due to predictor selection. One simple solution is to include all the platform available PMC events. However, with a large number of events, there will be an increased overhead for PMC configuration and control, especially caused by multiplexing of PMCs. An experimental result of this overhead is shown in Figure 3-2 (a) to (d). In this experiment, an embedded platform is configured to decode four different frames, (a) to (d), of a video stream with several PMCs monitoring up to 15 events during the decoding process. Each plot on the left in Figure 3-2 (a) to (d) shows the time to decode one frame of the cideo stream (y-axis) when the platform is configured to monitor different number of PMCs (x-axis). In addition, the CPU time plot only considers the processor time to decode while the total time plot includes, on top of that, any other activities of the CPU and its idle time.

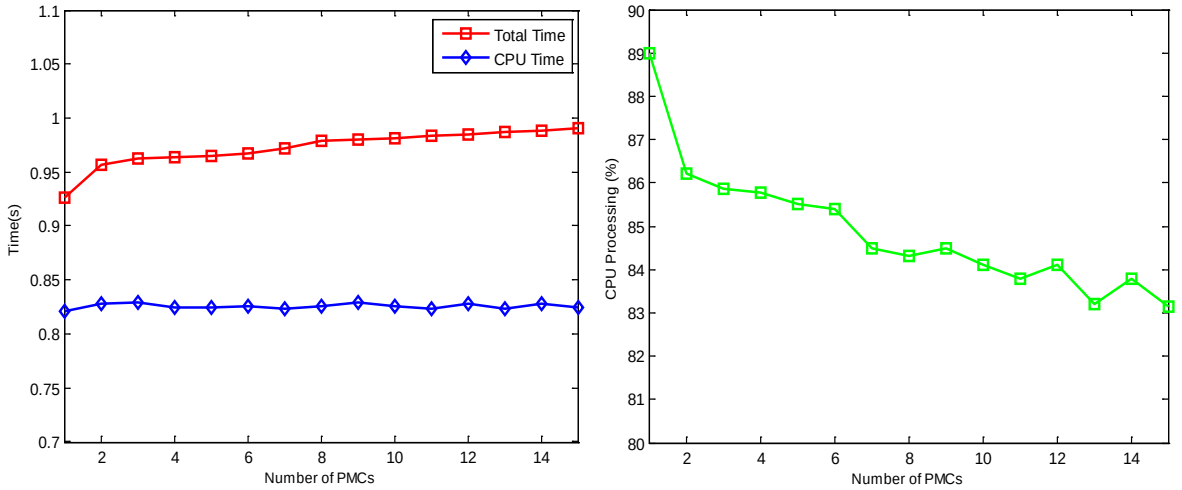


Figure 3-2-(a)

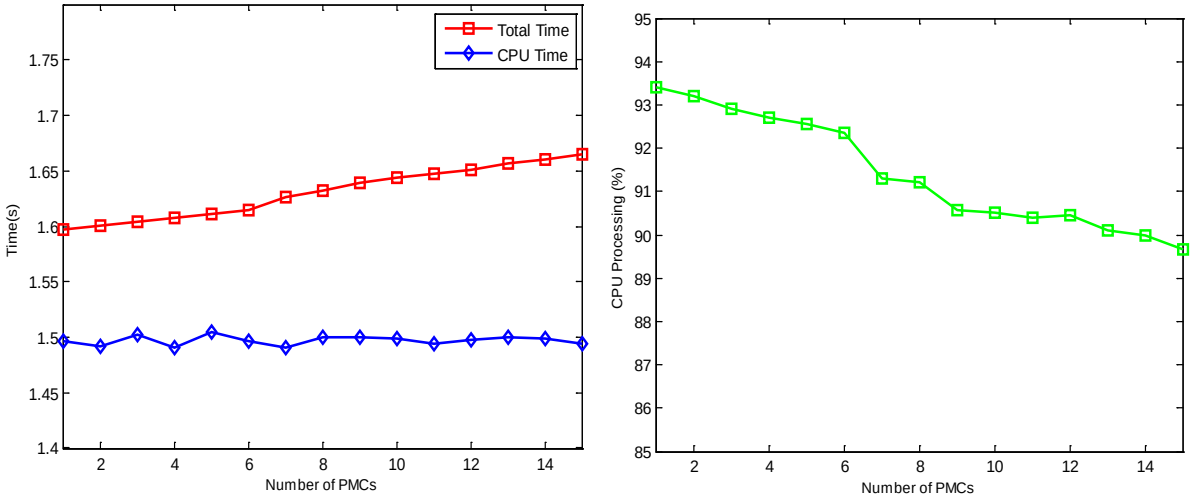


Figure 3-2-(b)

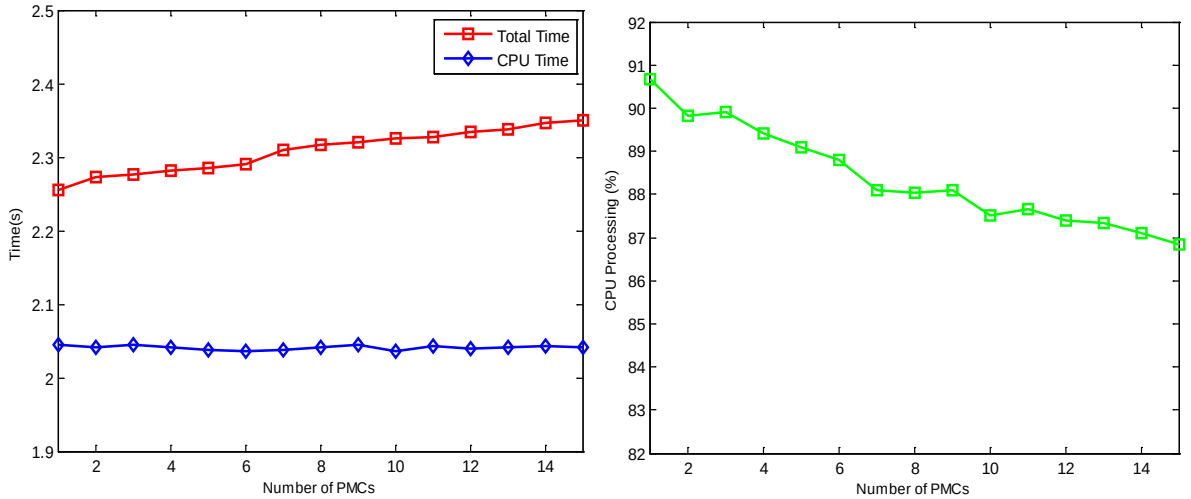


Figure 3-2-(c)

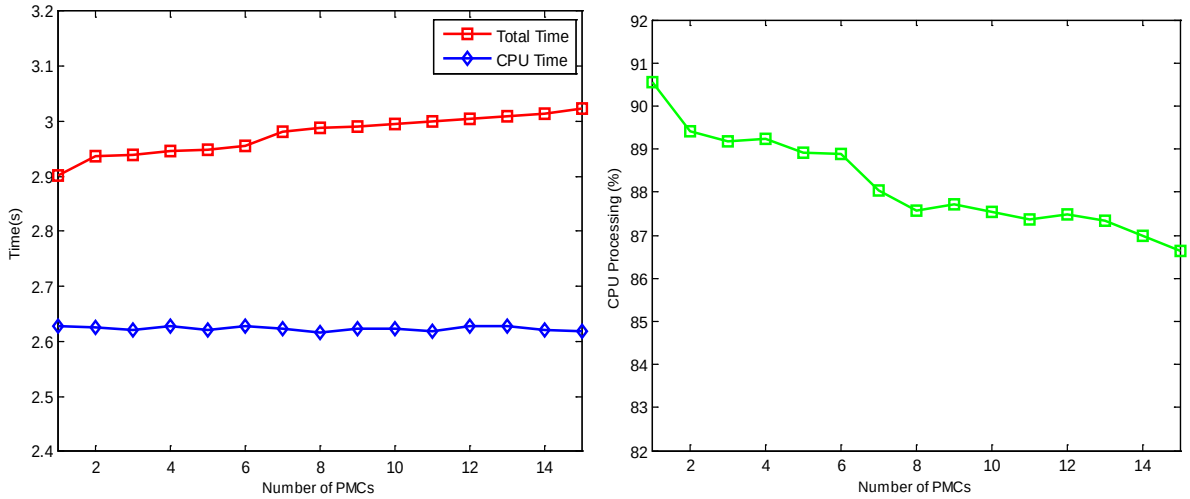


Figure 3-2-(d)

Figure 3-2 PMC Overhead

As can be noticed, the CPU time to decode a frame is almost constant while the total time increases with the number of PMCs. Plots on the right in Figure 3-2 (a) to (d) show the percentage of CPU time employed in PMC management. As can be seen, the decoding rate decreases by 4% when using 15 PMCs instead of using just one. This result suggests the need of limiting the number of PMC events taken into account. Since the size of the PMC event set needs to be reduced, the selection of PMC events is a problem that is worth to study.

The core concept of a PMC-driven model is to relate the energy behavior to the occurrence of several events, which are closely dependent on the architecture features and the platform monitoring capabilities. Model accuracy is strictly dependent on an elaborated selection of PMC-events, which may differ from platform to platform due to the uniqueness of each platform. Thus, a proper set of PMC events may not achieve the same accuracy when applied to a new platform. Formally, this problem is considered as a generalization problem.

Figure 3-3 shows an accuracy comparison on three models for two embedded platforms: P board (PB) and B board (BB).

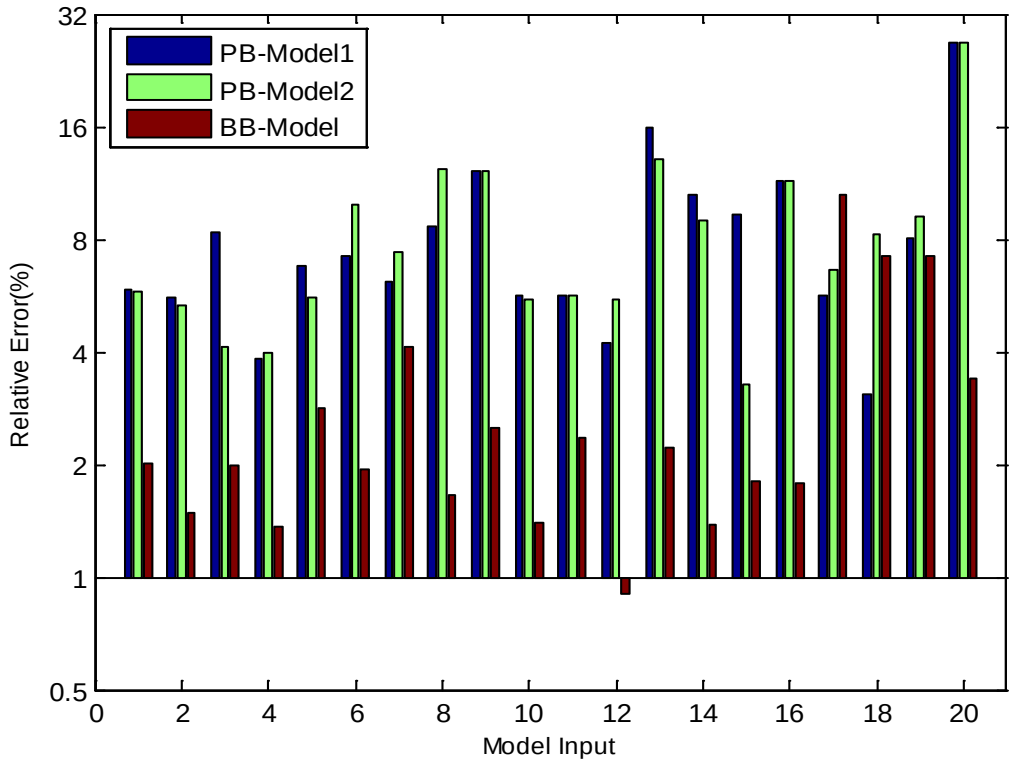


Figure 3-3 A Comparison on Model Performances in Two Platforms

In this example, 30 simple programs, such as a video decoder and encoder and an audio decoder and encoder, are used as benchmarks. 10 of them are employed as training data, namely B_T and other 20 ones are used to evaluate model accuracies, namely B_V. The embedded system BB is configured to monitor all the board-available events. During the executions of programs from the B_T group, both the measured energy consumptions and events samples are obtained. Four PMC events whose correlation coefficients of energy consumption are larger than 0.8 are selected as the model predictors, i.e., Instruction translation look-aside buffer misses (TLB_IM), conditional branch instructions taken (BR_TKN), store instructions (SR_INS), and total cycles (TOT_CYC). With these four PMC events, a model to estimate the energy consumption of the BB embedded system is built. When building the estimation models for the PB platform, the B_T group is also employed as the training data and the procedure of selecting PMC events is not repeated. This set of predictors on BB is used as a referrence to build the energy estimation models for PB. The difference is that in the PB platform, the BR_TKN event is not available. This event has been replaced with the number of branch instructions (BRN_INS). Then, with this change, the PB-model 1 is built. Also, a second model, the PB-Model2, is built with only the TLB_IM, SR_INS, and TOT_CYC events. The performances of these three models are evaluated with the programs from the B_V group. The relative errors between the estimated and the measured energy consumptions are calculated. As can be seen in Figure 3-3, no PB models achieves good performance compared with that of the BB model. In the latter case all the estimation

errors are maintained below 10%. Note the log scale in the Y axis for relative errors presentation has because of the large differences.

This comparison shows that an elaborated set of PMC events on one platform may not be practical on another one. Thus, model generalization needs to be solved. Modern processors have been implemented with different hardware architectures, instruction sets, pipeline depths, specific acceleration circuits and different instruction cycles. These variances indicate different data flows during the processing which will cause different contributions to the whole energy consumption. Therefore, more typical events need to be distinguished for each processor to achieve higher estimation accuracy.

3.1.2. Multi-collinearity Problem

In addition to the overhead introduced when dealing with a large number of PMC events, their correlations also need to be considered. In fact, there is no absolute definition of the correlation degree, which is a concept somehow based on the experimental experience. Table 3-1 lists a correlation degree scale as a function of the value of correlation coefficients [43].

Table 3-1 Correlation Coefficient vs Correlation Degree

Value of Correlation Coefficients	Correlations Degree
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 3-2 An Example of Internal Correlation of PMC Events

	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	E11
E1	1.00	0.1307	0.6990	0.5694	0.7044	0.5326	0.7260	0.4800	0.5079	0.4732	0.7718
E2		1.00	-0.2145	0.2953	-0.0894	0.1237	0.2082	-0.3449	-0.5140	-0.4640	-0.1763
E3			1.00	0.6043	0.8902	0.5442	0.5815	0.5226	0.7020	0.6323	0.7131
E4				1.00	0.7042	0.6788	0.6788	0.3285	0.3431	0.2667	0.3037
E5					1.00	0.5453	0.8221	0.3182	0.5319	0.7161	0.7183
E6						1.00	0.5110	0.4928	0.4734	0.2937	0.3810
E7							1.00	0.1657	0.2802	0.5749	0.6828
E8								1.00	0.9053	0.4093	0.5058
E9									1.00	0.6847	0.6212
E10										1.00	0.7513
E11											1.00

Table 3-2 shows an example of the internal correlation among 11 PMC events. The values in Table 3-2 are obtained from the selection process of PMC events for the BB platform mentioned above. As can be seen, several events are strongly correlated to each other, e.g., $r(\mathrm{E3}, \mathrm{E5})$, $r(\mathrm{E1}, \mathrm{E7})$, $r(\mathrm{E5}, \mathrm{E7})$, $r(\mathrm{E1}, \mathrm{E11})$, $r(\mathrm{E3}, \mathrm{E11})$, and $r(\mathrm{E5}, \mathrm{E11})$ are over 0.7, which represents a strong relationship. This indicates a possible multi-collinearity phenomenon in PMC events. Multi-collinearity has two main harms on estimation accuracy [44][46][47]:

- Multi-collinearity can lead to unstable solutions. When the training values of independent variables have slight changes, the coefficients of the built models may drastically change in magnitude and sign. This brings potential risks in applying the models in practical cases.
- Multi-collinearity can cause regression coefficients not to appear significant. As a consequence, important variables may be dropped.

A potential risk of multi-collinearity is the accuracy and the stability degradation with regard to the regression coefficient technique, which means that the presence of multi-collinearity produces bad estimates on model parameters but this fact does not imply that the fitted model always produces unsatisfactory predictions. How well a model performs depends on the purpose of the model. Usually, a model could be employed:

- To illustrate the relationship between the predictors (independent variables) and the response (dependent variable);
- To predict the response of future observations.

If the primary purpose of a model is for prediction, the relationship between predictors and response is not strictly required as long as the model is able to accurately represent the outcome trends. Thus, multi-collinearity is less harmful for a predictive model. It will be a problem if the modeling purpose is interested in both the prediction and how the individual predictor influences the response. Because it is inherently difficult to tease apart the individual impacts if two or more predictors are correlated. A regression model uses the information from variation between predictors and their corresponding variation in the response to make the estimations. Each predictor may contribute with less information for estimating its individual impact if multi-collinearity exists and therefore the effective amount of information to assess the predictor's effect is reduced.

To develop an energy-aware management and optimization, it is mandatory first, to understand how energy is impacted by different events and next, to make energy-efficient decisions. This is to say that the energy estimation model should clearly represent the relationship between predictors and response, i.e., the system activities and the energy consumption, to avoid misleading the internalization of the energy cost of different operations. In particular, the model should locate the

energy hot spots which are quite useful to determine how to optimize the limited energy budget. Therefore, there is a need to suppress the redundant PMC events to release the multi-collinear influence in order to build models which distinguish the impacts of each PMC event on the energy consumption. As a consequence, the information provided by the estimation model can be used by the energy-aware manager to wisely guide the application execution for energy saving.

3.2. Problem Solutions

As stated in the previous section, how to select the PMC events to be used in the energy estimation model among the different candidates is an important research point. If an energy estimation model includes all the available PMC events in the platform, most of the details of the application will be covered. However, a large number of PMC events, on the one hand, increases the complexity of on-line modeling and sampling time, on the other, introduces serious multi-collinearity problems. Thus, it is necessary to have a methodology to suppress the multi-collinearity problems by reducing the number of PMC events without losing the captured application behavior features. Identifying the PMC events which are strong related to energy consumption is the primary requirement of the energy estimation modeling process. In this context, a PMC-filter is proposed in this dissertation. It includes two parts, one is to identify the most appropriate PMC events and another one is to suppress the multi-collinearity problem.

3.2.1. PMC Event Selection

PMC event selection is a step to eliminate those events with a weaker contribution to energy consumption. In this step, the Spearman's rank correlation, r_{S_i}, has been employed because the overall distribution of the sample data is unknown. A threshold, α, of r_{S_i} is set to identify the PMC events with large energy correlation and to eliminate, from the initial set of PMC events, those whose coefficients are below α. This step can be simply described as the following pseudo code in Figure 3-4.

```
Set threshold α for energy correlation detecting;
for ∀ event ∈ {Event_original}
        if r_Si(event_i, energy) > α
           event_i → {Eevent_e1}
        end
end
```

Figure 3-4 Pseudo Code for Eliminating Weakly Energy-related PMC Events

In this step, how to judge the variable dependence with the correlation coefficient is the main concern. Figure 3-5 shows how two PMC events, issued instructions (TOT-IIS) and L1 data cache accesses (L1-DCA), relate to energy consumption.

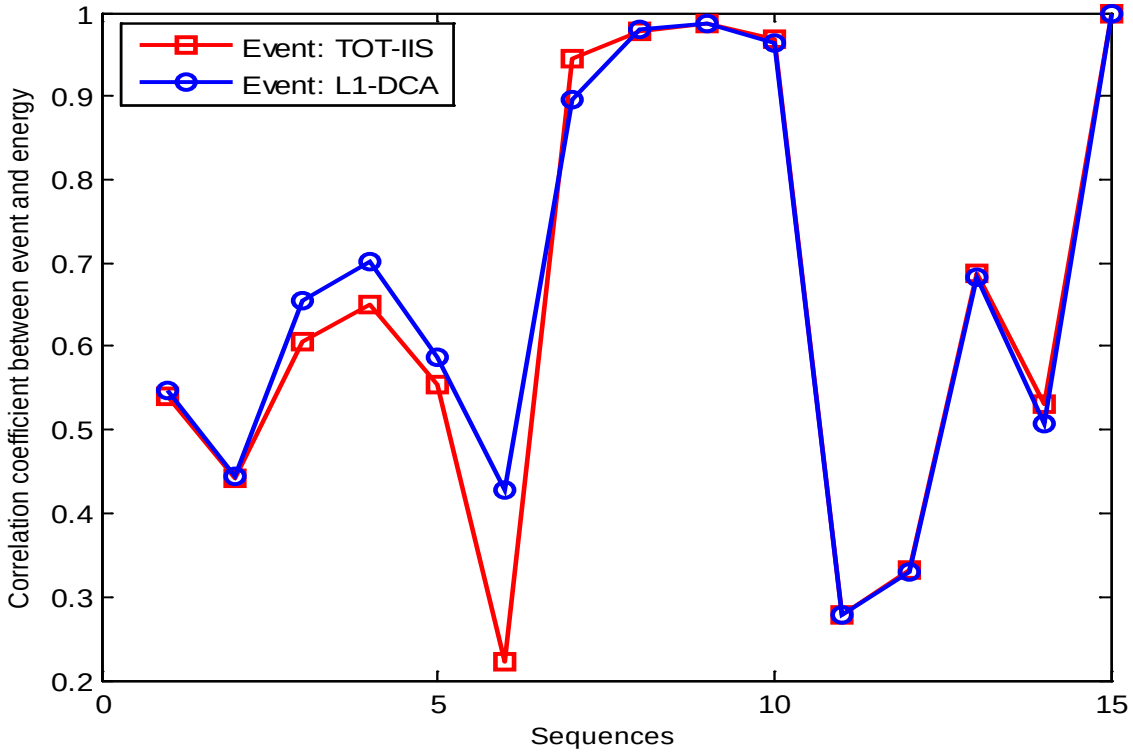


Figure 3-5 Relationship between PMC Event and Energy Consumption

In this example, 15 different video sequences are decoded on the PB platform, which is the same commercial embedded board used for the experiments discussed before in section 3.1.1. During the decoding processes, two events, TOT-IIS and L1-DCA, are monitored by the PMCs and the real energy consumptions are measured. Generally speaking, each event contributes to the energy consumption, thus an accurate model should capture all their impacts. However, as can be seen in Figure 3-5, the relationship varies from one video sequence to another. In some cases, these two events both are highly correlated to the energy while in some others, the correlations decrease to a middle level, and in some extremely cases, they have weak relationships. Therefore, a quite high event selection threshold may likely drop from the models those events that are highly related to the energy consumption. Similarly, a too low threshold will add to the model many events and will introduce a high overhead due to PMC sampling. Thus, the event selection threshold is recommended to be set at the middle level, i.e., the threshold could be set in a range from 0.4 to 0.6. This is to say that the events which have moderate relationship with energy should be maintained.

3.2.2. Multi-collinearity Suppression

As far as the system integrity and continuity concern, there are dependencies among PMC events, which are known as multi-collinearity. Figure 3-5 shows a potential collinearity between TOT-IIS and L1-DCA because the relationships between their energy consumptions has a quite similar pattern. Several PMC events can be highly correlated, i.e. the information provided by one of them can be predicated or explained by the others. In the previous section, it has been explained that the correlation coefficient shows the relationship between two variables. In other words, if two variables are perfectly correlated, they provide the same information to build an estimation model. Multi-collinearity makes more difficult to distinguish the PMC events that are relevant for the energy consumption because of the redundant information among variables.

A method to better interpret the correlation of two variables is to calculate their coefficient of determination (R^2), which is the square of their correlation coefficient. R^2 reflects the percentage of the variance of one variable that 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, $r_{YX} = 0.7$, then the coefficient of determination, R^2_{YX}, equals to 0.49. This means that 49% of the variance of variable Y can be explained by variable X [43]. The stronger the correlation of two variables, the more variance of one variable can be explained by another and the more information of one variable can be represented by another one. However, $R^2_{YX} = 0.49$ also means that 51% of the information cannot be replaced. This is because even these two variables have a strong correlation ($r_{YX} = 0.7$), there are still reasons that cause the differences between them. The idea of the shared variance can be intuitively shown in Figure 3-6. The gray area stands for the shared variances of two variables. The larger it is, the stronger they are related.

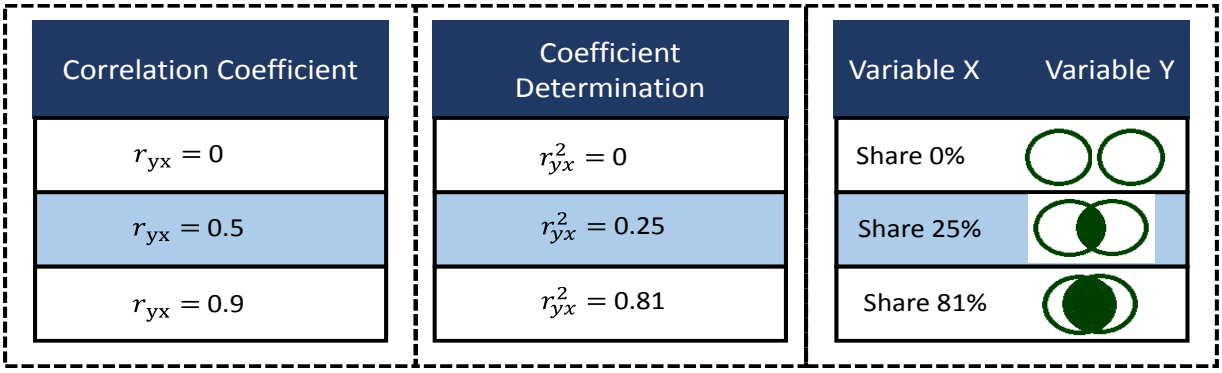


Figure 3-6 Shared Variance

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 25% information. While the third situation shows that one circle almost perfectly covers the other one due to the quite high

correlation between them. During the model fitting, the correlation information among PMC events is targeted to reflect more energy variation features within fewer PMC events. A simple example below will give a better explain on this goal:

Considering an accurate energy estimation model with two independent variables, Y and X, namely $E = aY + bX + \varepsilon$. Assuming that 75% of the features in Y can be explained by variable X, thus the variable Y can be estimated by variable X expressed in equation 3-1:

$$Y = cX + \varepsilon_{x \to y} \tag{3-1}$$

Where ε is the estimation error due to the 25% of unexplained features of Y. Thus the energy model can be rewritten as equation 3-2:

$$E = a(cX + \varepsilon_{x \to y}) + bX + \varepsilon = (ac + b)X + (a\varepsilon_{x \to y} + \varepsilon) \tag{3-2}$$

If the relationship between X and Y is stronger, the value of item $a\varepsilon_{x \to y}$ will be smaller, which means the error caused by using X to represent Y is smaller, and thus the whole error $a\varepsilon_{x \to y} + \varepsilon$ will be smaller. A high correlated case also indicates a reduction of the effective sample size. Since X and Y share 75% of their variance, thus only 25% of the information (i.e., a fourth) provided by the samples can be used to model the impact of each variable. As a consequence, 75% of the redundant information could be dropped and the effective sample size could be reduced.

There is no absolute threshold to distinguish if multi-collinearity causes harmful influences. The variance inflation factor (VIF), a widely used multi-collinear indicator [47], is going to be employed in this thesis. It provides a reference to evaluate how much variance of the coefficient estimation is being inflated by multi-collinearity. The VIF can be expressed as $VIF_i = \frac{1}{1-R_i^2}$. Assuming that there are P independent variables, R_i^2 is the coefficient of determination of the regressing variable x_i on the other P -1 independent variables. VIF is able to identify and separate the influences of distinct factors on the variance of the coefficient estimation. A large value of VIF_i indicates a serious multi-collinearity among independent variables. In this thesis, the following steps have been designed to iteratively refine the PMC set, namely, P_{e1}, using the VIF values as a reference:

- Calculate the VIF values of all PMC events in P_{e1}.
- Set a certain threshold, β, to be used to find out those PMC events whose $VIFs$ exceed this threshold. And, from them, find out one event, PE_a, which has the largest VIF. If there are several PMC events with the same largest VIF, the one with the smallest energy correlation

coefficient will be chosen.

- Determine one PMC event, PE_b, who has the closest correlation with PE_a.
- Eliminate one PMC event between PE_b and PE_a from P_{e1}, the one which has smaller energy correlation coefficient.
- Recalculate the VIF values of remaining PMC events and repeat the process from step 2 until all the VIF values are below threshold β. The resulting PMC events, as a new set of P_{e2}, are the most important ones as far as the energy correlation concerns.

This procedure can be outlined in the following pseudocode:

```
Set threshold β for VIF;
do{
        for ∀ event_i ∈ E_e1
                get VIF(event_i);
                if (VIF(event_i) > β)
                        event_i → E_tmp;
                end
        end
Find event_i ∈ E_tmp have largest VIF
        event_i → E_L;
Find event_a ∈ E_L has smallest r_Sa(event_a, energy)
∀ event_i ∈ {E_e1 − {event_a}}, ∃ event_b has largest r_Sa(event_a, event_b)
event_del = r_Sa(event_a, energy)>r_Sb(event_b, energy)? event_a: event_b;
E_e1=E_e1-{event_del};
} while(∃ evnet ∈ E_e1: VIF(event_i) > β)
End
```

Figure 3-7 Pseudo Code of Multi-collinearity Suppression

3.2.3. Design Flow of an Energy Estimation Model

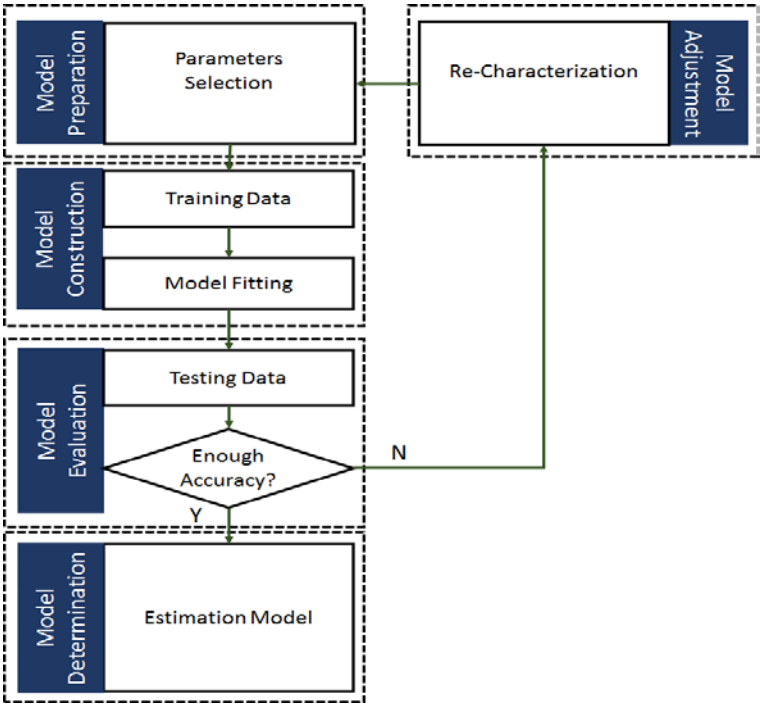


Figure 3-8 Design Flow of an Energy Estimation Model

A full energy estimation model is based on the power measurement. It provides correlations between system activities and energy consumption. Generally, to build a high-level energy estimator, two main steps are needed: parameters determination and model construction. Figure 3-8 shows the design flow of an energy estimation model. Note that the device whose energy wants to be estimated is called device under test (DUT). To estimate the energy, the parameters of the model, which are the samples of a set of PMC events, are obtained during the executions of different benchmarks. The selections of model parameters have been explained in detail in the previous two subsections. Then, in order to fit the energy model, the current of the whole system or the current of specific components together with the supplied voltage have to be measured and considered as model response. The energy estimation model is constructed by a proper fitting method. As it has been mentioned in section 2.2.3, the linear regression and the MARS methods are two candidates to process the fitting procedure. Once the coefficient of each independent variable is set, the model is able to estimate the energy from model inputs. In addition, the real measurements are also used to assess the model accuracy. 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, it can be used within the energy manager to provide information needed by the energy-optimizing strategies.

3.3. Conclusion

The accuracy of a PMC-driven model is highly related to the selected PMC events which reflect the application behavior features. Reduced modeling bias and accurate approximations are achieved when a greater number of counters are involved. However, a large number of PMC events also increases the model complexity and sampling time. An accurate PMC-driven model needs an elaborated selection of PMC events service as the explanatory variables (independent variables), thus this method faces a generalization problem. In addition, several PMC events can be highly correlated, i.e. the information provided by one of them can be predicted or explained by the others. This correlation, also known as multi-collinearity, makes more difficult the selection of the PMC events that are actually involved in the consumption of energy. A potential risk of multi-collinearity is the accuracy and the stability decrease with regard to the regression coefficients. In this chapter, a PMC-filter is proposed to solve these two problems. It includes two steps; the first step automatically identifies the most appropriate PMCs with no requirement on any specific detailed knowledge of the employed platform and the second step uses variance inflation factor (VIF) to suppress the redundant PMC events to release the multi-collinear influence.

PART C

Chapter 4: Energy Optimization and Reconfiguration Techniques

Chapter 5: Energy Optimization based on Functional-oriented Reconfiguration

4. Energy Optimization and Reconfiguration Techniques

As discussed in chapter 1, multimedia applications have become widespread with the advent of wireless communications. Its development presents two big challenges. One is the energy issue. Unlike the traditional multimedia applications, electronic mobile devices in the wireless environments are usually powered by battery and exhibit their energy-intensive features. Their functions are restricted by the limited amount of energy. Energy issue has been a critical point which strongly affects computation capability and video compression quality of mobile devices. The other challenge is the implementation complexity. The continuous evolution of more complex and advanced video coding standards has greatly impeded their efficient specification and implementation, and delayed the time to market. Reconfigurable design has been proposed to address these design expectations: deployment time reduction of new standards, implementation flexibility, dataflow programming, and platform generalization. In this chapter, existing research work on energy optimization and reconfigurable design will be presented.

4.1. Energy Optimization Techniques

Despite the continuous advances in chip technology, CPU clock rates have reached to an upper limit and the new architecture designs change to increasing the computational performance by increasing the number of cores. Nevertheless, CPU clocks are high and the power consumption problem is still serious in multicore based systems. Low-power research has attracted significant attentions on embedded multimedia application in electronic mobile devices, especially on wireless video streaming and playback applications due to their high energy-consuming characteristics.

4.1.1. Power Impact Issues

The power consumption on digital ICs is generally divided into two categories: static and dynamic power. The main source of static power is the leakage power which is not related with the IC switching activity. Static power is the power required to maintain the circuit in the same logic state. The total static power can be directly obtained by measuring the voltage dropping across a small calibrated resister located in the supply path. Dynamic power is the internal power generated by transitions of chip signals during system operations. The power consumption of a single circuit cell [48] is expressed in equation 4-1:

$$P_{all} = P_{static} + P_{dynamic} = V \cdot I + \alpha \cdot C_L \cdot V^2 \cdot f_{clk} \quad 4\text{-}1$$

Where, V stands for the supply voltage, I is the static current, α is the factor of node switching activity which is defined between 0 and 1, C_L is the load capacitance, and f_{clk} is the clock frequency.

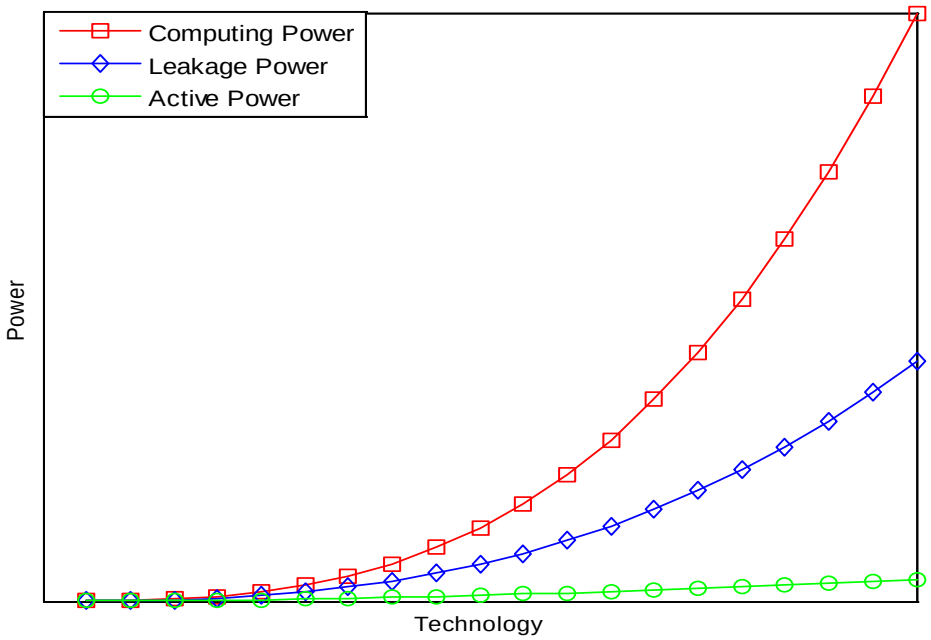


Figure 4-1 Power Consumption Trend [49]

Figure 4-1 shows the plot of three power consumption trends with regard to the progress of technology density, i.e., computing, leakage, and active power. These three power consumptions increase at a rate of k^3, $k^{2.7}$, and $k^{1.9}$, respectively, where k represents the scaling factor. In this case, it is set to 1.4. The computing power is related to the computing density which is defined as the maximum possible number of computations per area and time unit. Note that this trend is based on the assumption of a continuous increasing of clock frequency and voltage supply. With this in mind, many low-power design methods have been proposed to suppress this increase. Despite the fact that the growth of clock frequency has been limited lately, the excessive power consumption is still a bottleneck in IT industry.

Static power is the unavoidable power consumed by semiconductors. It is independent of the workloads, but is determined by the threshold voltage and transistor size. General approaches have difficulties to reduce the static power. On the other hand, dynamic power is proportional to the charge and discharge frequency of stray capacitances of logic units, which is usually optimized by each specific design. From equation 4-1, power impact issues can be summarized into four areas: operating voltage, load capacitance, switching activity, and operating frequency. Reduction of these four factors is the general objective of energy consumption reduction techniques. Note that there is a certain limit on voltage reduction because voltage will inevitably impact the speed and stability of any circuit. For example, when the voltage is reduced, the circuit delay increases and, as a consequence, the system

clock frequency must be reduced. Low clock frequency causes a system to run too slowly and therefore, to miss the predetermined deadlines. As a consequence, the performance of the system will be unacceptable compromised. Voltage reduction is always combined with other methods to optimize energy efficiency.

4.1.2. General Low-Power/Energy Optimization Techniques

Low-Power/Energy optimization techniques can be classified into different levels in the basis of their action objects. From low to high, the optimization levels are: technology level, gate level, circuit level, register-transfer level (RTL), architecture level, and system level. In this thesis, the system level is defined as high-level optimization while the others are classified as low-level approaches.

4.1.2.1. Low-level Optimization Techniques

Low-level approaches are investigated to directly solve the problem from the hardware point of view. In the following, typical techniques of each level are introduced.

Basic technology-level low-power designs techniques are the reductions of supply voltage and transistor size. In addition, the technological progress makes multilevel circuit layout possible. A scaled-down technology achieves the power reduction by increasing system integration degree to reduce circuit delay, inter-chip communication, and device capacitance [50] [51]. With the rapid development of technology, the ratio of chip area to package area is closed to 1:1 which dramatically shortens circuit delays and improves system reliability.

Regarding gate-level optimization techniques, logic gates are designed with low-power characteristics and structures [55] [56]. Professional development tools and software are employed to facilitate power optimization design. For example, a design tool [57] [58] can convert several two-input gates to three-input or more-input gates if the optimization configuration is enabled. A gate with more inputs can effectively reduce power consumption by reducing the number of logic gates and layout complexity. Another common used approach is to optimize pin permutation [59]. Pin permutation achieves low-power design by connecting the high transition-rate signal to a pin with less work load, or connecting a high-load pin with a low transition-rate signal to reduce switching activity.

Circuit-level low-power designs techniques focus on dynamic power. They typically consider physical capacitances and circuit switching frequencies. At this level, power optimization techniques include dynamic CMOS and asynchronous circuits [52] [53]. Dynamic CMOS logic gates memorize data by maintaining CMOS transistors in the state of high resistance. They can effectively reduce the number of device components and thus reduce the load capacitance. The advantages of asynchronous

circuits are modularity and combinability. Unlike synchronous circuits , they do not require global clock, but only need to use handshaking signals and FIFOs to interconnect different modules. Thus, the power consumption caused by the clock jitters of high frequency signals can be reduced. Comparing to synchronous circuits, the asynchronous circuits have greater potentials to reduce the power consumption [54].

Low-power technologies at the register transfer level (RTL) mainly reduce the power consumption by decreasing the glitch-spurious switches of registers. At this level, optimization is mainly achieved through clock gating, memory access, and operand isolation [60]-[62]:

- Clock gating is a commonly used method to decrease dynamic power. For sequential logic circuits, the main source of system dynamic power is due to their frequent clock transitions combined with the large parasitic capacitances of their outputs. Clock gating uses AND/OR gates to control the switch of the system clock. Its main idea is to switch off the clock source of idle modules in order to avoid unnecessary transitions caused by pending signals. In actual circuits, when the input clock of an idle module is disabled, data access of its subsequent modules is also turned off, e.g., if the clock of a register is disabled, the combinational logic connected to the outputs of this register will be in the quiescent state. Thus, the overall power consumption of the circuit will be significantly reduced.
- Memory is an indispensable component in almost any system. It also consumes a large share of the whole energy budget. A method based on block-memory-access divides the available memories into a plurality of parts and a selection signal will be decoded to access the targeted block. A reasonable design of address buses and chip selection signal will only select the required memory block to avoid energy consumption from un-selected blocks.
- Operand isolation technique is a method which makes the module input to be zero and maintains the output unchanged when the system is waiting. Outside stimulus will not act on system until the system is reactivated. In this way, unnecessary switching activities are avoided.

Architecture-level methodologies are considered when circuits are implemented [63]-[68]. At this level, IC designers focus more on the resource allocation and scheduling and balance among chip size, speed, reliability, and power consumption. It is well known that voltage reduction is one method to reduce power consumption. Its disadvantage is that circuit delay is significantly increased. A conclusive low-power design techniques should meet circuit efficiency while reducing voltage. To achieve this objective, the two most popular techniques, parallel structure and pipeline, are applied [65]. Parallel structure decomposes workloads to different processing units, e.g., datapaths or

mutlicores. In a two-core or two-datapath system, the operation frequency can be half of the original one. In addition, the circuit can use a lower operating voltage. The advantage of parallel structures is that they can increase system performance without increasing voltage and frequency [66]. This brings opportunities for energy-efficient design in energy constrained devices [67]. In embedded systems, it has become very popular to integrate multi-core SoCs into the system architecture [68]. This design, on the positive side, increases hardware utilization and leads to higher energy efficiency and performance. On the negative side, the leakage power influence must be considered. Since the leakage power is proportional to the silicon die area, parallel structures increase the circuit size and thus may incur in more leakage power consumption. Pipeline structures are essentially a parallel structure. It divides an instruction cycle into different steps in such a way that different steps can be executed concurrently. However, pipeline design has high complexity, it inserts additional registers among each step and thus increases the area and load capacitances, resulting in additional power consumption. Another important component presents on architectures is system bus which may also contribute to energy reduction. There have been many bus coding algorithms for different bus structures.

4.1.2.2. High-Level Optimization Techniques

Low-level approaches can achieve positive results. But these methods have large design costs and are usually implemented at the design stage. Therefore, they cannot be continuously improved after the device implementation. This lack of flexibility makes difficult to further optimize the energy consumption. Different research works have reported potential gains on energy efficiency by using work load adaption techniques at high level. At this level, efficient methods are employed to manage either task states or hardware components to reduce the energy consumption. These methods will be described in the following.

A. Power Management

Power management aims to effectively allocating and managing power resources to avoid excess consumption. This method has a significantly positive impact on systems, especially on battery-powered handheld devices. According to different principles and objectives, power management can be divided into two groups: state-based and performance-based power management. State-based methods take the advantages of the low-power states supported by devices to achieve power reduction [69] [70]. The main idea is to use the system idle state to save power and to quickly wake up the system when it is needed. Under the precondition to satisfy user requirements, performance-based algorithms decrease power consumption by dynamically adjusting system parameters such as voltage and frequency [71].

Early research mainly focuses on state-based algorithms for low-power design. As chip technology advances, especially when the frequency and voltage of processors become scalable,

performance-based research is gradually becoming a hotspot. Besides the feasibility of frequency and voltage scalability, the main reason to promote the progress of performance-based algorithms is the broader scope of applications regarding those of the state-based methods. Essentially, state-based methods are one of the extreme cases of performance-based methods [72]. For the more complex applications, their characteristic of long-term continuous computing leads to an inapplicability of state-based algorithms. This is because the high complexity and high calculation amount determine that the processing unit cannot maintain in a long period of idle state and more important, a frequent switching itself leads to an increase of energy consumption.

DVFS (Dynamic Voltage and Frequency Scale) is one of the most successful approaches of performance-based optimizations. Experiments were shown that most of the time, programs are running without fully utilizing the processing ability of processors [73]. Regarding to electronic mobile devices, they rarely demand high-performance for basic applications, even for applications such as high-resolution video playback and video call, the system only need to be changed to a high-performance state for a limited amount of time. It is not necessary to always use the highest voltage and frequency in the systems. In order to significantly reduce system power consumption, the DVFS technique dynamically scales the voltage and frequency of processors based on the demand computation of tasks. DVFS must satisfy the time deadline constraint when scales the voltage and frequency levels. Many researches [75]-[79] have conducted their researches to focus on accurate prediction of workload to improve the effectiveness. Common methods are based on algorithm-specific information, compiler analysis, and runtime prediction. However, two characteristics of applications, i.e., the non-stationary behavior and runtime distribution, make it impossible to perfectly predict workload even with the most complex predictors. In work [80], they tried to solve this problem by using a runtime distribution-ware workload prediction. They partitioned the software program into program regions and profiled runtime information, i.e., computational cycle and memory stall time, and updated the statistical parameters of the runtime distributions. Then, they set the voltage and frequency while satisfying the hard real-time constraints.

B. Energy-aware Operating Systems

One possibility to optimize energy is through the management of individual components. In any advanced computing device, the operating system (OS) fully controls the device including the work mode, hardware states, and running applications. The concept of energy-aware OS was proposed in the late 90s with two Linux-based OSs for laptops: Odyssey [81] and ECOSystem [82]. Odyssey estimates future demands of resources and energy to adapt the quality of service delivered to users. Similarly, ECOSytem focuses on the balancing between performance and energy consumption to save energy. It consideres the battery discharge rate as an indication of energy consumption. A share policy

related to tasks' currencies was used to schedule tasks and to preserve energy for more important tasks. In energy-aware OSs, energy is considered as the first-class resource [83]. Recently, battery faces more serious insufficiency to support energy-hungry applications due to the energy limitations of smart mobile devices. Energy-aware OSs, such as Cinder [84] and ErdOS [85], have again been proposed for mobile devices. Cinder achieves energy efficiency by application of isolation, subdivision, and delegation based on the energy accounting and power modeling. Different from other energy saving philosophies, ErdOS includes resources states, usage patterns, and user habits into management policies to achieve more flexible and efficient results [86]. It is worth mentioning Nemesis OS, which was an OS designed for multimedia applications [87]. According to application costs and utilities, as well as the congestion of system resources, this OS builds an energy allocation model and assignes credit to each application when accessing resources. By these means, applications can adapt their energy consumption to save energy.

In conclusion, energy-aware operating systems provide benefits for energy saving by managing the available resources rather than allowing applications to manage these resources.

4.1.3. Video Coding Specific Power Optimization Techniques

General low-power design techniques can be applied to the design of video codecs. In addition to those general ones, video specific features and considerations can provide more possibilities for low power design. Note that in most video coding standards, the encoding process is not specifically standardized. It is free to design an encoder as long as the generated encoded stream can be decoded as it is described in the standard. The decoding process is an inversion of the encoding process. The optimization at the encoder end can also impact the energy consumption of the decoding processes. In this chapter, no matter the optimization is for encoder or decoder, it is uniformed as the video coding optimization. Note that one stage that cannot be exactly inverted is the quantization stage because it is a non-invertible process which loses information. The following discusses low-level and high-level optimization techniques in video coding.

4.1.3.1. Low-Level Optimization Techniques

At the integrated circuit level, Liu et al. [88] have optimized algorithms in IDCT, deblocking filter, and prediction units to reduce processing cycles, memory size, and access frequency. Other low-power designs have included the use of constant multipliers [89], reduced transitions in datapath [90], and self-adaptive techniques [91]. These approaches were mostly applied to RTL level.

At architecture-level, different impacts on energy consumption of video coding have been deeply investigated. The reason why pipelines and parallel architecture enhanced the energy efficiency was

studied with performance and energy considerations in mind [92]. This study has provided a good guide for energy optimization design. A power-aware motion estimation design was presented in work [93]. The motion estimation (ME) has multiple modes and was supported by a specific VLSI architecture which reduced external accesses caused by video content and thus further reduced the power consumption. An energy-aware processor architecture design methodology within the balance of power, throughput, and area was specified for a H.264/AVC codec [94]. The design guidelines included pipeline organization and granularities, parallelism, and memory architecture. Parallel architectures of integer ME and fractional ME was proposed for memory access reduction [95]. It also achieved power scalability and hardware efficiency.

An interesting result of digital signal processor (DSP) usage was shown in paper [96] to guide the processing element selection for video decoding applications on embedded heterogeneous platforms. These platforms usually contain several General purpose processors (GPPs) and DSPs. GPP is a processor that is not implemented to particular languages or programs while DSP processor, is specialized to process particular types of operations to provide better performance-energy properties. In the previous mentioned work, they indicated that the required video bit-rate and resolution greatly impacted on the tradeoff of performance and energy. GPP could be a best choice in many cases, especially for those cases of low video bit-rate and resolution because of a considerable processing overhead in the case of DSP decoding, which might lead to degradation in performance and energy efficiency.

Furthermore, compared to other applications, video codecs process larger amount and more various types of data. Cache optimizations for multimedia applications have been proposed by many researchers. Z.Y. Xu el. al. [97] indicated that multimedia applications typically had a data block strategy which had good reusability. Thus, the structures and rules of traditional cache were still valid for multimedia. S. H. Wen et. al. [98] divided cache memories into three parts: instruction loop buffer, instruction cache, and scratchpad memory data buffer. In addition, they utilized the multi-bus mode which included one instruction bus and several data buses. Data were directly passed to scratchpad memory for processing usage through DMA in order to reduce the system energy on data searching and loading. J. H. Kim [99] proposed two new cache structures for optimizing data access of motion compensation in an H.264/AVC decoder. One structure was called index-separate-direct-mapping cache which mapped one page of the main memory into two continuous lines of cache. Another was called circular cache which only stored the necessary part of data instead of using the full cache line. Both structures could reduce the demand of memory bandwidth and improve the system performance and thus the system power consumption was decreased.

4.1.3.2. High-Level Optimization Techniques

A. Complexity-Based Methods

Different from other applications, video processing is usually modularized and standardized. This uniformity makes the low-power design on video start to reduce the computational complexities of algorithms to decrease power consumption.

Chu-Hsing Lin at el. [100] observed several energy issues. They observed that the fast scene changes resulted in consumption increase. They proposed that videos could be encoded with higher bit-rate to have better video quality because by doing this, it paid a lower penalty in power consumption than increasing the video quality by increasing the resolution.

Landge et al. [101] proposed a wavelet-based video decoder using hardware independent complexity metrics. The metrics were derived from the frequency of basic blocks executions and captured video content features and encoding parameters. They could be translated into platform-specific metrics to determine for the optimal voltage and frequency configuration for energy optimization.

Work [102] analyzed computational complexity of main functional blocks in an H.264/AVC encoder. They stated that ME occupied around 98% of the total computational complexity of the encoding process. A large reduction of power consumption could be achieved by reducing ME the complexity. In [103], a multi-mode content-dependent ME algorithm was proposed for power-aware video coding. Based on the predictions and judgments of motion complexity, the ME execution was switched among one of the four searching modes: full search, adaptive search range, adaptive enhanced four-step search, and three-step search. Those modes with lower search range decreased the computational complexity, while the image quality in terms of PSNR (Peak Signal to Noise Ratio) was dropped a little accordingly.

Wang et al. [104] improved the Sum of Absolute Transformed Differences (SATD) algorithm by using the linear transform and the fixed-spatial relationship of predicted pixels in intra mode. This fast SATD could be applied after Sum of Absolute Differences (SAD), eliminating unwanted intra prediction modes to significantly reduce computational burden. Another intra prediction improvement was proposed by Hsia et al.[105]. In this work, the prediction of the 4x4 intra blocks in improved based on partial sampling prediction and symmetry of adjacent angle modes.

The analysis in [106] showed that the entire computation for prediction mode selection could be reduced if those less probable modes were skipped for computation. In this analysis, a fast coding mode selection in H.264/AVC encoders was proposed. Similarly, Grecos et al. [107] showed that a

high speed-up could be achieved if the computationally intensive prediction modes were not performed. They proposed an inter-mode decision scheme for P slices in an H.264/AVC codec by using smoothness constraints, neighborhood information, and a set of skip mode conditions. Also, Kim et al. [108] proposed an algorithm for inter-model determination based on the rate-distortion cost of the tracked MB for the current MB.

B. DVFS-Based Methods

DVFS methods are also wildly used on video coding energy optimization. W. H. Yuan et.al proposed a GRACE-OS [109], an OS which implemented a low-power scheduler. This algorithm reduced the processor energy consumption while guaranteed the performance. It dynamically predicted the work load of tasks based on real-time statistics to achieve internal-task DVFS. Although this method obtained good results of energy consumption reduction and deadline guarantee, it increased the frequency of scale and did not consider the variation of task arriving time due to network transmission. Subsequent researches continuously developed this idea and included energy-efficient scheduling [110]-[113] to achieve better results.

Several parameters represent coding features. Thus, they can be used to predict the workload. Soner.Y et. al [114] proposed a stochastic modeling and optimization framework to perform dynamic voltage scaling in multicore systems by capturing the spatial and temporal variability of tasks such as frame type, compressed frame data size, and the computational workload to decode a frame. In the previously mentioned work, four models to capture the task statistical characteristics were introduced. Different tasks could be mapped into different cores with various voltages. This scheme minimized the average energy consumption while guaranteed the time constraints. Similarly, in [115], three efficient DVS techniques were presented for an MPEG decoder. A simulator was developed to take workload predictions based on the number of IDCT computations required by each frame.

Conventional DVFS methods used the expression *WCET/D* as the metric to set the frequency to meet the deadline constraints, where *WCET* is the worst case execution time and *D* is the time to deadline. Research works showed that these pessimistic methods could lose opportunities to further reduce energy consumption. WCET rarely occurs in reality because the execution time may present different kinds of statistical distributions. Work in [116] indicated that more energy reduction was obtained by setting the frequency as the ratio of average execution time to the time to deadline as long as the deadline condition was guaranteed.

From the architecture point of view, the voltage and frequency adjustments should avoid the cyclic nature of dependencies in executing tasks to lose the throughput constraints. In work [117], a methodology was proposed for such cyclic dependent tasks. This methodology assumed WCET to

identify the executions that could be slowed down at an off-line step and utilized the slacks arising from tasks that finish their executions before the WCET to execute new tasks. Thus, the energy consumption was reduced while the throughput constraints were satisfied. Similarly, in work [118], they focused on energy optimization with the consideration on transition overhead, inter-core communication, and discrete voltage levels. They proposed a two-phase approach. In the first phase, they proposed a coarse-grained parallelization algorithm which utilized a set of independent sub-tasks instead of the original periodic dependent tasks. In the second phase, a genetic scheduling algorithm was proposed to search and find the best schedule to optimize energy consumption.

C. Scale-based Optimization Techniques

More recent researches showed that video quality and performance did not increase linearly as the computational complexity increases. As shown in Figure 4-2, user perceptions of video quality and performance always meet a saturation point and little improvement is achieved beyond this point [119].

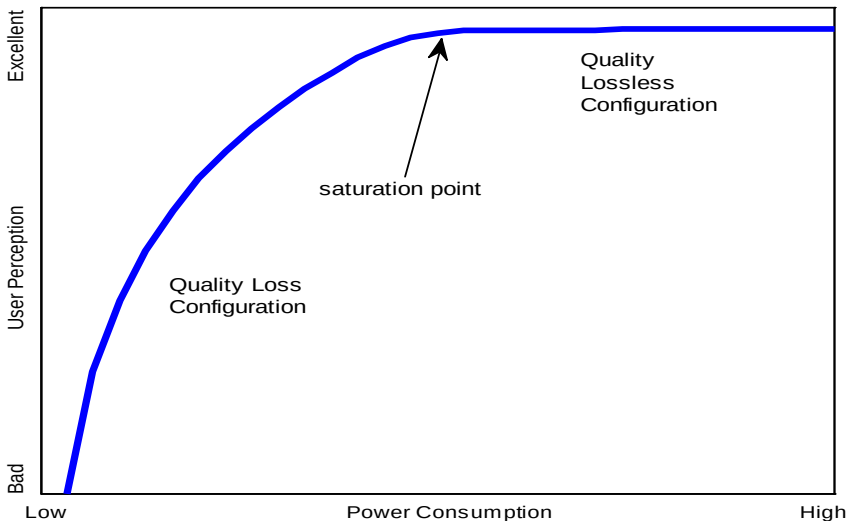


Figure 4-2 User Perception VS. Power Consumption [119]

This conclusion has attracted researchers to focus more on rate-distortion-complexity optimization rather than ceaselessly persist in computational complexity to achieve better user perception. The goal of rate-distortion-complexity optimization is to, heuristically, select power-aware algorithms to adaptively tradeoff between user satisfaction and power consumption based on video content features and battery states. For example, a configurable video coding system [120] was proposed to use dynamic parameterization in ME based on the tradeoffs derived from input and output signal statistics. In [119]- [122], the proposed system could dynamically configure it based on battery status, content complexity, user preference, and operating environment to prolong the battery lifetime and meet resource constraints of target platforms.

This adaptive idea has been standardized as Scalability Video Coding (SVC) as an extension of the H.264/AVC codec. It can efficiently encode video streaming to fit various bandwidths of transmission channels and computational resource constraints of decoding devices. SVC structures video bit streams into different layers corresponding to different qualities, frame rates, and resolutions. Thus it can provide flexible encoding solution for temporal, spatial, and quality scalability. Proper layer design can provide a power-aware feature for codec design [123]-[125].

Some new optimization schemes including user experience has been studied for on-line streaming. In[126], they managed the downloading cache based on the user view history and the network condition to minimize unnecessary active periods to save energy. In addition, video codec optimization can integrate various techniques. In[126], they presented a scalable H.264 Ultra-HD video codec engine that used various low power optimization techniques across architecture, design, circuit, software and system. A special custom buffer was designed to enable accessing up to 36x4 pixel aligned at any vertical pixel position to simultaneously reduce the shared level-2 memory accesses. Meanwhile, a task management scheme dynamically switches among the used hardware accelerators.

4.2. Reconfiguration on Video Coding

4.2.1. Implementation Complexity on Video Coding

Along with the significant development of video codec, user demands of video are increasing in diverse directions. Users have higher requirements on video quality, as well as flexibility and scalability of the video products. Therefore, the implementation of the new video coding generation has reached a consensus on higher coding performance and greater flexibility from both theoretical and practical points of view.

In conventional video coding, each standard defines a different format. A standard is the only bridge to achieve communication among codecs. That is to say, a codec can only be implemented to encode/decode video streams in accordance with the provisions of standards. This greatly reduces the adaptive capacity of codecs. Some standards define different profiles and levels with sectional differences to satisfy more user requirements. As a consequence, a decoder which is satisfied to particular profiles or levels only supports particular categories of applications [128]. This increases the burden of network transmission. Standards were implemented in the past as a monolithic specification with predefined functionalities, applications, and platforms [10]. Although coding designers have considered the overlap among encoder tools, however, since codec systems have been frequently implemented in highly parallel computing platforms, it is inevitably to introduce additional

complexities during the implementation processes, such as global variables, non-optimal process arrangements, and inefficient data structures. Continuous evolutions of more complex and advanced standards have seriously impeded their efficient specifications and implementations. To meet various demands for multimedia communication, traditional video codec standards are facing difficulties such as longer time to market, difficult to quickly implement, and lack of flexibility. With the consideration of these difficulties, the concept of reconfiguration was proposed to improve the reusability and flexibility of codecs.

4.2.2. Reconfiguration Techniques

4.2.2.1. Definition of Reconfiguration Techniques and their Hierarchy

Reconfigurable techniques were derived from reconfigurable computing, which was first proposed by Professor Gerald Estrin in 1960 [129]. Reconfigurable computing is a computing mode between software and hardware computing. It achieves the approximated performance to hardware while maintains implement flexibilities as software implementations. The main technical basis of reconfigurable computing is reconfigurable devices. The internal hardware circuits of reconfigurable devices are determined by configuration information, thus, the hardware resources can be programmed by dynamically calling or modifying the configuration.

From another perspective, reconfigurable techniques have the capability to change or improve system functionalities or produce new functionalities by re-connecting the existing functional units. To emphasize this purpose, this capability is stated as functional reconfiguration, or functional-oriented reconfiguration. In this case, the reconfiguration process does not focus on the physical implementation details (e.g., implemented platform, software or hardware computing, how to configure the reconfigurable devices). In fact, functional-oriented reconfiguration pays more attention on how to change system functionalities. It performs in an earlier stage than the physical implementation stage. It is like a conceptual tool to provide functionally correct designs at a higher level. A functional-oriented process is independent of its implementation platform.

Figure 4-3 summarizes the relationship among different levels of reconfigurable systems. A reconfigurable implementation can be divided into physical, rule, and system layers. The physical layer consists of processing devices such as the configurable processor, e.g., DSP or FPGA. It is the infrastructure for a reconfigurable platform. The rule layer defines the methodologies and techniques to achieve a reconfigurable system, such as task scheduling, software and hardware division as well as communication system. System layer is higher than physical and rule layers. It is based on user-oriented requirements to provide system-level reconfiguration schemes. Functional-oriented reconfiguration is implemented at this level. It aims to optimize applications to satisfy performance

requirements or to change the connection of functional units to implement new functionalities. This process is platform-independent. Any updated or new functionality will then be converted to the specifically concrete implementation on any hardware platform by automatic conversion tools.

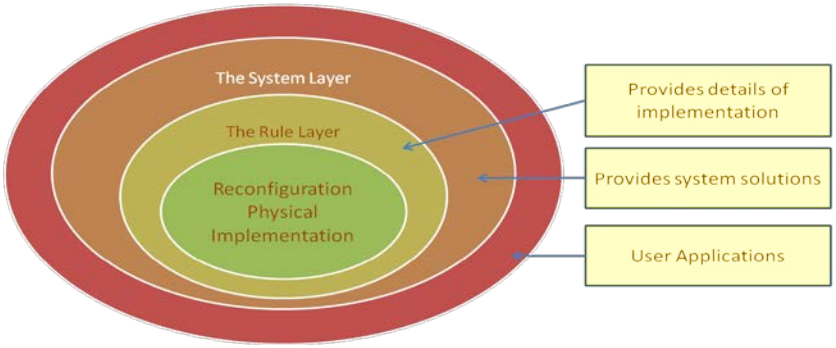


Figure 4-3 Relationship among Different Levels of Reconfigurable Systems

4.2.2.2. Implementation Techniques of Reconfigurable Computing

Reconfigurable computing can be implemented by a variety of methods, which can be included in any of the following categories: software reconfiguration, programmable hardware reconfiguration, and logic structure that is dedicated to the execution phase.

Software reconfiguration, which can significantly reduce the utilization of hardware, is the most commonly used method. A software reconfiguration may achieve running time reduction and flexibilities when the design is changed. An inappropriate software reconfiguration may increase the power consumption and become an unpractical scheme. With regard to the performance, most designers prefer to implement all the performance-constrained functions by hardware. Hardware reconfiguration techniques can be divided into static and dynamic ones. Static reconfiguration techniques refer to the static overloading of logic functions, i.e., to change the logic functions by simultaneously downloading the configuration files and information to the programmable devices in addition to the external control flow. When the system is running, reconfiguration is not allowed and the process of reconfiguration cannot be interrupted. Static techniques are not competent for applications whose circuit may often need to be changed, such as a video streaming computing. On the other hand, dynamic reconfiguration techniques have a caching logic, which can quickly modify the global or local circuit logic by means of an external logic which controls the layout, routing, and resource allocation. Dynamic reconfiguration can be processed in clock cycles, not affecting the overall system operation. Although compared to static techniques, dynamic ones have already improved in efficiency and flexibility, they still have a potential risk to cause a performance decrease and the power consumption increase if the reconfiguration lasts a long time for implementing complex logic functions.

Reconfigurable computing approximately achieves the design flexibility of software designs implemented on GPPs and the performance efficiency of designs implemented on application specific integrated circuits (ASICs). Since Xilinx Company introduced its first Field Programmable Gate Array (FPGA) in mid-1980s [130], reconfigurable technology has been rapidly developed and promoted. FPGA is an integrated circuit designed to be able to make substantial changes after manufacturing at run time. Currently, researches in reconfigurable computing involve many aspects such as software and hardware platforms, operating systems, programming languages, compiler tools, and implementation algorithms. These issues will not be extended since they are out of the scope of this thesis.

4.2.2.3. Implementation Techniques of Functional-oriented Reconfiguration

A reconfigurable computing device can implement and modify a specific functionality. Functional-oriented reconfiguration is an abstract model independent of the platform details. The purpose of functional reconfiguration is to reduce the complexity of an initial design, improve the code quality, optimize the structure, and facilitate the modification and extension of functionalities. The core idea of functional reconfiguration is to take into account of new environments, requirements, and functionalities of software systems in advance and, as a consequence, to improve the flexibility, reliability, and development efficiency of a design to avoid the redesign of the whole system.

A functional-oriented scheme aims to provide a portable solution for any platform. Thus, its implementation should be portably translated from the high-level representation to native machine code suited to the architecture of the underlying platform. Figure 4-4 shows an example of functional-oriented implementation. A virtual machine (VM) is employed to provide the independence of platform. Meanwhile, to shorten the time of translation from high-level presentation to machine code, quick compiler techniques, especially JIT technique, is employed to obtain this speedup.

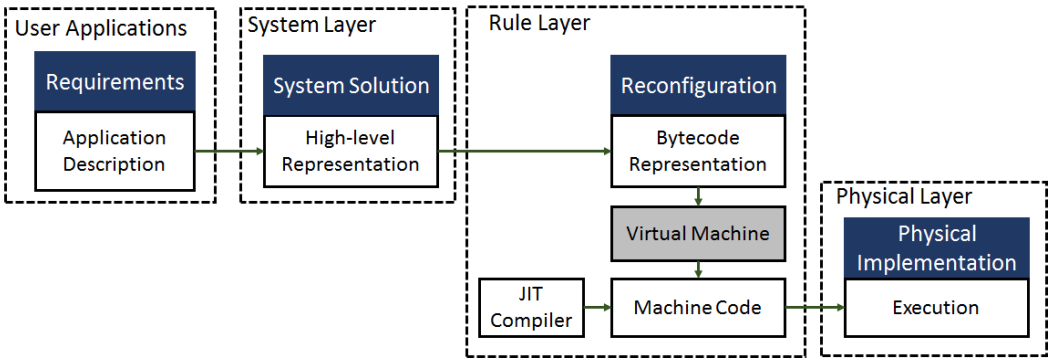


Figure 4-4 Functional Reconfiguration Framework

Functional reconfiguration techniques are briefly introduced in what follows.

A. Virtualization

Generally speaking, a compilation process is an inevitable step from source code writing to application binary generation. Compilation is actually processed by either a compiler or an interpreter. The former converts the human readable programs to the executable machine codes, the so-called binary or backend codes, which can be directly executed on a particular hardware.

The whole compilation procedure is separated into four stages: high-level source code, intermediate representation (IR) (e.g., object files or bytecode), sequence of operation, and final execution. The difference to use a compiler or an interpreter is the moment to trigger the execution procedure. After translating the source code into IR form, compiled languages (such as C, C++ and FORTRAN) use a linker to generate the complete machine code, which is a series of basic operations to control the target processor to execute its corresponding work. Interpreted languages (such as Java, Python and Ruby) also experience these steps. The difference is the following. A compiled language first saves those basic operation sequences generated from the source code, and then, executes them together by one single command; an interpreted language “throws” these basic sequences to an execution unit and immediately produces the actions. The observed phenomenon is that a compiled language needs firstly compilation and then execution, while an interpreted language can be directly “executed” from the source codes.

In addition, for those compiled languages, the generated object files are specified for one particular processor architecture. For example, the files for the ARM architecture cannot be used for a MIPS processor. Source files are translated into the target processor instructions during the compilation. Thus, the same source code needs to be re-compiled to be executed on another processor. For any interpreted language, a compilation procedure is also needed. But the generated IR files are platform independent, and they can be translated into specific processor instructions during the execution processing. Therefore, the source code can execute on various platforms without intermediate conversion. The process of translating IR files to target processor instructions is completed by a virtual machine (VM).

Although interpretation and compilation are the two main methods by which applications are implemented, they are not mutually exclusive, especially for modern languages. The terms of “interpreted language” or “compiled language” only mean that the canonical implementation of that language is an interpreter or a compiler, respectively. Currently, the virtualization-oriented technology further eliminates their boundary. A high level language is ideally an abstraction independent of particular implementations. Virtualization technology includes an application, the VM, and additional software responsible for implementing the program execution. A VM is an abstract layer outside the hardware layer to achieve the cross-platform or cross-instruction-set implementations as long as the

compiler translates the program codes into the IRs of the VM. It brings more flexible practices on code optimization. Essentially, the JVM (Java Virtual Machine) of Java and the CLI (Common Language Infrastructure) of C# are VMs rather than simple programming languages. They have an stack-based instruction set architecture, which is hardware independent. Applications based on VMs are run on a computer using an interpreter or a Just-In-Time (JIT) compiler or both. Details of JIT compilers will be described later. By using a VM and a very-high-level input program representation, these systems are able to provide platform portability and security services in addition to reasonable performance.

B. Just-in-Time Compilers

As mentioned before, to achieve real-time reconfiguration, fast compilation and execution are required. This is to say, an application will not be compiled and translated into native instructions like classical compilers do. Instead, the compiler should recompile the functional units as less as possible. To achieve this goal, target application is first translated into an IR format. Then, IR files will be read line by line and executed by an intermediate engine from a VM. As discussed in the previous section, an interpreter saves the compilation time, but its execution efficiency is much lower than a compiled binary program. Thanks to the JIT technique, the execution time has been much better improved.

The JIT compilation is a combination of interpretation and compilation, having the advantages and drawbacks of both. JIT compilation interprets instructions one by one, but it will cache the translated codes to reduce recompilation overhead. JIT compilation is a form of dynamic compilation and it is particularly suited to dynamic programming languages. A JIT engine takes IRs from a compiler front-end and produces machine code to execute it on-the-fly. Note that JIT does not target to parse the syntax neither to provide a runtime library support. It typically takes the conjunction with front-end engines and a suite of libraries to generate IRs and to provide the environment for code execution.

JIT has two features that help it to achieve the reconfiguration goal: (1) efficient, especially during the run-time; (2) easy extension. Compilers use abstract syntax trees (ASTs) to translate and compile source codes. In computer science, an AST is a tree structure to represent the abstract syntactic structure of source codes. The word abstract means that not every detail of the real syntax will be represented in the AST. Modifying ASTs requires updating existing modules, which makes difficult to support older codes when new features are added. JIT can add new frontends without adding new node types to the AST. This feature provides to JIT an, easy extension as far as the implementation concerns. Although JIT engines have essentially identical goals, they may implement different approaches for processes such as IR optimization, machine code emission, and code

rewriting on different architectures. And the cost of expensive processes should be controlled in order to maintain performance benefits from native code creations.

4.2.3. Functional-Oriented Reconfiguration on Video Coding

4.2.3.1. Video Decoding Process Framework

Different video standards vary in their algorithms, but their similarities can be summarized as the following three aspects:

- Video algorithm operations are based on MBs. The structure of MBs is shown in Figure 4-5 as an example. The pixel size of a MB may be different but MB operations are all designed with parallelism in mind.

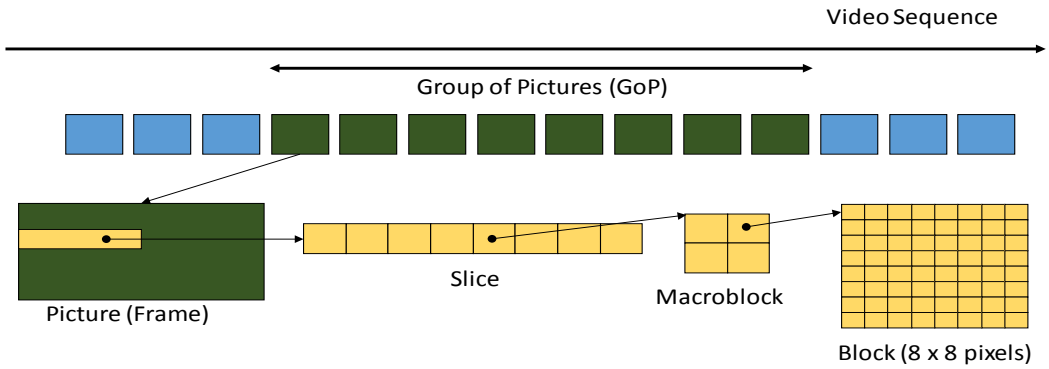


Figure 4-5 Video Sequence hierarchy

- Video data are processed by structured operations. Each pixel within the block has the same dependencies and relationships to other pixels.
- Each conventional codec system of the MPEG series is based on the same processing procedure, which can be structured into several main functional blocks as shown in Figure 4-6. This framework consists of a syntax parser, a residual decoding module, a motion compensation module, and a frame buffer module for prediction. The syntax parser extracts the syntax structure of the input streaming to obtain the control information and residual data. The residual module completes the inverse quantization and inverse transformation of the residual data. The motion compensation module and the buffer module will implement the picture prediction.

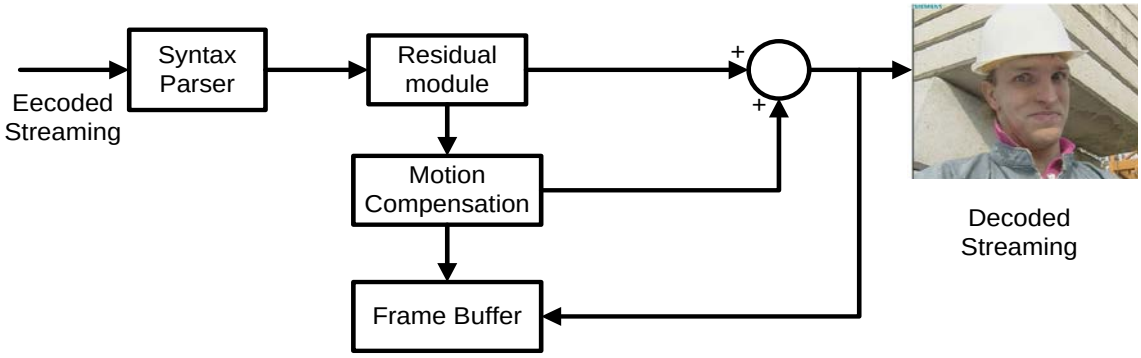


Figure 4-6 Hybrid Decoder Framework

A simplified block diagram of an H.264/AVC video decoder is shown in Figure 4-7.

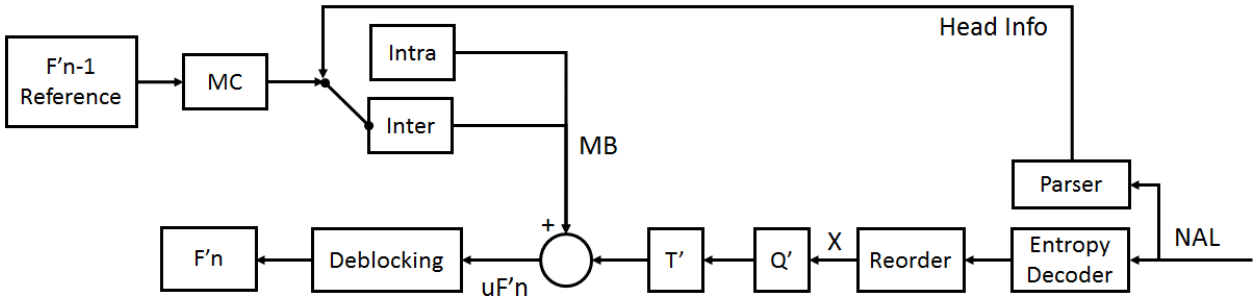


Figure 4-7 Simplified Block Diagram of an H.264/AVC Video Decoder

The encoded stream comes in the form of Network Abstraction Layer (NAL) packets. To decode the contents of a video NAL packet, the decoder first performs the entropy decoding and then reorders the decoded data to obtain the quantized coefficient array X. The residual difference Dn' is obtained after inverse quantization (Q') and transform (T') of X. The decoder parsers the header information of the streaming data to determine if intra or inter decoding process must be carried out, at a macroblock (MB) basis. For inter predicted MBs, a predicted macroblock (MB) is obtained by the Motion Compensation engine (MC) based on information on one or more previously decoded frames ($F'n-1$). For intra MBs, a predicted MB is obtained based on the information on previously decoded MBs from the same frame. After adding this MB to the residual to obtain picture $\mu F'n$, then the decoder carries out a loop filter (Deblocking) to complete the decoding and, at last, outputs the decoded video in YUV format. The reconstructed picture will be stored in a frame buffer and may be used as one of the reference pictures for next frames.

4.2.3.2. Reconfigurable Video Coding Standard

As it has been introduced above, video processing can be divided into different procedures. Thus, it is possible to implement new coding functionalities just changing any of the procedures without modifying the whole decoding framework. In this context, the Reconfigurable Video Coding (RVC)

standard has been proposed by MPEG [131]. MPEG RVC has proposed the creation of a flexible and configurable video coding framework. RVC defines each coding procedure as a coding tool. It builds a sharing mechanism on these coding tools based on either non-compliant or compliant MPEG standards to promote the development of multimedia middleware. RVC has introduced the idea of functional-oriented reconfiguration to video fields. It aims to dynamically reconfigure decoders according to different requirements and specific applications with high flexibility and scalability [131]-[133].

A. RVC Implementation

Unlike other standards which have been developed case-by-case in monolithic textual descriptions (e.g, C/C++ reference software), RVC is based on the concept of Abstract Decoder Models (ADMs), which takes advantage of reusable and restructurable coding tools to implement multiple decoders or even a new decoding scheme different from any existed coding standard. The use of ADM allows a generic representation on multiple coding specifications. It supports the dynamic combination of different coding tools with a high flexibility, reusability, and scalability. Traditional MPEG standards provide different profiles to balance between decoder compression performance and its implementation complexity. To enable an appropriate profile selection, a normative description has been included into the media syntax (bitstream). RVC has further extended the profile-based formalism. The decoder fundamental algorithms can be combined in an arbitrary way in the RVC standard by adding side-information into the encoded bitstream. Two MPEG standards are applied in this context [131]: 1) MPEG-B part 4, which generally defines the framework and the standard languages used to describe the components of the framework and 2) MPEG-C part 4, which defines the video tool libraries (VTLs) employed in existing MPEG standards. These two standards are continuously evolving with new amendments for upcoming or future decoder descriptions.

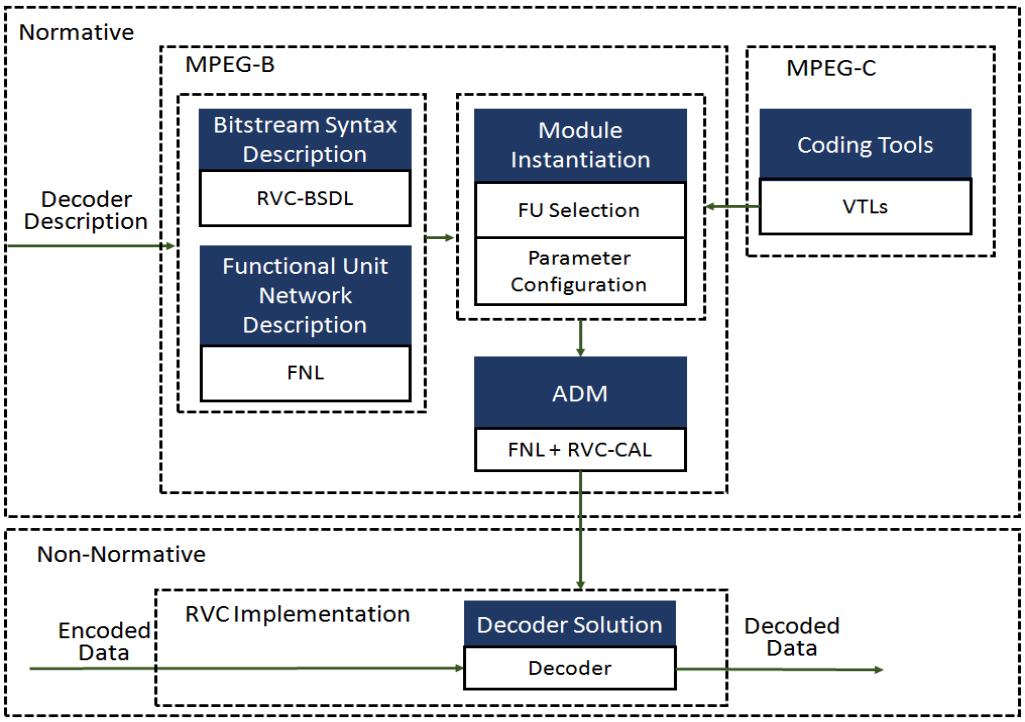


Figure 4-8 RVC Framework

As shown in Figure 4-8, RVC framework aims to produce an ADM that represents the specification of a decoder profile. The ADM allows passing the reconfiguration information, referred to a decoder description, from encoder to decoder. An RVC bitstream includes two parts: the original encoded media content and the decoder description. The decoder description includes two kinds of information: Bitstream Syntax Description (BSD) and Functional Unit Network Description (FND). BSD information describes the syntax structure of the encoded bitstream. For example, the syntax elements, and the attributes of syntax elements such as length, number and the occurrence order of these elements. FND information describes the employed codec tools and their inter-connections. RVC framework has standardized VTLs to represent each functional unit (FU) as a coding tool. Currently, all the FUs are drawn from existing MPEG standards but the VTLs can be updated with any new coding tool. When a decoder description is received, a decoder parser will first analyze this information to know the structure of encoded bitstream by parsing the BSD and to know the connection of employed FUs by parsing the FND. According to the result of this analysis, the decoder will be implemented by selecting and connecting the corresponding FUs from VTLs.

Each FU has a textual specification that defines its purpose and a reference implementation expressed in a standardized language called RVC-CAL Actor Language (RVC-CAL) [134]. In this language, an FU is defined as an actor, which is an encapsulated entity including input and output interfaces, parameters, and an internal finite state machine. One actor cannot modify the internal state of another one. The only form of interaction among actors is to send tokens through connection

channels. An actor may have several actions. An action is a computation process which may consume the input tokens and may produce output tokens. Each action has its own enable condition, input tokens, and current actor state. After an action executing, the actor internal state may be changed. The name of the FUs is normative to distinguish two kinds of usage in the VTL:

- The algorithmic (ALGO) coding tools, such as the Inverse Discrete Cosine Transform (IDCT) and the Inverse Quantifier (IQ);
- The data management (MGNT) tools, such as data multiplexers or demultiplexers.

RVC-CAL is trying to express the parallelism and modularity of decoder algorithms, which is a suitable attribute to implement RVC technique on a wide variety of platforms, from multi-core GPPs to FPGAs. ADMs are built as block diagrams with an XML dialect, the so-called XML Dataflow Format (XDF). FUs are processing entities and their connections represent the data flow. With the normative standard libraries of FUs and decoder descriptions, an ADM is able to define a new decoder or a decoder based on existing standards. It expresses a configuration to form a decoder. This configuration corresponds to an oriented graph in which vertices are the required FUs and edges are the communication dependencies between FUs. Figure 4-9 gives an example of a decoder configuration. An instance of FU is defined by its identifier and its name attribute. It can optionally assign values to the parameters of an actor.

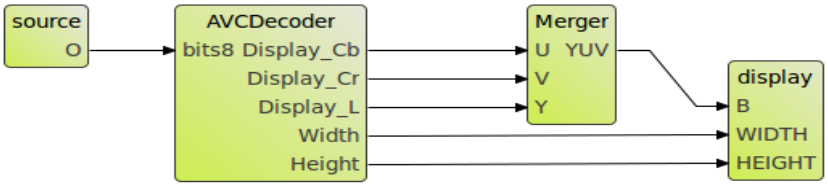


Figure 4-9 MPEG RVC Configuration of Decoders

The FNL expresses the network for one decoder configuration. An FNL defines 3 types of edges: (1) between an input port of a network and an instance (input); (2) between an output port of an instance and an input port of another instance (Connection); (3) between an output port of an instance and the output port of a network (Output). As an example, Figure 4-10 illustrates the FNL description of network in Figure 4-9.

```
<?xml version="1.0" encoding="UTF-8"?>
<XDF name="Top_mpeg4_part10_CBP_decoder">
    <Instance id="source">
        <Class name="org.sc29.wg11.common.Source"/>
    </Instance>
    <Instance id="Merger">
        <Class name="org.sc29.wg11.mpeg4.part10.cbp.display.Mgnt_Merger420_AVC"/>
    </Instance>
    <Instance id="display">
        <Class name="org.sc29.wg11.common.DisplayYUV"/>
    </Instance>
    <Instance id="AVCDecoder">
        <Class name="org.sc29.wg11.mpeg4.part10.cbp"/>
    </Instance>
    <Connection dst="AVCDecoder" dst-port="bits8" src="source" src-port="O"/>
    <Connection dst="Merger" dst-port="U" src="AVCDecoder" src-port="Display_Cb"/>
    <Connection dst="Merger" dst-port="V" src="AVCDecoder" src-port="Display_Cr"/>
    <Connection dst="Merger" dst-port="Y" src="AVCDecoder" src-port="Display_L"/>
    <Connection dst="display" dst-port="WIDTH" src="AVCDecoder" src-port="WIDTH"/>
    <Connection dst="display" dst-port="HEIGHT" src="AVCDecoder" src-port="HEIGHT"/>
    <Connection dst="display" dst-port="B" src="Merger" src-port="YUV"/>
</XDF>
```

Figure 4-10 FNL Description of the Network in Figure 4-9

B. Advantages of RVC

RVC has been introduced mainly on the basis of the following aspects:

- To support diverse video content and formats. A variety of video standards have been widely used in different situations. Since media materials are in varied formats, applications require a device that can dynamically change its operating mode to adapt to media contents without restarting. For example, IPTV was clearly defined in its design requirements to allow the use of bitstreams of several standard types (Figure 4-11-a) in the transmission end. To be able to decode all the bitstreams, currently, there are two main solutions at the receiver end: one is to use the transcoding technique, and another is to use multiple decoders. Transcoding increases the decoding time and causes a progressive loss of quality for each successive generation due to the cumulative compression degradations. While using multiple decoders, on one hand, the system complexity is greatly increased especially when it needs to switch among a variety of codecs; On the other hand, it is necessary to pre-specify the bitstream transmission order because the hardware devices can not dynamically adapt their functional units to decode the different bitstreams. How to enhance the flexibility of the receiver end and to dynamically adapt to different user demands have become meaningful research issues. RVC framework solves this problem through reconfiguration. As shown in Figure 4-11 (b), the transmission channel allows transferring the bitstream with various standards. All decoders are constructed to be conformed to the MPEG-B standard. Depending on the decoder description, a decoder can be constructed by selecting FUs within only one MPEG standard VTL, by using FUs from several MPEG standard VTLs, or by combining MPEG VTLs and non-MPEG VTLs together. RVC provides a unified platform for video codec technology to relieve the incompatibility of different standards.

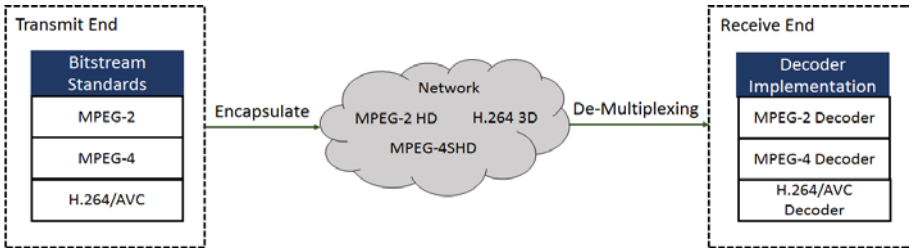


Figure 4-11 (a) Traditional Solution

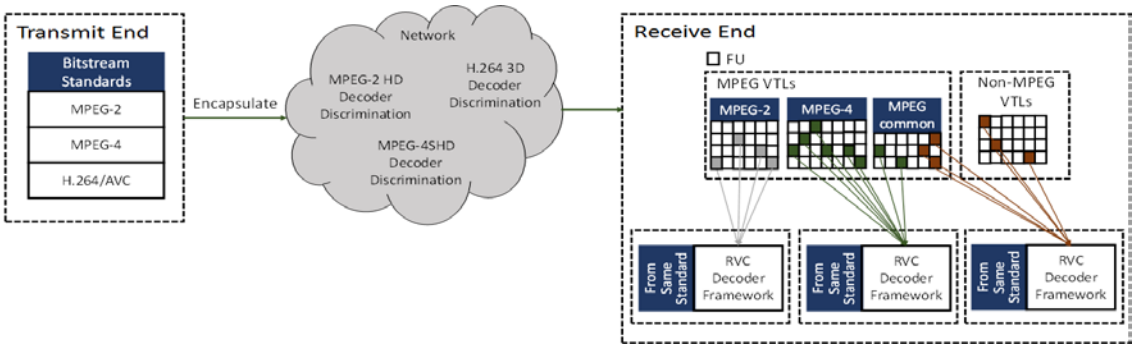


Figure 4-11 (b) RVC Solution

Figure 4-11 Two Solutions to Manage Multiple Bitstreams

➢ To shorten the design period and to avoid repeating designs. As mentioned before, current mainstream video standards have many coding tools in common (e.g., transformation, quantization and intra-prediction). The successful video standards have presented common design methodologies and functionality partitions. This similarity demonstrates a possibility of FU reutilization. It is also possible to introduce new tools with similar structures.

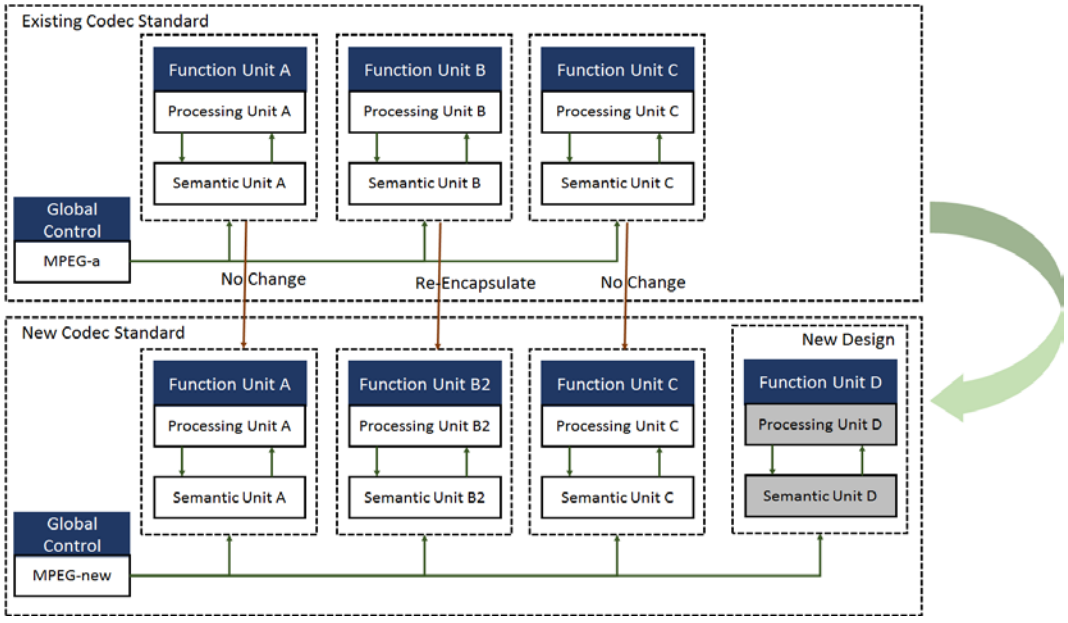


Figure 4-12 An Abstraction of RVC Framework for a New Standard Development

Figure 4-12 illustrates how a new video coding standard, MPEG-new, can be designed based on an existing one, MPEG-a. The necessary modifications are to re-encapsulate the

FU-B and to design new FU-D while the FU-A and FU-C can be reused without any change. RVC scheme significantly facilitates a new standard implementation. It provides a platform which defines standards from the perspective of coding tools to improve the reusability and introduces new features and technologies to systems. MPEG RVC aims to reduce the technical barriers among video coding standards by unifying and combining different functional units of various coding toolsets.

In addition, with the fast technology developments, video coding will move towards a new stage. MPEG and VCEG have launched a new standard for a new generation coding research [135]. Comparing to the current algorithms, new standards will provide higher resolutions and significant improvement in compression efficiency while increasing the coding complexity [136]. Larger data and faster processing speed are big challenges of existing video equipment and implemented technologies. It has become worthy of studying on how to quickly adapt the codecs themselves to new coding technologies and the inclusion of new coding tools.

- To facilitate the demand-oriented designs. With the continuously increasing demands for products, personalized design is becoming more attractive. Traditional video codec design defines a number of standards for choice, which limits users' preferences within the range of defined standards. In some situations, users would rather define their own decoders to avoid unnecessary complexities. For example, H.264/AVC standard defines the syntax elements and various coding tools. In a certain application, only part of these elements or tools needs to be used. However, all of them have to be supported by the decoder to meet an H.264/AVC standard. In addition, with the development of video codec technologies, many new encoding tools with high performance or low complexity have attracted the vendors. But they have not been included into the existing standards, thus they cannot be correctly decoded by the existing devices, unless a new standard is developed. One feature of RVC is to quickly implement and include new coding tools. The only thing to pay attention is to ensure that the input and output interfaces of new coding tools can be correctly connected to the decoding network without influencing the data flow.

Along these lines, the development of MPEG RVC codec standard has gradually become a hot topic in the video coding field. MPEG RVC framework will provide a unified platform for video codec technology. The aim is to reduce cost of the new technology development, relieve the incompatibility of different standards and enhance the promotion of the new video standards.

4.3. Conclusion

Motivated by the pervasive use of multimedia applications on battery-powered portable devices, the latest video coding standards have been developed to enable higher data compression rates and decoding efficiency. This continuous evolution towards more complex and advanced standards has greatly impeded their efficient specification and implementation from both energy-constraint and design standardization points of view. This chapter includes two parts. They present a detailed introduction on the related work on two topics: energy optimization and reconfigurable design. In the first part, energy optimization techniques have been first introduced at different levels, including low-power designs for the special case of video coding. The second part is related to reconfigurable design, which is a new philosophy motivated by the requirements of high flexibility and scalability. Many implementation techniques have been studied for reconfigurable design. In particular, reconfigurable design on video coding is defined in this thesis as functional-oriented reconfiguration. The MPEG reconfigurable video coding standard is described and its advantages are described.

5. Energy Optimization based on Functional-oriented Reconfiguration

Chapter 1 has addressed the motivation to optimize the energy consumption of streaming applications running on battery-powered mobile devices. Video codec design no longer only focuses on performance improvement and response time reduction but gradually more attention is being shifted to energy efficiency designs. Chapter 4 has discussed various low-power design techniques proposed by the research community, ranging from low to high level and from general to video focused purposes. This chapter will address an energy optimization methodology on video applications with the goal of balancing energy consumption and quality of service based on the functional-oriented reconfiguration, which, as an available mechanism, has shown its simplicity and flexibility on video codec design. On top of that, functional-oriented reconfiguration provides a new notion of energy optimization. Effectively, it can quickly assimilate different new FU-based low-power designs, and adjust a decoding scheme to adapt it to different battery conditions and user preferences. In this chapter, an energy-aware codec manager, independent to platforms and codec standards, is proposed. Besides, problems and objectives of energy optimization and management are first stated in section 5.1. Afterwards, the feasibility of energy control using reconfigurable video coding is discussed in section 5.2. Then, the energy-aware manager will be introduced in section 5.3, and, finally, in section 5.4, the conclusion will be drawn.

5.1. Problems and Objectives of Video Energy Optimization

New trends on video coding design focus on energy efficiency and optimization for longer battery lifetime. The reason for this tendency is that the gradually growing demands for data rates and enhanced functionalities result in much more complex coding algorithms which consume a significant part of the battery energy. Dynamic adaption, either on computing resource allocation or video quality, has become an attractive topic. A considerable amount of research works have shown positive and practical solutions. Recently, MPEG proposed a new ad hoc group known as Green MPEG to address energy issues in decoder standardization [137]. Green MPEG proposes a concept referred to as Green metadata, which could be extracted from either the video encoder or the pre-processor and used at the receive end to reduce the power consumption. The green metadata can be used at the decoding end. An additional power optimization module processes the green metadata information and applies the appropriate operations for power-consumption control. If a feedback channel is available, the metadata could be sent back to adapt the encoder operations [138].

For energy-efficient decoding, Green MPEG distinguishes two sets of Green metadata: Complexity Metrics (CM) metadata and Decoding Operation Reduction Request (DOR-REQ) [138]. A decoder may use CM metadata combined with DVFS technique to scale the voltage and operating frequency for power savings. For example, the CM metadata is embedded into the bitstream and is extracted at the receiver to indicate frame-decoding complexity. According to this indicator, the power optimizer module will set the correct operating voltage and frequency of the CPU, which could reduce the power consumption while guaranteeing the decoding deadline. In a point-to-point application, the remote encoder may use the DOR-REQ metadata to modify the encoding complexity of the encoded bitstream and thus, the decoder can reduce its local power consumption due to the decoding complexity reduction. The proposal of this thesis is similar to the metadata idea. An energy-aware manager detects the remaining energy and takes into account the energy estimation from an energy estimator as a signal to switch from the current configuration to another one. At the same time, the energy-aware manager will ask the encoder to adapt and produce compliant bitstreams.

Green metadata is the additional information that enables energy reduction on the basis of four aspects: decoder power consumption, display power consumption, media selection for joint decoder as well as display power reduction and quality recovery after low-power encoding. In past MPEG meetings, a research group from Samsung proposed a display adaptation for power reduction [139]. Their methodology achieved power saving by scaling the backlight to reduce display power consumption while still producing the same perceived quality. In this section, the problems of the adaptive low-power design will be discussed and according objectives are proposed.

5.1.1. Problem Statement

The processing and transmission of video data occupy a dominant position in multimedia communications research. The intrinsic large amount of information of video data challenges storage, processing and transmission technologies. Video coding technologies focus on how to improve the coding efficiency to reduce the binary rate and, as a consequence, to meet the channel bandwidth. For video compression, its main objective is to approach efficiencies close to the Shannon distortion limit by means of advanced complex algorithms and coding techniques. In addition, with the development of wireless communication technologies, especially the third/four-generation of mobile communication systems, the channel bandwidth has significantly increased. This improvement has made possible to process and transmit multimedia data over wireless channels in real time [140] [141]. However, the high computational complexity involved leads to high energy consumption, which is unacceptable for energy-constrained mobile devices. Limited by energy and bandwidth, real-time video data processing and transmission have three requirements:

- Video data compression ratio need to meet bandwidth limitations.
- Reconstructed video should maintain a certain quality after decompression.
- The energy consumed by video data decoding should be maintained within a certain range to guarantee the battery life time, especially for handheld terminals.

Obviously, these three requirements are mutually restrained. For example, blind compression to meet the channel bandwidth requirement may cause an unsatisfactory distortion. Conversely, too much attention on quality will make the process not to comply with the bandwidth requirement. In addition, although efficient video compression strongly reduces the amount of data transmitted to reduce the transferring energy consumption, on the other hand, decoding highly-compressed encoded bitstreams requires higher computational complexity and more energy supply. Therefore, video coding design for mobile devices aims to overall consider the relationship among bandwidth (binary rate), video quality (distortion) and energy consumption. In this context, energy awareness can be added to a configurable codec. A configurable codec can adaptively evolve its energy consumption state as a tradeoff between video quality and energy.

As introduced in chapter 4, many system level energy solutions have been proposed for energy-aware behavioral adaption and resource control. The core idea of these solutions can be considered to exploit a system "slack" [49]. For example, adaptive voltage scaling is a process variation slack and clock gating is a temporal slack. Discovering new types of slack can introduce new possibilities for energy efficiency improvement. For example, low-power designs that offer execution alternatives can be considered as user experience or execution slacks. Scalable algorithm design can be considered as an example of this kind of slack.

Investigations of energy optimization suffer from two limitations:

- One is that these existing approaches have been designed for specific codec standards or implemented on particular platforms. These methodologies depend too closely on human intuition and specific codec knowledge. They take advantage of the features of their target application, but, at the same time, they are limited in terms of the capacity to develop and extend to new standards. Flourishing markets of mobile devices and video applications may introduce new coding tools with higher complexities when implementing a specific optimization approach. Reconfigurable coding techniques provide a unified methodology for video coding design. They can adjusts different operations modes based on the condition and environment changes.
- Another one is that they are typically restricted to either encoders or decoders. In the literature [142], there are two design paradigms: either to assume the encoder has enough

energy and computing resources or to shift the majority of the workload to the decoder. With the advanced wireless networks, a growing number of applications could be benefited from the use of energy management concurrently in both the encoder and decoder sides. For example, both encoder and decoder are energy constrained when running a video chat application on Smartphones. One difficulty to apply this control globally is that there is always a predefined consistency between the compressed video data to be recognized by decoders. This limit could also be overcome using functional-oriented reconfigurable frameworks. Effectively, new decoders can be built from the decoder description encapsulated into the encoded data.

In different situations, users may have different definitions of optimization. For example, with a fully charged battery and WLAN connection, users may prefer to pursue streaming with high quality. On the contrary, limited by battery capacity or network transmission bandwidth, users are more likely to accept encoded streaming with low quality when the emphasis is put more on the information transmitted. In this context, a flexible solution would be more attractive for users. Combing functional-oriented reconfiguration techniques and energy optimization could be a new direction to achieve flexible and universal solutions on energy efficiency.

5.1.2. Objectives

Wireless communication promotes on-line video watching and thus, to achieve a satisfactory user experience, both video content and network transmission adaptation are needed. This fact indicates that video application design has been shifting lately to systematically consider the energy allocation between computation and communication. This is to say, an ideal design will be able to jointly control the computational complexity parameters during video coding and transmission according to real-time conditions and constraints, such as video content characteristics, network constraints, battery capacity, and distortion requirements. Then, the objective is to implement a framework to manage video energy consumption by means of reconfigurable decoders and adapting encoder parameters and algorithms to extend the battery life time. During execution time, the framework monitors the running conditions to provide online information to conduct a proper energy management.

5.2. Feasibility of Energy Control of Video Coding

The basis of video reconfiguration is the scalability and substitutability of each functional unit.

5.2.1. Features of Video Streaming Computing

Video decoding is a complex application which demands high amount of calculation. Each functional unit requires a certain calculation amount. Moreover, their calculation amounts present big differences according to streaming content, frame type, code rate, quantization parameter (QP), texture complexity, and motion variation.

To show the differences, the HEVC decoder is executed on the PB board, which is one of the two embedded boards used in the previous discussed experiments. The detailed description about this board will be given later in Chapter 7. Figure 5-1 (a) to (d) present the average, variance, maximum, and minimum numbers of computing instructions sampled by a PMC during the decoding of 100 frames for four video contents, namely ducks, harbor, mobile, and, hall encoded at different QPs while keeping other parameters the same for the same content. QP with smaller value indicates finer quantization and higher computing complexity. The number of instructions is employed as an indicator of computing complexity.

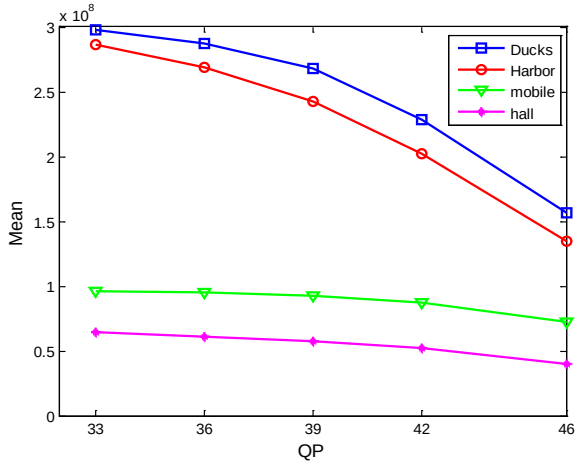


Figure 5-1-(a) Average Number of Instructions

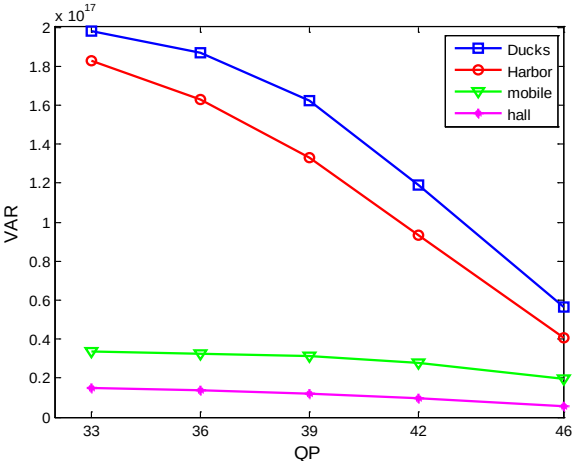


Figure 5-1-(b) Variance Number of Instructions

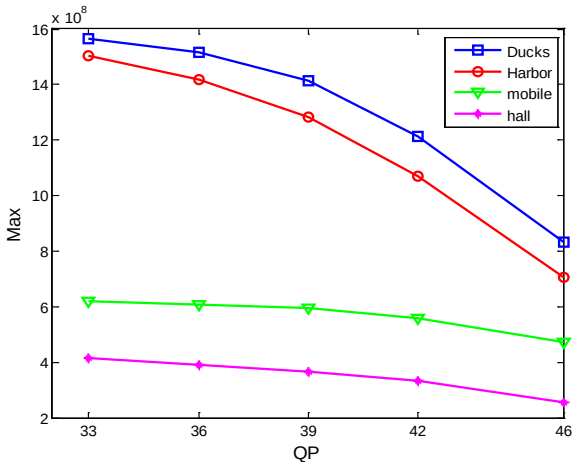


Figure 5-1-(c) Maximum Number of Instructions

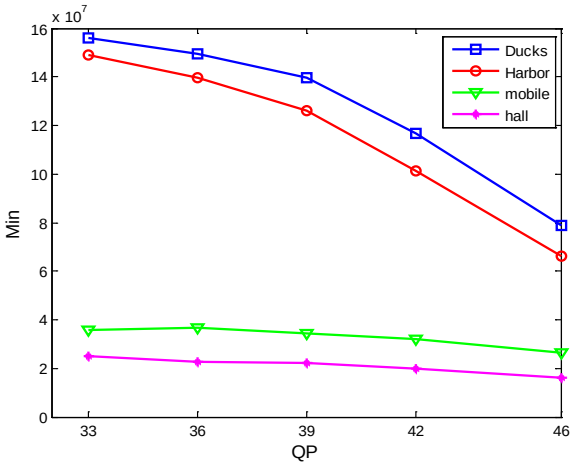


Figure 5-1-(d) Minimum Number of Instructions

Figure 5-1 Instruction Variation of Different Contents Encoded with Different QP

The same as in Figure 5-1, Figure 5-2 (a) to (d) show the instruction variations when the same four video contents are encoded by different frame type combinations, i.e., only I frame, one I frame followed by all P frames, alternant I and P frames, and alternant I, P, and B frames. An I frame is intra-coded, that is, no other frames will be needed as a reference to decode it. P frame stands for predicted picture which refers to previous frame to avoid storing unchanging image information. Likewise, B frame is Bi-predictive picture which obtains data reference from both previous and following frames to further compress data. I frames are less compressible and typically requires higher amount of calculation for encoding and decoding.

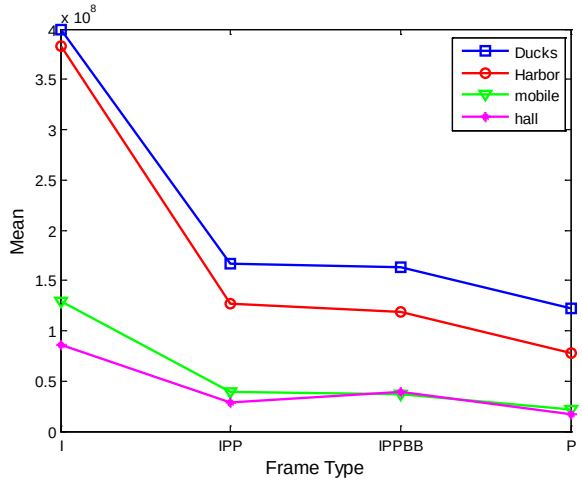


Figure 5-2-(a) Average Number of Instructions

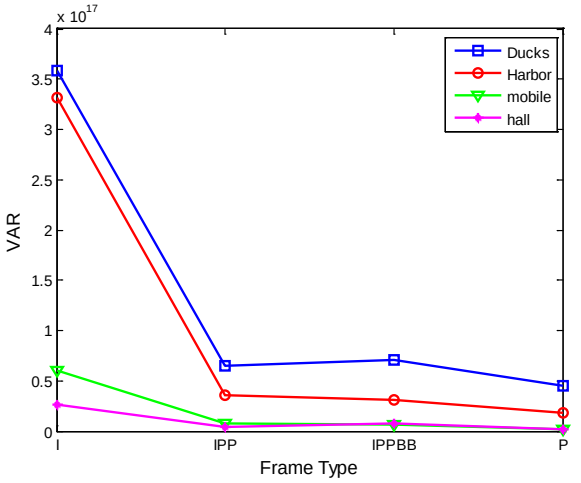


Figure 5-2-(b) Variance Value of Instructions

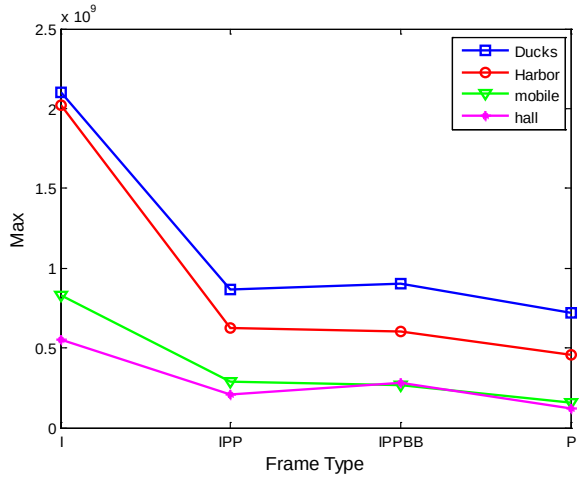


Figure 5-2-(c) Maximum number of Instructions

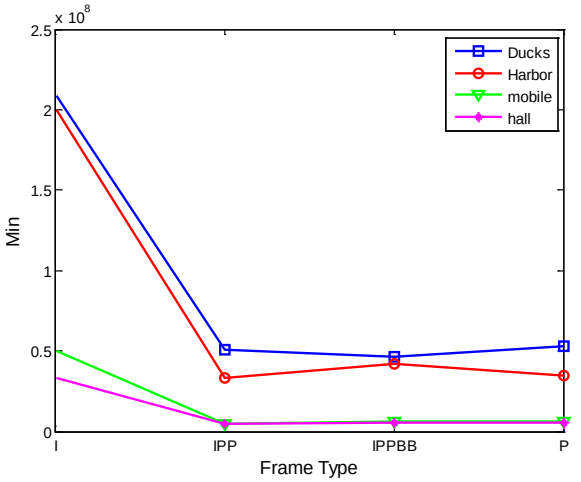


Figure 5-2-(d) Minimum Number of Instruction

Figure 5-2 Instruction Variation of Different Contents Encoded with Different Frame Types

As can be seen from these figures:

- Video decoding is a task which demands great computational complexity. To decode a frame, eight orders of magnitude, i.e., hundreds millions, of instructions are executed.
- Computation complexity varies among different encoding parameters. For example, with the

same content encoded by different QPs, the average, maximum, or minimum values of the number of executed instructions decrease along QP increases. This pattern can also be observed in the case of frame type changes. It is worth mentioning that for a piece of encoded sequence, the QP may be fixed while it can include several frame types.

- Computation complexity varies in the overall decoding process. Comparing the maximum and minimum number of instructions obtained during the decoding of the same sequence, the difference can reach to 17.8 times. In addition, the variance locates in high orders of magnitude indicating a great difference among sampled data. There are several reasons for this variation. First, the different types of frame lead to a great difference on processing modes. Secondly, the complexity of texture and the intensity of motion compensation impact on computation complexity. Generally speaking, more complex texture and more intensive motion lead to more residual and motion information. During the decoding process, more data need to be processed which addresses more amount of computation.

5.2.2. Feasibility of Energy Control of Video Coding

Video streaming with different encoding computation complexities always lead to different energy consumption. Two decoders from MPEG-4 Part 10, namely CBP and PHP are executed to decode four video sequences on the same platform, the one mentioned in subsection 5.2.1. The CBP decoder implements the constrained based profile and the PHP decoder implements the progressive high profile. Each sub-figure in Figure 5-3 stands for the energy consumption of a different video content. As can be seen, there are obvious differences of energy consumption between the two decoders when they are decoding streams encoded with the same sequence.

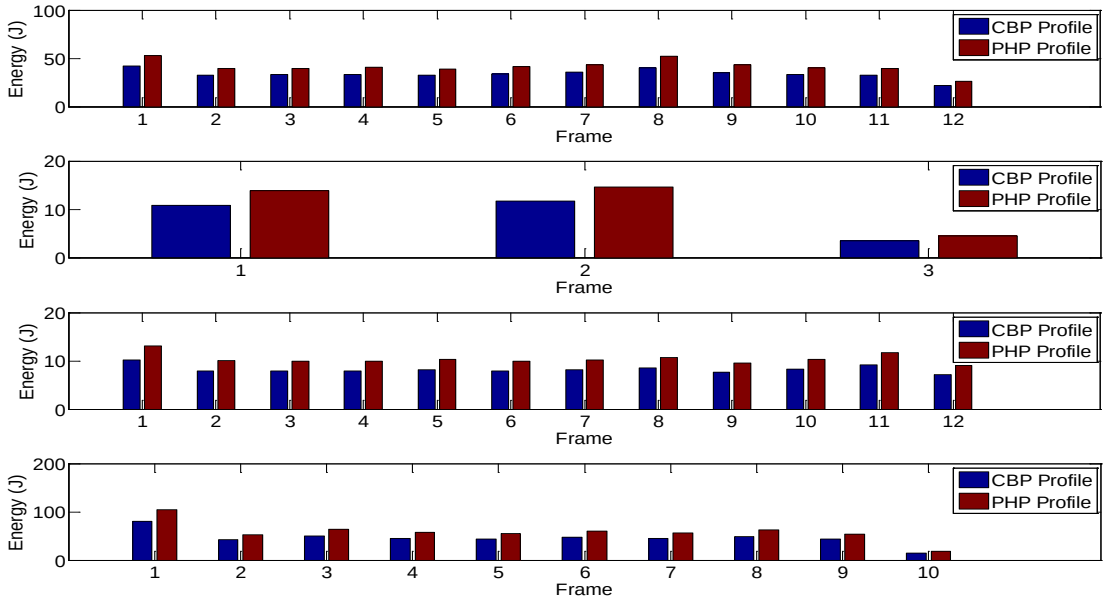


Figure 5-3 Energy Consumption Comparison on Two Decoders

The largest difference between CBP and PHP is the algorithm of entropy coding, which follows the transformation and quantization processing to remove statistical redundancy [143]. The CBP decoder only supports context-adaptive variable-length coding (CAVLC) while PHP decoder supports both CAVLC and context-based adaptive binary arithmetic coding (CABAC). One main idea of entropy coding is to relate the length of codeword to symbol frequency by using variable length coding (VLC). CAVLC [144] uses many VLC tables and selects proper tables according to the context that has been transmitted. CABAC [145] combines an adaptive binary arithmetic coding technique and a well-designed context model with full consideration of statistical characteristics of video streaming to flexibly complete lossless coding under the condition of knowing the model probability distribution of existing syntax elements. Compared to CAVLC, CABAC achieves better compression ratio but introduces more computational complexity.

The aforementioned results suggest that there is a potential space to adjust video coding energy consumption by changing the coding complexity. The amount of energy required in video processing should be properly used rather than be assigned indiscriminately. With the information provided from energy awareness, a decoder can be configured with less complexity or at appropriate spatial and temporal resolutions to yield the best perceptual quality. In this situation, RVC framework shows benefits from its high flexibility. All encoding details are passed to the decoder as side information. By parsing this information, a decoder can replace different FUs, thus, changing the decoding algorithms and energy consumptions.

5.3. Proposal

5.3.1. Energy-aware Framework of Reconfigurable Video Coding

The proposed energy optimization module currently focuses on the decoder end based on the conceptual view of the original RVC framework. It includes two additional units, namely energy-aware manager and PMC-based energy estimator. The whole framework of energy-aware RVC is shown in Figure 5-4. The energy-aware manager provides decisions on how to switch among different decoder descriptions to reduce the energy consumption during the decoding process. And the energy estimator estimates the energy consumption during a certain time interval. The estimation results are passed to the energy-aware manager together with the battery state-of-charge to provide a metric that is used to reconfigure a decoder.

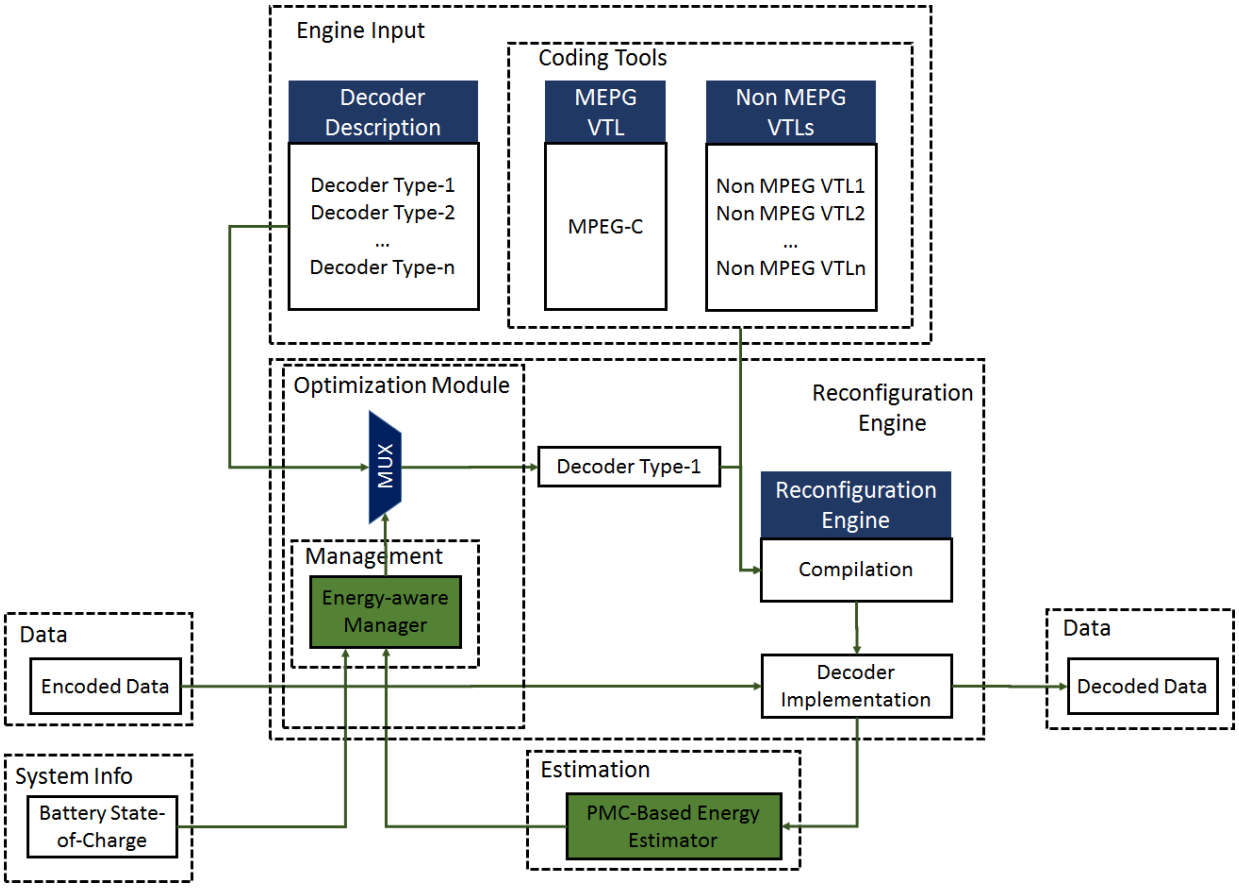


Figure 5-4 Proposed Energy-aware RVC Framework

To complete a reconfiguration, a reconfiguration engine is needed. It has two inputs, a decoder description, which is also known as decoder configuration, and VTLs. Usually, VTLs can be represented in different forms according with the kind of engine that has been employed. The engine interconnects the necessary functional units to form a complete decoder and translate this decoder network to byte code format. Again, this format depends also on the engine type but it is platform independent.

5.3.2. Energy-aware Management

The core question of energy-aware management is how to choose the proper functional units. This problem can be mathematically described as following:

The energy-aware manager assumes a decoder can be structured with a set of FUs, $D = \{fu_1, fu_2, \ldots, fu_p\}$ chosen from a finite set, $FU = \{fu_1, fu_2, \ldots, fu_q\}$. The manager operates the reconfiguration with an operation mode k, selected from a set $\mathcal{K} = \{k_1, k_2, \ldots, k_m\}$, which is featured by a tuple $(\omega_1, \omega_2, \ldots \omega_n \,|\, \sum \omega_i = 1)$ defining the user preference. The operation mode can be extended by including additional dimensions to the given parameters. For example, it can be restricted to the computation complexity, bitrate, and quality. How to choose the operation mode depends on

maximizing the overall system gain, G, and battery life time, T, while ensuring decoder function validity, i.e.,

$$\max G = g_i^{k,j}(D_j, l) \text{ and } \max T = \sum t_i$$
$$s.t. \sum e_i^j(D_j) \leq \alpha E \quad (5\text{-}1)$$

Where,

- g_i^k *is* a model of system gain under the preference mode $k \in \mathcal{K}_i$, being defined as equation 5-2:

$$g^{k,j}(D_j, l) = \frac{\omega_c \times l}{C(D_j)} + \frac{\omega_b \times l}{B(D_j)} + \frac{\omega_q \times Q(D_j)}{l} \quad (5\text{-}2)$$

- l presents the level of energy-saving and larger value means that greater efforts should be carried out to optimize energy unitization on battery life extension;
- $C(D_j)$, $B(D_j)$, and $Q(D_j)$ stand for decoder computing complexity, bitrate, and image quality, respectively; $C(D_j)$ and $B(D_j)$ are inverse proportional to system gain which means a decoder is more expectedly designed with less computational complexities and bit rate but a higher image quality is always targeted, as shown in the proportional relationship $Q(D_j)$ item;
- t_i is the time to decode one frame;
- e_i^j is the energy consumption of decoding one frame;
- E is the total battery capacity;
- $\alpha \in (0,1]$ is a user-defined parameter establishing the limit of the energy budget for video applications.

The model of system gain can be applied to both native-stored bitstreams and on-line streams because the energy optimization module will automatically choose a compatible decoder description to reconfigure the decoder according to the computed system gain. One difference for these two decoding situations is that native decoding does not need to consider the energy impact from data receiving and network condition. However, currently, the system gain is facilitated by only considering computational complexity, i.e., $\omega_c = 1, and\ \omega_b = \omega_q = 0$ for both cases. In addition, the on-line streaming case has a possibility to gain further energy saving by communication with the encoder side. This idea follows the idea of Green Metadata. Figure 5-5 shows an example of this concept for a point-to-point application (e.g., a video conference). In this situation, each terminal device contains a receiver for decoding and has a feedback channel. The receiver sends an energy-

aware message from its local decoder to the remote encoder. This message will inform the remote encoder to adjust its encoding parameters with the concern of the battery life of its client, i.e., the device for decoding the bitstream. The remaining battery life is determined by the client based on the energy consumption of the current representation it is using, which can be estimated by an energy model. Periodically, the energy optimization model computes the energy saving level (ESL) (Step 1 in Figure 5-5). In this thesis, ESL is defined as $\frac{1}{B_r/E_r}$, where B_r denotes the remaining battery budget for video applications and E_r is the current energy-consuming rate which is estimated by the energy estimation model. The expression $\frac{B_r}{E_r}$ shows the time it takes the whole energy budget to be consumed at the current rate, and thus a smaller value suggests a more pressing necessity of energy reduction. That is why ESL is defined as a reciprocal. The ESL is sent by the client to the encoder through the feedback path between the transmitter and receiver (Step 2 in Figure 5-5). The ESL is extracted at the encoding end (step 3 in Figure 5-5) to translate the energy saving request into a new configuration of the encoder (Step 4 in Figure 5-5), so that it can produce a streaming which complies to the ESL (Step 5 in Figure 5-5). In this way, each encoder can adapt the complexity of the encoded stream as a function of the battery level of the other device communicating with it (Step 6 in Figure 5-5).

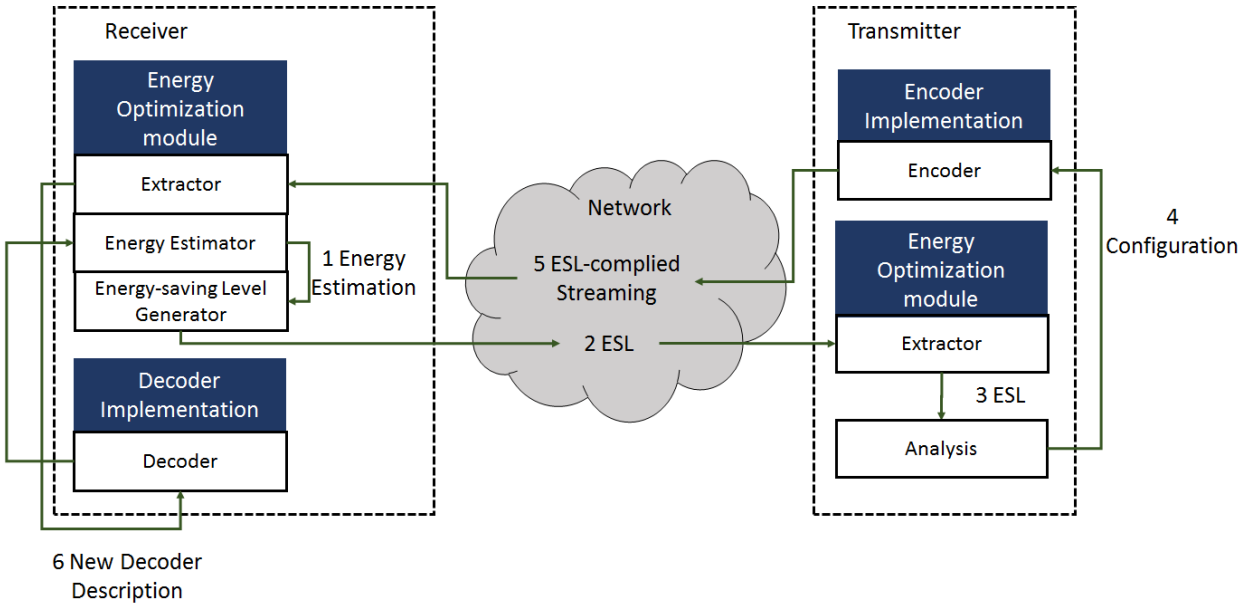


Figure 5-5 An example of the Usage of ESL Information

5.4. Conclusion

Rather than keeping improving the compress ratio and processing ability of video coding, dynamic adaption has been newly considered to meet the requirements of longer battery lifetime. An ideal design will be able to jointly control the computational complexity parameters during video

coding and transmission according to real-time conditions and constraints. This chapter has proposed an energy-aware optimization model to manage and optimize the energy consumption based on the RVC specification. This idea is follow up of the conceptual framework of Green MPEG. The optimization module performs as an energy-aware manager of energy consumption and services at the decoder end. It takes into account the energy consumption ratio to determine how to reconfigure the decoder while providing the largest system gain, i.e., lower computational complexity and bit rate, and higher image quality. In this thesis, the system gain is only determined as computational complexity.

PART D

Chapter 6: Experimental Study-case Infrastructure

Chapter 7: Implementation

6. Experimental Study-case Infrastructure

This thesis focuses on an energy optimization method based on functional-oriented reconfiguration. To evaluate its capacity on energy saving, RVC specifications running on embedded platforms are chosen as the experimental study case. Figure 6-1 shows the infrastructure of the study-case. The aim of this chapter is to introduce each element of Figure 6-1: the reconfiguration engine and development environment of the RVC framework, the PMC tool, the modeling assistant tool, the hardware platforms, and the benchmarks.

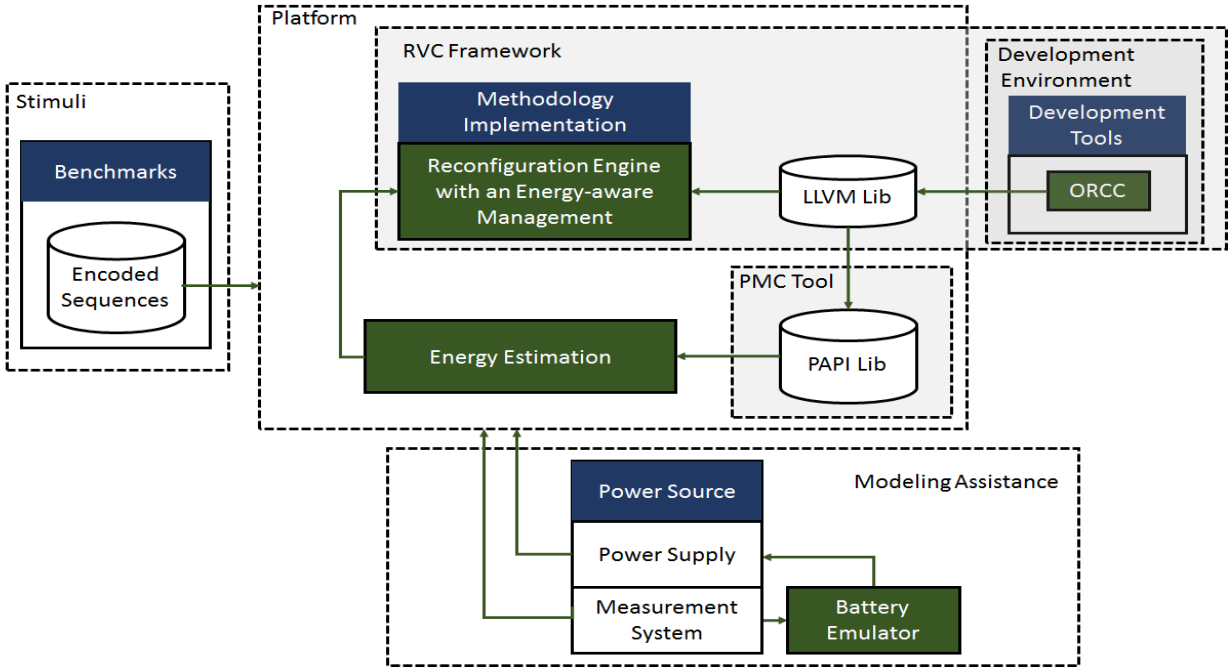


Figure 6-1 Experimental Study-case Infrastructure

6.1. Reconfiguration Engine and Development Environment of RVC Framework

The principles and advantages of the RVC framework have been introduced in previous chapters. In summary, along with the coding development process, the video coding standard has a constant goal: to achieve a bit-rate as low as possible while maintaining the best possible quality. The performance improvements on video coding have been achieved at the expense of additional computational complexity. The continuous evolution of more complex and advanced standards has greatly impeded their efficient specification and implementation. In attempt to facilitate innovation in video coding design and to quickly integrate successful algorithms into existing standards, the reconfiguration mechanism has been introduced into the video field and already shown its features of

flexibility and scalability. It is believed that these features can help to save energy with awareness of energy consumption. In the following, the reconfiguration engine and the development environment used in this thesis are introduced.

6.1.1. Reconfiguration Engine

6.1.1.1. Low Level Virtual Machine

Targeting for code portability, a virtual machine (VM) infrastructure is mandatory for the reconfiguration engine. Performance and portability is a crucial point for VM choice. VMs such as Java and Python have high portability because they are written in standard C and rely a lot on their own libraries. However, at the same time, the VMs for high-level languages, such as JVM for Java or CLR for C#, are more than twice slower than the equivalent C compilation [147]. LLVM is another choice with the efficiency consideration in mind. LLVM was originally developed by Chris Lattner at the University of Illinois, Urbana-Champaign, as a register-based compiler framework [148]. It is implemented in a level lower than typical VMs. Thus, it could provide an infrastructure to easily port any VM to a platform that already supports LLVM.

Essentially, LLVM is a compiler architecture rather than a compiler. It can be considered as a library to help designer to build compilers. The LLVM compilation procedure consists of three stages: high-level language frontend interpretation, intermediate optimization and backend code generation. The frontend converts high-level languages, e.g., the RVC-CAL language in RVC framework, to LLVM intermediate representation (LLVM IR). A frontend only needs to be responsible for syntax analysis, validation, and error diagnosis for the source code. After the fronted translates the source code into LLVM IR, the intermediate optimizer is responsible for LLVM IR optimization. The optimizer is based on the LLVM virtual instruction set and is independent of the compiler frontend and backend. It provides the language-independent optimization and CPU-aimed codes generation. The backend code generator converts optimized LLVM IR into machine codes corresponding to the specified target processor.

LLVM has its own format of IR. All the LLVM IRs are finally compiled to assembly language of the specific platform. Then these assembly codes are executed by the native assembler and linker to generate executable shared libraries. The whole LLVM architecture is shown in Figure 6-2.

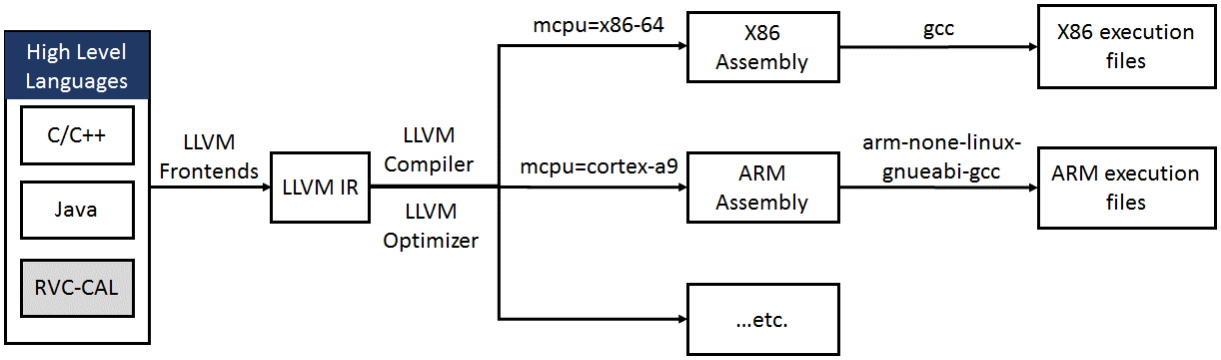


Figure 6-2 LLVM Framework [148]

LLVM IR is the key point for LLVM optimization and native code generation. The LLVM virtual instruction set is close to an assembly language. The abstract mechanism of LLVM IR is an infinite set of virtual registers. They are coded in a Three Address Code (3AC) form and a Static Single Assignment (SSA) form. A 3AC code is an intermediate code which has at most three operands and is typically decomposed into a four-tuple: (Operator, Operand1, Operand 2, result). SSA is a refinement of the 3AC form. SSA form ensures each variable is assigned exactly once. Note that existing variables have different copies. A new assigned variable is indicated by the original name and a subscript, so that every definition gets its own version. The SSA form is independent of high-level programming languages and the target architecture syntaxes. Based on 3AC and SSA forms, LLVM can simplify the value transfer through virtual registers and memory by only using load and store operations. As a consequence, LLVM can produce much faster and more efficient executions.

In addition, due to the standardized LLVM IR, the LLVM optimizer can be reused for any new programming language or device. This is a general procedure without any modification. LLVM optimization is achieved through various passes. In LLVM framework, a pass is an operation on a unit of IR. Each pass is a node to perform a part of the transformations and optimizations work. All the passes make up the compiler. Passes can be classified into analysis, transform, and utility passes. LLVM optimization and conversion work is done by a number of passes. Each pass is a node which is responsible for the optimization or transform. Pass framework has a very good reusability. Developers can choose from existing passes to build their own optimization and transformation. They can also rewrite new passes to implement their solutions. The reason is that each pass is independent, so a new pass does not need to take consideration on the implementation of previous passes. Thus, developers can easily achieve their desired effects. Then, LLVM, being a lower level VM, is easier to port to different OS and hardware architectures.

As discussed above, LLVM fits all the expectations of the RVC framework implementation: high portability and efficiency. Thus, it has been chosen to be employed into the RVC framework to support reconfigurable decoding. Note that in RVC specifications, the coding algorithms are

implemented by the RVC-CAL programming language. The LLVM compiler does not provide the corresponding frontend for the CAL language. Therefore, the LLVM IR is generated by an open RVC-CAL compiler which will be introduced in the next section.

6.1.1.2. Just-in-time (JIT) Adaptive Decoder Engine

Two RVC-specific components support RVC ADMs. The first component consists of VTLs described as an LLVM representation. The second component works as a layer of the LLVM compiler. Ideally, this compiler should be a JIT-implemented compiler. Besides these two components, an LLVM-based Just-in-time adaptive decoder engine (Jade) is developed to manage the description of ADMs and the connection of VTLs to produce decoders. On top of that, a configuration engine in Jade works to select the required coding tools from the VTLs based on the network description of an RVC specification. Then, Jade produces an implementation of the corresponding decoder and a model of execution in an imperative bytecode form according to the LLVM environment. Finally, the LLVM compiler gets and translates the produced bytecode into native code for its execution on the platform. As such, Jade is responsible to dynamically load and execute ADMs, to schedule among different decoder descriptions, and to manage the execution of the final decoder. Therefore, it the proposed energy-aware manager can be implemented as an additional unit to determine when and how to select the different FUs with regard to the current energy awareness.

6.1.2. Development Environment

Table 6-1 lists the necessary tools and libraries of RVC-CAL development environment.

Table 6-1 Summary of Tools and Packages

Tools and Libararies	Functionalities
ORCC	A plugin for Programming languages translation
Graphiti	A graphical tool to build the XDF network
Xtext	A tool for development of programming languages
SDL	An open source library to facilitate multimedia implementation
Cmake	An advanced platform-crossed compilation tool for source code management and compilation
Eclipse IDE	An integrated development environment for RVC-CAL developing
Java-JRE and Java-JDK	Supports for Java running environment and development environment
SVN	Application and projects version control and source codes management

The recommended operating system is the Ubuntu series due to its abundant third-part libraries.

6.1.2.1. Open RVC-CAL Compiler

The RVC-CAL reference language, used for MPEG RVC specification, has been designed as a subset of the CAL language. Compared to CAL, RVC-CAL retains a high level of abstraction to

describe actors, but reduces its expressivity on types, operators, and functionalities that cannot be easily integrated into hardware platforms. Open RVC-CAL Compiler (ORCC) [146] is an open source compiler that can be loaded as an Eclipse plugin to provide a complete Integrated Development Environment (IDE) dedicated for designing, analyzing, and transforming RVC specifications. The most important feature of ORCC is to convert CAL codes to any other programming languages, such as Verilog, VHDL, C/C++, Java, and LLVM IR. This makes that the high-level CAL language can be easily adapted to any platform (Figure 6-3). Note that ORCC is only employed to generate the source code. The assembly or executable code for the target platform are obtained by other tools.

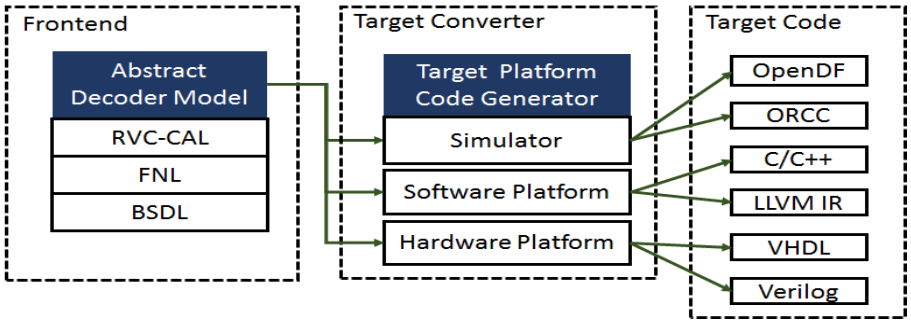


Figure 6-3 ORCC Framework

The converters of ORCC are called backends. In Table 6-2, the existing ORCC backends and their implementation status against the video decoders are summed up.

Table 6-2 RVC Specifications on ORCC Backend [146]

	MPEG-4 Part 2 SP	MPEG-4 Part 10	MPEG-H Part 2	JPEG
C	√	√	√	√
HLS	√	×	×	-
Jade	√	√	√	-
LLVM	√	√	√	-
Promela	√	-	-	-
Simulator	√	√	×	√
TTA	√	×	√	-
Xronos	√	×	-	√

To generate LLVM IRs for RVC-CAL specifications, ORCC provides its embedded specific converter for the RVC-CAL language. This IR converter permits that the body of actions is decomposed in the form of load and store instructions as Static Single Assignment (SSA) form. In addition, the represented actor structure is still equivalent as the original one which contains its name, pattern, a list of actions and the Finite State Machine (FSM). This is to say, at the moment of performing a translation from an RVC-CAL FU into LLVM IR, the original high-level information must be kept. The full translation has two steps: one is a specific ORCC frontend that parses a chosen network and translates it to an ORCC specific IR. The second is a dedicated ORCC backend to

generate the targeted language based on the representation generated from the previous step. For the requirements of Jade, a new LLVM-based backend, named Jade backend, has been developed to produce LLVM IR of the VTLs.

6.1.2.2. Graphiti

Graphiti is a graphical tool infrastructure which provides the graphical representations and editing possibilities [149] [150]. An example of an RVC specification based on Graphiti of Eclipse IDE is presented in Figure 6-4. Graphiti provides a fast and easy network building through visual programming. The decoder description can be hierarchically defined by Graphiti as an XDF network. The vertices of the XDF network have three forms: input port, instance and output port. The instance can be assigned to a sub-network or an actor, which is the minimal unit in RVC framework. The edges between two vertices represent the data flow. They will be instantiated as virtual FIFOs. Graphiti allows a very easy and clear method to view the whole project.

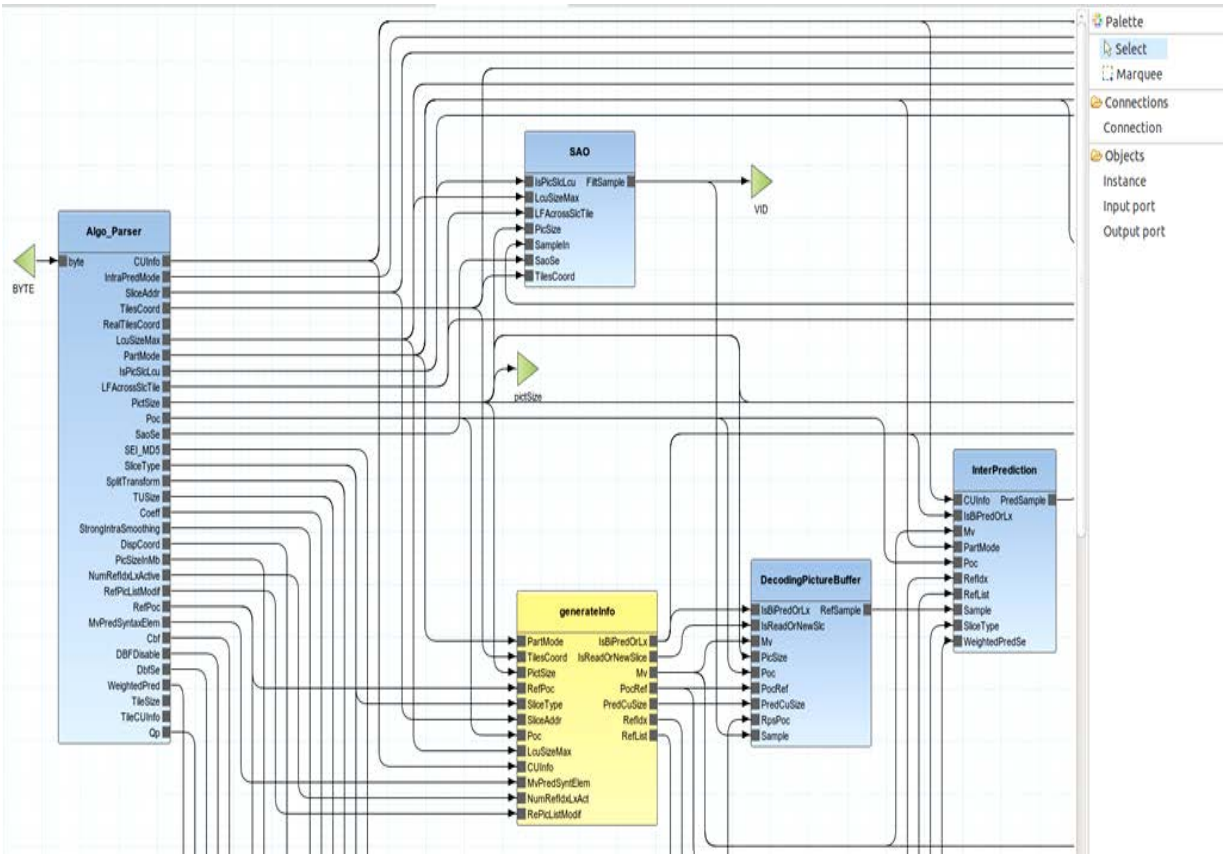


Figure 6-4 The Graphical FU Network Editors in Eclipse IDE

6.1.2.3. Xtext Tool

The Xtext tool is a text editor employed as an advanced RVC-CAL editor [149] [151]. It provides features such as syntax coloring, content assist and code correction for increasing the

efficiency of actor development. Moreover, with ORCC plug-in, the development environment is able to parse all actors and build their interior dependence on-the-fly, which facilities the error detection.

6.1.2.4. Simple DirectMedia Layer

Simple DirectMedia Layer (SDL) is an open source cross-platform library designed to provide a common abstract layer to hardware components via OpenGL and Direct3D [152]. SDL is designed in C language and provides several low level controls on images, audio, and I/O peripherals. It allows developers use the same or similar codes to develop any application on multiple platforms (e.g., Linux, Windows and Mac OS). SDL is currently widely used for developing games, simulators, media player, and other multimedia applications.

6.1.2.5. Cross Platform Make

Cross Platform Make (Cmake) is an advanced platform-crossed compilation tool for C/C++ projects [153]. Cmake uses a simple syntax to describe the compilation process of multiple platforms and can output a variety of forms of make files or project files. The configuration file of Cmake is named as *CmakeLists.txt*, which is a set of Cmake scripts to manage all the components of the project. Cmake does not directly build the final executable file, but it generates the standard build files (such as the *Makefile* for Unix or *projects/workspaces* for Windows), and then it executes the application in accordance with general compilation approaches. Another feature of Cmake is to support directory hierarchies and applications that depend on multiple libraries.

The main goal to use Cmake in this thesis is to compile and install Jade in the target environment. In addition, it is also used to compile the codes that converted by C/C++ backend. Note that Cmake is more like a tool to facilitate source code management and completion rather than a compiler. Cmake is OS-dependent and the calling of a real compiler is embedded into the configuration file of Cmake. For a Windows-based platform, the Visual Studio compiler could be employed as the tool to compile and debug the code, while for Linux-based platforms, GCC-based method is the most widely used tool to obtain the executable files.

6.1.2.6. Eclipse

Eclipse is a free integrated development environment with the original design for java software development. ORCC is implemented in Java as an Eclipse plugin. In this project, depending on user needs, either Eclipse IDE packages for C/C++ developers or for Java developers can be employed. Meanwhile, ORCC requires a Java environment. The Java Runtime Environment (JRE) is required with at least the version 1.6. of Sun's JRE. OpenJDK is recommended on Linux [154].

6.1.2.7. Apache Subversion

Apache Subversion (SVN) is a version control system for open-source projects under the Apache license. It can be used to maintain the RVC-based applications and Jade project [155].

6.1.3. Building Procedure of an Energy-aware Decoder

Figure 6-5 illustrates the building procedure of an energy-aware decoder with the ORCC infrastructure in the context of this thesis.

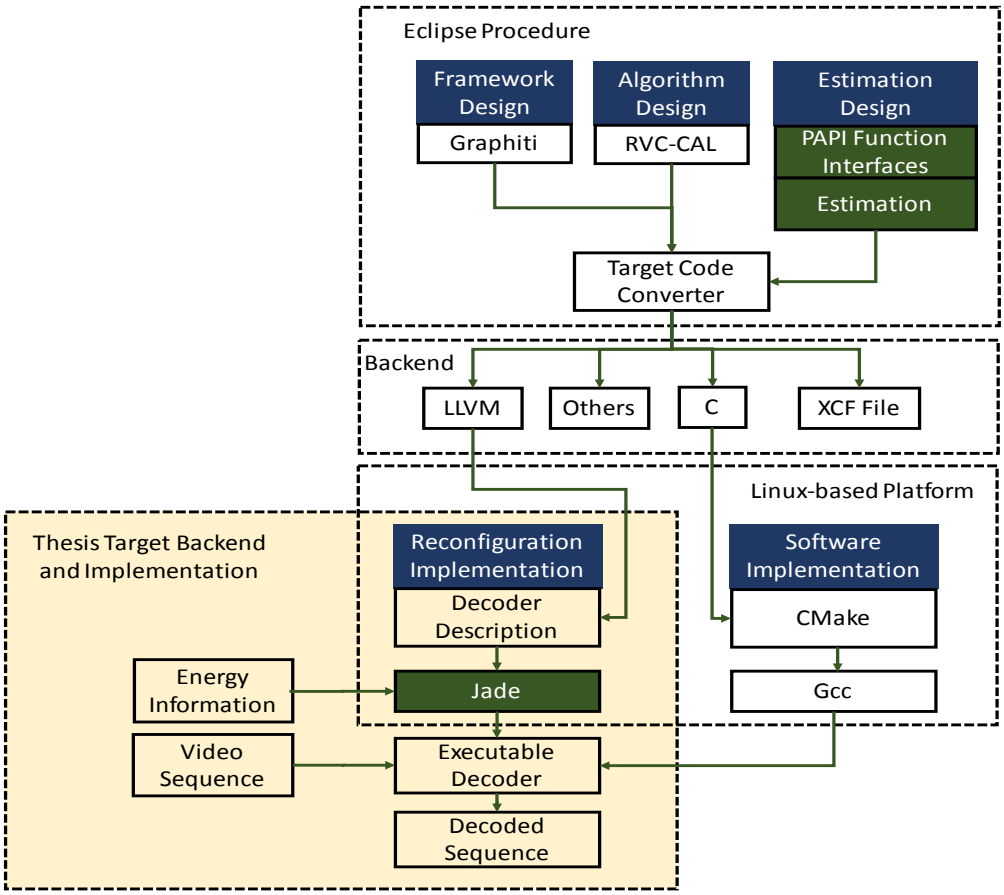


Figure 6-5 General Working Procedure

The ORCC plugin in Eclipse IDE works as the frontend to generate source codes in format of different backends. To use Jade, ORCC provides a specific frontend to generate the LLVM IR. Then, Jade is needed to achieve the on-line reconfiguration. The energy estimation model is included into the original RVC framework, so the converted code does not need to be modified and can be directly used as the input of Jade. Finally, the converted code with the network description will form the executable decoding process to decode the encoded video sequences. Along the decoder execution, the energy information will be provided to Jade to dynamically switch to a new decoder description, and if the feedback channel is available, Jade will pass the energy metadata to the encoder side to adapt the encoding parameters.

6.2. PMC Programming Tool

Performance Application Programming (PAPI) is a third-party tool that provides a methodology to use PMCs for most major microprocessors. PAPI can be divided into two layers as shown in Figure 6-6:

- Framework Layer. The framework layer consists of APIs in low and high levels and machine independent support functions. This abstraction layer provides portability across different platforms. It uses the same routines with similar argument lists to control and access PMCs.
- Component Layer. The component layer defines and exports a machine independent interface to machine dependent functions and data structures. These functions are defined in the components, which may use kernel extensions, operating system calls, or assembly language to access the hardware performance counters on a variety of subsystems. PAPI uses the most efficient and flexible of the three, depending on what is available on the platform.

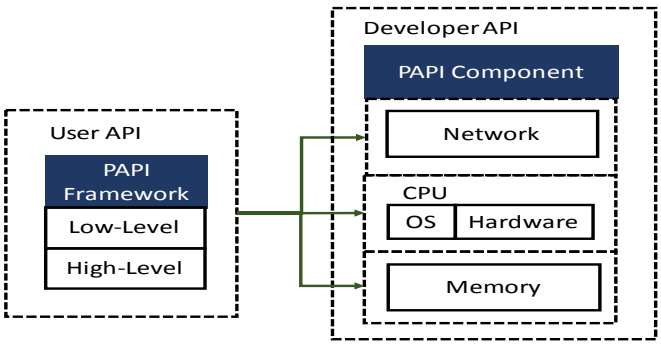


Figure 6-6 PAPI Architecture

PAPI provides two interfaces to the underlying counter hardware: a simple, high-level interface for the acquisition of simple measurements and a fully programmable, low-level interface for user with more sophisticated needs. The high-level interface simply provides the ability to start, stop and read specific events, one at a time. The low-level API of PAPI is employed to manage hardware events for fine-grained measurement and control of the PAPI interface. Using low-level API rather than high-level one benefits from efficiency and functionality. Low-level API is also featured with the ability to obtain executable and hardware information and to set options for multiplexing and overflow handling. The advanced features beyond simple event counting from low-level APIs are:

- Multiplexing. PMCs are generally a scarce resource. There are often much more events of interest than counters to count them on. Multiplexing is one way to relieve this dilemma. When a microprocessor has a limited number of hardware counters, multiplexing overcomes

this limitation by subdividing the usage of counter hardware over time (timesharing) among a large number of performance events. Multiplexing allows more events to be counted than there are limited physical counters. However, when timesharing is employed, the measurement of the existing counters results in some loss in precision [156]. Note that in this case no single event is measured for the full analysis time. When a physical counter is switched to monitor another event, the counting report of the previous event mapped on this counter is estimated by its history information. And, unavoidably, multiplexing incurs a small amount of overhead when switching events. In other words, the more events are multiplexed, the more likely is that the results will be statistically skewed. The amount of time spent in the measured regions should be greater than the multiplexing time slice times the number of events measured in order to get acceptable results. The default time slice for multiplexing is currently set at 100000 microseconds.

- Parallel Programming. PAPI can be used with parallel as well as serial programs. The parallel usage is based on threads. A thread is an independent flow of instructions that can be scheduled to run by the operating system. Multi-threaded programming is one form of parallel programming where several controlled threads are executed concurrently in the program. All threads execute in the same memory space, and can therefore work concurrently on shared data. Threads can run in parallel on several processors, allowing a single program to divide its work among several processors, thus running faster than a single-threaded program, which runs on only one processor at a time. In PAPI, each thread is responsible for the creation, start, stop, and read of its own counters. When a thread is created, it inherits PAPI information or state from the calling thread unless PAPI usage is explicitly specified. PAPI supports threading agnostically by allowing the user to specify the function that returns the current thread ID.
- Overflow. Most processor can generate an interrupt when a PMC exceeds a threshold. PAPI provides the overflow handler to allow the user to take periodic measurements. If a sample value exceeds the predefined threshold, then the interrupt handler will be called by the current context with additional arguments. These arguments will help the user to determine which event causes the overflow and at what location in the source code the overflow occurred.

PAPI only tracks “hardware events”, the occurrence of signals onboard the microprocessor. It does not count system calls, software interrupts or other software events. Currently, PAPI only supports thread level measurements with kernel or bound threads. There are two kinds of events defined in PAPI:

- Preset Events. As a part of PAPI, there is a predefined set of events, namely preset events,

which represent a common implementation. Preset events are a common set of CPU events which are more general, relevant and useful for application performance tuning. These events are typically found in many CPUs that provide performance counters and give access to the memory hierarchy, cache coherence protocol events, cycle and instruction counts, functional unit, and pipeline status. A preset can be either directly available as a single counter or derived using a combination of counters. PAPI defined a set of about 100 preset events for CPUs. However, some ones may be unavailable on any particular platform. A given CPU will implement a subset of those, often no more than a few dozens. PAPI provides interfaces to determine exactly which preset events are available on a target platform. With this predefined set, same source code will count similar and possibly comparable events when running on different platforms. If a programmer chooses to use this set of standardized events, then the source code of PAPI does not need to be changed and only a fresh compilation and link is necessary.

- Native Events. Each processor has a number of events that are native to its specific architecture. There are generally more native events available than the number of them that are mapped onto the PAPI preset events. For some specific components, native events are generally the only available option. PAPI provides access to native events on all supported platforms through the low-level interface. Even if no preset event is available that exposes a given native event, native events can still be accessed directly.

Native events have the advantages that they are comprehensive and cover all the platform available events. To use native events effectively, one should be very familiar with the particular platform in use. In addition, the native event codes and names are platform dependent, so native codes for one platform are not likely to work for other platforms. Although the modification for using native events is easy, in order to make the model be more general, this thesis has chosen to use the preset events. A number of PAPI functions are employed to automatically detect existing events on the target platform from the predefined event list.

6.3. Modeling Assistant Tool

To show the performance of the proposed energy-aware manager, experiments on real platforms with a modeling assistant tool are necessary. This assistant tool includes a measurement system and a battery emulator. A measurement system is needed to carry out the voltage and current measurements for the estimation modeling. Instead of a time-consuming battery charging procedure, a battery emulator is employed [157]. As shown in Figure 6-7, the measurement system consists of a PC-controlled battery emulator connected to a programmable power supply. The chosen power supply is

the Agilent 66321D Mobile Communications DC Source [158], which includes a digital voltmeter to take measurements. The adjusted voltage output is plugged into a DC/DC converter module to raise the regular voltage of the battery to the operating voltage of the embedded platform.

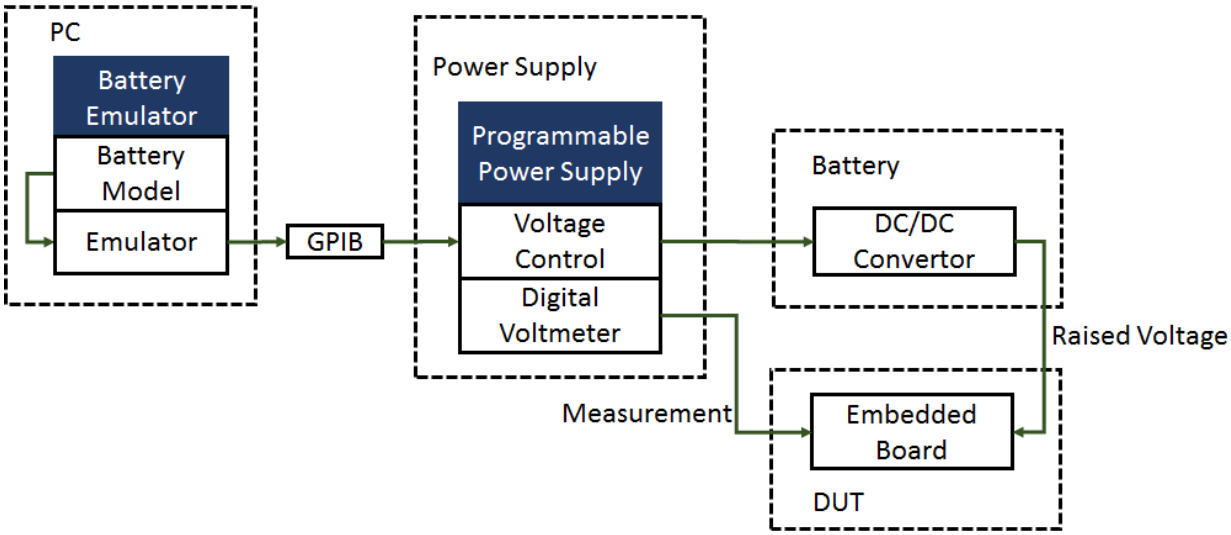


Figure 6-7 Block Diagram of the Measurement System

The battery emulator is a PC-based controller. It must be able to control the power supply to simulate the voltage drop of a battery based on the measurements of the current consumed by the platform. To achieve this, an accuracy battery model must be previously specified.

For any battery, during discharge, current within the battery is carried on by ions moving from negative to positive electrodes through the non-aqueous electrolyte and separator diaphragm. To simulate the battery, a polynomial regression model is chosen. This model in equation 6-1 describes its role as a sum of polynomials, which are dependent on the state of discharge (SoD) of the battery [157]. Q is the battery capacity and $i(t)$ is the current delivered by the battery. V is the output voltage of battery, R_{int} is the internal resistance of battery and c_k are the regression coefficients. Finally, n is the order of the equation. To build the model, only the characteristic parameters of the battery are needed. By using this model, the discharging curve of any battery can be emulated with averaged errors below 2% [157].

$$SoD = \frac{1}{Q}\int_0^t i(t)dt + SoD_0$$
$$V(SoD) = \sum_{k=0}^{k=n} c_k SoD^k - R_{int} i(t) \qquad \text{6-1}$$

As shown in Figure 6-8, a battery simulator based on the Labview software, from National Instrument, has been developed to implement the battery emulator. The estimated voltage is passed to an Agilent power supply via IVI drivers and VISA drivers to simulate the battery.

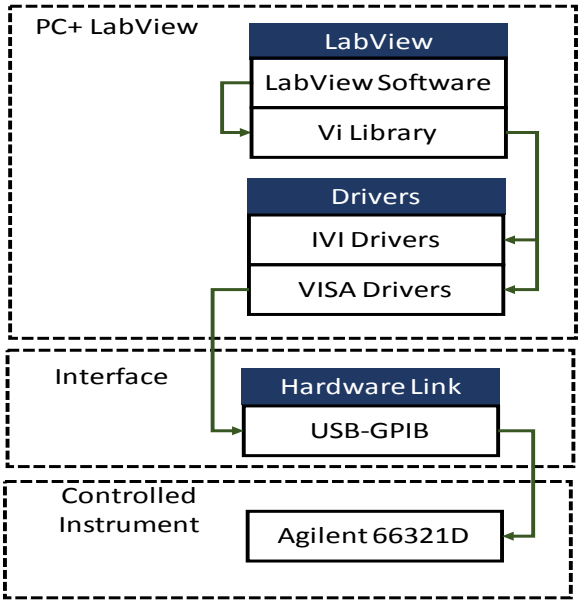


Figure 6-8 Measurement System Layers [157]

The usage of this emulator is quite simple. Figure 6-9 shows the Graphical User Interface (GUI) of the battery emulator. Its usage is described following in detail in Appendix A.

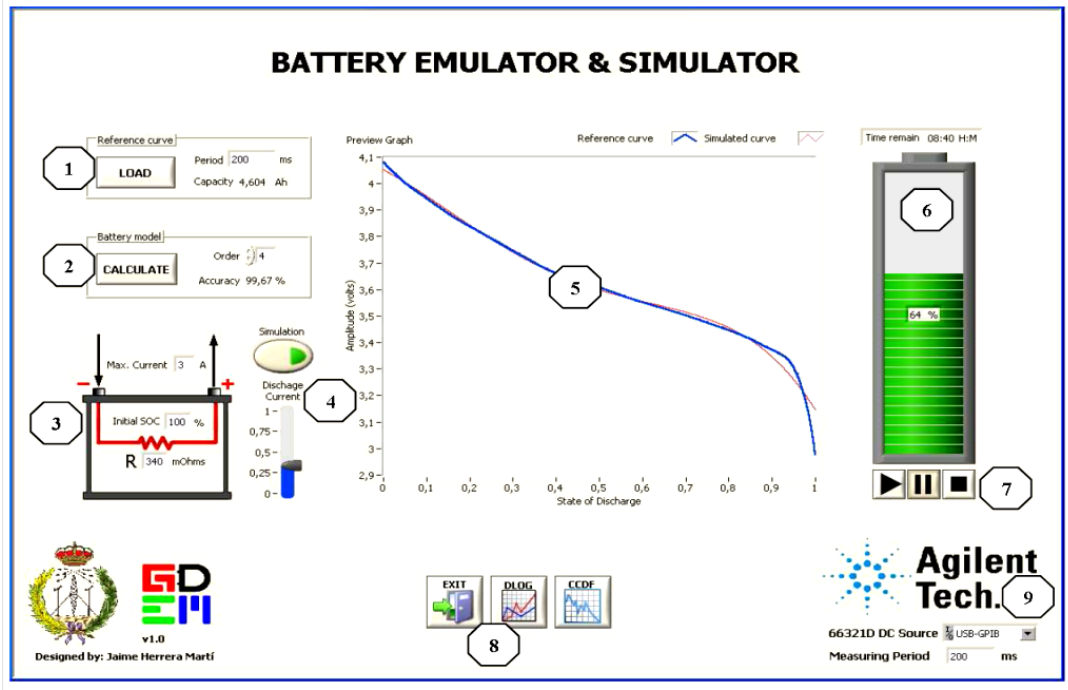


Figure 6-9 GUI of the Battery Emulator and Simulator [157]

6.4. Platforms

In order to get a comprehensive analysis of the energy consumption behavior in CPU and memory components in a controlled environment, two embedded platforms are considered instead of a real mobile device. These platforms have been selected because they both have ARM processors, which come from the same processor architecture family as those processors presented in actual mobile devices. In addition, they are open development platforms with detailed technical information on the chips and a large open-source support community, which facilitates the embedded software development.

6.4.1. Description of the Platforms

In this project, two embedded systems, PandaBoard [159] and BeagleBoard [160], are employed targeting the evaluation of the proposed energy-aware manager. According to their manufacturer, both boards are designed as a prototyping vehicle for mobile application developments. In particular, they are aimed at projects targeting battery-powered mobile devices.

These two boards consist of a rich set of resources to run a wide range of applications. The PandaBoard has been implemented in various versions. The focus of this project is PandaBoard ES which is built with a TI OMAP4460 processor containing two ARM Cortex-A9 cores running at up to 1.2 GHz. Other onboard devices are a C64x DSP, a PowerVR SGX540 GPU, and a DDR2 SDRAM with size of 1 GB. In addition, PandaBoard ES is rich in peripherals: 10/100 Ethernet, 802.11 b/g/n wireless module, Bluetooth, USB 2.0, stereo audio in and out, dual-display output, and expansion headers for I^2C, LCD, and camera. Persistent storage is via an SD card cage. BeagleBoard has a similar architecture but is a weaker version of PandaBoard. The employed version is BeagleBoard XM which is based on a TI OMAP3530 application processor consisting of an ARM Cortex-A8 core clocked up to 720 MHz, a 2D/3D graphic engine featured to a SGX510 GPU, a TMS320C64x+DSP processor core, and 512 MB low-power LPDDR RAM memory. The connectivity and peripherals are similar to those of the PandaBoard except the supports of wireless and Bluetooth modules and an HDMI interface.

Table 6-3 describes the main features of them.

Table 6-3 Platform Features of PandaBoard and BeagleBoard [159]

<table>
<tr><td>Component</td><td colspan="4">Features</td></tr>
<tr><td rowspan="3">Processor</td><td colspan="2">PandaBoard ES (OMAP 4460)</td><td colspan="2">BeagleBoard XM (OMAP 3530)</td></tr>
<tr><td colspan="2">Two ARM Cortex-A9 cores
Two ARM Cortex-M3 microprocessor</td><td colspan="2">One ARM Cortex-A8 core</td></tr>
<tr><td colspan="4">Digital Signal Processor (DSP)
Image and Video accelerator
Image Signal Processor (ISP)
2D/3D graphic accelerator</td></tr>
<tr><td rowspan="3">Memory</td><td colspan="2">PandaBoard ES</td><td colspan="2">BeagleBoard XM</td></tr>
<tr><td colspan="2">1 GB DDR2 SDRAM</td><td colspan="2">512 MB LPDDR RAM</td></tr>
<tr><td colspan="4">SD/MMC Card Cage</td></tr>
<tr><td rowspan="3">Connector</td><td colspan="3">Video</td><td>Audio</td></tr>
<tr><td>PandaBoard</td><td>BeagleBoard</td><td rowspan="2">3.5mm, L+R Out
3.5mm, Stereo In</td></tr>
<tr><td>High-Definition Multimedia Interface (HDMI), Type A</td><td></td></tr>
<tr><td rowspan="3">Communication Interface</td><td colspan="2">PandaBoard</td><td colspan="2">BeagleBoard</td></tr>
<tr><td colspan="2">2.4 GHz 802.11 b/g/n WIFI
Bluetooth V2.1</td><td colspan="2"></td></tr>
<tr><td colspan="4">10/100 Ethernet</td></tr>
<tr><td>USB Port</td><td colspan="4">USB 2.0 OTG Port
USB Host Ports</td></tr>
<tr><td>Expansion</td><td colspan="4">General Purpose Expansion (I2C, USB, MMC,DSS...)
Camera Expansion Connector</td></tr>
<tr><td>Debug</td><td colspan="4">14 Pin JTAG
UART/RS-232 Port
GPIO Pins</td></tr>
<tr><td>User Interface</td><td colspan="4">Switches
Reset Button</td></tr>
</table>

6.4.2. PMCs on ARM Platforms

The Performance Monitoring Unit (PMU) of Cortex-A9 processor provides six PMCs to monitor the events of processor and memory components, 2 of them can be used simultaneously. In Cortex-A8 processor, there are 4 PMCs and 2 of them can be simultaneously used. Both processors provide a coprocessor (CP15) to manage PMCs and their control registers [175][177]. The purpose of CP15 is

to control and provide status information for functions implemented in the processor. Its main functions of the system include the controls and configurations of the overall system, the configurations and managements of the cache and memory units, the preloading engine for L2 cache, and the system performance monitoring, which is the function used in this thesis. Unfortunately, on ARM/Linux platforms, these PMCs are restricted to access from the user-space by default. Trying to access PMCs from the user space will cause an exception of illegal instruction violation. This problem can be easily solved with a user-written driver to connect the user-space functions to PMC operations. Moreover, it is necessary to bind the monitored PMC events to one thread to evaluate its performance without the interference from other threads. If PMCs are configured directly through the configuration register, they will monitor all the occurrences of the interested events but not distinguish which thread operations triggers them. Thanks to the PAPI tool, the PMC usage has been facilitated. PAPI tool and its features have been introduced in section 6.2 in detail. The Table 6-4 (a) to (c) below list all the available preset events on PandaBoard and BeagleBoard.

Table 6-4 Introduction of the Common Preset Events

Table 6-4 (a) On Both Prototype Boards

Events		Events Description
Cache Access	PAPI_L1_DCA	L1 data cache accesses
	PAPI_L1_DCM	L1 data cache misses
	PAPI_L1_ICM	L1 instruction cache misses
Conditional Branching	PAPI_BR_MSP	Conditional branch instructions mispredicted
	PAPI_BR_INS	Branch instructions
Instruction Counting	PAPI_TOT_INS	Instructions completed
	PAPI_TOT_CYC	Total cycles
Data Access	PAPI_SR_INS	Store instructions
	PAPI_LD_INS	Load instructions
TLB Operations	PAPI_TLB_DM	Data translation lookaside buffer misses
	PAPI_TLB_IM	Instruction translation lookaside butter misses

Table 6-4 (b) On PandaBoard

Events		Events Description
Floating Point Operations	PAPI_FP_INS	Floating point instructions
Instruction Counting	PAPI_HW_INT	Hardware Interrupts
	PAPI_TOT_IIS	Instructions issued
	PAPI_VEC_INS	Vector/SIMD instructions

Table 6-4(c) On BeagleBoard

Events		Events Description
Cache Access	PAPI_L1_ICA	L1 instruction cache accesses
	PAPI_L2_TCM	L2 total cache accesses
	PAPI_L2_TCM	L2 total cache misses
Conditional Branching	PAPI_BR_TKN	Conditional branch instructions taken
Instruction Counting	PAPI_STL_ICY	Cycles with no instruction issued

6.4.3. Component Classification and Energy-related Events

Most embedded systems are single-board computers (SBCs) which are complete computers built on a single circuit board. There are no exact design standards for an embedded system. An embedded system consists of various physical components and can be easily extended. General speaking, those devices of an embedded system can be divided into five main categories: computation, storage, communication, buses and I/O (as shown in Figure 6-10).

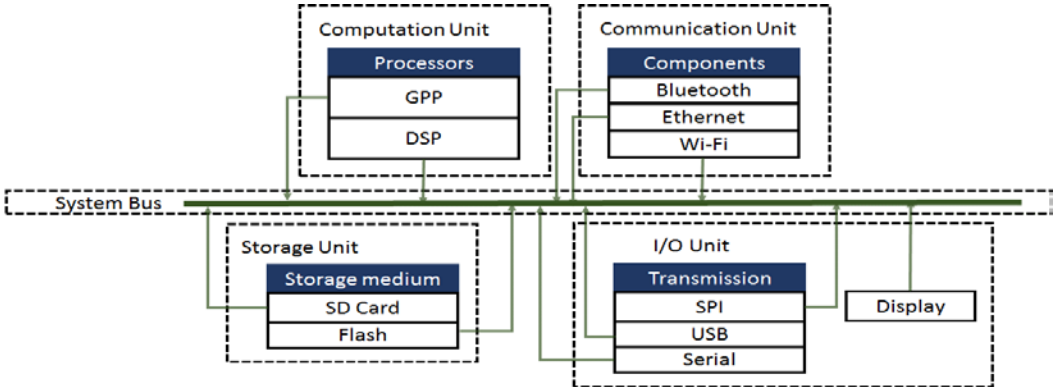


Figure 6-10 High-Level Overview of the Embedded System Architecture

Each of these categories has its unique functionality that cannot be replaced by another one. Therefore, each category can be considered as an independent component which deserves a specific analysis of its energy consumption. The energy consumption of each component can be obtained by observing the representative events. Note that the representative events differ from each category in the embedded system.

6.4.3.1. Computation

As the most complex component, there are many details need to be considered for the computation unit. Processors have been implemented with various hardware architectures, instruction set, pipeline depth, specific acceleration circuits, and instruction cycles. These variances have

different contributions to the whole energy consumption. The execution of each instruction consumes a baseline energy. Additional units such as branch prediction, cache, and pipelining are implemented to accelerate the processing speed and they also contribute to the energy consumption. Usually, more typical events need to be distinguished from each processor to make higher accurate energy estimation.

6.4.3.2. Communication

A communication component may change among several states to complete a communication operations. The transfer and traffic are the two most important states. The energy of the traffic state can be estimated by counting the number of time intervals in which the component stays in this state and the energy of the transfer state can be predicted by the number of transferred bytes.

6.4.3.3. Storage

The total energy can be predicted by the amount of transferred data. In other words, the prediction is computed by directly multiplying the transferred size or the bandwidth and the access times. Usually, a complex storage component may have several states that consume different amount of energies. A more accurate model also considers the energy consumption during the state transitions. In embedded systems, SD card or flash are usually used as the storage component which are not as complicated as the hard disc. Thus it can be assumed that each access of the SD card or flash has the same energy consumption, and their energy is mainly related with the access times, which can be estimated through the number of L2 data cache misses.

6.4.3.4. Buses

Since embedded systems have a fixed bus frequency, the energy consumption of buses can be estimated by the number of bus activities and the bus width.

6.4.3.5. I/O devices

Most of the I/O devices have several states which consume different amount of energy. Their energy consumption mainly depends on the number of I/O requests and the according state.

To simplify the work, this dissertation mainly considers the computation, memory and main peripheral units. All the units can be estimated through a PMC-based approach, which facilitates the model implementation. Note that the energy consumption of the peripheral components is presented by their interface operations, and each interface is assumed to consume the same amount of energy.

6.5. Benchmarks

Four representations of decoders standardized by MPEG RVC are employed as the test bench: the Simple profile (SP) from the MPEG-4 part 2 standard [161]-[163], the Constrained Baseline Profile (CBP), the Progressive High Profile (PHP) from the MPEG-4 part 10, which is also named as AVC/H.264 [164]-[166], and the HEVC Main Profile (MP)[167]. All the sequences come from the JVC conformance sequences. They are widely used in research and display a wide variety as far as the amount of spatial detail and movement concerns. The following is a brief introduction about these four decoders. They belong to two video coding standards: MPEG 4 and HEVC.

6.5.1. MPEG-4

MPEG-4 is a video coding standard designed for rich multimedia. It provides various codec tools with excellent compression capability. MPEG-4 uses a number of new technologies such as shape encoding and adaptive discrete cosine transform (DCT) to greatly improve the coding efficiency. MPEG-4 consists of several standards which are termed as different parts. Part 2 and Part 10 are employed in this project.

6.5.1.1. MPEG-4 Part 2

MPEG-4 Part 2, also known as MPEG Visual, is a DCT based standard defined to provide higher compression efficiency with new compression tools such as combination of motion-compensated prediction and scalar-quantized DCT coefficient coding [162]. Video applications are ranged from low-quality and low-resolution requirements to high definition preference, thus, video standards are grouped with a set of capabilities in a manner appropriate for various applications. Each profile is declared with different code in the encoder to allow a decoder to recognize the applied constraints and requirements to correctly decode the stream. MPEG-4 Part 2 has 21 profiles ranging from simple one to advanced one. Among them, the simple profile (SP) has been implemented in RVC framework. SP is designed to applications that constrained by low bit rate and low resolution conditions.

6.5.1.2. MPEG-4 Part 10

MPEG-4 Part 10 was jointly developed by ITU-T and MPEG, and is commonly referred to H.264 or advanced video codec (AVC). It is based on the advantages of the previous standards such as H.263+, MPEG-4 Part 2, and integrates their successful experience. It still uses the traditional hybrid coding framework but introduces new features, such as multiple reference frames, multi-block types, integer transform, intra-prediction, and other new compression technologies, to achieve significant improvement of coding efficiency. AVC offers the lowest bitrate for a given quality among any previous codec. The performance improvement leads AVC to become the most widely used

standard for video products and services where quality and compression efficiency are paramount. For example, digital television broadcasting, video real-time communication, and network video streaming transmission.

H.264/MPEG-4 AVC has been a promotional technology for digital video in almost every area and has substantially displaced the older standards within their existing application domains. However, the coding method of AVC relies on the fact that the computational power and memory have much progressed on the latest generation of high-performance hardware. With the consideration of hardware cost and power consumption, AVC also offers different profiles to control the degree of sophistication in codec. Profiles of AVC can be generally divided into baseline, main, and high groups. Each group includes several profiles sharing some common features. Two profiles have been implemented in RVC framework:

A. Constrained Baseline Profile

Constrained Baseline Profile (CBP) shares the common features between baseline, main, and high profiles. It is primarily designed for low-cost applications or additional fault-tolerant applications, such as video conference, and mobile video.

B. Progressive High Profile

As a member of the high profile group, progressive high profile (PHP) supports all types of frame and offers best compression ratio but without supporting the field coding features. It is typically used for broadcast.

6.5.2. HEVC

With the increasing diversity of multimedia services, users have become more demanding with regard to broadcast resolution and video experience. Moreover, with the growing popularity of electronic mobile devices such as smart phones and tablets, the traffic and transmission needs are giving rise to increasing challenges on the networks. H.264/AVC has been proved to be insufficient for data compression of high definition sequences. The coding efficiency needs to be further improved. Therefore, High Efficiency Video Coding (HEVC) was proposed essentially to address these issues. HEVC is a successor of H.264/AVC and particularly focuses on two key points: one is to provide higher compression quality and video resolution and another one is to facilitate the parallelism for multi-core architectures. HEVC doubles the data compression ratio compared to H.264/MPEG-4 AVC at the same level of video quality but the computational complexity increases from two to ten times [167]. HEVC partitions picture into coding tree units (CTUs) to achieve a better

parallel processing. The size of the CTU is selected by the encoder based on the sampling schema and syntax elements.

The first version of HEVC standard was completed and published in early 2013. Its main profile of HEVC has been implemented in RVC framework. In this work, it is employed as the benchmark for HEVC standard.

6.6. Conclusion

This chapter describes the whole infrastructure of the study-case used in this thesis work. It firstly introduces how the reconfiguration video coding engine works as well as the development environment including the necessary tools, libraries and working procedures to implement an energy-aware manager embedded in video decoders. Afterwards, energy modeling related tools, i.e., the PMC programming tool and the modeling assistant tool are introduced. Then, a study on the experimental platforms used in this thesis has been presented with descriptions of platform features, available PMCs, and a simple component classification. Finally, four benchmarks are introduced. They are four typically used video coding profiles specified by RVC, corresponding to the simple profile of MPEG-4 part 2, the constrained baseline profile and the progressive high profile of AVC/H.264, and the main profile of HEVC.

7. Implementation

Chapter 6 has introduced the whole system structure of the study case. In this chapter, the implementations related to energy optimization and management are described in detail. To enable the management, an estimation model, which is based on the PMC mechanism, is needed to provide energy awareness. Thus, the PMC control tool, PAPI, should be inserted into the original decoder. The better solution is to integrate PAPI APIs into the ORCC framework. In this way, PAPI APIs can be modularly operated and more importantly, PAPI APIs can be automatically included into the generated backend code, keeping the platform independence of ADMs. Besides energy awareness, the energy-aware manager, which takes charge of making reconfiguration decisions, is the core design of this implementation. This manager accurately and efficiently makes decisions to reconfigure the decoder and inform the encoder to adapt its encoding parameters. A accuracy means, on one hand, that the current capacity of the battery and the rate of energy consumption are accurately predicted, and, on the other hand, users' preferences are accurately considered by the manager. Efficiency means that the manager does not introduce an overhead spoils the decoder performance.

7.1. PAPI Integration

The predicted energy consumption is an important reference for the proposed energy-aware manager to make the optimization decisions. As discussed in section 2.2.4, the modeling method based on performance monitoring counters is chosen in this thesis work due to its simplicity and generalization. To facilitate the configuration and usage of PMCs, PAPI is employed in this work. Readers are kindly recommended to read section 6.2 to get more details of this tool. In this section, the integration of PAPI APIs into the ORCC framework is described. In fact, since ORCC can convert the RVC-CAL specifications into other backend formats, such as C/C++, VHDL and LLVM IR, these PAPI APIs can be manually inserted into the generated code after converting. In doing so, all the inconvenience will be brought to designers. First of all, not only those source files used by PAPI APIs need to be modified, but also the configuration file needs to be modified in order to include the correct PAPI library path. And more importantly, the modification procedure must be repeated any moment the RVC-CAL representation requires any tiny adjustment, even if those ones independent from the PAPI APIs. This procedure is time-consuming and loses the benefits of the abstraction provided by ADMs. Therefore, it is worthy of considering to directly integrate PAPI APIs into the original RVC-CAL framework.

7.1.1. Integration

7.1.1.1. Framework with PAPI Integration

One of the most important features of the CAL language is the concept of Dataflow Process Network (DPN) [173]. DPNs provide an efficient modular method to build systems using unidirectional FIFOs instead of synchronization primitives such as mutex or semaphores [174]. An actor of an RVC-CAL specifications can be considered as a particular case of a DPN. Based on the concept of DPNs, it is easy to integrate PAPI functions into the RVC-CAL structure to take PMC event samples during decoder execution. To do so, start and stop signals are needed to control PMCs counting. Different actors can trigger and send these signals. Figure 7-1 shows an example of a decoder network that has been modified to include PAPI function calls for a case in which the overall decoder performance wants to be monitored using the PMCs. As can be seen, the start signal is sent at the decoder initial state to enable the PMCs to take samples. A stop signal sent from the display actor stops those PMCs. In this case, PMCs periodically (e.g., after decoding every 25 frames) sample events occurred from all the fired actors during the period. Afterwards, PMC statistics are stored.

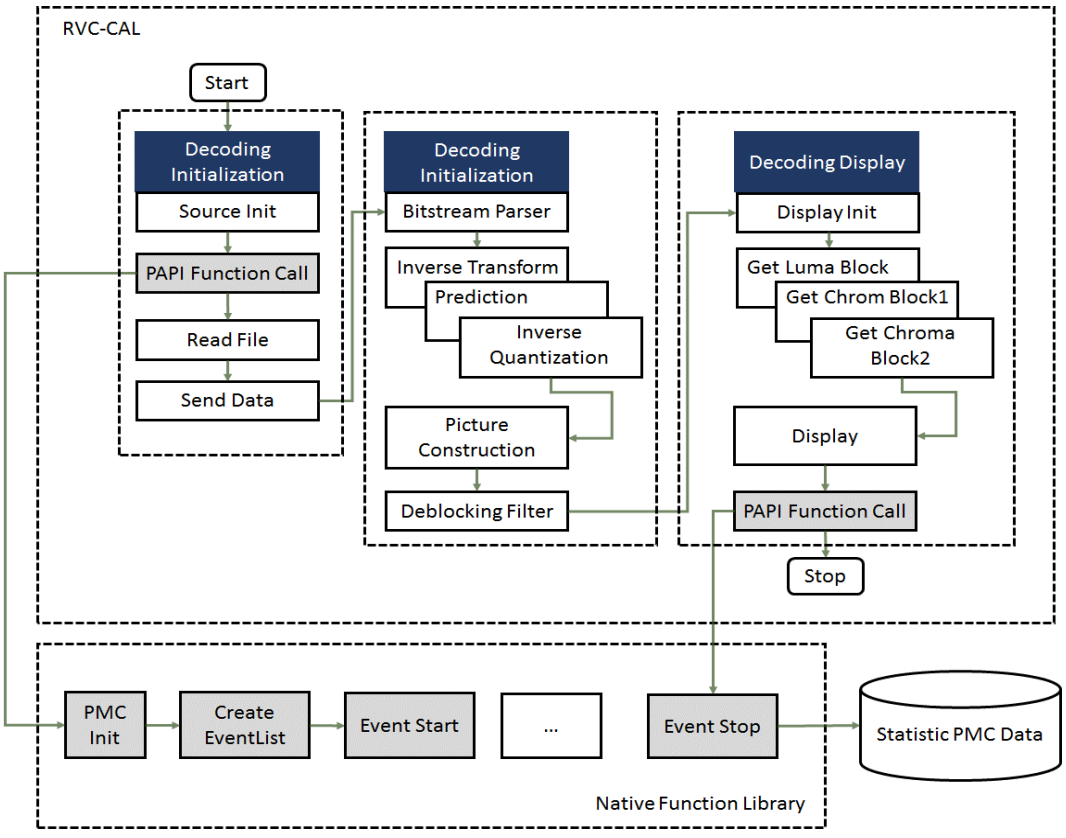


Figure 7-1 PAPI Tool Integration

7.1.1.2. Integration Primitives

ORCC provides a mechanism to call native functions from a local library once those functions are declared in ORCC code as native ones. Thus, instead of translating PAPI functions from C language to RVC-CAL language, only the PAPI interfaces are needed to be inserted into the RVC framework. As a consequence, the PAPI interfaces are automatically merged into any code generated by various backends, such as C/C++ and LLVM IR.

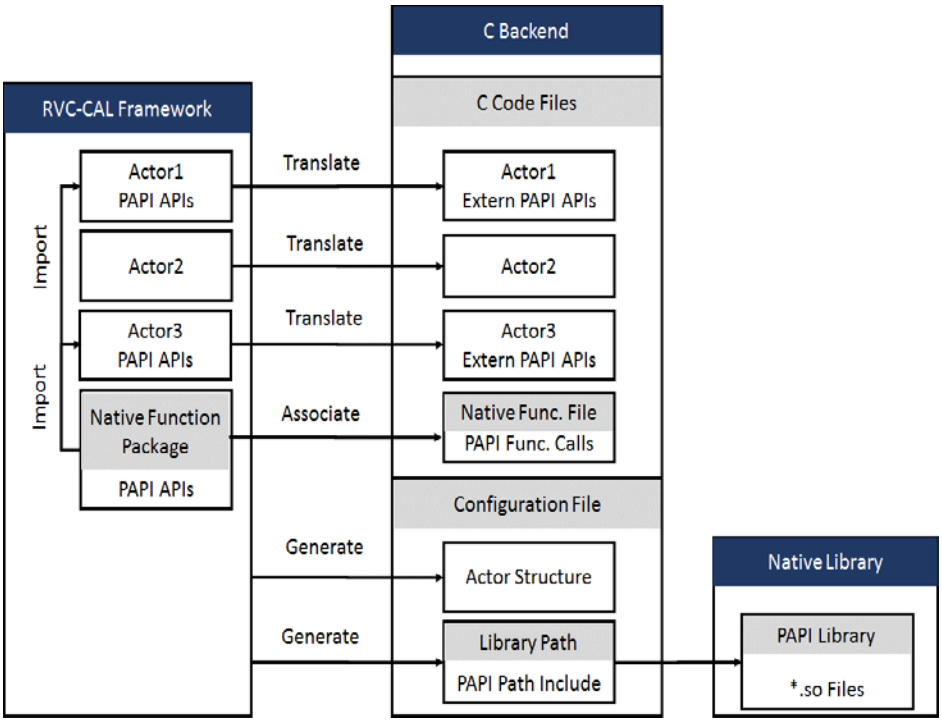


Figure 7-2 Native Function Mechanism

This mechanism of native functions can be shown in Figure 7-2. At the RVC-CAL framework, PAPI API functions are marked as native functions (with the label "@native") in a single package. This package can be imported into any actor where the PAPI API functions are called. Then, the modified RVC-CAL representation will be translated into a target backend, e.g., C backend. Automatically, actors are translated into "C" files. On the other hand, native function are associated to actual C implements of the PAPI API functions. Header files of the PAPI function calls are included into the automatically generated C files. All PAPI API functions are referenced as external functions in each actor. In addition to translate the "C" files, ORCC also generates a configuration file which indicates the structure of all the "C" files and, more importantly, configures the paths to all packages and libraries.

The following primitives are PAPI API functions that have been integrated into the RVC-CAL framework:

```
@native procedure event_init() end
```

- A procedure to initialize the structures employed to monitor the performance counters. In this procedure, the PAPI library will be first initialized. This initialization checks memory states, hardware support, and system call status. It is known that there is a limitation of the physical number of PMCs that can be simultaneously employed, thus, in order to monitor more events, a multiplexing scheme will be enabled and initialized next. At last, the thread support will be initialized, which will make PMCs to monitor and record only the events caused by the function with specific thread ID.

```
int ( size=32) eventCodeSize = *;
@native procedure event_create_eventList(    int eventCodeSize,
                                             int(size=32) eventCode[eventCodeSize],
                                             int threadID) end
```

- A procedure to add a list of events defined in eventCode array to the PAPI event set. This set will be used to configure PMCs control registers to make PMCs monitor and record corresponding events. In this procedure, an empty event set will be first created. The size of the PAPI event list is defined by the variable eventCodeSize, and note that the event set should be assigned by the thread labeled as threadID. This event set can be bounded to a component, which is set as CPU by default. Note that new components can be created. If the multiplexing mode is initialized, the standard event set need to be converted to a multiplexed one. After checking the availability and validness, each defined event will be added into the event set.

```
@native procedure event_start(int threadID) end
```

- This procedure will start monitoring the previously created event set assigned to the thread labeled as *threadID*.

```
int ( size=32) eventCodeSize = *;
@native procedure event_stop (    int eventCodeSize,
                                  int(size=64) PMC[eventCodeSize],
                                  int threadID) end
```

- This procedure will stop monitoring the previously created event set assigned to the thread labeled as *threadID* and store the sample values of each PMC in an array named *PMC*. These values indicate the numbers of occurrences of the monitored events and will be used for energy estimation.

```
@native procedure event_destroy_eventList(int threadID) end
```

- A procedure to delete the event set assigned to the thread labeled as *threadID*. In this procedure, all the events are removed from the PAPI event set and their profiling is turned off. The memory associated with the event set will also be de-allocated.

7.1.1.3. Integration Implementation

There are two ways to insert PAPI primitives. One is to take the PAPI primitives as a new action. Another one is to insert those primitives into existing actions.

A. As New Actions

Figure 7-3 shows an example of integrating PAPI primitives into an RVC-CAL actor as new actions.

```
init: action ==>
do
   event_init(THREAD_ID);
   event_create_eventList(eventCodeSetSize, eventCodeSet, THREAD_ID);
   event_start(THREAD_ID);
end
papi_done: action ==>
do
   event_stop(eventCodeSetSize, PMC, THREAD_ID, PAPI_TITLE);
   event_destroy_eventList(THREAD_ID);
   PAPI_LIST := 0;
end
schedule fsm INIT:
  INIT          (init)          --> start;
  start         (read.avail)    --> readpix0;
  readpix0      (readpix.l)     --> readpix1;
  readpix1      (readpix.u)     --> readpix2;
  readpix2      (readpix.ul)    --> pixdone;
  pixdone       (write)         --> PAPI_DONE;
  PAPI_DONE  (papi_done)        --> INIT;
end
```

Figure 7-3 PAPI Interface for an RVC-CAL Actor

All the actions are constrained using an internal FSM to impose a partial order among action tags. In this case, once the actor is scheduled (or fired), the *init* action is first enabled. The last action of the actor (*papi_done*) is the one that stops the PMC sampling and reads the samples. However,

these two PAPI actions, *init* and *papi_done* can be scheduled after any other actions as long as the *init* action is executed before the *papi_done* action. In this way, PAPI can be integrated into any actor if the developer wants to focus on any specific unit. In addition, more functionality can be added in either the *init* or the *papi_done* action.

B. In Existing Actions

Similarly, those PAPI primitives can also be simply inserted within actions. Figure 7-4 shows partially the code as of an example that monitors the display action in the *displayYUV* actor.

```
initialize ==>
        do
                if isEnergy_aware_src()=1 then
                        event_init();
                        event_create_eventList(eventCodeSetSize, eventCodeSet, thread_ID);
                        event_start_src(thread_ID);
                end
        end
displayPicture: action ==>
        guard
                nbBlockGot >= pictureSizeInMb,
                (displayYUV_getFlags() & DISP_ENABLE) != 0
        do              ... ... /* Display related Functions */
                nbFrameDecoded := nbFrameDecoded + 1;
                if isEnergy_aware()=1 then
                        EstCounter := EstCounter +1;
                        if EstCounter =25 then
                                event_stop(eventCodeSetSize, PMC, thread_ID);
                                IsEventStarted := false;
                                EstCounter := 0;
                                source_pause(PMC);
                                event_start(thread_ID);
                                IsEventStarted := true;
                        end
                end
        end
```

Figure 7-4 PAPI Interface for an RVC-CAL Decoder in DisplayYUV Actor

In Figure 7-4, at the initialization phase, if the *energy_aware* mode is enabled, functions to configure PMCs are called through PAPI primitives. The energy-aware mode is explained more in detail, later, in the section 7.2. Note that the initialization action is only executed once the first time an actor is fired. The decoded frame will be displayed when the corresponding action, *displayPicture*, is scheduled. All the actions executed during the PMC-working period are monitored by PMCs. After decoding a certain number of frames (e.g., 25 frames), the PMC sampling will be stopped.

Afterwards, the statistic PMC data will be passed to the energy model to estimate the energy consumption during this period. The estimation procedure will be completed in the *source_pause* function. Then, PMCs will start sampling again for the next 25 frames.

Essentially, there is no large difference between these two methods. In the RVC-CAL framework, each time one actor is fired, only one of the actions will be executed according to the current state of the FSM. In other words, from the moment PMCs are started until they are stopped, not only the actors where PAPI primitives are inserted are scheduled, but other actors may also be fired. Thus, PAPI functions monitor in this way the behavior of several actors rather than only one. In the case shown in Figure 7-3, after the *papi_done* action is accomplished, the corresponding actor is scheduled out from the enable state. PAPI functions are then started again until this actor is re-fired to trigger the *init* action. Unlike the case shown in Figure 7-4, PMC monitoring is achieved in a non-continuous way.

7.1.2. The Dependence of PAPI and OS

The PAPI tool is used to facilitate the configuration and utilization of PMCs. It provides a universal interface to hide the hardware details. However, there is still one tool layer which is not independent of the operating system and hardware. In this case, to successfully use the PAPI tool on embedded platforms, OSs patches or specific configurations are needed.

7.1.2.1. OS Patch on PandaBoard

Although PMCs are available in the PandaBoard, they cannot be directly used. The basic reason is that on the PandaBoard, PMCs are not only enabled by the performance monitoring unit (PMU), but are also determined by the PMU/CTI (Cross Trigger Interface) interrupt. Cortex-A9 processor has a functionality called cross trigger, which uses the events of one module to trigger the behavior of another module. CTI connects all the modules that generate trigger events to the Cross Trigger Matrix (CTM) to achieve the cross trigger.

To better understand the kernel patch for PMC support, the concept of Linux interrupt is briefly introduced at first. Linux interrupts can be divided into two types: soft interrupts and hard interrupts. Soft interrupts are implemented by the signal mechanism. However, to support PMCs, hard interrupts are needed. The interrupting device sends hard-level signal to the interrupt controller through the interrupt bus to inform the OS that an interrupt has been generated. Then, the OS will detect the kind of interrupt and the index of the interrupt bus by checking the state register and the status bit of the interrupt controller. To use the interrupt bus, the device needs to send an interrupt requirement (IRQ). Then, the OS will decide to response to the IRQs based on their priorities. Thus, the patch should be able to let the OS to detect the PMU interrupts.

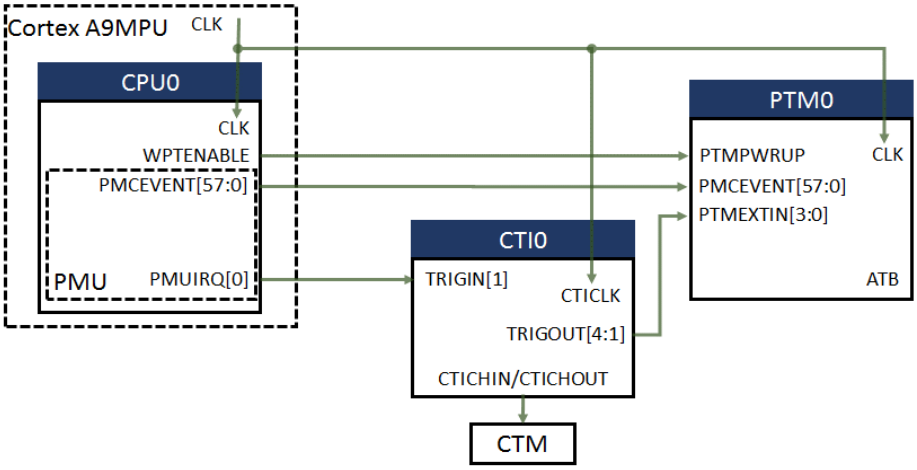


Figure 7-5 Block Diagram of the Cortex-A9 CTI Connections

Figure 7-5 shows a simplified block diagram showing the interconnection between the PMU, the CTI and the PTM [175][176]. PTM is the abbreviation of Program Trace Macrocell, a module that performs real-time instruction flow tracing. Except cycle counters, the PTM can use all available PMC events through its extended external inputs (PTMEXTIN). Among all the extended external inputs, two of them are used to access PMCs and each one could independently select one of the PMU events to monitor. Trace tools could use the information generated by the PTM to reconstruct the execution of all or part of a program. Based on this connection, the main tasks of the patch are:

- To add PMU Support;
- To add power management (PM) support. Hooks to initialize the hardware at run-time are available to support dynamic PM through the ARM PMU driver. Without having these runtime PM hooks, the configuration of the PMU hardware would be lost when low power states are entered and hence would prevent PMU from working;
- To implement the route from PMU IRQs to CTI IRQs. CTI enables the debug logic, the embedded trace macrocell (ETM), and the PMU. The ETM is part of the PTM. It is a real-time trace module and provides the instruction and data tracing of a processor. The CTI is connected to a number of trigger inputs and trigger outputs. Each trigger input can be connected to one or more trigger outputs. The base address of the CTI is not fixed and can be different for specific system implementation. However, the offset of any particular register from the base address is fixed.

7.1.2.2. OS Configuration on BeagleBoard

Unlike PandaBoard, BeagleBoard uses the Cortex-A8 processor. The Cortex-A8 processor implements the ETM instead of the PTM. Similarly to the relationship of the PTM and the PMU,

PMC events are all available for the ETM through the extended external inputs (EXTIN). Each PMC event is mapped to one of the two extended external inputs.

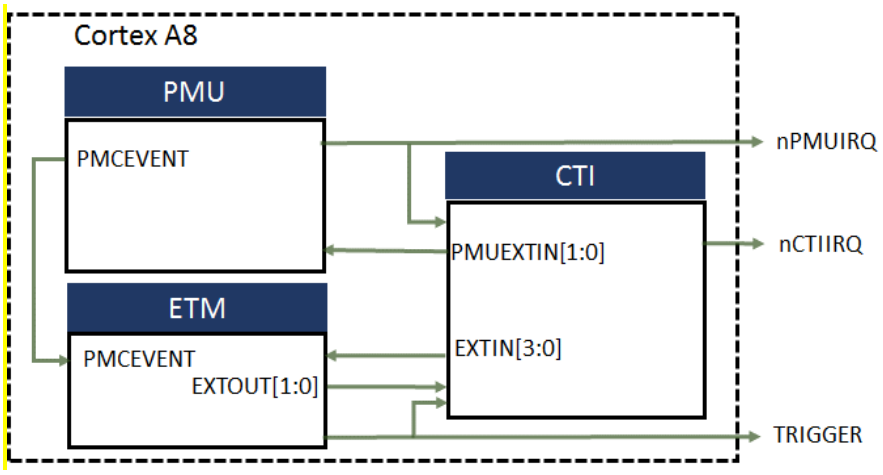


Figure 7-6 Block Diagram of the Cortex-A8 CTI Connections

The interconnections between the PMU, the CTI, and the ETM are shown in Figure 7-6 [177]. Note that the difference between the Cortex-A9 and the Cortex-A8 is that, for the latter, the interrupt of the PMC can be generated by the PMU itself without passing the CTI. The processor will assert the pin *nPMUIRQ* if the PMU generates an interrupt. This pin can be routed to an external interrupt controller for prioritization and masking. Since PMU IRQ can be directly detected by the OS as long as the debugging unit is enabled, there is no need to patch the OS to support PMC events. Thus, to use the PMC events on the BeagleBoard, the procedure would be as follows:

- Enable performance events and counters
- Enable OMAP 3 debugging peripherals to enable the according hardware.

7.2. Implementation of the Energy-aware Manager

7.2.1. Implementation of Energy-aware Events in Jade

As introduced in section 6.1.1.2, the Just-In-Time adaptive decoder engine (Jade) is responsible for the decoder reconfiguration and scheduling. The proposed energy-aware manager is an additional unit of Jade. Jade provides three operation modes to implement a decoder, namely command line, console, and scenario. The scenario mode is the most powerful one to manage decoder configuration and execution through a JSC (JavaScript configuration file)-formed file. To configure the initial settings of decoders, this file specifies a list of pre-defined XML events. In this file, each line is an XML event to be executed to perform different functions. Table 7-1 lists the most commonly used XML events.

Table 7-1 List of Jade Events in Scenario Mode

XML Events	Functionality
Load	Load and store a given decoder network with a specific identifier
Start	Start the decoder. Parameter 'id' is the indicator of the employed decoder network to configure the decoder
Pause	Pause a given decoder
Set	Reconfigure an existing decoder with a new decoder network
Wait	Put Jade in wait mode for a given period (in second)
Remove	Remove a given decoder network
Stop	Stop and exit Jade

To implement the energy-aware management, three specific XML events should be added into the original list.

1. <Mode />

This event allows users to set Jade to work in either energy-aware or non-energy-aware (normal) mode. If Jade is set in energy-aware mode, all users' preferences are recorded as the guidelines to make reconfiguration decisions.

2. <Enable />, <Disable />

The energy-aware manager will be enabled if the energy-aware mode is set in the <Mode /> event procedure. It is disabled if the non-energy-aware mode is set. However, the energy management might be enabled or disabled at any moment while Jade is running. These two events are used to enable or disable the energy-aware mode after the <Mode /> event procedure.

Figure 7-7 shows an example of scenario configure file in which the proposed events have been employed. In this example, first, Jade provides an interface to let users set the Jade work mode (Line 4) and corresponding parameters. Then Jade loads two different decoder descriptions and set the identifier of each one with the parameter '*id*' (Line 7 and 8). After the successful return of the *Load* event, users can also disable the energy-aware mode if it has been enabled before (Line 11). Jade executes then the decoder based on the corresponding decoder description with the parameter '*id*' which is defined in the *Load* event (Line 14-17). The encoded sequence is defined by the parameter '*input*' with the access path. Users can use the *Enable* event to run Jade in energy-aware mode (Line 20) again. If users do so, Jade will set the current total energy amount (it can be provided by the battery monitor). During decoder execution, the energy estimation model periodically provides estimated when a certain number of frames are decoded, and the energy-aware manager will compare it with the low battery threshold. Once a low battery state is detected, although users have set the same decoder to decode the two available sequences (Line 23 and 24), Jade will automatically load

and reconfigure another decoder determined by the video manager to decrease the energy consumption. After decoding all sequences, Jade will stop and exit (Line 27 and 28).

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <JSC>
3         <!--This is a new event which lets users to choose the running mode of Jade: Normal mode or Energy-aware Mode-->
4         <Mode/>
5
6         <!--Two decoder description are loaded-->
7         <Load id="1" xdf="/VTLs/PHP.xdf" />
8         <Load id="2" xdf="/VTLs/CBP.xdf" />
9
10        <!--This is a new event. Even though at the beginning Jade is set in energy-aware mode, this event can disable this mode-->
11        <Disable />
12
13        <!--Here the parameter 'id' specifies the decoder employed to decode the sequence.-->
14        <Start input="/HighResolution/SequenceH.264" id="1" threaded="1"/>
15        <Pause />
16        <Start input="/MediaResolution/SequenceM.264" id="1" threaded="1"/>
17        <Pause />
18
19        <!--A new event to enable energy-aware mode-->
20        <Enable />
21
22        <Start input="/HighResolution/SequenceH.264" id="1" threaded="1"/>
23        <Pause />
24        <Start input="/MediaResolution/SequenceM.264" id="1" threaded="1"/>
25        <Pause />
26
27        <Stop id="1"/>
28        <Stop id="2"/>
29 </JSC>
```

Figure 7-7 Scenario Specification for JADE with the Event Extension Proposal

7.2.2. Implementation of Energy-aware Management Metric

To optimize and manage the energy consumption, the management metric has been introduced in section 5.3. The metric depends on the system gain related to the energy-saving level, computing complexity, bitrate, and image quality. In this work, the system gain has been simplified as indicated in equation 7-1, i.e., the system gain $g(D_j)$ is proportional to the inverse of the computing complexity $C(D_j)$.

$$g(D_j) = \frac{1}{C(D_j)} \tag{7-1}$$

This is to say, the energy-aware manager will always reconfigure the decoder with the lowest complexity one once it detects the low-battery situation. The complexity can be indicated by the id of the decoder description, e.g., a larger id indicates a less complex decoder.

7.2.3. Implementation of the Energy-aware Manager

Another challenge of this thesis is to efficiently control the decoder to reduce the overhead introduced while decoding. The core of Jade is the Just-In-Time (JIT) compiler. JIT exploits the LLVM IR to achieve real-time decoder reconfiguration.

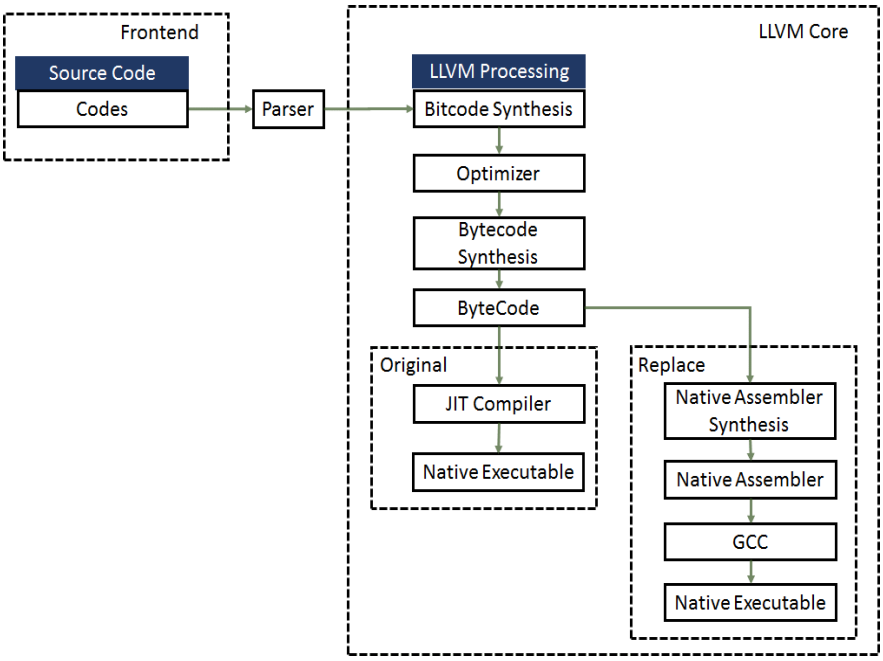


Figure 7-8 LLVM Interaction with JIT and GCC Compilers

Currently, JIT cannot perfectly support the ARM architecture and thus the JIT compiler cannot correctly generate an executable binary. As an alternative, Jade uses GCC to generate the binary file from the LLVM IRs. Figure 7-8 shows the relationship between LLVM, JIT and GCC. However, using the GCC compiler, all the VTLs in the form of LLVM IRs are recompiled and every FU is linked together as a monolithic file, even those FUs that have been already compiled in previous decoders and reused in the current decoder. This seriously increases the reconfiguration time. The whole process of reconfiguring a decoder is shown in Figure 7-9, including the backend translation and executable binary file generation. Arm-fix is a parameter used to enable GCC instead of JIT to complete the compilation.

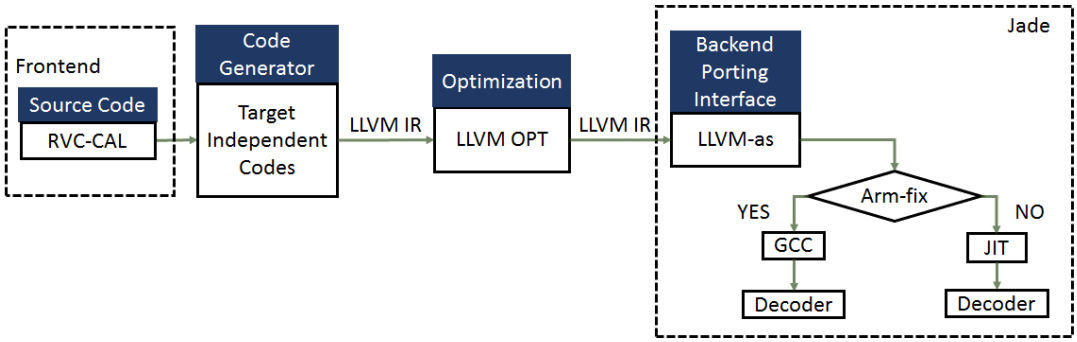


Figure 7-9 RVC Specification Implementation Process

GCC is a static compiler. The decoding processing is executed by Jade after the source code is compiled and linked to form a single binary file (*tempDecoder*). In this case, *tempDecoder* is a child process forked by Jade. To communicate and synchronize Jade and *tempDecoder*, the inter-process synchronization mechanism is used.

7.2.3.1. Relationship and Communication Structure between Jade and the Decoder

As far as the communication concerns, Jade works as a parent process. After the preparatory work to execute the decoder, Jade forks the child process. This child process will execute a system call from the "exec" family to execute different programs from its parent process. In the case of Jade, the child process is the decoder, named as *tempDecoder*. If the energy-aware mode is disabled, during the execution of *tempDecoder*, Jade will change into the wait status and be woken up until *tempDecoder* finishes. A simplified relationship between Jade and the decoder can be seen in Figure 7-10.

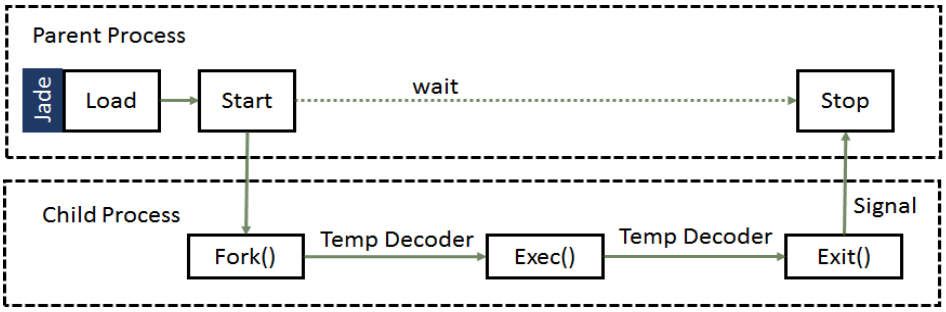


Figure 7-10 Relationship between Jade and Decoder in Energy-aware Disable Mode

However, the situation when the energy-aware mode is enabled is more complicate:

- First, as a parent process, Jade can only be woken up by the return or exit signal sent from its child process, i.e., *tempdecoder*. In this way, the energy-aware manager, which is implemented as a unit in Jade, cannot timely obtain energy update information from the *tempdecoder*. An alternative solution is to change Jade to the pause status instead of wait. This allows waking up Jade through various signals. Meanwhile, while the energy-aware manager makes the management decision, *tempdecoder* is also in the pause status waiting for the signal from Jade (to continue the decoding processing or to be killed due to the low-battery situation). A simple way to change the process is to call the pause function, which suspends the program execution until a signal arrives.
- Second, the energy estimation value needs to be shared between Jade and *tempdecoder*. Unlike the shared data among threads that belong to the same process, there is no direct way to pass parameters among different processes. Therefore, a shared memory mechanism is employed to pass the data between Jade and its child process, *tempdecoder*.

Based on this pause-status mechanism, the decoder control of Jade based on the decisions from the energy-aware manager is summarized in Figure 7-11.

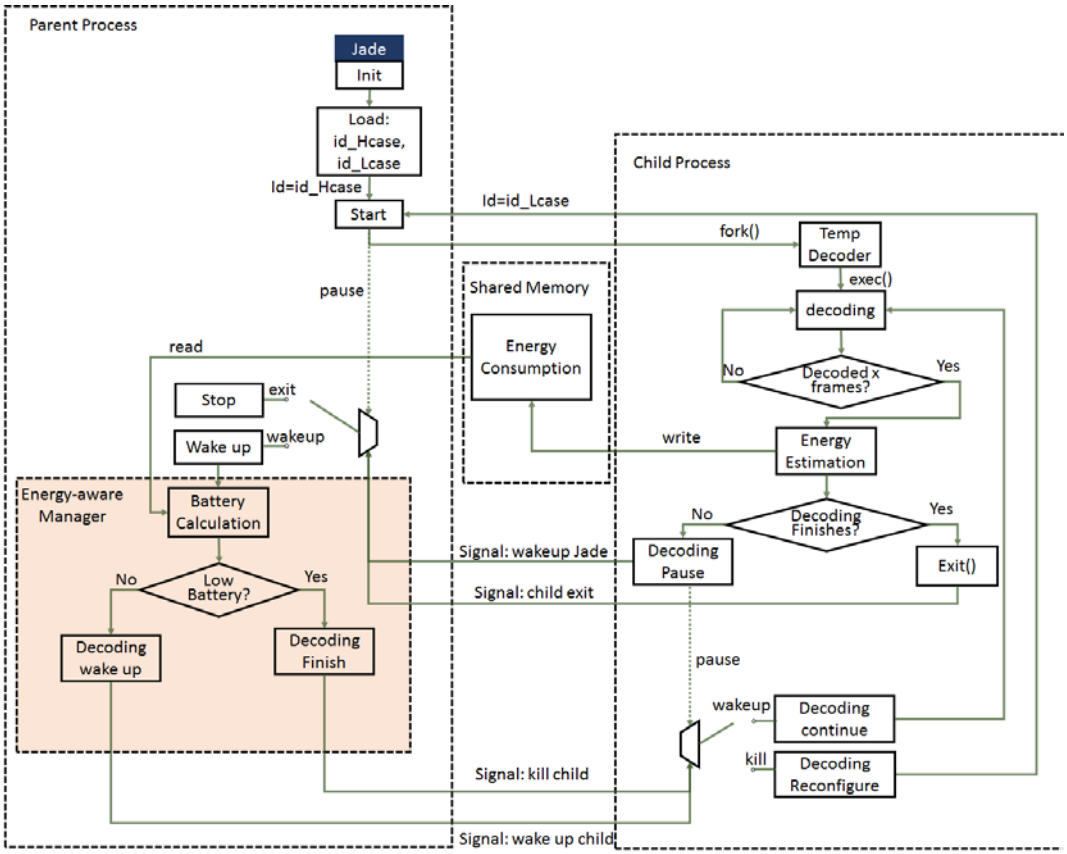


Figure 7-11 Relationship between Jade and the Decoder in Energy-aware Enable Mode with Pause-wake Mechanism

However, experimental results show that this pause-wake mechanism introduces an unacceptable performance decrease. The number of decoded frames per second decreases 70%. This is because each time a process is waken up from its pause status, a sequence of steps is carried out, including:

- To load this process from kernel space to put it to user space;
- To resume its last executed information to CPU and registers.

This is a quite time-consuming procedure. Therefore, to avoid the overhead of context switching, a simple *while (1)* loop is employed instead of the *pause* status. And all the shared information, such as energy estimation values and low battery flags, are passed through the shared memory. If the low battery signal is true, the decoder calls the exit function and terminates. In this case, there are two conditions to break the *while (1)* loop, one is the low battery state and another is the decoder finish signal. The former one is controlled by the energy-aware manager and the latter one is sent by the *tempdecoder* and automatically detected by Jade based on the signal mechanism. A low-battery break will cause the energy-aware manager to inform Jade to reconfigure the decoder with a lower-power decoder description while a finishing-break will make Jade to load a new sequence to decode. The final communication scheme between Jade and the *tempdecoder* is shown in Figure 7-12.

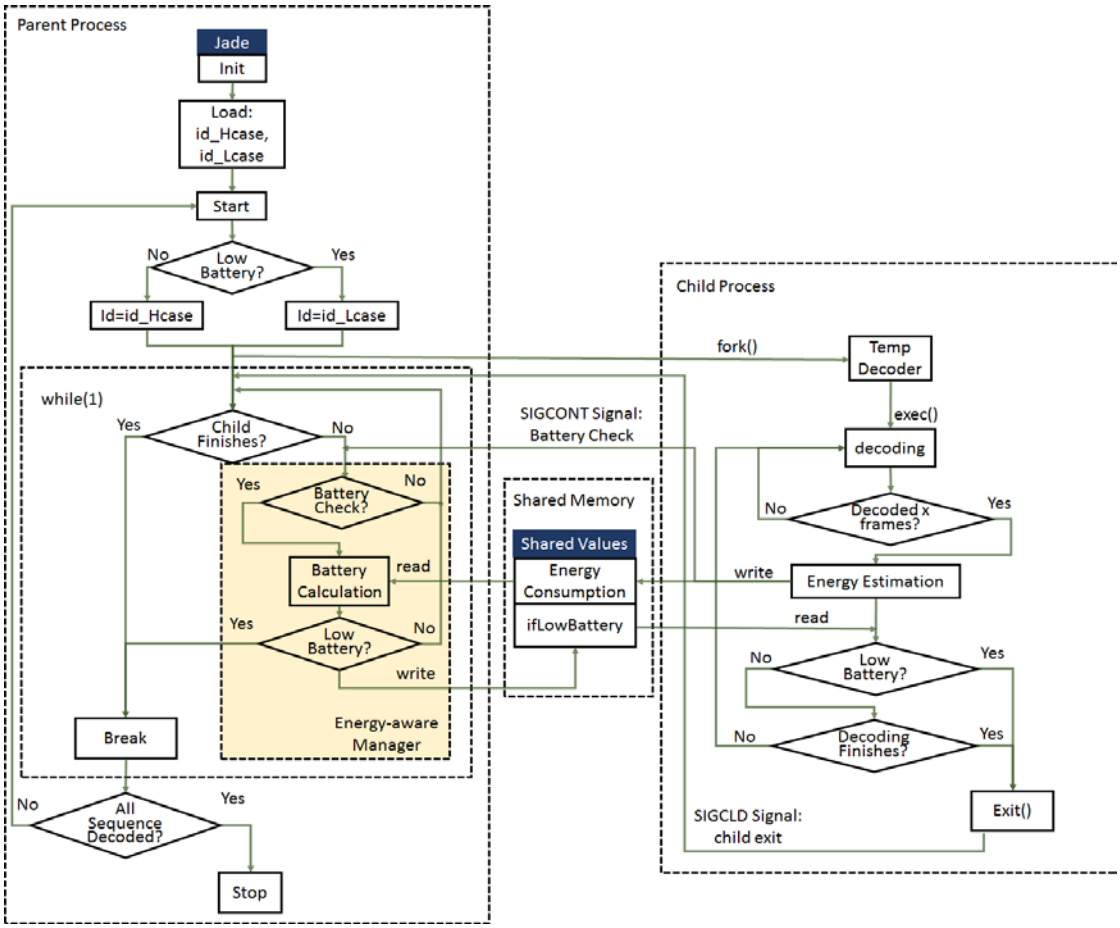


Figure 7-12 Relationship between Jade and Decoder in Energy-aware Enable Mode with while (1) mechanism

7.2.3.2. Implementation Details of the Communication Scheme

A. Signal Mechanism

The core of the communication between Jade and the *tempdecoder* is the signal mechanism provided by the Linux OS. The mechanism is used for event notifications among asynchronous processes. In Linux, a common way to communicate processes is the *signal* channel. A process can send a signal to a different processes using the *kill()* system call with prototype:

```
int kill(pid_t pid, int sig)
```

This system call will send the signal with number *sig* to the process with process ID *pid*. Signal numbers are small positive integers, which may vary from one platform architecture to another one. There are a set of pre-defined signals and the corresponding default actions. In Linux, by default, a SIGKILL signal kills a process, a SIGSTOP signal stops the process and a SIGCONT signal resumes a stopped process. When a process receives a signal, a default action will occur, unless the process has been arranged to handle this signal. A handler for a user-defined action can be set up with prototype:

```
typedef  void (*sighandler_t) (int);

(sighandler_t) signal (int sig, sig_handler);
```

This prototype sets up the routine *sig_handler* as a handler for a signal with number *sig*. When a signal arrives, the signal handler is invoked to interrupt the current program. When the signal handler returns, the interrupted activity will be continued. In the proposed implementation, as shown in Figure 7-12, there are two signals used. Every time the *tempdecoder* finishes decoding a certain number of images, the energy consumption of this period is estimated and stored into the shared memory. Meanwhile, the decoder will send the SIGCONT signal. The handler of the SIGCONT signal is to inform the manager to check the battery status with the estimated energy information. If the decoder has finished its work and exits, a SIGCLD signal sends and its handler sets a child-finished flag as true to break out Jade from the *while(1)* loop.

B. Shared Memory Mechanism

Signals are just employed to inform or notify a process about what has happened in another process, but they are not used to pass data. Considering that parent and child processes have their own data sections, the simple usage of global variables cannot be utilized to pass data among different processes. Thus, a shared memory mechanism is employed to pass data between Jade and the *tempdecoder*. Shared memory schemes are efficient methods to pass data between programs because data do not need to be copied among communicating processes. One process creates a memory portion and other processes (if they are allowed) access it.

In Linux, a process creates a shared memory segment using the function with prototype:

```
shmget(key_t key, int size, int flag):
```

This function returns the ID of the created shared memory if the creation is successful or -1 if an error happens. Parameter *key* is a non-negative integer to identify each section of shared memory. It is typically set as the constant *'IPC_PRIVATE'*, which lets the kernel choose a new key. The *keys* of shared sections are system-wide, and their values continually increase to a maximum value and then wrap around to zero. Parameter *Size* is the size of shared memory segment in bytes and the argument *flag* specifies the initial access permissions and creates the control flags. This function call can also get the *ID* of an existing shared segment when a process requests sharing an existing memory portion.

Once a shared memory segment has been created, a process can attach this segment to its address space by calling the *shmat()* function. Once successfully attached, the process can read from or write to the segment according to the permission configuration requested during the attach operation. A

shared memory segment is described by a control structure with a unique *ID* that points to an area of physical memory. The identifier of the segment is called *shmid*. The prototype of the *shmat* function is:

```
shmat(int shmid, void *addr, int flag).
```

It will return a pointer to the shared memory if the mapping is successful or -1 on error.

In addition, the shared memory can be controlled by a system call as:

```
shmctl(int shmid, int cmd, struct shmid_ds *buf)
```

The parameter *cmd* can be one of *IPC_STAT*, *IPC_SET* or *IPC_RMID*. *IPC_STAT* fills the buffer with the structure of shared memory and obtained the status of the shared memory specified by *shmid*. *IPC_SET* can change the status of the shared memory. *IPC_RMID* will remove the shared memory segment from the system once the last process which has attached to this segment terminates or detaches from it.

Jade and the *tempdecoder* use shared memory to pass information. Jade creates a piece of shared memory with a particular key number and attach itself to this space. The decoder can then get the ID of this shared segment with the same key and attach to it. Then, both of these two processes can access the memory segment to share the energy related information and low battery flag. Finally, the shared memory is removed by Jade and the decoder.

7.3. Conclusion

In this chapter, the proposed implementation of the energy optimization and management based on the functional-oriented reconfiguration is described in detail. As far as Figure 5-4 concerns, the complete implementation includes the energy estimation model and an energy-aware manager for decoder reconfiguration. As thus the implementation consists of solutions for these two parts:

- Firstly, to estimate the energy consumption of various decoders, the PMCs need to be easily configured and enabled. Two methods to integrate PAPI API functions into any actor of ORCC framework are introduced. One is to create new actions to operate on PMCs and another is to insert those API functions into the existing actions of the actors. Both methods can avoid the modification of the generated target source code. A tiny difference is that with the second method, continuous sampling of PMCs can be maintained. In addition, the necessary OS patches and configuration operations to enable PMC usage are introduced.
- Next, the discussion focuses on how to implement the energy-aware manager. This manager

is an additional unit of the tool Jade. To enable it, three additional events of Jade, namely mode, enable, and disable events are implemented. The mode event selects the work mode of Jade while the enable and disable events set Jade to work in energy-aware mode or normal mode, respectively. In the energy-aware mode, the proposed energy-aware manager is enabled. Besides work mode setting, the core operation of this manager is to decide when to reconfigure once the low battery state is detected. In the implementation of this thesis, the decision is simplified. The manager chooses the decoder with the lowest complexity to reconfigure. Another part of the implementation in this chapter focuses on the communication between Jade and the decoder. Jade controls the decoder execution and the decoder passes energy information to the manager. A signal-based inter-process communication scheme is employed for the communication. Data is passed between the two processes through a shared-memory scheme. This implementation method achieves the efficiency to make reconfiguration decisions.

PART E

Chapter 8: Results

8. Results

The objective of this thesis work is to provide an energy optimization and management mechanism on video coding applications to extend the battery life. This mechanism includes an energy estimation model and an energy-aware manager based on the functional-oriented reconfiguration engine. The functional-oriented reconfiguration, stated in this thesis, is one of the reconfiguration techniques which is platform-independent and aims to improve system functionalities or produce new functionalities by re-connecting the existing functional units. In this chapter, the experimental results, including the validation and evaluation of the model, the verification of the energy-aware manager implementation and the battery life time extension will be given in three parts: for the first part, the results related with the selection of PMC events will be firstly presented. Then, starting with the selected PMC event set, an analysis will be conducted to guide the training data selection. This guideline will be treated by more tests to prove its capability of improving the model accuracy. Finally, the overhead of the estimation model will be given to show its real performance; for the second part, the modifications of the reconfiguration engine to implement the energy-aware manager will be testes and verified; at last, the potential battery life extension by combining the estimation model and the energy-aware manager will be shown.

All the experiments are carried out on two embedded platforms, the PandaBoard platform and the BeagleBoard platform, running a Linux 3.8.0 kernel. In the following parts of the chapter, “PB” and “BB” are used as the name of each platform. Note that the PB has been patched to add the performance monitor unit interrupts to support PAPI. Four decoders, the simple profile of the MPEG4 Part 2 decoder, a progressive high profile decoder implementation and a constrained baseline profile decoder implementation of the MPEG4 advanced video coding standard, and a main profile of the high efficiency video coding decoder have been considered as the benchmarks. Seventy-eight, forty-one and seventy-one conformance sequences have been applied to test each standard, respectively. They are configured with the common test conditions such as different spatial resolutions, frame combinations, slice types, quantization parameters, frame rates, and entropy coding methods. SP, CBP, PHP and MP are defined as the short name of each decoder, respectively. More details of the experiment infrastructure has been introduced in chapter 6.

8.1. Model Validation and Evaluation

To help the energy-aware manager makes the reliable determination on decoder reconfiguration, an accurate estimation model is necessary. Followed by the methodology described in chapter 3, the modeling procedure is:

- PMC events selection, which uses a PMC-filter to select the more suitable set of events;
- Model fitting, which employs either linear regression or MARS regression to build the energy estimation model.

Once a model is built, its validation and evaluation are conducted to demonstrate the model performance. There are two aspects to assess:

- Accuracy assessment, which compares estimation results and measurement results to assess the model accuracy.
- Efficiency assessment, which merges the estimation model into the decoder to test the decoder performance decrease.

The results of these four steps will be described in detail in the next sub-sections.

8.1.1. Common Explanations of the Experiments

Before moving into the details about the experimental results, the common explanations will be given in short.

8.1.1.1. Model Description

The energy estimation model proposed in this paper is expressed in equation 8-1.

$$E_M = \bar{P} \times T_i + f(PMC_s) \qquad 8\text{-}1$$

The first addend is the baseline energy. Given that either the OS running on the platform or any of its idle devices always consume a certain amount of energy, it is necessary to set this bottom line as the product of the system idle power ($\bar{P}$) and the execution time (T_i). The second addend in this equation, the so-called incremental energy, denotes the energy consumption enforced by the system activity. This term is the one estimated from the observation of the PMC events and its values are usually much smaller than the ones of the baseline energy. Usually, the modeling procedure needs to use external power measurement to profile coefficients for each event. Typically, these values are directly obtained with the hardware measurements. The measurement system has been described in section 6.3. This measurement system is also used to evaluate the accuracy of the estimation models.

8.1.1.2. Models Relative Errors

The mean absolute percentage error (MAPE), which is the percentage of the difference between the estimated energy value and the measured value is calculated to visually show the modeling accuracy as equation 8-2.

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{y_t - \widehat{y_t}}{y_t}\right| \times 100\% = \frac{1}{n}\sum_{t=1}^{n}|\widetilde{a_t}| \times 100\% \quad 8\text{-}2$$

Where y_t is the measured value, $\widehat{y_t}$ is the predicted value, and n is the number of the fitted points, i.e., the number of frames of this sequence.

To assess the overall accuracy of each model, the MAPE distribution will be used. It is calculated as the pseudocode in Figure 8-1. Any test sequence (seq_j) from the benchmark set is trained to build a model ($Model_j$). The MAPE error ($Err^k_{MAPE_j}$), when this model ($Model_j$) is employed to estimate the energy of each test sequence, will be calculated from the estimation energy ($Model_j(seq_k)$) and the measured energy ($Meas(seq_k)$). Then, all the MAPE errors of this model will be averaged to present the accuracy of the model, where *N* is the number of benchmarks. Finally, *N* averaged MAPE errors will be distributed into different error levels. For example, errors lower than 10% is an error level and errors ranging from 10% to 20% is another error level. In this thesis, an estimation model, whose averaged MAPE error is less than 10%, will be considered to be an accurate model.

For ∀ seq ∈ {benchmarks}
    $Model_j = f\left(PMCs_{(seq_j)}\right)$
    For ∀ seq ∈ benchmarks
        $Err^k_{MAPE_j} = MAPE(Model_j(seq_k), Meas(seq_k))$
    End
    $AvgErr_j = avg(\sum_{k=1}^{N} Err^k_{MAPE_j})$
End
Distribution ($Err^k_{MAPE_j}$)

Figure 8-1 Pseudocode to Calculate MAPE Distribution

8.1.2. PMC Events Selection

As discussed before, the set of PMC events is a key point to build a PMC-driven model from two considerations: modeling overhead and captured application characteristics. Some works [26][27][29] before used only total number of instructions (TOT_INS) as the only parameter to estimate the energy consumption. However, this simple model does not show good accuracy when employed for complex applications. In this subsection, the modeling results from a TOT_INS-based energy estimation model for a video decoder model running on the PB platform will be shown. PMCs are set to monitor TOT_INS with the period of decoding one frame. Both, linear and MARS regression methods

(described in section 2.2.3) are conducted and four decoders, SP, CBP, PHP, and MP (described in section 6.5) are employed as the benchmarks. SP, CBP, PHP, and MP are used as their abbreviations, respectively, for simplicity. Among the encoded sequences of each decoder standard, each one of them is selected as training data to obtain the model parameters once and the model generated with these parameters is evaluated by all sequences.

As can be seen in Table 8-1 (a) to (d), model accuracy decreases when the decoder complexity increases. For the PHP and MP decoders, less than 15% models can achieve the accuracy lower than 10%, which is unacceptable in real applications.

Table 8-1 Average Error Distribution of Models Based on TOT_INS

Table 8-1(a) SP Decoder

Methods	<5%	5%-10%	10%-20%	20%-30%	>30%
Linear	7.41	22.22	40.74	18.52	11.11
MARS	14.81	33.33	37.03	7.41	7.41

Table 8-1(b) CBP Decoder

Methods	<5%	5%-10%	10%-20%	20%-30%	>30%
Linear	5.26	17.54	36.84	22.82	17.54
MARS	12.28	12.28	19.30	24.56	31.58

Table 8-1(c) PHP Decoder

Methods	<5%	5%-10%	10%-20%	20%-30%	>30%
Linear	2.78	11.11	30.56	25.00	30.56
MARS	2.78	8.33	27.78	25.00	36.11

Table 8-1(d) MP Decoder

Methods	<5%	5%-10%	10%-20%	20%-30%	>30%
Linear	3.03	6.07	39.39	12.12	39.39
MARS	1.56	6.25	34.38	10.94	46.87

These results are analyzed in the following discussion. In modern processor, the total number of instructions includes both, issued and replayed instructions. Replaying is a technique which is employed by multi-threaded processors to avoid stalling the pipeline when a long latency event occurs. A pipelined processor will continue to issue instructions followed an issued instruction as long as they do not depend, or their dependencies can be resolved by forwarding results. This mechanism makes pipeline continuously process instructions, so the instructions will be executed quickly. However, for complex algorithms, it is more frequent that instructions encounter long latency events, such as a load operation that generates misses in the cache, a conflict with a shared port or

communication among different functional units. In these cases, to allow independent instructions executing immediately without waiting periods, instruction replay technique solves this problem by squashing the instructions in the pipeline and beginning the execution of instructions from a different thread. This will cause problems in the energy estimation. As discussed in section 6.2, PAPI binds its observation on specific threads. This is to say, those instruction assigned to different threads will not be counted by the PMC although they are caused by the same application. Thus, to use only the total instruction count will hide the cases when the long latency occurs. Usually, long latency is always accompanied by data missing, data unavailability or mis-prediction, which may result in more energy consumed by this instruction than other immediately completed instructions. Therefore, a model must be able to capture the energy consumption using the information of various types of events to give a more accurate prediction. Otherwise the model will not be able to correctly compute the energy estimation.

Before using the PMC-filter, the correlation threshold, α, to eliminate the weak energy-related PMC events and the VIF threshold, β, to filter the multi-collinear PMC events, need to be set. Correlation coefficient values in the order of 0.1, 0.3 and 0.5 are experientially considered as weak, medium and strong, respectively [44]. Similarly to the proposal in [32], the threshold α has been set to 0.5. As a common empirical rule, VIF values larger than 10, from 5 to 10 and less than 5 are usually considered as belonging to sequences with high, medium and low multicollinearity, respectively. As far as the energy characteristics concern, the threshold β has been experimentally set to 10. Both two thresholds are set with the medium values in order to ensure that the retained PMC events capture most of the energy characteristics of the applications running on the platform.

After the threshold setting, the proper PMC event set needs to be identified. All PMCs sample the pre-defined events corresponding to the platform along the decoder execution. According to the Cortex-A9 technical reference manual [177], there are only two PMCs that can be simultaneously used. To relieve the limitation of the number of events that can be sampled, the multiplexing technique provided by PAPI is employed. It is needed to point out that this timesharing method causes a small loss in precision [40].

Table 8-2 Selected Events and functionality

PMC Events	Monitor Events Description
TOT_INS	Instructions completed
L1_DCM	Level 1 Data Cache Miss
HW_INT	Hardware interrupts

For the test-bench and video sequences previously described, three platform energy-related PMC events (See Table 8-2) have been selected following the procedure detailed in section 3.2. The

selected events reflect three important energy-consuming activities: processor activity, memory access and peripheral operations. TOT_INS is proportional to the decoder execution time. It might be interpreted as the average energy consumption of decoding a frame. A larger value means a longer run time and a larger amount of required energy. Given the stream-like nature of a decoder, the selection of the L1_DCM event reflects the fact that the L1 data cache is one of the most active units from the energy consumption point of view. The selection of this event is reasonable because video decoders always exert a particular burden on cache due to their high data rates and large sizes. Especially for the streaming video, data cache performance decreases because the continuous video data obtained from the network decreases the spatial locality. However, the L1_ICM is not selected because the decoding algorithms usually consist of tight loops that are repeatedly used. HW_INT is proportional to the activity of peripherals during the decoder execution. This event is used to capture the activities outside the CPU and memory domain. It is worth to mention that these HW_INT-based events model the peripherals activity in a coarse way because it is assumed that every peripheral activity consumes the same energy. However, the models implemented in this work have an acceptable accuracy because: 1) the PandaBoard platform is configured as the minimal system which only enables a minimal set of devices to avoid introducing more complicated issues; 2) the architecture of the embedded systems are not as complex as the one of the desktop systems, thus the energy consumption of peripheral activities can be assumed to have the same value.

8.1.3. Modeling Techniques Analysis and Comparison for the PHP Decoder Use-case

After PMC event selection, next step is to build models based on this PMC event set. Usually, a modeling procedure is an iterative process which needs to be repeatedly adjusted to determine the final parameters. Modeling validation and error analysis are the two major steps in the modeling process. Modeling validation is the first step to assess model performance because the estimation results will guide the underlying investigation and the prediction ability of a model is the precondition to provide good answers. Error analysis on the estimation results is carried out to adjust the model performance. An incisive analysis is the guide to modify modeling assumptions and to shorten the repeating time of achieving the required accuracy. In the following, the first modeling validation results, results analysis, and the validation results after model pruning will be orderly presented. All the results are obtained by running the PHP decoder on the PB platform with the all the sequences compatible with this decoder.

In general, it is impossible for any quantitative prediction to achieve hundred-percent accuracy. The estimation error is the deviation from actual values, which determines the estimation accuracy. To compare the accuracy of the estimation models obtained during the modeling process (potential

estimation models), three colors (red, orange and green) are employed to visually represent the MAPE of each model at three levels: greater than 20%, from 10% to 20%, and less than 10%, respectively. Each row in Figure 8-2 represents the MAPE for all the sequences, including the training sequence used to obtain the model parameters, when an estimation model is applied.

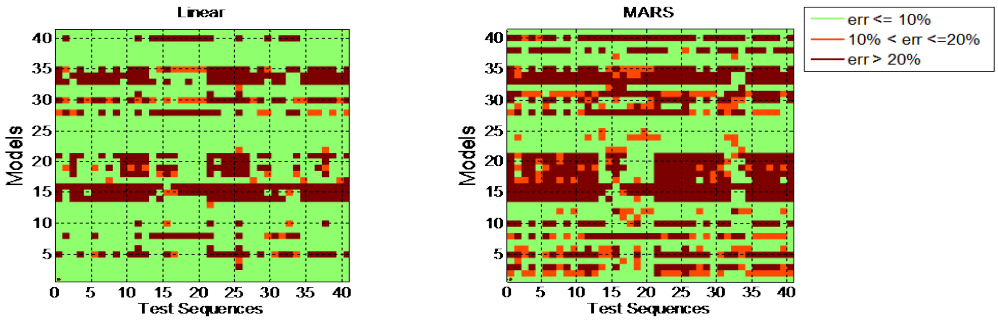


Figure 8-2(a) LR **Figure 8-2(b) MARS**

Figure 8-2 Average Error

From Figure 8-2, it can be noticed the different accuracy achieved from the potential estimation models. Surprisingly, as can be seen in Figure 8-2 (a) and Figure 8-2 (b), the more complex MARS methodology behaves worse than a simple linear regression method. Unlike the previous work [179], Figure 8-2 (b) indicates that increasing the model complexity does not necessarily imply a gain in accuracy.

A good model should have similar estimation results on training data and test data. In these figures, the diagonal line presents the estimation error of each model tested by its training data. Obviously, the estimation results should be quite good. But not every row in which this point locates is all in green. In some extreme cases, the rows are almost completely in red. This means that there are several models that perform quite well during the model training while lost their prediction abilities with other input sequences, especially for MARS models. This phenomenon is called over-fitting. Over-fitting usually refers to a model becoming overly complex in order to get consistent hypothesis while resulting in a poor generalization ability. Usually, there are some reasons to cause over-fitting:

- Extraction errors of modeling samples (in this case, the PMC samples on the PMC events), including (but not limited to) too few samples, sampling method errors, not enough consideration of operational scenarios and characteristics, with the effects that the sample data cannot effectively represent the behaviors of the response.
- Too large interference of noise data.
- Logical assumptions that made at the model training step are no longer appropriate for the real practices. Any prediction model is build and applied based on several assumptions.

Common assumptions include: historical data can be used to speculate future behavior; application scenarios and context do not have significant changes, training data is similar to the application data, and so on. If the above-mentioned assumptions are against to the situations of the real practices, then, the model based on these assumptions cannot be applied effectively.

- Too many explanatory variables.

In this experiment, the input data of linear fitting and MARS fitting are the same, and the linear method shows a relative good predictive capacity, which excludes reasons of accuracy decreased due to number of explanatory variables and noise interference. Meanwhile, the application scenarios and context do not significantly change, the modeling data and application data share a certain similarity, and therefore the third mentioned factor leading to over-fitting can also be excluded. Therefore, the reason of over-fitting can be attributed to the first factor. Specifically, it is because the training data cannot represent all the features of the prediction objects.

An example based on PB platform below compares the different estimation results of MARS and linear regression when they are applied to the data outside the training interval in order to better understand how the over-fitting phenomenon affects more on the accuracy of piecewise fitting method. Figure 8-3 shows the histogram of the values of each PMC event for an individual training sequence (also from the PHP test sequences). MARS method fits its basis functions based on the data distribution of the training sequence. The two knots t_- and t_+ of the cubic function of each PMC event are listed in Table 8-3 and marked as red lines in Figure 8-3. As can be seen in the figure, most of the events belong to the intervals defined by their knots.

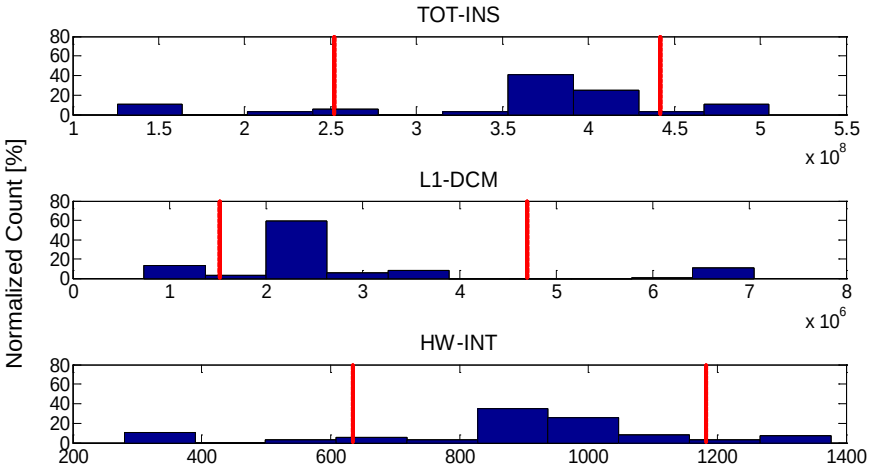


Figure 8-3 PMC Value Sequence Histogram

Table 8-3 Basis Function Knots

PMC Events	KNOT $\mathbf{t}_-$	KNOT t_+
TOT_INS	2.52×10^8	4.42×10^8
L1_DCM	1.51×10^6	4.70×10^6
HW_INT	634.50	1182.50

Figure 8-4 shows the actual energy consumed and the estimated energies drawn from a linear and a MARS model as a function of the TOT_INS value. As can be seen, when the values of the TOT_INS event of the test data are located within the interval of the training data values (i.e., $[2.52\times10^8, 4.42\times10^8]$), the MARS method has a quite good estimation. Since the MARS method is designed to employ local linear fits, it may adapt too close to its training data and, therefore, to lose its predictive capabilities when a new data set is involved. Thus, the basis functions responsible of this behavior produce uncontrolled estimation errors when applied to sequences whose PMC value range is located far away from the basis function knots.

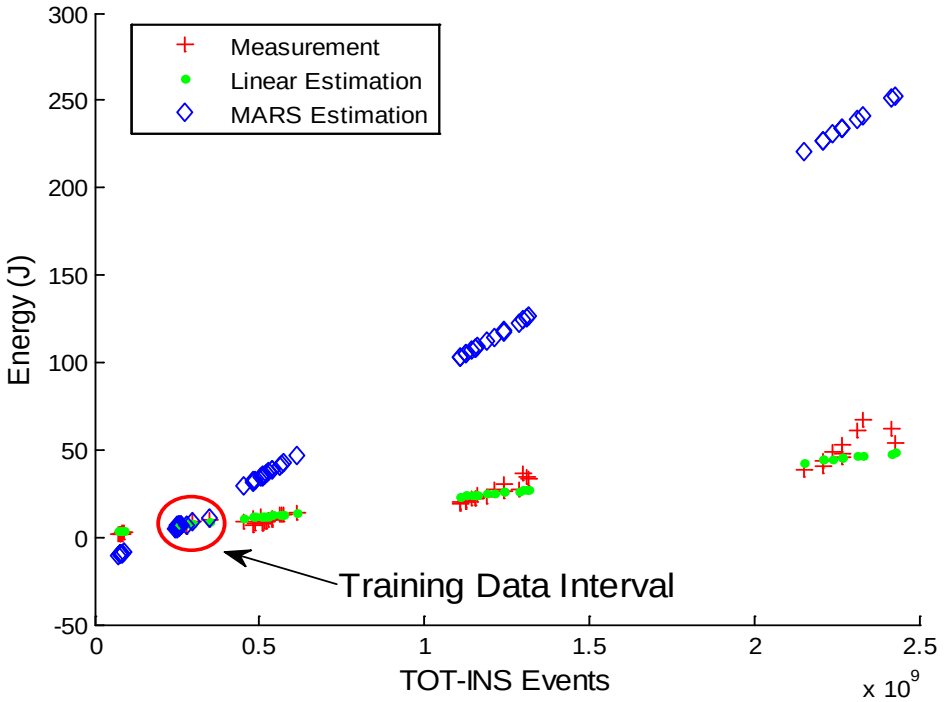


Figure 8-4 Comparison of the Linear Model and the MARS Model

In fact, over-fitting problem also exists when using the linear method, but is not as serious as in the MARS method because these two methods are different in how to reflect data trends. Linear method uses a straight line mapping of the independent variables to the dependent variable and tries to fit a model based on the statistical properties of the training data to have an overall minimal mean square error. While MARS introduces the idea of piece-wise accuracy which only considers accurate if the data fall within the current range. Furthermore, although each basis function of MARS is linear,

MARS permits one variable to be estimated with several basis functions and the superposition of all the basis function may eventually present in the form of polynomial. It is not difficult to imagine that when the actual observations fall outside of the prediction interval, the increase/decrease speed of the polynomial response will be much greater than that of the linear response. If the current model cannot correctly reflect the data trend, this will inevitably result in an inaccurate prediction. Therefore, MARS presents a much worse performance than linear regression when over-fitting exists.

As discussed above, one solution to avoid serious over-fitting phenomena is to employ training data with more varieties. In order to better understand how to select the training sequences to improve the model predictability and stability, the conformance video sequences used in this work have been classified into the following two criteria: the average energy consumption per frame and its coefficient of variation. It has been shown in [37] that complexity parameters such as frame size and rate are proportional to the average energy consumption of a sequence. The video decoder behavior is related to the complexity of the input sequence which could be well reflected by the average consumption per frame. Therefore, the better the matching between the frame average energy consumption of the training and test sequences, the better the accuracy. The variation of the frame energy consumption can be represented by its coefficient of variation (CV) [180], which is a statistical normalized measure of dispersion. The CV is defined as the ratio of the standard deviation, σ, to the mean, μ.

Based on the measurement of the energy consumption on the tests carried out using the PHP decoder, the average frame energy and the CV values are experimentally divided into four groups. For the former, the groups are the following: {<=0.7J}, {0.7J, 1J}, {1J, 3J} and {>3J} while for the latter, they are {0, 10%}, {10%, 20%}, {20%, 80%} and {>80%}. Figure 8-5 (a) and (b) show the maximum and average estimation errors of each of the 16 resulting groups for the linear and MARS regression methods, respectively. The x and y axis reflect the CV and average frame energy partitions, respectively. The blue color means that no sequence has been allocated into the partition while the other three colors have the same meaning as in Figure 8-5 as far as the relative average energy estimation error concerns.

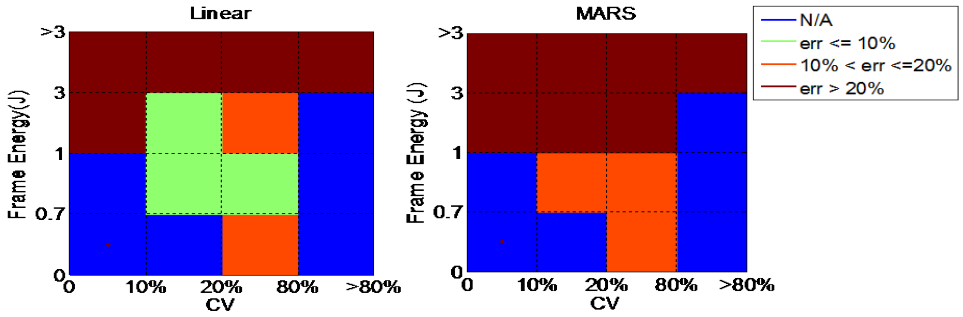


Figure 8-5 (a) Maximum

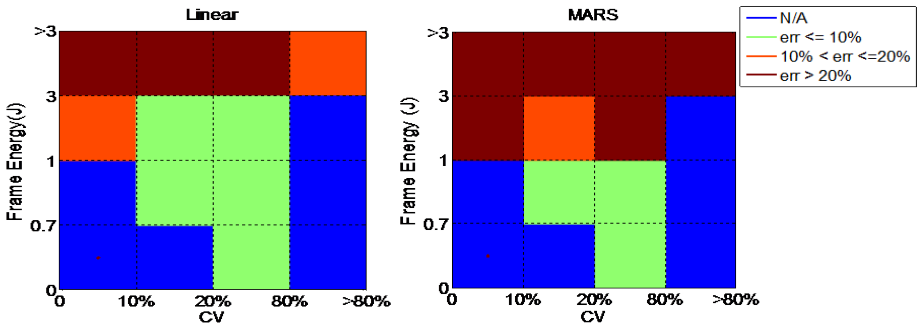


Figure 8-5 Average

Figure 8-5 Estimation Error of Each Group

Although the linear regression and MARS regression methods have different performance, one common observation from Figure 8-5 is that the best models are those whose training sequences have frame average energy and CV values in the medium intervals. Effectively, the training data used by both methods must be located in a sufficiently wide interval as not to provide a wrong tendency. For instance, small CV values represent sequences with relatively stable energy consumption. In other words, it is very likely that the concerned sequence consists of fewer types of frames. On the other hand, high CV values represent sequences which include different types of frames. While the former sequences provide a model that lacks the capacity of capturing the behavior of more complex sequences, the latter sequences derive models which adapt too closely to the training data which lose its generalization ability. For the frame average energy values, similar arguments can be reasoned.

The analysis above suggests that the combination of various conformance sequences can have a better chance to achieve better model accuracy than a single conformance sequence. As a consequence, new models have been built combining sequences which belong to each resolution group (QCIF, CIF and HD) and have medium CV values. Models have been tested with all the combined sequences and the results are shown in Figure 8-6. In this figure, three colors (red, orange and green) are employed to represent the average relative energy estimation errors at three levels: greater than 10%, from 10% to 5% and less than 5%, respectively. The maximum and average errors are detailed in Table 8-4 (a) and Table 8-4 (b), respectively. In these tables, the columns show the number of models (in percentage) distributed on each of the three level intervals defined above.

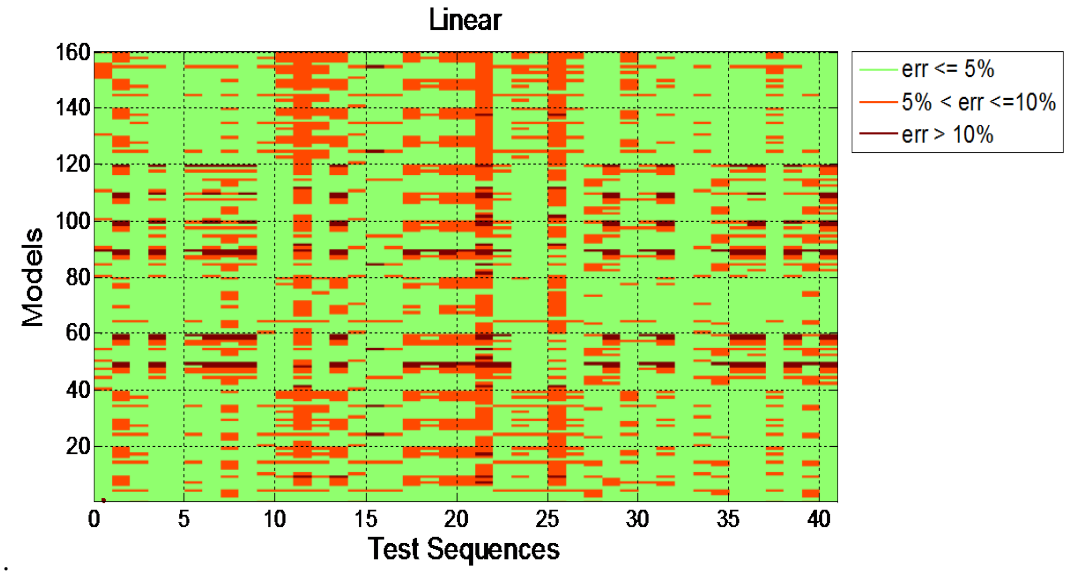


Figure 8-6 (a) LR

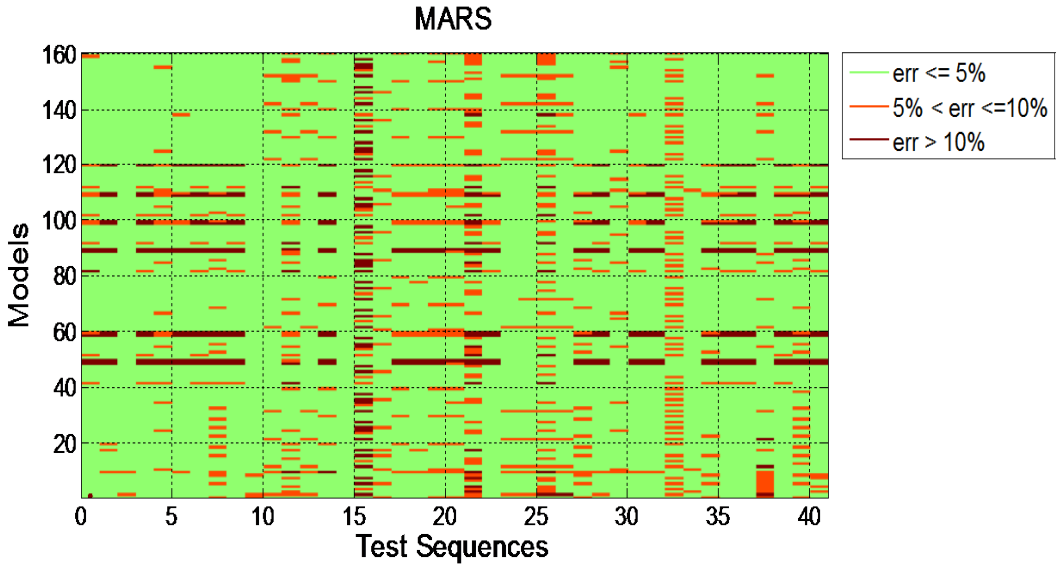


Figure 8-6 (b) MARS Average

Figure 8-6 Average Error for Model Based on Combined Training Sequences

Table 8-4 (a) and (a) show clear performance improvements in both modeling methods when combined training sequences are employed instead of the individual sequences. Besides, the resulting MARS models are able to obtain more accurate results than those of linear models.

Table 8-4 Error Distribution of Models Based on Combined Training Sequences

Table 8-4 (a) Maximum

Methods	<5%	5%-10%	10%-20%	20%-30%	>30%
Linear	0.00%	38.75%	41.25%	18.12%	1.88%
MARS	0.00%	45.63%	45.62%	8.75%	0.00%

Table 8-4 (b) Average

Methods	<5%	5%-10%	10%-20%	20%-30%	>30%
Linear	69.37%	26.87%	3.76%	0.00%	0.00%
MARS	83.13%	15.00%	1.87%	0.00%	0.00%

In Figure 8-7, the estimation errors of the linear and MARS regression models trained by the same combined training sequence, which is one of the combined training sequences used in the experiment whose results is shown in Figure 8-6, have been compared with more details. Note that the x axis denotes all the frames of this combined sequence, and the y axis shows the relative estimation error of each frame. As can be seen, the MARS method provides more accurate estimation than the linear one. It is worth to point out that all average estimation errors from the MARS model are less than 6.5%. This is because of the piecewise characteristic of MARS. Since a combined training sequence captures more decoder behaviors than a single one, an accurate model better represents the different relationships of the decoder complexities and their energy consumption. These relationships are determined by the slope of each basis function. Therefore, the MARS model can better match different decoder behaviors. On the contrary, the linear regression lacks of the predicted capability to relate the non-linearity to the linearity.

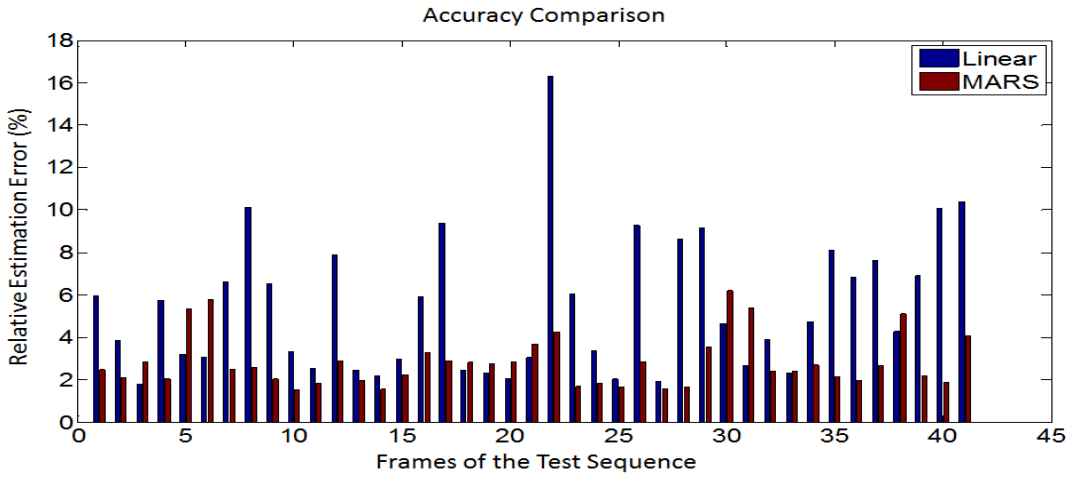


Figure 8-7 LR and MARS Comparison based on One Combined Training Sequence

8.1.4. Modeling techniques Extension

For a deeper evaluation of the model accuracy, the energy consumptions of different decoders have been estimated. Besides the previously employed PHP decoder, new experiments are conducted with the SP decoder, the CBP decoder, and the MP decoder. Note that given the memory limitations of the BeagleBoard, the platform fails to run the CBP, PHP, and MP decoders. Thus, only the MPEG4Part2@SP decoder is tested on the BeagleBoard and to keep the accuracy analysis, the PandaBoard has been employed with the SP decoder, the CBP decoder, and the MP decoder.

Table 8-5 (a) shows the average distribution of the errors of the energy estimation model when the SP decoder is used to decode the set the test sequences on the PandaBoard platform. Models that utilized combined training sequences have good performance with both linear and MARS methods, i.e., the 95% of the models have an average estimation error smaller than 10%. This SP decoder has also been tested on the BeagleBoard platform. The test results are summarized in the Table 8-5 (b). In this case, almost the totality of the models have an average estimation error smaller than 5%. This is because on one hand, the PMC events available on each platform are different, which leads to the model has different errors, and on the other hand, the platform with higher complexity (i.e., the PandaBoard) could be more difficult to obtain its energy estimation model as accurate as the model for the platform with lower complexity (i.e., the BeagleBoard).

Table 8-5 Average Error Distribution of Models Based on Combined Training Sequence of SP Decoder

Table 8-5 (a) On PandaBoard

Methods	<5%	5%-10%	10%-20%	20%-30%	>30%
Linear	76.00%	19.58%	4.42%	0.00%	0.00%
MARS	66.78%	28.99%	4.23%	0.00%	0.00%

Table 8-5 (b) On BeagleBoard

Methods	<5%	5%-10%	10%-20%	20%-30%	>30%
Linear	99.23%	0.77%	0%	0%	0%
MARS	100%	0%	0%	0%	0%

Table 8-6 shows the distribution of the average errors of the energy estimation models when the CBP decoder is used to decoder the set of test sequences. The two fitting methods, linear and MARS, can approximately achieve an amount of 90% and 94% of the models with an average relative estimation error smaller than 10%.

Table 8-6 Average Error Distribution of Models Based on Combined Training Sequences of CBP decoder

Methods	<5%	5%-10%	10%-20%	20%-30%	>30%
Linear	10.74%	78.52%	10.74%	0.00%	0.00%
MARS	14.05%	79.34%	6.61%	0.00%	0.00%

Most existing energy modeling methods assume a linear model. However, the relation between power consumption and system statistics is essentially non-linear. The non-linearity introduces errors when a linear model is used. With high-CV sequences, non-linearity is more serious than in the case of low-CV sequences. As a piecewise method, the MARS method considers the linearity within each data interval, therefore, averaging the global errors. For the CBP decoder, the CV values of the video

sequences vary over a larger range than those of the PHP decoder (from 4% to 190% in PHP decoder and from 4% to 310% in CBP decoder). Thus, the linear method almost double the number of models with an estimation error larger than 10% compare to the MARS method.

Table 8-7 Average Error Distribution of Models Based on Combined Training Sequences of MP decoder

Methods	<5%	5%-10%	10%-20%	20%-30%	>30%
Linear	39.39%	57.30%	3.31%	0.00%	0.00%
MARS	16.67%	77.78%	5.55%	0.00%	0.00%

Table 8-7 shows the distribution of the average errors of the energy estimation models when the MP decoder is used. In this case, both linear and MARS methods have similar estimation accuracy. That is, more than 95% of the models achieve an average error smaller than 10%. Again, this result also shows the CV values of the video sequences impact the estimation accuracy. In this group, all the sequences have a moderate variation of their CV values, ranging from 1% to 20%. Thus, although the linear method has less flexibility than the MARS method, it is still able to capture the sequence characteristics.

In addition, since the PandaBoard has a multi-core processor, the PMC-based energy estimation models have also been verified when the decoder is executed using the two cores. Given that the RVC framework divides the decoder algorithm into several FUs, a proper mapping of the FUs increases the decoder performance. An example of partitions of the four tested decoders (SP decoder, CBP decoder, PHP decoder, and MP decoder) is shown in Figure 8-8 (a)-(c), respectively.

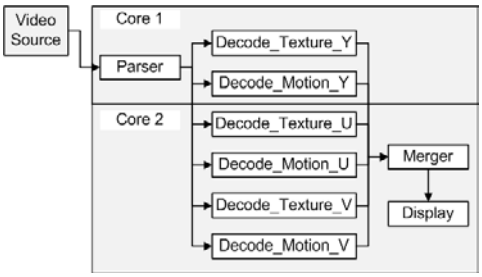


Figure 8-8 (a) MPEG4 Part 2 SP

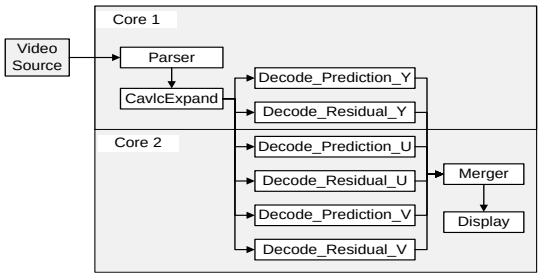


Figure 8-8(b) MPEG4 Part 10 CBP/PHP

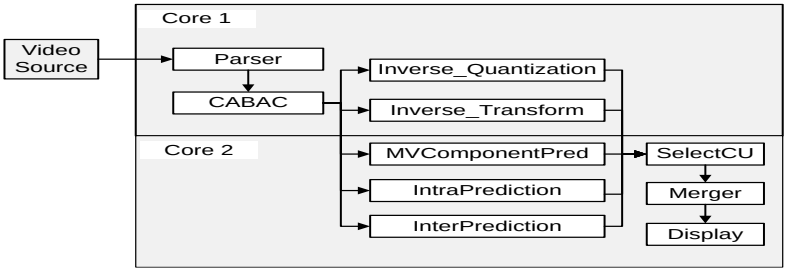


Figure 8-8 (c) MPEG MP

Figure 8-8 Decoder Partition

The contribution to the whole energy consumption on each core is defined by the number of FUs mapped to each core. Thus, to train the sub-model of each core, the dynamic energy model employed is the one indicated in equation 8-3:

$$E_M = \bar{P} \times T_i + E_{c1} + E_{c2} = \bar{P} \times T_i + f_1(PMC_{c1}) + f_2(PMC_{c2}) \quad 8\text{-}3$$

As the same as equation 8-1, $\bar{P}$ and T_i denote the system average idle powers and $Decoder_i$ execution time, respectively. E_{cj} is the estimated incremental energy of $Core_j$. Since the measurement system takes the measurement for the whole platform, it is needed to divide the measured incremental energy into two parts, and each core will use one part to train its own model. A share of the total measured incremental energy is allocated to each core. After several experiments with the decoders, the weights are experimentally set as 1:6 (C1:C2), i.e., $Core_1$ is assumed to consume one-seventh of the total incremental energy and $Core_2$ consumes six times more energy. The estimation results are shown in Table 8-8 (a) to (d).

Table 8-8 Average Error Distribution of Models Based on Combined Training Sequences in Two Cores

Table 8-8 (a) SP Decoder

Methods	<5%	5%-10%	10%-20%	20%-30%	>30%
Linear	78.09%	18.43%	3.48%	0.00%	0.00%
MARS	56.37%	39.06%	7.53%	0.00%	0.00%

Table 8-8 (b) CBP Decoder

Methods	<5%	5%-10%	10%-20%	20%-30%	>30%
Linear	14.17%	67.08%	18.75%	0.00%	0.00%
MARS	53.75%	37.50%	8.75%	0.00%	0.00%

Table 8-8 (c) PHP Decoder

Methods	<5%	5%-10%	10%-20%	20%-30%	>30%
Linear	20.84%	63.78%	15.38%	0.00%	0.00%
MARS	26.68%	68.60%	4.72%	0.00%	0.00%

Table 8-8 (d) MP Decoder

Methods	<5%	5%-10%	10%-20%	20%-30%	>30%
Linear	92.05%	4.55%	3.40%	0.00%	0.00%
MARS	52.56%	39.10%	8.33%	0.00%	0.00%

As can be seen in these tables, when the complexity of the decoder architecture increases, the estimation difficulty also increases. Comparing the percentage of the models which have errors less than 10% in single-core and multi-core modes, the estimation accuracy slightly decreases for the MARS method in the multi-core mode. However, MARS method can still keep this percentage larger

than 90% in all estimation cases. Differently, the modeling performance of the linear method varies from decoder to decoder. For the simple SP decoder, linear method can keep similar accuracy compare to its predictive ability in single-core mode. With the increase of the decoder complexities, the inter-core communication may result in a nonlinear relationship between PMCs and the energy consumption. Thus, more than 15% of the models based on the linear method have an average error larger than 10% for the CBP and PHP decoder. It is surprising that for the MP decoder, linear method has a quite good performance in the multi-core case. One reasonable explanation is that, as an embedded system, the processing capacity and speed of PandaBoard is limited. To decode a MP sequence, which has high computational complexity, the system may be saturated what, in turn, could be expressed as a linear relationship between PMCs and energy consumption.

8.1.5. Model Computation Speed

In this section, the impact of the implementation of the energy estimation model on the decoder performance is analyzed. The number of decoded frames per second (FPS) is employed as the metric to reflect the decoder performance. The FPS reduction rate obtained when the decoder implements the energy estimation model on PB platform is listed in the Table 8-9. The testing video sequences employed are: hit001 and jvc009 (CIF and QCIF, respectively) for the SP decoder; BA2_Sony_F (QCIF) and HCBP1_HHI_A (CIF) for CBP and PHP decoders; BQSquare (416x240) and PartyScene (832x480) for the MP decoder. As can be seen, PMC monitoring and energy estimation have slight influence on the decoder performance. The largest decoder performance decrease for linear and MARS are 3.87% and 3.91% when the decoders are executed using one core, and 3.99% and 4.04% when two cores of PB processor are both enabled for decoding.

Table 8-9 Energy Estimation Impact on FPS (%)

Method		SP		CBP		PHP		MP	
		QCIF	CIF	QCIF	CIF	QCIF	CIF	QCIF	CIF
Linear Estimation	Single-Core	3.87	2.38	2.58	1.94	1.92	1.83	1.55	1.17
	Multi-Core	3.91	2.31	2.18	2.03	2.02	1.94	1.21	1.04
MARS Estimation	Single-Core	3.99	2.40	2.84	1.98	1.96	1.75	1.60	1.19
	Multi-Core	4.04	2.37	2.64	2.03	1.94	1.85	1.60	1.06

In addition, the ratio of the process processing time to the total execution time is used as an indicator to intuitively show the PMC overhead. The processing time is the CPU time when the processor executes the decoder thread, which does not count the occupation of the CPU by other processes and the hardware interrupts. The total execution time is the duration from the decoder start point to its end point. It includes the processing time, PMC operation time, and other operation time during the decoder execution such as thread switch and OS system calls which are caused by

introducing the PAPI functions. It does not distinguish which event or which process is running on the CPU.

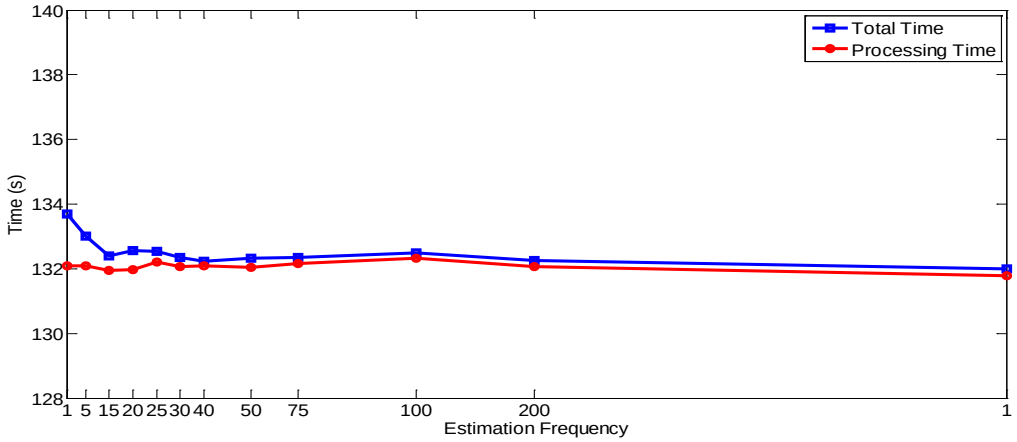


Figure 8-9 (a) PHP Decoder

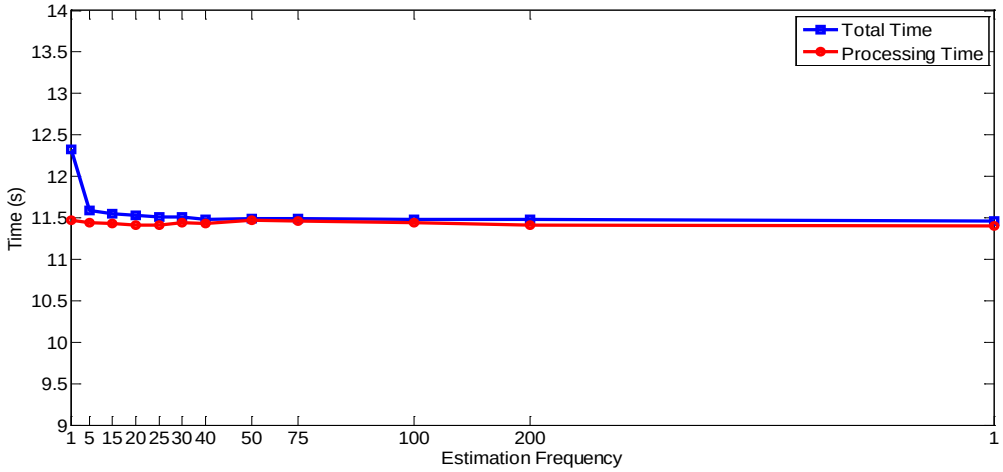


Figure 8-9 (b) CBP Decoder

Figure 8-9 Modeling Overhead

In Figure 8-9, it can be seen that the PAPI functions cause extremely little influence on the decoder performance. The x-axis is the estimation frequency, i.e., to estimate energy every certain number of frames. And the y-axis is the time (total and processing). It is worth to point out that both the processing time and the total time vary from one execution to another due to some un-deterministic issues like operating system interaction, program layout, crossing page boundaries, unaligned instruction fetches, and hardware interrupts, etc. However, this variation fluctuates within a narrow range. The numbers shown in the above figures have been averaged by repeating the same decoder and sequence for 10 times. As can be seen, the processing time concentrates around 132s and 11.4s in two profiles, respectively. The largest modeling overhead happens when the estimation is carried out at a frame basis, which is 1.34% and 6.50%, respectively. When the estimation frequency decrease, the overhead also decreases and in both decoders, in the other situations, all the modeling overhead are lower than 1.50% of the total execution time.

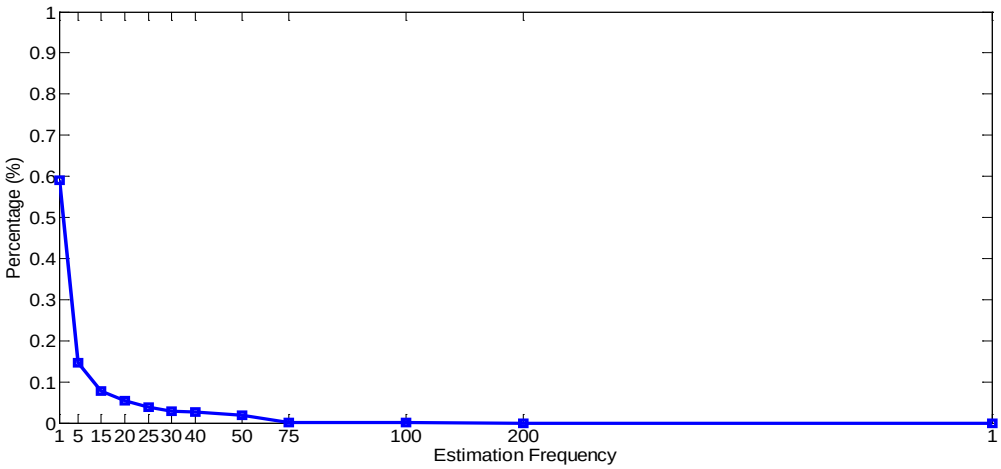


Figure 8-10 Model Computing Time Percentage

Above figures show the overall overhead caused by all the operations and interrupts related to the activity of energy estimation, to evaluate the overhead of modeling computation, a simple method by inserting time stamps can be used. The steps to do the energy estimation are: (1) Stop the PMCs sampling; (2) Do the estimation; (3) Start again the PMCs to sample. The time stamp can help to compute the execution time of these three steps. As shown in Figure 8-10, the computing time in less than 0.6% even in the worst case, i.e., to do the estimation every frame. Thus, the PMC-based method would be able to estimate the energy consumption on-line with a small performance cost.

8.2. Verification of the Energy-aware Manager Implementation

Before illustrating how well the energy-aware manager performs on battery life extension, the implementation of new primitives and the modifications on Jade to implement the energy-aware manager should be verified. Jade is the reconfiguration engine developed to manage both the description of ADMs and the connection of VTLs to produce decoders. On-the-fly reconfiguration to adapt the current energy constraint, ideally, needs a feedback path between the sender and receiver. The energy-aware manager should extract the information from the bitstream to decide the corresponding network connection. It has been introduced in chapter 7 that Jade can work at its scenario mode to execute different events through JSC-formed configuration files. Thus, Jade will pre-load all the decoder networks listed in the configuration file. Various encoded sequences are pre-defined to be loaded accordingly to different battery levels to simulate the metadata-based adaption. It is worth to note that the scenario mode is an emulation of a continuous video streaming decoding. The following figures will show the event configurations and their execution results. Note that only important information is captured to directly present the consequences.

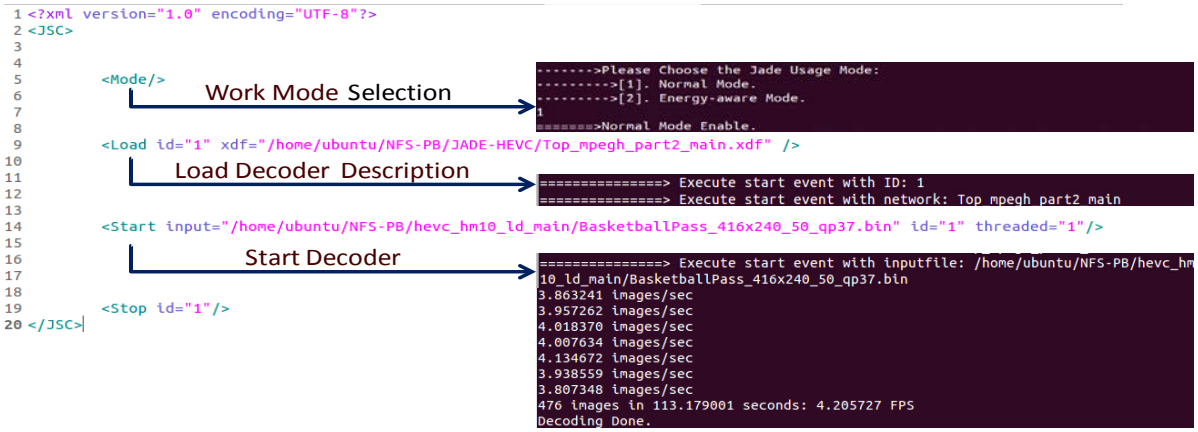


Figure 8-11(a) Normal Mode

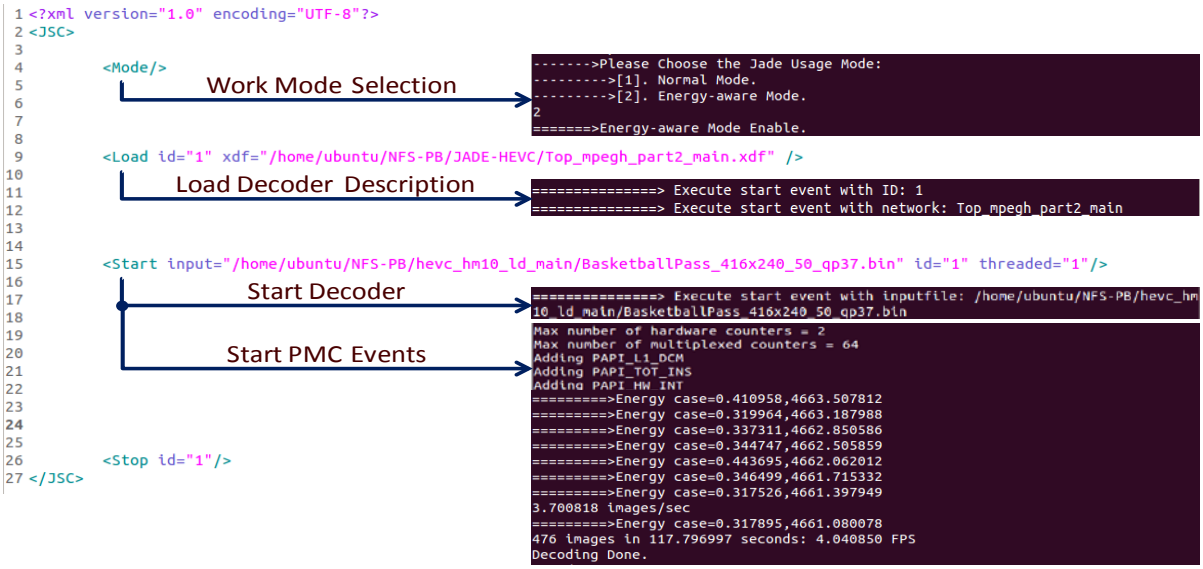


Figure 8-11 (b) Energy-aware Mode

Figure 8-11Verification on Mode primitive

Figure 8-11 (a) and Figure 8-11(b) show how the “Mode” primitive works. Before loading the decoder description and starting to decode a video sequence, the mode event is defined for the work mode. As can be seen in Figure 8-11 (a), if the normal mode is selected, Jade will enable neither the energy estimation nor the energy-aware manager and will not interfere on the decoding process once a decoder is started. The decoder continuously decodes the video sequence until it is finished. Instead, in the case shown in Figure 8-11(b), the energy-aware mode is enabled and the video manager initiates the PMCs. As a result, an estimation of the consumption of each frame is obtained. The remaining battery capacity is thus estimated.

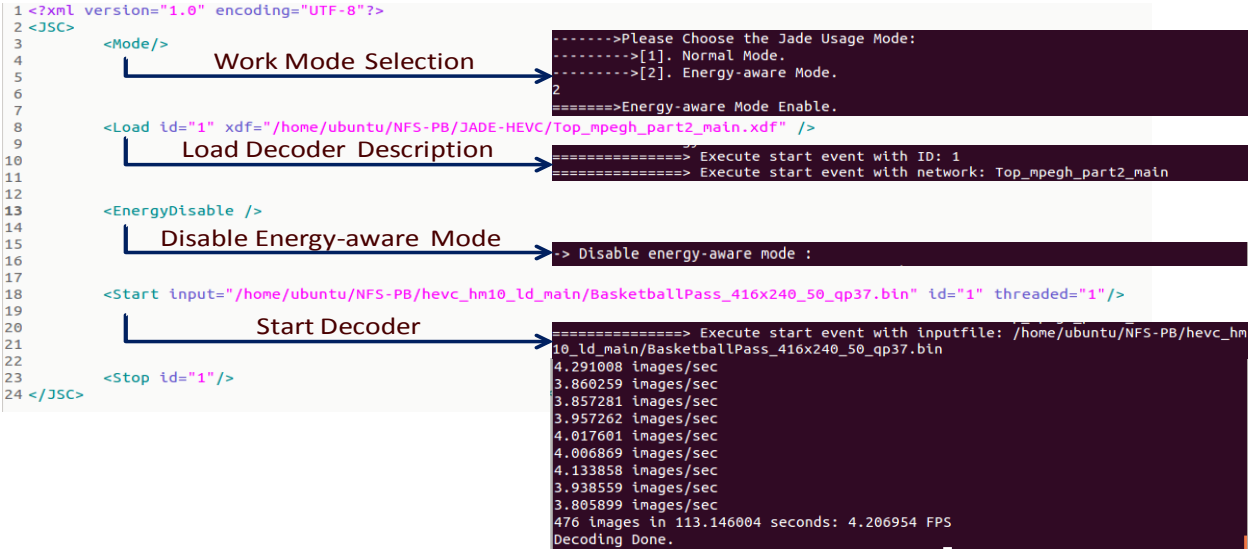


Figure 8-12 Verification on Energy Disable primitive

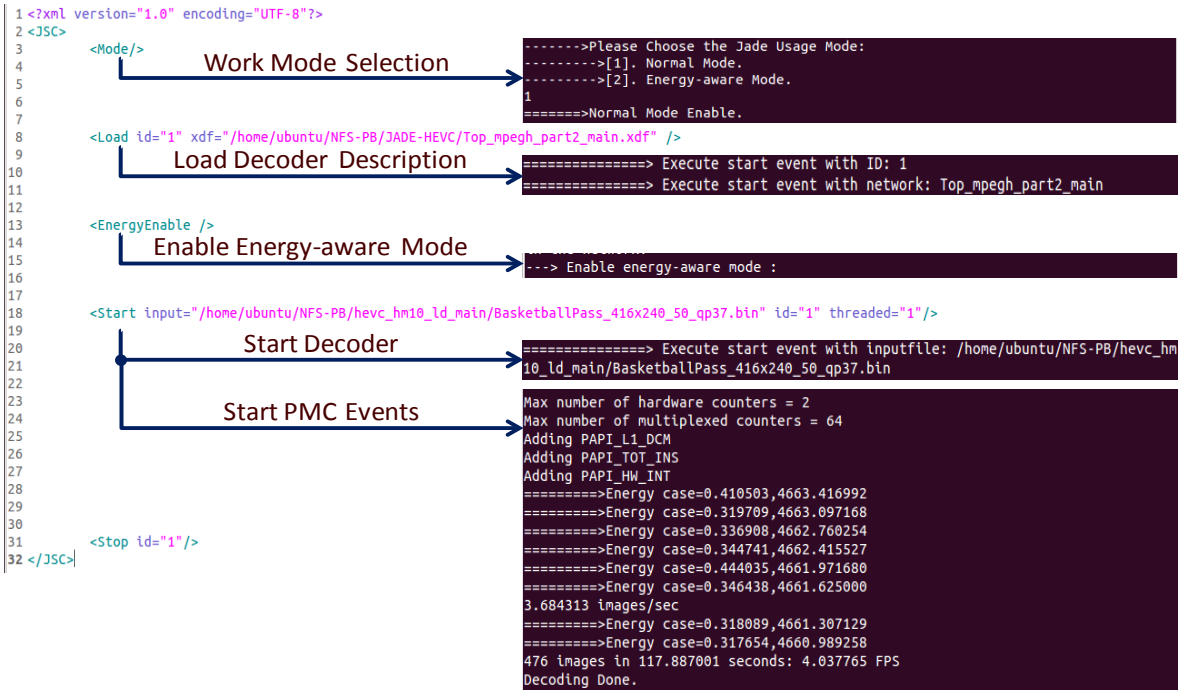


Figure 8-13 Verification on Energy Enable primitive

Figure 8-12 and Figure 8-13 show the proper results of the “EnegyDisable” and “EnergyEnable” primitives, respectively. “EnegyDisable” is used to disable the energy-aware mode. As can be seen in Figure 8-12, although the video manager is set to work at the energy-aware mode, the “EnegyDisable” event can still configure it to work in normal mode. A similar result can also be seen in Figure 8-13 to enable the energy-aware mode after executing the “Mode” event.

Figure 8-14 and Figure 8-15 will generally illustrate how the energy-aware manager controls the decoding process. Note that in the JSC-formed configuration files, the parameter “id” of “Load”

event is the identifier of the decoder description and it will be passed to the “Start” event to define which decoder is executed.

Figure 8-14 Verification on Reconfiguration Control Part 1

Low-battery state is achieved after a long time decoding period. As can be seen in Figure 8-14, the energy-aware mode is disabled, and thus the complete execution of the sequence decoding is accorded to the JSC-formed configuration file, i.e., only the decoder with “id” 1 is executed. Differently, Figure 8-15 shows the decoding control under the energy-aware mode.

Figure 8-15 Verification on Reconfiguration Control Part 2

In this case, as can be seen in Figure 8-15, the decoder has been reconfigured by Jade once the battery capacity is detected lower than the current threshold. Four thresholds, 250J, 100J, 40J, and 0.2J, are pre-defined here as examples. Note that the total battery capacity can be set to any value through the battery emulator. At the full battery stage, the configuration file specifies that the decoder with id 1, i.e., the PHP decoder, is the one to be employed. As can be seen in Figure 8-15, the printed message shows that this decoder is started. Once the remaining battery is lower than the first threshold, the decoding process will stop and the energy-aware manager will inform Jade to

reconfigure a new decoder with another id, which is calculated by the management metric. In this example, the new decoder with id 2 is selected, i.e., the CBP decoder. Please note the blue arrows in Figure 8-15. Here is the difference between the configuration file and a real execution. The configuration file defines only to use the PHP decoder, while the energy-aware manager controls to use the CBP decoder. Two more reconfigurations occur when it is detected that the battery level is below the other two thresholds.

8.3. Battery life Extension

Since the energy estimation model has demonstrated its accuracy on different decoders and the functionality of the energy-aware manager has been verified, to include them into the RVC framework for energy saving appears feasible. Experiments to show the battery life extension have been conducted on the PB.

8.3.1. Experiment on Decoder Reconfiguration

Previous analysis [178] has reported that the HEVC standard may increase the complexity by up a factor of two compared with the AVC/H.264 standard. Thus, an AVC/H.264 decoder could be a good candidate when the remaining battery level falls below a certain threshold. In the first experiment, the PHP decoder and the CBP decoder are employed to test the energy improvement by changing the decoder description. The MP, PHP and CBP decoders are switched between each other based on an energy-aware management decision.

With the same energy budget (i.e., 3600 Joules, in this experiment) set by the battery emulator, a clear playback extension is shown in Figure 8-16, when Jade runs in the energy-aware mode. The black solid line denotes the pre-defined energy threshold, and in this case, it represents one third of the total energy budget (i.e., 1200 Joules). The other three red-plus, green-point, and blue-diamond lines illustrate how Jade operates in normal mode and two energy-aware modes, respectively. In all cases, encoding parameters such as the quantization parameter (QP) and resolution have been adjusted to run the test with similar quality. The turning point in the plot corresponds to the situation in which the remaining battery level is detected at less than the predefined threshold. In all three test cases, the energy is consumed at the same rate before the turning point. Without any energy considerations, Jade continues running the same decoder (MP), and the battery runs out at *1448s*. In the energy-aware mode, two reconfiguration decisions were tested. The green-point line shows the battery life extension created by switching from a MP decoder to a less energy-consuming decoder, i.e., the PHP profile of H.264/AVC standard. In this situation, the battery lasts for *1656*s, an increase of a *14.4%* with respect to the previous operational time. The blue-diamond line provides another energy-saving possibility by

reconfiguring the decoder to the CBP profile. The battery runs out at *1764*s, which achieves an increase of a *21.8%* in the operational time compared with that of the MP decoder. In the two energy-aware cases, the potential for energy efficiency improvement has been achieved by reconfiguring the HEVC decoder to become the H.264/AVC one. Additionally, the result also shows a *6.5%* (from *1656s* to *1764s*) increase of the playback time when Jade uses the CBP profile instead of the PHP profile during the reconfiguration. This is because the PHP decoder has relatively high compression ability and supports higher quality applications than the CBP decoder, at the expense of a large amount of energy consumption.

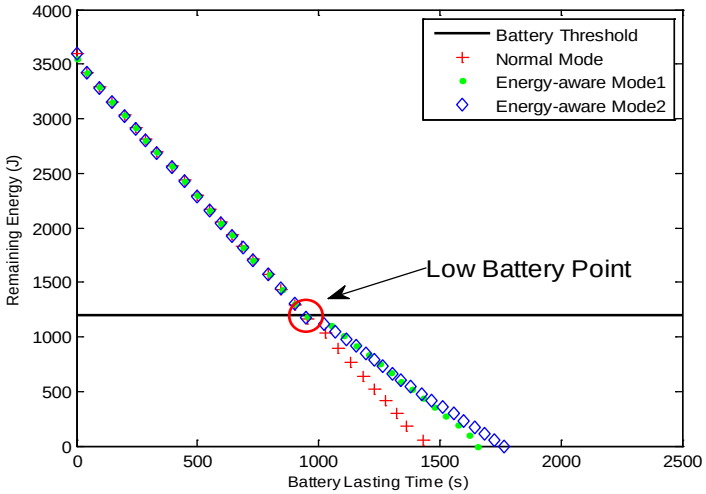


Figure 8-16 Energy Efficiency Improvement by Reconfiguration

8.3.2. Experiment on Coding Parameter Change

One of the goals of this work is to achieve energy efficiency improvement via decoder reconfiguration. However, in point-to-point or broadcasting scenario, the energy-aware manager might take the decision to choose a battery lifetime extension by informing the encoder to reconfigure its encoding parameters. When the energy-aware manager detects a low-battery state, it will message the encoder to encode the raw data with less complexity. The block diagram of a simplified diagram of a video encoder/decoder is shown in Figure 8-17.

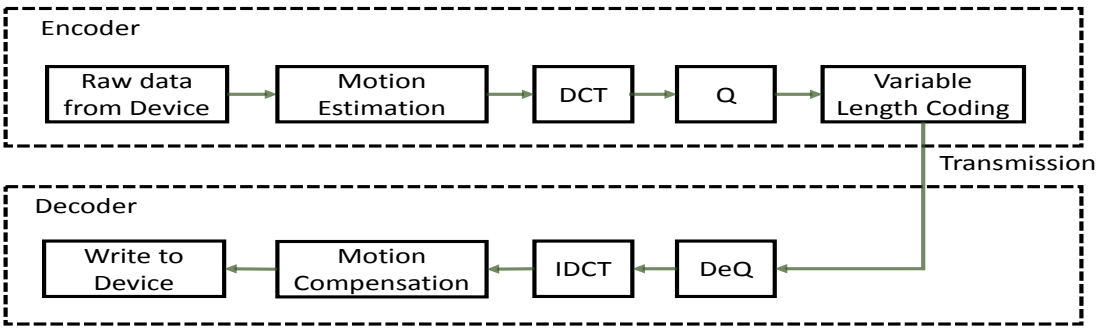


Figure 8-17 MPEG Codec Algorithm

As can be seen, decoding is a reverse processing of encoding, thus, the energy consumption of the decoder is affected by the way the encoder performs on video data compression. An example of influence on computational complexity by encoding with different algorithms has been partly shown in section 5.2.2. More details about factors that impact on the sequence quality and encode/decode efficiency are listed below:

- The period of the I frames, i.e. the distance between the two I frames. Decreasing the period of I frames will improve the objective video quality, but it will also increase the network load.
- Number of P and B frames. The encoding/decoding of P and B frames introduces more computational load, regarding to the encoding/decoding of I frames. As an example, if the P to I frame ratio is increased, more energy will be consumed in both, the encoding and the decoding process, because I frames do not require motion estimation and compensation as P frames do.
- Data Rate. It is one of the most important elements to control the image quality. Generally speaking, with the same resolution, the larger data rate, the lower compression rate.
- Resolution. Both, spatial and temporal resolutions are primary factors in determining data rate. General speaking, a higher resolution requires a higher encoding/decoding workload as well as more energy.
- Quantization Parameter (QP). QP is one of the most important encoding parameters. QP has very much influence on the image quality. As an example, in work [132], a decrement on the QP from 32 to 27 produces a 3dB increment on the PSNR (Peak Signal to Noise Ratio), approximately. In addition, the lower QP value also results in higher energy consumption.
- Entropy Coding. Different entropy encoding/decoding schemes will introduce different computational complexity and energy consumption. As an example, the H.264/AVC standard defines two kinds of entropy coding: CABAC and CAVLC. CABAC is a lossless encoding which provides good quality. CAVLC uses less CPU resources, but it affects the image quality. Note that if the entropy coding is changed, the decoder needs to be reconfigured.

Note that factors listed above include two kinds of impact issues. One is the encoding algorithm (entropy coding) and the other one is the coding parameters (all the factors except entropy coding). In this case, the energy-aware manager could inform the encoder when and how to adjust and optimize the parameters or encoding tools to meet the energy constraints. Note that if the encoder changes the coding parameters, the decoder does not need to be reconfigured. The following two experiments are conducted by changing the coding algorithm and coding parameter, respectively, when the energy-aware manager detects the low battery states.

Figure 8-18 shows the energy savings created by changing the QP parameters of a PHP-supported sequence under the same experimental environment as the previous one in section 8.3.1. The red-plus and green-point lines show how energy is consumed when the video stream is encoded with two different QP parameters, respectively. By increasing the QP value from 27 to 32, the playback time is extended from *1426s* to *1623s*, i.e., a *12.1%* improvement.

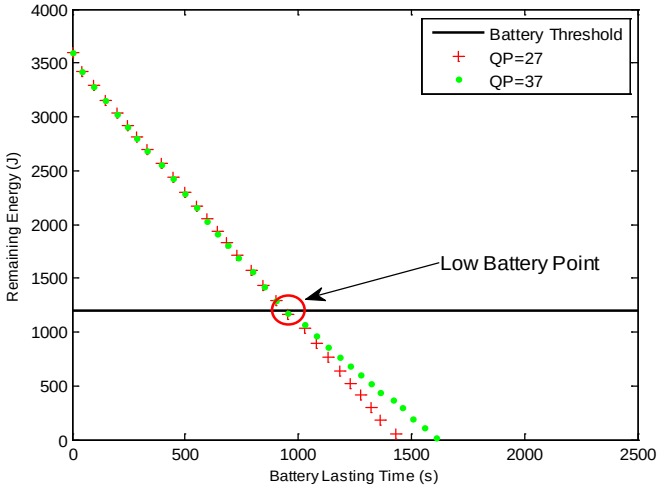


Figure 8-18 Energy Efficiency Improvement by Changing the QP

A proper selection and adaption of encoding parameters may have a moderate influence on energy consumption. Since a decoder is a connection of various functional units, both a combination of FUs replacing and parameter adaption could achieve more efficiency on battery lifetime extension.

8.4. Conclusion

This chapter assesses the thesis proposal, which is an energy management and optimization mechanism including an energy estimation model and an energy-aware manager based on a functional-oriented reconfiguration engine, namely jade.

The energy estimation model for decoders has been first validated and evaluated with a set of selected PMC events on the PB platform. Conducting an analysis on this result, a suggestion to select training data to achieve more accurate models is introduced. This suggested method is repeated on the PB and BB platforms. All the results show good accuracy following the guide of training data selection, i.e., more than 90% of the models achieve an average error smaller than 10%, especially when using the MARS fitting method. In addition, the overhead of PMC monitoring and energy estimation is proven to have slight influence on the decoder performance. Based on the accurate energy estimation model, the energy-aware manager is able to potentially save energy consumption during the decoder execution. The modifications of Jade to include the energy-aware manager are first tested and verified. Results show that Jade can correctly respond under the different test cases. As a

consequence, by combining the estimation model and the energy-aware manager into the RVC framework, the experimental results show a good potential of energy efficiency improvement with increases of about 14.4%, 21.8%, and 12.1%, respectively in three different test cases.

PART F

Chapter 9: Conclusion and Future Work

9. Conclusion and Future Work

The capabilities of wireless mobile devices have been growing on full-scale except for their energy source, the battery, which has experienced a relatively slow development. Therefore, it has been a critical issue to optimally use the limited battery energy under certain performance requirements and quality of service. This thesis work has focused on the energy optimization and management mechanism for video decoding applications based on functional-oriented reconfiguration to reduce their energy consumption, and thus, to extend the battery lifetime. The functional-oriented reconfiguration is one of the reconfiguration techniques which is platform-independent and aims to improve system functionalities or produce new functionalities by re-connecting the existing functional units. An energy estimation model and an energy-aware manager have been implemented to support the energy consumption control. The obtained results show a good potential to increase the battery life time. In this chapter, a conclusion of the thesis work and future research direction will be drawn.

9.1. Conclusion

9.1.1. Motivation and Results of the Proposed Energy Optimization and Management Mechanism

Along with the great developments of semiconductor and wireless communication technologies, mobile devices such as Smartphones and tablets have been blended into people's daily life and promote a great progress on multimedia utilization. Video streaming playback decoders, which occupy a large percentage of multimedia applications, have been mainly designed to optimize the decoding speed during a long time. However, the usefulness of mobile devices is severely limited by the battery capacity which is far behind of the devices demands. Battery lifetime is an important factor to assess user's satisfaction on a mobile device. A failure to guarantee the user desired lifetime can significantly degrade the user experience on a product, and make it unacceptable.

Motivated by the energy constraints, the research interests have been shifted from pursuing maximum performance to tradeoff between energy and performance for energy saving. For the streaming applications which usually execute for long periods, energy-saving is especially important. This thesis has addressed this problem with the goal of reducing the energy consumption. A technique for energy optimization and management through the functional-oriented reconfiguration of video decoders has been proposed. Reconfigurable video is a new design philosophy based on data flow and parallelism. It aims to provide a uniform framework to facilitate the design of next generation video codecs and the consistency of encoders and decoders among various coding standards. Unlike traditional monolithic designs, reconfigurable video designs are able to replace or add functional units

without disorganizing the whole decoder network and thus it is proved to be a flexible technique to satisfy various user demands. Reconfigurable video design leads to a new direction for energy optimization on video streaming applications. One of the contributions of this dissertation to the state-of-the-art in energy-aware design on video coding is to introduce reconfiguration to the energy saving field to provide a flexible solution without specific details from either the hardware platform or the used coding standards. In this work, video decoders are implemented through the reconfigurable video coding (RVC) standard which allows connecting functional units from video tool libraries (VTLs) to form a complete video decoder. This framework allows a high degree of flexibility and scalability as the encoders and decoders can dynamically adapt themselves based on both the current battery capacity and user preferences to achieve a better utilization of the limited energy in multimedia applications.

The proposed energy optimization and management mechanism is implemented at the decoder end. It consists of an energy-aware manager implemented as an additional unit of the reconfiguration engine, an energy estimation model, and if available, a feedback channel connected to the encoder end. The reconfiguration engine is a tool to reconfigure the decoders. During video streaming playback, the energy-aware manager estimates the battery lasting time based on the monitored battery level and the predicted energy-consuming rate from the energy estimation model. Once it is detected that the remaining battery is not sufficient for user desirable lasting time, it will inform the reconfiguration engine to reconfigure the decoder for consuming less energy. If the feedback channel from the decoder to the encoder is available, the manager can inform the encoder unit to change either the encoding parameters or the encoding algorithms for energy-saving adaption.

Although the ultimate objective of energy saving is achieved by efficiently switching the decoder implementation, how and when to reconfigure the decoder for energy consumption adaption is extremely important. Considering that a model to estimate the energy consumption is very helpful to lead to elegant and correct energy management decisions, this work has launched an accurate and practical energy estimation model for the CPU, memory, and basic processor peripherals based on the use of PMCs. In most modern processors, PMCs are implemented as special-purpose registers to monitor the occurrences of several events such as the cache miss and hardware interrupt and can be managed by high-level tools. Although the PMC-based modeling technique has been widely used in many research works, a departure from previous researches in this thesis work is to propose a methodology for PMC events selection without manual intervention, which supports multiple hardware platforms and video coding standards. In particular, a PMC-filter is implemented to automate the selection of the most appropriate PMC events that affect energy consumption and reflect the energy behavior of applications. Furthermore, a detailed study on the influence of the training data

on model accuracy has been presented for better model building. The modeling methodology has been evaluated on different underlying platforms, single-core and multi-core, and different characteristics of workload, including the use of MPEG4 Part2 SP, MPEG4 Part10 CBP, MPEG4 Part10 PHP and MPEG HEVC decoders. All the results show a good accuracy and low on-line computation overhead.

Besides the energy estimation model, an energy-aware manager has been implemented to take the charge of the energy consumption control. How to choose the proper decoder for reconfiguration when the low battery situation occurs is the main challenge to design the management metrics of the energy-aware manager. A good decision should comprehensively consider the decoder computational complexity, the image quality, and the desirable battery lasting time. In this work, this decision has been simplified. To choose a new decoder, only the computational complexity has been considered. In addition, both new events of the reconfiguration engine and the required modifications on the engine to implement the energy-aware manager have been implemented. The reconfiguration engine provides three usage modes to implement a decoder, namely command line, console, and scenario mode. The scenario mode is the most powerful one to manage decoder configuration and execution. In this mode, the engine uses pre-defined XML events to manage the decoder configuration and execution. Three events, namely "Mode", "Enable", and "Disable", have been added to enable the energy-aware manager for energy optimization and management. The modifications on the reconfiguration engine focus on the communication between the engine and the decoder. A signal-based inter-process communication scheme and a shared-memory scheme have been implemented to communicate and pass data between them, respectively. These implementations have been verified to test the engine correctly responds to the reconfiguration decisions made by the energy-aware manager under the different test cases.

Integration of the energy estimation model into the RVC framework along with the energy-aware manager included in the reconfiguration engine achieves easy and flexible reconfiguration management for the energy saving criteria. The experimental results indicate a possibility to lengthen the battery lifetime in three energy-aware test cases: reconfiguring the HEVC decoder to a PHP or a CBP H.264/AVC decoder, and adjusting the QP coding parameters. The experimental results carried out on different test cases show a good performance of the proposed energy-aware optimization mechanism, which allows significant increases in the battery lifetime by functional reconfiguring the decoders.

9.1.2. Exploitation of Implementing the Modeling Method on FPGA Systems

On the GPP-based embedded systems, the information provided by PMC events works as a profiler of the system behavior. However, FPGA systems may require a more hardware-aware profiler. Some contributions have been achieved in works [169] [170]. The University of Toronto has implemented a snooping software profiler to count the total number of cycles during an application execution [169]. In this work, they have developed an on-chip, real time, FPGA-based profiler, Airwolf, for the Nios II processor to be synthesized on Altera FPGAs. Airwolf works similarly as PAPI. It inserts software drivers around the software functional blocks to enable or disable particular counters implemented in Airwolf. These FPGA-based profiling tools have similar functionalities as PMCs embedded processors or general purpose processors. They can be considered as a proxy of the system behavior and used to estimate the system energy consumption. Therefore, the PMC-based modeling methodology is also applicable to the FPGA systems with their profiler tools.

In addition, in spite of the fact that the PAPI-based implementation of the proposed methodology addresses software solutions, the scope of the PMC-based energy estimation methodology could be enlarged to drive hardware-based RVC tools and methodologies such as the dynamic partial reconfiguration (DPR) of FPGA [181] or the multi-dataflow composer tool (MDC) [182]. Effectively, to achieve run-time hardware reconfiguration, the MDC tool assembles several specifications and inserts multiplexers to switch the data-flow through a shared set of actors, while the DPR permits the reconfiguration of specific parts of an FPGA. To exploit the proposed methodology, tools to directly insert specific PMCs into hardware description language (HDL) specifications exist [183]. Once PMCs are inserted and the generated HDL code is synthesized, neither multiplexer selection nor partial reconfiguration prevents the event count gathering to obtain the energy estimations.

9.1.3. Publications

Journals:

- **R. Ren**, E. Juárez, C. Sanz, M. Raulet and F. Pescador, “Energy-Aware decoder management: a case study on RVC-CAL specification based on just-in-time adaptive decoder engine,” IEEE Trans. on Consumer Electronics, vol. 60, no.3, pp. 499-507, Aug. 2014.
- J. Wei, **R. Ren**, E. Juarez and F. Pescador, “A Linux Implementation of the Energy-based Fair Queuing Scheduling Algorithm for Battery-limited Mobile Systems,” IEEE Trans. on Consumer Electronics, vol. 60, no.3, pp.267-275, May 2014.
- **R. Ren**, J. Wei, E. Juárez, M. Garrido, C. Sanz and F. Pescador, “A PMC-driven methodology for energy estimation in RVC-CAL video codec specifications,” Signal Processing:

Image Communication, vol. 28, no. 10, pp. 1303–1314, Nov. 2013.

Conferences:

- **R. Ren**, E. Juárez, C. Sanz, M. Raulet and F. Pescador, "Energy-Aware Decoders: a Case Study Based on an RVC-CAL Specification," Conf. on Design and Architectures for Signal and Image Processing, Oct. 2014.
- **R. Ren**, E. Juárez, C. Sanz, M. Raulet and F. Pescador, "On-line Energy Estimation Model of an RVC-CAL HEVC Decoder," pp.63-64, Int. Conf. on Consumer Electronics, Jan.2014.
- **R. Ren**, E. Juárez, C. Sanz and F. Pescador, "On-line Energy Estimation Methodology for RVC-CAL Video Codec Specifications," Conf. on Design of Circuits and Integrated Systems, Nov.2013.
- **R. Ren**, E. Juárez, C. Sanz, M. Raulet and F. Pescador, "System-Level PMC-driven Energy Estimation Models in RVC-CAL Video Codec Specifications," Conf. on Design and Architectures for Signal and Image Processing , pp.55-62, Oct.2013.
- **R. Ren**, E. Juárez, F. Pescador and C. Sanz, "A Stable High-Level Energy Estimation Methodology for Battery-Powered Embedded Systems," IEEE Int. Symp. on Consumer Electronics , pp.1-3, Jun. 2012.

MPEG Meetings:

- **R. Ren**, E. Juárez, M. Raulet, J.G. Wei, M. Garrido et al, m33115: Energy-aware Reconfiguration based on a Just-im-Time Adaptive Decoder Engine (JADE). ISO/IEC JTC1/SC29/WG11, 108th MPEG Meeting Document Register, Mar.-Apr. 2014.
- E. Juárez, **R. Ren**, M. Raulet, J. G. Wei, M. Garrido et al, m31243: Performance Monitoring for Energy Estimation in RVC-CAL Description. ISO/IEC JTC1/SC29/WG11, 106th MPEG Meeting Document Register, Oct.-Nov. 2013.
- E. Juárez, **R. Ren**, J. G. Wei, M. Raulet, M. Garrido et al, m28171: Proposal of a Decoder Energy Management Scheme with RVC. ISO/IEC JTC1/SC29/WG11, 103th MPEG Meeting Document Register, Jan. 2013.
- E. Juárez, **R. Ren**, J. G. Wei, M. Raulet, M. Garrido et al, m25903: RVC Inverse Transform FU for HEVC. ISO/IEC JTC1/SC29/WG11, 101st MPEG Meeting Document Register, Jul. 2012.

9.2. Future Work

As proposed in this thesis, to efficiently extend the battery lifetime through the energy optimization and management, two aspects, first, the accurate estimations on energy consuming rates and remain battery capacities and, second, the appropriate reconfiguration decisions are required. The future work will be focused on these two aspects.

From the accuracy point of view, first of all, a system-wide energy estimation model is needed. The test cases considered in this thesis have been carried out using two embedded platforms without the consideration on processer, memory and peripheral units. As discussed in the work [168], the hardware performance information is not only available on processors, but also scattered at the whole platform. For example, other important components such as the graphics processing unit, the memory interface and the network interface also have hardware to monitor various events that can supply information about the system performance. This kind of counters will provide additional information to include into the energy model for a heterogeneous system. The third-party tool, PAPI, used in this methodology, is able to extend the utilization of the modeling methodology. A branch of the PAPI tool, named as the component branch (or PAPI-C), has been designed to count events beyond the CPU. PAPI-C applies APIs to configure and count events related to other components such as network or memory controllers, power or temperature monitors or even specific processing units. In the energy estimation model introduced in this paper, an event which records the number of hardware interrupts is employed to include the information related to with the energy consumption on the peripherals. This event gives the model a general system-wide estimation not only limited to the CPU and memory components. However, more specific models can be individually built with more component details. Moreover, the battery discharging characteristics and a precise battery discharging model under different thermal conditions still need an exhaustive investigation to give a more accurate estimation on the remaining battery level.

From the management point of view, as a first step on battery life extension based on reconfiguration, the energy-aware manager in this thesis only considers the reduction on the computational complexity to save energy when it makes the reconfiguration decisions. However, the problem can be extended in various directions. Image quality and network delay are two QoS requirements for user satisfaction on streaming applications. To provide a low-distortion and high compression ratio decoder, high computational complexities are needed. In turn, this leads to high energy consumption, which is against the original intention of the energy-aware manager. These three issues, the compression ratio of the video data, the video quality, and the battery life time are mutually restrained. Thus, to achieve satisfactory battery life time both, video quality adaption and network transmission adaption, should be taken into account. The energy-aware manager should comprehensively consider how to balance the computational complexity, the network bit-rate constraint, and the distortion of multimedia delivery. This tradeoff depends on the relationship and interaction between computational and channel parameters, thus, it is worth to do research on management metrics allowing to built a more complete management metric which jointly selects video source parameters and channel parameters based on the video content characteristics, available network resources, underlying network conditions and user preferences. In addition, the energy-aware

manager can include other low-power design methods such as dynamic voltage and frequency scaling for further energy reduction.

As described above, the management point will be the heart of the future research. It comprehensively considers different elements to provide more efficient and practical energy management strategies.

Appendix A: Introduction of Battery Emulator Usage

Figure 6-9 Figure- 1, the Graphical User Interface (GUI) of the battery emulator and Figure-2, the DLOG graph for current and voltage observations, are two mainly used interface used for taking the measurements. The usage is described following:

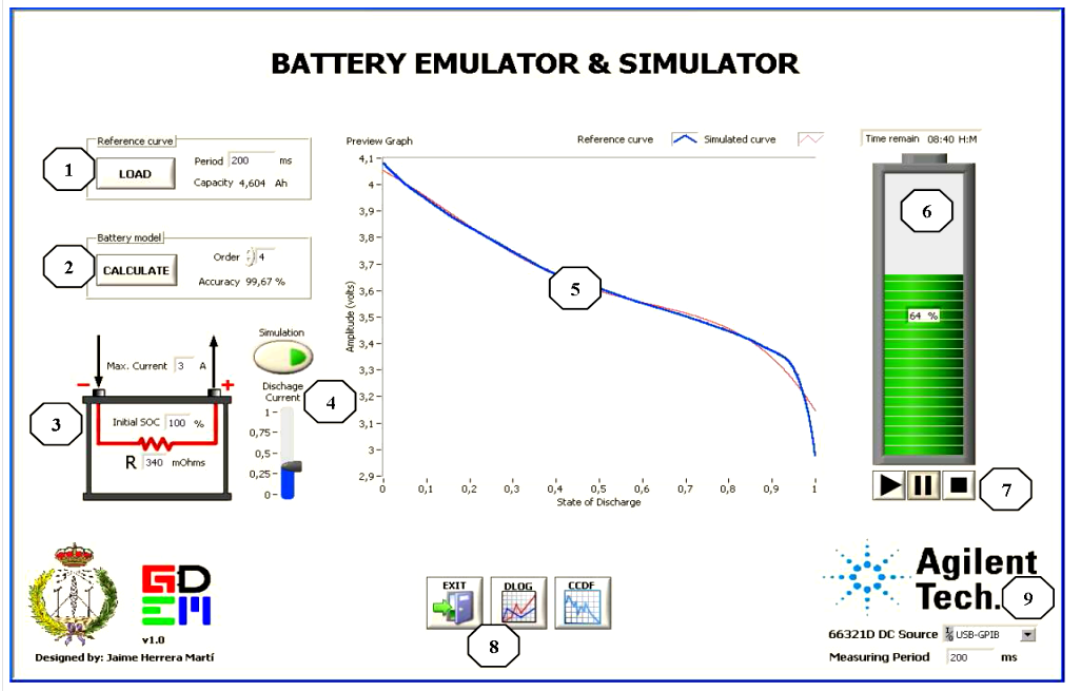


Figure- 1 GUI of the Battery Emulator and Simulator [157]

- Icon 1: To load the reference curve. First, enter the period used to sample the reference curve of the battery state of charge, e.g., 200ms, and then press the button 'LOAD' to load the curve.
- Icon 2: To calculate the battery model. Enter the degree of the polynomial reference curve and then press the button 'CALCULATE' to calculate the model. Depending on the introduced degree, the model accuracy is visually shown and the higher the degree, the more precise the model will be. The range of degree is set between 2 and 10 in this emulator.
- Icon 3: To introduce the battery parameters. This interface is used to set the battery internal resistance (bottom), the initial state of the battery (middle) and the maximum current that the battery can deliver (top). The capacity of the battery is not introduced because it has already been obtained from the reference curve.
- Icon 4: To choose the usage mode. If the simulation mode is enable, the program will emulate the behavior of the battery according to the values of the current that is introduced through controlling the 'discharge current' slider (bottom). If the simulation is disable, the

program will emulate the behavior of the battery according to samples of the actual current measured by the power supply.

- Icon 5: Graphical preview. The reference graph appears in blue in this chart and the battery model is shown in red.
- Icon 6: Battery charge indicator. Indicates the charge status of the battery and the remaining lifetime according to the current consumed at this moment.
- Icon 7: Emulation control buttons. Press 'PLAY' to start the battery emulator, press 'PAUSE' to temporarily stop it and press this button again to continue, press 'STOP' to stop the simulation.
- Icon 8: Function buttons. Press 'EXIT' to close the program, press 'DLOG' to switch the window of the graph of current and voltage evolution and press 'CCDF' to show the cumulative distribution of the current.
- Icon 9: Power supply control. Choose from the pull-down list the connect port of the power supply. Make sure that the power supply is switched on and the driver of National Instruments is enable. Enter the current sample period for current and voltage sampling.

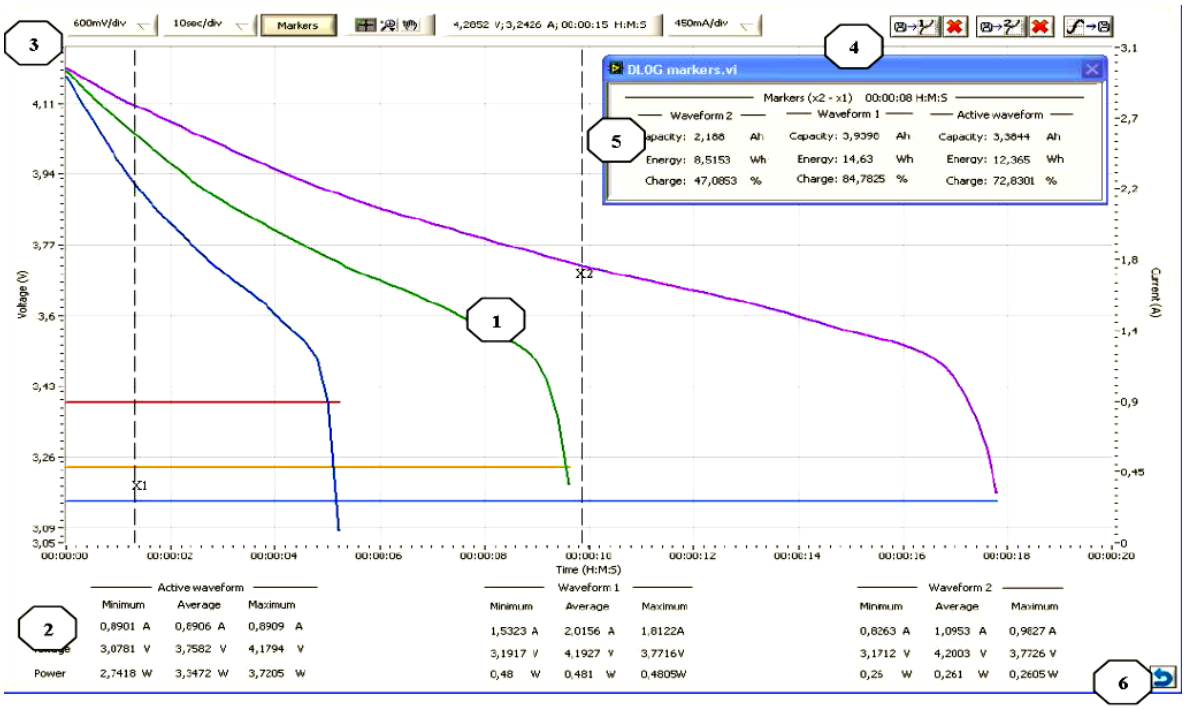


Figure- 2 Graph of the Current and Voltage Evolution [157]

- Icon 1: Graphical representation. There are three axes in this graph: voltage axis (left), time axis (bottom) and current axis (right). Three different curves are presented in the graph. The "Active curve" (voltage is shown in blue and current is in red) is the one being measured. In addition, two reference curves can be loaded trough the "Open curve" button located at the top-right corner in this window. Current sampling curve can also be saved into a text file

trough the “Save curve” button.

- Icon 2: Curve information indicator. These tables indicate the minimum, average, and maximum of current, voltage and power of the active curve and the two reference curves, respectively.
- Icon 3: Graph control. Three pull-down lists can control the scale of the axes of voltage (mV/div), time (HMS/vid) and current (mA/div), respectively. The “Markers” button opens a pop-up window with additional information between two markers (X1 and X2). There is also an indicator of the mouse position which is given by the coordinate of three axes.
- Icon 4: Curve button. Three buttons are used to load two reference curves button and to save the current curve.
- Icon 5: Pop-up window shows the battery capacity, energy consumption and remain battery between X1 and X2 markers for each curve.
- Icon 6: Button to return the initial window.

References

[1]. D.D. Sheldon, “How the Internet Has Revolutionized Video Marketing,” A Senior Thesis Submitted in Partial Fulfillment of the Requirements for Graduation in the Honors Program Liberty University, Fall 2013.
[2]. Cisco report, “Cisco Visual Networking Index: Forecast and Methodology 2010-2015.”
[3]. ComScore whitepaper, “The 2013 Europe Digital Year in Review”.
[4]. ComScore whitepaper, “The 2013 US Digital Year in Review”.
[5]. R. B. Mamatha and N. Keshaveni, “Comparative Study of Video Compression Techniques-H.264/AVC,” Int. Journal of Advanced Research in Computer Science and Software Engineering, vol.4, no.11, pp. 874-877, 2014.
[6]. A. Carroll and G. Heiser, “An analysis of power consumption in a smartphone,” Proc. of the 2010 USENIX Conf.on USENIX annual technical Conf, 2010.
[7]. G. P. Perrucci, F. H. P Fitzek, and J. Widmer, “Survey on Energy Consumption Entities on the Smartphone Plarform,” IEEE 73rd Vehicular Technology Conf, pp.1-6, May 15-18, 2011.
[8]. S. Kim, H. Kim, J. Hwang, J. Lee, and E. Seo, “An event-driven power management scheme for mobile consumer electronics,” IEEE Trans. Consumer Electron., vol. 59, no.1, pp. 259-266, Feb. 2013.
[9]. B. Li, and S. Park, “Energy efficient burst scheduling in mobile TV services,” IEEE Trans. Consumer Electron., vol. 59, no.1, pp. 24-30, Feb. 2013.
[10]. I. Richardson, M. Bystrom, S. Kannangara and M.F. Lopez, “Dynamic Configuration: Beyond Video Coding Standards”, in Proc. IEEE Int. Conf.on System n Chip, Sept, 2008.
[11]. J. M. Donelan, Q. Li, V. Naing, J. A. Hoffer, D. J. Weber, and A. D. Kuo, “Biomechanical Energy Harvesting: Generating Electricity During Walking with Minimal User Effort,” Journal of Science, vol.319, no.5864, pp. 807-810, 2008.
[12]. V. Tiwari, S. Malik and A. Wolfe, “Power Analysis of Embedded Software: A First Step Towards Software Power Minimization”, IEEE Trans on VLSI Systems, vol.2, no.4, pp. 437-445, Dec. 1994.
[13]. V. Tiwari, S. Malik, A. Wolfe, and M. T. C. Lee, “Instruction Level Power Analysis and Optimization of Software,” Journal of VLSI Signal Processing, vol.13, no.2, pp. 1-18, 1996.
[14]. M. T. C. Lee, V. Tiwari, S. Malik, and M. Fujita, “Power Analysis and Minimization Techniques for Embedded DSP Software”, IEEE Trans. on VLSI Systems, pp. 123-135, bol.5, no.1, 1997.
[15]. B. Klass, D. E. Thomas, H. Schmit, and D. F. Nagle, “Modeling inter-instruction energy effects in a digital signal processor,” in Proc. of the Power Driven Microarchitecture Workshop in Conjunction with the 25th Int. Symp. Computer Architecture, June 27-July 1, 1998.
[16]. A. Sama, J. F. M. Theeuwen, and M. Balakrishnan, “Speeding up power estimation of embedded software,” in Proc. of the Int. Symp. on Low Power Electronics and Design, pp. 191–196, 2000.
[17]. 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.
[18]. J.Laurent, N.Julien, E. Senn, E. Martion, “Functional level power analysis: An efficient approach for modeling the power consumption of complex processor”, Proc. of the Conf.on Design, automation and test in Europe, vol.1, pp.10666, Feb.2004.
[19]. E. Senn, J.Laurent, N. Julien and E. Martin, “SoftExplorer: Estimation, characterization and optimization of the Power and Energy Consumption at the Algorithmic Level”, IEEE power and timing modeling, optimization and simulation, 2004.
[20]. E. Senn, D. Chillet, O. Zendra, C. Belleudy, S. Bilavarn and et al., “Open-People: Open Power and Energy Optimization Platform and Estimator”, Euromicro Conf.on Digital System Design, pp.668-675, Sep. 2012.
[21]. T. Li, and L. K. John, “Run- time Modeling and Estimation of Operating System Power Consumption”, In Proc. of the ACM SIGMETRICS Int. Conf.on Measurement and modeling of computer systems,pp160-171, vol.31, no.1, June 2003.
[22]. G. González, E. Juárez, J. J. Castro and C. Sanz, “Energy Consumption Estimation of an OMAP-Based Android Operating System”, VLSI Circuits and systems Conf, pp.18-20, Apr.2011.

References

[23]. L. Benini, R. Hodgson, and P. Siegel, “System-level Power Estimation and Optimization”, Proc. of the 1998 Int. Symp. on low power electronics and design, pp.173-178, Aug. 1998.

[24]. L. Benini, and G. Micheli, “System-Level Power Optimization: Techniques and Tools”, ACM Trans. on design automation of electronic systems, vol.5, no.2, Apr.2000.

[25]. D. Sunwoo, H. Al-Sukhni, J. Holt and D. Chiou, “Early Models for System-level Power Estimation”, 8th Int. Workshop on Microprocessor Test and Verification, 2007.

[26]. Y. Cho, Y. Kim, S. Park and N. Chang, “System-Level Power Estimation using an on-chip Bus Performance Monitoring Unit ”, Proc. of the 2008 IEEE/ACM Int. Conf.on computer-aided design, pp.149-154, 2008.

[27]. F. Bellosa, “The benefits of event-driven energy accounting in power-sensitive systems”, Proc. of the 9th ACM SIGOPS European Workshop, pp.37-42, Sep. 2000.

[28]. A. S. Dhodapkar and J. E. Smith, “Managing Multi-Configuration Hardware via Dynamic Working Set Analysis”, Proc. of the 29th annual Int. Symp. on computer architecture, pp.233-244, May 2002.

[29]. T. Li, and L. K. John, “Run- time Modeling and Estimation of Operating System Power Consumption”, In Proc. of the ACM SIGMETRICS Int. Conf.on Measurement and modeling of computer systems, vol.31, no.1, pp160-171, June 2003.

[30]. G. Contreras and M. Martionosi, “Power Prediction for Intel XScale Processor Using Performance Monitor Unit Events,” Proc. of the Int. Symp. on low power electronics and design, pp. 221-226, Aug. 2005.

[31]. B. Goel, S. McKee, R. Gioiosa, K. Singh, M. Bhadauria, and M. Cesati, “Portable, scalable, per-core power estimation for intelligent resource management,” Proc. of the 2010 Int. Conf.on Green Computing, pp135 –146, Aug. 2010.

[32]. C. Lively, X. F. Wu, V. Taylor, S. Moore, H. Chang, and C. Su et al, “Power-Aware Predictive Models of Hybird (MPI/OpenMP) Scientific Applications on Multicore Systems”, Int. Conf.on Energy-Aware High Performance Computing, Sept. 2011.

[33]. V. Jimenez, F.J. Cazorla, R. Gioiosa, M. Valero, C. Boneti, 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.228-24, Sept. 2011.

[34]. 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 Conf.on & Int’l Conf.on Cyber, Physical and Social Computing, pp.27-34, Dec.2010.

[35]. X. Lu, E. Erkip, Y. Wang and D. Goodman, “Power Efficient Multimedia Communication Over Wireless Channels”, IEEE Journal on Selected Areas in Communications, Vol.21, No.19, pp. 1738-1751, Feb. 2004.

[36]. X. Lu, T. Fernaine and Y. Wang, “Modeling power consumption of a H.263 video encoder”, Proc. of the Int. Symp. on Circuits and Systems, vol.2, pp. 77-80, May.2004.

[37]. X. Li , Z. Ma and F. C. A. Fernandes, “Modeling power consumption for video decoding on mobile platform and its application to power-rate constrained streaming”, Visual Communications and Image Processing, pp.1-6, Nov. 2012.

[38]. Y. Benmoussa, J. Boukhobza, E. Senn, and D. Benazzouz, “Energy Consumption Modeling of H.264/AVC Video Decoding for GPP and DSP,” Euromicro Conf.on Digital System Design, pp. 890-897, Sept.2013.

[39]. Z. Ma, H. Hu and Y. Wang, “On Complexity Modeling of H.264/AVC Video Decoding and Its Application for Energy Efficient Decoding,” IEEE Trans. on Multimedia, vol.13, no.6, pp. 1240-1255, 2011.

[40]. V. Weaver, “perf_event programming guide”, http://web.eecs.utk.edu/~vweaver1/projects/perf-events/programming.html.

[41]. perfmon2, http://perfmon2.sourceforge.net/docs_v4.html.

[42]. K. Pearson, “Notes on regression and inheritance in the case of two parents,” Proc. of the Royal Society of London, vol.58, pp.240-242, 1895.

[43]. N. J. Salkind, “Statics for People Who (Think They) Hate Statistics”, SAGE Publications, Inc.

[44]. D. A. Freedman, “Statistical Models: Theory and Practice,” Cambridge University Press. p. 26, 2009

[45]. J. H. Friedman, “Multivariate Adaptive Regression Splines”, Journal of The Annals of Statistics, vol. 19, no. 1, pp. 1-67, Mar. 1991.

References

[46]. P. G. Hoel, “Introduction to mathematical statistics,” Wiley Series in Probability and Statistics, 1984.

[47]. T. Baguley, “Serious Stats: A guide to advanced statistics for the behavioral sciences,” Palgrave Macmillan, 2012.

[48]. F. N. Najm, “Transition density: A New Measure of Activity in Digital Circuits,” IEEE Trans. on Computer-Aided Design, vol. 12, pp. 310–323, Feb. 1993.

[49]. C. M. Kyung and S. Yoo, “Energy-Aware System Design: Algorithms and Architectures,” Chapter Introduction, pp.3, Springer, 2011

[50]. A. Baschirotto, V. Chironi, G. Cocciolo, S. D’Amico, M. De Matteis, and P. Delizia, “Low Power Analog Design in Scaled Technologies,” in Topical Workshop on Electronics for Particle Physics, pp.103-110, 2009.

[51]. G. Aditya and R. Ramana, “Design and Analysis of Low Power Generic Circuits in Nano Scale Technology”, Int. Journal of Electronics and Communication Engineering, vol.5, no.1, pp. 35-47, 2012.

[52]. R.Malathi and G.Prabhakaran, “A Low Power based Asynchronous Circuit Design using Power Gated Logic,” Int. Journal of Scientific Research and Management Studies, vol.1, no.1, pp.11-18,

[53]. C.A. Fabian and M.D. Ercegovac, “Input Synchronization in Low Power CMOS Arithmetic Circuit Design,” in Proc. of the Asilomar Conf.on Signals, Systems and Computers, pp.172-176, Nov. 1996.

[54]. S.Hauck, “Asynchronous Design Methodologies: An Overview,” Proc. of the IEEE, vol.83, no.1, pp.69-93, 1995.

[55]. A.G.M. Strollo, E. Napoli, and D. De Caro, “New Clock-gating Techniques for Low-power Flip-flops,” in Proc. of Int. Symp. on Low Power Electronics and Design, pp.114-119, 2000.

[56]. V. Tiwari, S. Malik, and P. Ashar, “Guarded evaluation: pushing power management to logic synthesis/design,” IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol.17, no.10, pp.1051-1060, Oct. 1998.

[57]. Power Compiler Reference Manual. Synopsys, Inc., Nov. 2000.

[58]. B. Chen and I. Nedelchev, “Power compiler: a gate-level power optimization and synthesis system,” in Proc. of IEEE Int. Conf. on Computer Design, pp.74-79, Oct. 1997.

[59]. C. W. Kang and M. Pedram, “Technology Mapping for Low Leakage Power and High Speed with Hot Carrier Effect Consideration”, Proc. of the Asia and South Pacific Design Automation Conf, pp.203-208, Jan.2003.

[60]. S. P. Mai, C. Zhang, J. Chao, and Z.H.Wang, “Design and Implementation of a DSP with Multi-level Low Power Strategies for Cochlear Implants,” High Technology Letters, vol.15, no.2, Jun. 2009, pp.141-146.

[61]. N. Banerjee, A. Raychowdhury, S. Bhunia, H. Mahmoodi, and K. Roy, “Novel Low-overhead Operand Isolation Techniques for Low-power Datapath Synthesis, ” IEEE Trans. on Very Large Scale Integration Systems, vol.14, no.9, pp.1034-1039, 2006.

[62]. H. Li, S. Bhunia, Y. Chen, K. Roy, and T. N. Vijaykumar, “DCG: Deterministic Clock-gating for Low-power Microprocessor Design, ” IEEE Trans. on Very Large Scale Integration Systems, vol.12, no.3, pp. 245-254, 2004.

[63]. P. Meier, R. Rutenbar, and L. Carley, “Exploring multiplier architecture and layout for low power,” in Proc. of the IEEE Custom Integrated Circuits Conf, pp.513-516, May 1996.

[64]. D. Li, P. H. Chou, N. Bagherzadeh, and F. Kurdahi, “Power-Aware Architecture Synthesis and Optimization for Mission-Critical Embedded Systems”, Hardware/Software Codesign: specification languages, interfaces and integration, partitioning, synthesis

[65]. D. Kirovski, M. Potkonjak, “System-level synthesis of low-power hard real-time systems”, Proc. of the Conf. on Design Automation, pp.697-702, 1997.

[66]. D. Markovic, V. Stojanovic, B. Nikolic, M. Horowitz, and R. Brodersen, “Methods for true energy-performance optimization,” IEEE Journal of Solid-State Circuits, vol. 39, no. 8, pp. 1282–1293, 2004.

[67]. A. Wang and A. Chandrakasan, “Energy-efficient DSPs for wireless sensor networks,” IEEE Signal Processing Magazine, vol. 19, no. 4, pp. 68–78, 2002.

[68]. C. H. van Berkel, “Multi-core for mobile phones,” Proc. of the Conf. on Design, Automation and Test in Europe, pp. 1260–1265, Apr. 2009.

[69]. T. D. Burd, and R. W. Brodersen, “Energy Efficient CMOS Microprocessor design”, Proc. 28th. annual HICSS Conf, vol. I, pp 288-297, Jan. 1995.

References

[70]. M. Weiser, B. Welch, and A. Demers, “Scheduling for Reduced CPU Energy”, Proc. of the 1st USENIX Conf. on Operating Systems Design and Implementation, no.2, pp.68-74, 1994.

[71]. H. L. Chan, W. T. Chan, T. W. Lam, L. K. Lee, K. S. Mak. et al, “Energy Efficient Online Deadline Scheduling”, Proc. of the 18th ACM-SIAM Symp. on Discrete Algorithms, pp.795-804, 2007.

[72]. Y. Zhu and F. Mueller, “Feedback EDF Scheduling Exploiting Dynamic Voltage Scaling”, In Proc. IEEE Real-Time and Embedded Technology and Applications Symp, pp.84-93, May 2004.

[73]. D. Kirovski and M. Potkonjak, “System-level synthesis of low-power hard real-time systems”,. In Proc. Design Automation Conf., pp.697-702, 1997.

[74]. A. Azevedo, I. Issenin, R. Cornea, R. Gupta, N. Dutt, and et al, “Profile-based dynamic voltage scheduling using program checkpoints,” In Proc. of the Conf. on Design, Automation and Test, pp168, Mar. 2002

[75]. S. Bang, K. Bang, S. Yoon, and E. Y. Chung, “Run-time adaptive workload estimation for dynamic voltage scaling,” IEEE Trans. on Computer-Aided Design of Integrated Circuits Systems, vol.28, no.9, pp. 1334-1347, 2009

[76]. W. Y. Liang, S. C. Chen, Y. L Chang, and J. P. Fang, “Memory-aware dynamic voltage and frequency prediction for portable devices,” IEEE Int. Conf. on Embedded and Real-Time Computing Systems and Applications, pp. 229-236, Aug. 2008.

[77]. G. Dhiman and T. S. Rosting, “System-level power management using online learning,” IEEE Trans. on Computer-Aided Design of Integrated Circuits Systems, vol.28, no.5, pp. 676-689, 2009.

[78]. Y. Gu and S. Chakraborty, “Control theory-based DVS for interactive 3D games,” IEEE Design Automation Conference, pp. 740-745, Jun. 2008.

[79]. J. Kim, S. Oh, S. Yoo, and C. M. Kyung, “An analytical dynamic scaling of supply voltage and body bias based on parallelism-aware workload and runtime distribution,” IEEE Trans. on Computer-Aided Design of Integrated Circuits Systems, vol. 28, no.4, pp.568-581, 2009.

[80]. J. Kim, S. Yoo, and C.M. Kyung, “Energy Awareness in Processor/Multi-Processor Design,” pp.47-69, Springer Netherlands, 2011.

[81]. B. Noble, M. Satyanarayanan, and M. Price, “A Programming Interface for Application-Aware Adaptation in Mobile Computing,” in Proc. of the 2nd Symp. on Mobile and Location-Independent Computing, pp. 57–66, 1995.

[82]. C. S. Ellis, “The Case for Higher-Level Power Management,” in Proc. of the Seventh Workshop on Hot Topics in Operating Systems, pp. 162–167, 1999.

[83]. A. Vahdat, A. Lebeck, and C. S. Ellis, “Every joule is precious: the case for revisiting operating system design for energy efficiency,” in Proc. of the 9th workshop on ACM SIGOPS European workshop: beyond the PC: new challenges for the operating system, pp. 31–36, 2000.

[84]. A. Roy, S. M. Rumble, R. Stutsman, P. Levis, D. Mazieres, and et al, “Energy management in mobile devices with the cinder operating system,” in Proc. of the sixth Conf. on Computer systems, pp. 139–152, 2011.

[85]. N. V. Rodriguez and J. Crowcroft, “ErdOS: achieving energy savings in mobile OS,” in Proc. of the sixth Int. workshop on MobiArch, pp. 37–42, 2011.

[86]. N. V. Rodriguez, P. Hui, J. Crowcroft, and A. Rice, “Exhausting battery statistics: understanding the energy demands on mobile handsets,” in Proc. of the second ACM SIGCOMM workshop on Networking, systems, and applications on mobile handhelds, pp. 9–14, 2010.

[87]. R. Neugebauer and D. McAuley, “Energy Is Just Another Resource: Energy Accounting and Energy Pricing in the Nemesis OS,” in Proc. of the Workshop on Hot Topics in Operating Systems, pp. 67-72, May 2001.

[88]. T. M. Liu, T. A. Lin, S. Z. Wang, and C. Y. Lee, “A Low-power dual-mode video decoder for mobile applications,” IEEE Communications Magazine, vol.44, no.8, pp.119-126, 2006

[89]. N. J. August and D. S. Ha, “Low power design of DCT and IDCT for low bit rate video codecs,” IEEE Trans. on Multimedia, vol. 6, no. 3, pp. 414–422, 2004.

[90]. T. H. Tsai and D. L. Fang, “A novel design of CAVLC decoder with low power consideration,” in Proc. of the IEEE Asian Solid-State Circuits Conf., pp. 196–199, Nov. 2007.

References

[91]. K. Xu and C. S. Choy, "A power-efficient and self-adaptive prediction engine for H.264/AVC decoding," IEEE Trans. on Very Large Scale Integration Systems, vol.16, no.3, pp.302-313, 2008.

[92]. D. Markovic, V. Stojanovic, B. Nikolic, M. Horowitz, and R. Brodersen, "Methods for true energy-performance optimization," IEEE Journal of Solid-State Circuits, vol. 39, no. 8, pp. 1282–1293, 2004

[93]. D. Kwon, P. Driessen, A. Basso, and P. Agathoklis, "Performance and computational complexity optimization in configurable hybrid video coding system," IEEE Trans. on Circuits and System for Video Technology, vol.16, no.1, pp.31-42, 2006.

[94]. K. Xu, T. M. Liu, J. I. Guo, and C. S. Choy, "Methods for power/throughput/area optimization of H.264/AVC decoding," Journal of Signal Processing Systems, vol. 60, no. 1, pp. 131–145, 2010.

[95]. T. C. Chen, Y. H. Chen, S. F. Tsai, S. Y. Chien, and L. G. Chen, "Fast algorithm and architecture design of low power integer motion estimation for H.264/AVC," IEEE Trans. on Circuits and Systems for Video Technology, vol. 17, no. 5, pp. 568–577, 2007.

[96]. Y. Benmoussa, J. Boukhobza, E. Senn, and D. Benazzouz, "GPP VS. DSP: A Performance/Energy Characterization and Evaluation of Video Decoding," IEEE Int. Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, pp. 273-282, Aug. 2013.

[97]. Z. Y. Xu, S. Sohoni, R. Min and Y.M. Hu, "an Analysis of Cache Performance of Multimedia Applications," IEEE Trans. on computers, vol. 53, no. 1, Jan. pp. 20-38, 2004.

[98]. S. H. Wen ,H. J. Cui and K. Tang, "Low Power Embedded Multimedia Processor Architecture," Chinese Journal of Electronics,vol.16 , no.1, Jan. 2007.

[99]. J. Y. Kim, G. H. Hyun, and H. J. Lee, "Cache Organizations for H.264/AVC Motion Compensation," 13th IEEE Int. Conf. on Embedded and Real-Time Computing Systems and Applications, pp. 534-541, 2007.

[100]. C. H. Lin, J. C. Liu, C. W. Liao, " Energy analysis of multimedia video decoding on mobile handheld devices ", Int. Conf. on Multimedia and Ubiquitous Engineering, pp.120-125, 2007.

[101]. G. Landge, M. Van Der Schaar, and V. Akella, "Complexity metric driven energy optimization framework for implementing MPEG-21 scalable video decoders," In Proc. of the IEEE International Conf. on Acoustics, Speech, and Signal Processing, vol.2, pp.1141-1144, Mar. 2005.

[102]. T. C. Chen, Y. W. Huang, and L. G. Chen, "Analysis and design of macroblock pipelining for H.264/AVC VLSI architecture," In Proc. IEEE Int. Symp. Circuits Systems, pp. 273-276, May 2004,

[103]. S. S. Lin, P. C. Tseng, C. P. Lin, and L. G. Chen, "Multi-mode content-aware motion estimation algorithm for power-aware video coding systems, " in Proc. IEEE Workshop on Signal Processing Systems, pp. 239-244, 2004.

[104]. H. M. Wang, C. H. Tseng, and J. F. Yang, "Computation reduction for intra 434 mode decision with SATD criterion in H.264/AVC," IET Signal Process, vol.1, no.3, pp.121-127, 2007.

[105]. C. H. Hsia, J.S Chiang, Y. H. Wang, and T. Yuan, "Fast Intra Prediction Mode Decision Algorithm for H.264/AVC Video Coding Standard," Third Int. Conf. in Intelligent Information Hiding and Multimedia Signal Processing, vol.2, pp.535-538, Nov. 2007.

[106]. H. Kim and Y. Altunbasak, "Low-complexity macro block mode selection for H.264/AVC encoders," in Proc. Int. Conf. Image Process., Singapore, pp.765-768, 2004

[107]. C. Crecos and M. Y. Yang, "Fast inter mode prediction for P slices in the H.264 video coding standard." IEEE Trans. Broadcasting, vol.51, no.2, pp.256-263, June 2005.

[108]. B. G. Kim and C. S. Cho, "A fast inter-mode decision algorithm based on macro-block tracking for P slices in the H.264/AVC video standard," in Proc. Int. Conf. Image Process, pp.301-304, 2007.

[109]. Y. C. Wang, M. H. Lin, P. C Lin, "Energy Efficient Intra-Task Dynamic Voltage Scaling for Realistic CPUs of Mobile Devices," Journal of information science and engineering, vol.25, pp. 251-272, 2009.

[110]. W. Y. Lee, Y. W. Ko, H. Lee, and H. Kim, "Energy-efficient scheduling of a real-time task on DVFS-enabled multi-cores," Proc. of the 2009 Int. Conf. on Hybrid Information Technology, pp. 273-277, 2009.

[111]. Y. H. Wei, C. Y. Yang, T. W. Kuo, S. H. Hung, and Y. H. Chu, "Energy-efficient real-time scheduling of multimedia tasks on multi-core processors," Proc. of the 2010 ACM Symp. on Applied Computing, pp. 258-262, 2010.

[112]. R. Xu, R. Melhem, and D. Mosse, "Energy-Aware Scheduling for Streaming Applications on Chip Multiprocessors," Proc. of the 28th IEEE Int. Real-Time Systems Symp, pp. 25-38, 2007.

References

[113]. J. Cong and K. Gururaj, “Energy efficient multiprocessor task scheduling under input-dependent variation,” Proc. of the Conf. on Design, Automation and Test, pp. 411-416, 2009.

[114]. S. Yaldiz, A. Demir and S. Tasiran, “Stochastic Modeling and Optimization for Energy Management in Multi-Core Systems: A Video Decoding Case Study,” IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol, 27, no.7, pp.1264-1277, 2008.

[115]. Y. Tan, P. Malani, Q. Qiu and Q. Wu, “Workload prediction and dynamic voltage scaling for mpeg decoding,” Proc. Asia and South Pacific Conf. on Design Automation, pp.911-916, 2006.

[116]. S. Hong, S. Yoo, H. Jin, K. M. Choi, and et al, “Runtime distribution-aware dynamic voltage scaling,” Int. Conf. on Computer-Aided Design, pp. 587-594, Nov. 2006.

[117]. A. Kumar, A. Das, and A. Kumar, “Energy Optimization by Exploiting Execution Slacks in Streaming Applications on Multiprocessor Systems,” In Proc. of the 50th Annual Design Automation Conf., 2013.

[118]. Y. Wang, H. Liu, D. Liu, Z. W. Qin, Z. L. Shao, and E. H. M. Sha, “Overhead-aware energy optimization for real-time streaming applications on multiprocessor system-on-chip,” Journal of ACM Trans. on Design Automation of Electronic Systems, vol.16, no.2, pp.1-32, Mar. 2011

[119]. C. J. Lian, S. Y. Chien, C. P. Lin, P. C. Tseng, and L. G. Chen, “Power-aware multimedia: Concepts and design perspectives,” IEEE Circuits and Systems Magazine, vol.7, no.2, pp. 26–34, 2007.

[120]. P. Jain, A. Laffely, W. Burleson, R. Tessier, and D. Goeckel, “Dynamically parameterized algorithms and architectures to exploit signal variations,” Journal of VLSI Signal Processing Systems, vol. 36, no. 1, pp. 27–40, 2004.

[121]. H. W. Cheng and L. R. Dung, “A content-based methodology for power-aware motion estimation architecture,” IEEE Trans. on Circuits and Systems, vol. 52, no. 10, pp. 631–635, 2005.

[122]. S. W. Lee and C. C. J. Kuo, “Complexity modeling for motion compensation in H.264/AVC decoder,” in Proc. of the 14th IEEE Int. Conf. on Image Processing , vol. 5, pp. 313–316, Sept. 2007.

[123]. H. H. Chun, P. W. Hsiao, C. Tihao, and H. M. Hang, “Advances in the scalable amendment of H.264/AVC,” IEEE Communications Magazine, vol. 45, no. 1, pp. 68–76, 2007.

[124]. M. Paul, M. R. Frater, and J. F. Arnold, “An efficient mode selection prior to the actual encoding for H.264/AVC encoder,” IEEE Trans. on Multimedia, vol. 11, no.4, pp. 581–588, 2009.

[125]. W. Ji, M. Chen, X.H. Ge, P. Li, and Y Q. Chen, “ESVD: An Integrated Energy Scalable Framework for Low-Power Video Decoding Systems,” EURASIP Journal on Wireless Communications and Networking, pp.1-13, 2010.

[126]. H. Sanghvi, M. Mody, N. Nandan, M. Mehendale, S. Das, and et al, “A 28nm Programmable and Low Power Ultra-HD Video Codec Engine,” IEEE Int. Symp. on Circuits and Systems, pp.558-561, Jun. 2014.

[127]. X. Li, M. Dong, Z. Ma, and F. Fernandos, “GreenTube: Power Optimization for Mobile Video Streaming via Dynamic Cache Management,” Proc. of the Int. Conf. on Multimedia, pp.279-288, 2012.

[128]. P. Antoniou, V.Vassiliou, A. Pitsillides, A. Panayides, A. Vlotomas, and et.al., “Final report of Adaptive Methods for the Transmission of Video Streams in Wireless Networks,” University of Cyprus Department of Computer Science Networks Research Laboratory.

[129]. G. Estrin, “Organization of Computer Systems-The Fixed Plus Variable Structure Computer,” Proc. Western Joint IRE-AIEE-ACM Computer Conf., pp. 33-40, 1960.

[130]. Xilinx Company, http://www.xilinx.com

[131]. E. S. Jang, J. Ohm, and M. Mattavelli, “Whitepaper on reconfigurable video coding (RVC)”, ISO/IEC JTC1/SC29/WG11 Document N9586, Antalya, Turkey.

[132]. S.S. Bhattacharyya, J. Eker, J. W. Janneck, C. Lucarz, M. Mattavelli, and M. Raulet, “Overview of the MPEG reconfigurable video coding framework”, Journal of Signal Processing Systems, pp. 251-263, 2009.

[133]. M. Mattavelli, I. Amer, and M.Raulet, “The reconfigurable video coding standard [Standards in a Nutshell]”, IEEE Signal Processing Magazine, vol.27, no.3, pp. 159-167, 2010.

[134]. M. Wipliez, G. Roquier, and J. F. Nezan, “Software code generation for the RVC-CAL language,” Journal of Signal Processing Systems, vol.63, no.2, pp.203-231, 2011.

[135]. J. R. Ohm and G. Sullivanm, “Vision, applications and requirements for high efficiency video coding (HEVC),” document N11872, Daegu, South Korea, Jan. 2011.

[136]. M. Viitanen, J. Vanne, T.D. Hamalainen, M. Gabbouj and J. Lainema, “Complexity Analysis of Next-Generation HEVC Decoder,” IEEE Int. Symp. on Circuits and Systems, pp. 882-885, May. 2012
[137]. F. Fernandes, A. Segall, and M.Zhou, “Draft Call for Proposals on Green MPEG”, MPEG document, M28249, Geneva, Jan.2013
[138]. ISO/IEC JTC1/SC29/WG11/N13468, “Context, Objectives, Use Cases and Requirements for Green MPEG,” Jan. 2013
[139]. Z. Ma, M. Dong, F. C. A. Fernandes, and S. Hwang, “Display Power Reduction Using Extended Nal Unit Header Information, ” Patent Application number: 20130278834, Oct. 2013.
[140]. O. Oyman, J. Foerster, Y.J. Tcha, and S. C. Lee, “Toward enhanced mobile video services over WiMAX and LTE Update,” IEEE Communications Magazine, vol.48, no.8, pp.68-76, Aug. 2010.
[141]. G. Y. Li, Z. K. Xu, C. Xiong, and C. Y . Yang, “Energy-efficient wireless communications: tutorial, survey, and open issues,” IEEE Wireless Communications, vol.18, no. 8, pp.28-35, Dec. 2011.
[142]. D. Forte, A. Srivastava, “Energy and thermal-aware video coding via encoder/decoder workload balancing, ” Proc. of the ACM/IEEE Int. Symp. on Low power electronics and design, Aug. 2010.
[143]. X. Tu. Tran and V. H. Tran. “An Efficient Architecture of Forward Transforms and Quantization for H.264/AVC Codecs,” In REV Journal on Electronics and Communications, vol. 1, no. 2, pp. 122-129, 2011.
[144]. G. Bjontegaard and K. Lillevold, “Context-adaptive VLC (CVLC) Coding of Coefficients,” JVT Document JVT-C028. Fairfax, VA, 2002.
[145]. D. Marpe, H. Schwarz, and T. Wiegand, “Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression,” IEEE Trans. on Circuits and Systems for Video Technology, vol.13, no,7, pp. 621-636, 2003.
[146]. ORCC project. http://orcc.sourceforge.net/
[147]. J. Gorin, M. Wipliez, F. Preteux, and M. Raulet, “LLVM-based and scalable MPEG-RVC decoder,” Journal of Real-Time Image Processing, vol.6, no.1, pp. 59-70, 2010.
[148]. C. Lattner and V. Adve, “LLVM: a compilation framework for lifelong program analysis & transformation,” In Proc. of the Int. Symp. on Code Generation and Optimization, pp.75, Mar. 2004.
[149]. J. Gorin, H. Yviquel, F. Preteux, and M. Raulet, “Just-in-time adaptive decoder engine: a universal video decoder based on MPEG RVC,” Proc. of the ACM Int. Conf. on Multimedia, pp.711-714, 2011.
[150]. Grapgiti Tool. https://eclipse.org/graphiti/
[151]. Xtext Tool, https://eclipse.org/Xtext/documentation.html
[152]. Simple DirectMedia Layer, http://wiki.libsdl.org/FrontPage
[153]. Cross Platform Make, http://www.cmake.org/overview/
[154]. http://orcc.sourceforge.net/getting-started/install-orcc/
[155]. http://orcc.sourceforge.net/getting-started/get-applications/
[156]. V.M.Weaver, M.Johnson, K.Kasichayanula, J.Ralph, P. Luszcaek, and D. Terpstra el al., “Measuring Energy and Power with PAPI”, in Proc. International Workshop on Power-Aware Systems and Architectures, pp. 262-268, Sept. 2012.
[157]. J. Herrera, Desarrollo de un Emulador de Baterías para el Estudio del Consumo de la Tarjeta BeagleBoard, PFC EUITT-UPM, Jul. 2011.
[158]. User's Guide, Agilent Technologies Model 66319B/D, 66321B/D Mobile Communications DC Source, 2005.
[159]. OMAP4460 PandaBoard ES System Reference Manual, Revision 0.1, Sept. 2011.
[160]. BeagleBoard System Reference Manual Rev C4, Dec.2009
[161]. ISO/IEC 14496-2 (MPEG-4), “Information Technology-Coding of Audio-Visual Objects-Part 2: Visual, 2002
[162]. MPEG-4 Visual, http://mpeg.chiariglione.org/standards/mpeg-4/video
[163]. J. W. Janneck, I. Miller, D. Parlour, G. Roquier, M. Wipliez, and M. Raulet, “Synthesizing Hardware from Dataflow Programs: An MPEG-4 Simple Profile Decoder Case Study”, Journal of Signal Processing Systems, vol.63, no.2, pp. 241–249, May 2011.
[164]. ITU-T Recommendation H.264, ISO/IEC 14496-10, “Advanced Video Coding for generic audiovisual services”, 2005.

References

[165]. J. Gorin, M. Raulet, Y. L. Cheng, H. Y. Lin, N. Siret, and et al, “An RVC Dataflow description of the AVC Constrained Baseline Profile decoder,” IEEE Int. Conf on Image Processing, pp.753–756, 2010.
[166]. ITU-T,Recommendation T-REC, January 2012, “H.264.1 : Conformance specification for ITU-T H.264 Advanced Video Coding”.
[167]. M. T. Pourazad, C. Doute, M. Azimi, and P. Nasiopoulos, “HEVC: The New Cold Standard for Video Compression: How does HEVC compare with T.264/AVC?,” IEEE Consumers Electronic Magazine, pp.36-46, Jul. 2012.
[168]. S. Browne, J. Dongarra, N. Garner, K.London, and P. Mucci, “A Portable Programming Interface for Performance Evaluation on Modern Processors,” International Journal of High Performance Computing Applications, vol.14, no.3, pp. 189-204, Aug. 2000.
[169]. L. Shannon and P. Chow, “Maximizing System Performance: Using Reconfigurability to Monitor System Communications,” In Proc. IEEE Int. Conf. on Field-Programmable technology, pp.231-238, Dec. 2004.
[170]. J. G. Tong and M. A. S. Khalid, “Profiling Tools for FPGA-Based Embedded Systems: Survey and Quantitative Comparison,” Journal of Computers, vol.3, no.6, pp.1-14, Jun. 2008.
[171]. P. H. Chen, C. T. King, Y. Y. Chang, and S.Y. Tseng, “Multiprocessor System-on-Chip Profiling Architecture: Design and Implementation,” Int. Conf. on Parallel and Distributed Systems, pp.519-526, Dec. 2009.
[172]. Yu Bai, Priya Vaidya, “Memory characterization to analyze and predict multimedia performance and power in embedded systems, ”, ICASSP 2009
[173]. J. Eker and J.W. Janneck, “CAL language report,” ERL Technical Memo UCB/ERL M03/48, EECS Department, University of California at Berkeley, Berkeley, California, USA, Dec. 2003.
[174]. E. A. Lee and T. M. Parks, “Dataflow Process Networks”, in Proc. of the IEEE, Vol.83, no.5, pp.773-799, 1995.
[175]. Cortex-A9 Technical Reference Manual, revision: r3p0
[176]. CoreSight PTM-A9 Technical Reference Manual, revision: r1p0
[177]. Cortex-A8 Technical Reference Manual, revision: r1p1
[178]. J. Vanne, M. Viitanen, T. D. Hämäläinen, and A. Hallapuro, “Comparative Rate-Distortion-Complexity Analysis of HEVC and AVC Video Codecs,” IEEE Trans. on Circuits Syst. Video Technology., vol. 22, no.12, pp. 1885-1898, Dec. 2010.
[179]. J. D. Davis, S. Rivore, M. Goldszmidt, and E. K. Ardestani, “No Hardware Required: Building and Validating Composable Highly Accurate OS-based Power Models”, Microsoft Technical Report, 2011.
[180]. A. G. Bedeian and K.W. Mossholder, “On the use of the coefficient of variation as a Measure of Diversity”, Organizational Research Methods, vol.3, pp.285-297, 2000.
[181]. M. Hentati, Y. Aoudni, J.F.Nezan, M.Abid, O. Deforges, “FPGA dynamic reconfiguration using the RVC technology: inverse quantization case stud ,” Proc. of the Conf. on Design and Architectures for Signal and Image Processing, pp.108-114, Nov. 2011.
[182]. F. Palumbo, N. Carta, L. Raffo, “The Multi-dataflow Composer Tool: A Runtime Reconfigurable HDL Platform Composer,” Proc. of the Conf. on Design and Architectures for Signal and Image Processing, pp. 178-185, Nov. 2011.
[183]. A. G. Schmidt, N. Steiner, M. French, R. Sass, “HwPMI: An Extensible Performance Monitoring Infrastructure for Improving Hardware Design and Productivity on FPGAs,” Int. Journal of Reconfigurable Computing, 2012.