DEPARTAMENTO DE INGENIERÍA TELEMÁTICA Y ELECTRÓNICA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y SISTEMAS DE TELECOMUNICACIÓN

Control algorithms for energy optimization in multimedia hand-held devices

TESIS DOCTORAL

Qiong Tang

Master Northwestern Polytechnical University

DIRECTORES

Ángel Manuel Groba Gonzalez

Doctor Ing.de Telecomunicación por la Universidad Politécnica de Madrid

Eduardo Juárez Martínez

Docteur ès Sciences Techniques por el École Polytechnique Fédéral de Lausanne

2017

Resumen

La industria de la microelectrónica ha impulsado las capacidades de los dispositivos móviles multimedia, pero la batería, que es la única fuente de energía de este tipo de dispositivos, está experimentando un desarrollo relativamente lento. Por lo tanto, la determinación de cómo optimizar el consumo de energía de los dispositivos móviles bajo un requisito de rendimiento predefinido se ha convertido en un problema crítico. Además, según un informe reciente, el consumo de vídeo para tablets y teléfonos inteligentes creció un 35% en el año 2014 y ha crecido un 170% desde 2013. En realidad, la reproducción de vídeo móvil ha experimentado un crecimiento significativo del 2084% de 2011 a 2015. Como ejemplos de las tareas que mayor consumo de energía implican, la codificación, decodificación y presentación de secuencias de vídeo se encuentran entre los principales temas de investigación sobre la gestión de la energía en los sistemas multimedia. Además, cada nuevo estándar de vídeo también tiende a aumentar la necesidad de energía de las tareas de vídeo con respecto a las normas anteriores.

Esta tesis presenta una solución basada en algoritmos de control para la regulación del consumo de potencia bajo las limitaciones de capacidad de la batería de los dispositivos portátiles multimedia mientras se ejecuta una aplicación de decodificación de vídeo y se mantiene una calidad razonable de experiencia de usuario. Se propone un sistema general de control que incluye un subsistema de control de bucle cerrado en tiempo real y un gestor de control de potencia, y se ha implementado en el sistema operativo de una placa de desarrollo de bajo coste. En lugar de utilizar un sensor de consumo específico, se propone un estimador de potencia basado en eventos del sistema como señal de realimentación en el subsistema de bucle cerrado. El estimador de potencia obtiene periódicamente valores de cuenta de eventos significativos y calcula las estimaciones de consumo de potencia a través de modelos matemáticos. Este estimador de potencia se ha implementado en un kernel de Linux y se ha evaluado mientras se ejecuta una aplicación de decodificación de vídeo en una plataforma de desarrollo de sistemas empotrados. Posteriormente, antes de la implementación del sistema de control en tiempo real, se utilizan datos de estimación fuera de línea para obtener un modelo de sistema que permite la aplicación de métodos clásicos de teoría de control para analizar y diseñar

diferentes controladores. Los resultados de la simulación muestran que los controladores integrales mantienen la estabilidad del sistema y logran un error medio en régimen permanente nulo con tiempos de establecimiento cortos, incluso en presencia de ruido de estimación o perturbaciones. A partir de estos resultados de simulación, los controladores han sido implementados en el sistema de desarrollo y los resultados reales coinciden con los resultados de simulación. El sistema de control es capaz de regular la potencia consumida y la tasa de descarga de la batería en presencia de fluctuaciones en la demanda de consumo de energía del descodificador, lo que presenta buenos resultados para garantizar una determinada duración de la batería[1].

[1] La expresión "duración de la batería" se refiere en esta tesis al intervalo de tiempo durante el cual el dispositivo puede funcionar, partiendo de una batería completamente cargada hasta el agotamiento de la misma.

Abstract

The micro-electronics industry has been boosting the capabilities of multimedia mobile devices, but the battery, which is the only power source of most mobile devices, is experiencing relatively slow development. Therefore, determining how to optimize the energy consumption of mobile devices under a predefined performance requirement has become a critical issue. Besides, according to a recent report, tablet and smartphone video consumption grew 35% in the year 2014 and has grown 170% since 2013. Actually, mobile video playback has experienced a significant growth of 2084% from 2011 to 2015. As some of the most energy-consuming tasks, encoding, decoding and presentation of video sequences are among the main subjects of research on power management in multimedia systems. In addition, every new video standard also tends to increase the energy requirement of video tasks with respect to the previous standards.

This dissertation presents a solution based on control algorithms for power regulation under the limited battery capacities of multimedia hand-held devices while executing a decoder application and maintaining a reasonable quality of user experience. A control system, which includes a real-time closed-loop control subsystem and a power-control governor, is proposed and it has been implemented in the operating system of a low-cost development board. Instead of using any specific power sensor, a power estimator based on monitored system events of multimedia mobile devices is proposed as the feedback signal in the closed-loop subsystem. The power estimator periodically obtains significant-events count values and calculates power-consumption estimations through mathematical models. This power estimator has been implemented in a Linux kernel and evaluated while running a video decoder application on an embedded development platform. Afterwards, prior to the implementation of the real-time control system, off-line estimation data are used to get a system model, which enables the application of classic control-theory methods to analyze and design different controllers. The simulation results show that integral controllers keep the system stability and achieve null average steady-state error with short settling times, even in the presence of estimation noise or disturbance. From these promising simulation results, the controllers have been implemented in the development board and the real results match simulation results. The control system is able to regulate the power consumption and the battery discharge rate in the presence of fluctuations in

the decoder power-consumption demand, which presents good results to guarantee a certain battery lifetime[2].

[2] The term “battery lifetime” refers in this manuscript to the length of time for which the device can run, starting from a fully charged battery.

Acknowledgement

First, I would like to thank my supervisors, Dr. Ángel M. Groba and Eduardo Juárez, for their valuable feedback and support throughout this work. Their support, diligence, and commitment to high-quality research have contributed significantly to this thesis. I am also grateful to all the professors and lab mates in GDEM-CITSEM: César Sanz, Matías J. Garrido, Fernando Pescador and Pedro J. Lobo, for their efforts in providing and creating such a friendly and helpful working environment in the lab. Thanks also to my colleagues: Rong Ren, Jianguo Wei, Henry O. Cruz and Miguel Chavarrías. I have learned a lot from each of them; our friendships have made my time at GDEM an enjoyable and unique experience. Thanks to Paula and Enrique, I have benefited a lot through the cooperation and the exchange of ideas with them.

Finally, I would like to acknowledge and appreciate the financial support from China Scholarship Council (CSC) and GDEM. Besides, I would like to express my love and appreciation to my parents. Thanks for them to bring me up and support good quality of education for me. During these years, I met difficulties and setbacks that taught me to grow and hone my character. I benefit from the doctoral stage of study, and no matter where I work in the future, I will keep a serious and focused researcher, and strive to do everything for the community to contribute to their own strength. I would like to thank all my friends: Rong Ren, Jianguo Wei, Xiaomin Zhao, Meijuan Zhang, Henry, Jesús, Ana and Carmen, for their love and concern.

Content

List of Figures

List of Tables

List of Acronyms

AAPE = Average Absolute Percentage Error

API = Application Programming Interface

APPs = Applications

BRR = Backward Rectangular Rule

CCNT = Cycle Count

CNTENC = Count Enable Clear

CNTENS = Count Enable Set

CPU = Central Processing Unit

CTU = Coding Tree Unit

DCT = Discrete Cosine Transform

DPM = Dynamic Power Management

DSP = Digital Signal Processors

DVFS = Dynamic Voltage and Frequency Scaling

EVTSEL = Event Selection

FLAG = Overflow Flag Status

FRR = Forward Rectangular Rule

GPP = General Purpose Processors

GPU = Graphics Processor Unit

INTENS = Interrupt Enable Set

MARS = Multivariate Adaptive Regression Splines

MMU = Memory Management Unit

MPU = MicroProcessor Unit

NNZ = Number of NonZero

OPP = Operating Performance Point

PAPI = Performance Application Programming Interface

PC = Program Counter

PuC = Platform under Control

PCG = Power Control Governor

PCL = Performance Counter for Linux

PI = Proportional Integral

PID = Proportional Integral Derivative

PM = Power management

PMC = Performance Monitor Counters

PMCNT = Performance Monitor Count

PMNXSEL = Performance Counter Selection

PSM = Power-Saving Mode

PMU = Power Management Unit

QoE = Quality of Experience

QP = Quantization Parameters

QoS = Quality of Service

SoC = State of Charge

SP = Simple Profile

STB = Set-Top Box

SWINCR = Software Increment

TR = Tustin's bilinear Rule

USEREN = User Enable

VoIP = Voice over IP

Chapter 1 Introduction

1.1 Background and challenge

Currently, there is a pervasive utilization of hand-held terminal devices, such as mobile phones, tablets, smart watches and so on. The multimedia devices are essential in people's daily life. Smartphones and other types of hand-held sets with multimedia capabilities are increasingly utilized in communication and entertainment, and at the same time, mobile computing and communication technologies are also rapidly advancing. However, as the only power source of most mobile devices, the battery capacity has not experienced an equivalent increase. Therefore, optimally utilizing the limited battery energy on mobile devices under a predefined performance requirement becomes a critical issue. Mobile devices have already changed the habits of daily life due to its integration with some main multimedia functions, such as TV, radio, game console, camera, and video telephony and so on. With the support of various applications (APPs), the users can simultaneously watch online videos, carry out video, audio and text chat, or download video playbacks. The increased complexity and functionality in many mobile devices has motivated a transformation of the system usefulness assessment from a quality of service (QoS) approach to a quality of experience (QoE) approach [1][2].

In one report from Ericsson in 2016 [3], video viewing is being gradually switched from traditional big-screen devices to online-streaming on smartphones. Especially in teen behaviors, between 2011 and 2015, teens increased their TV/video viewing at home on smartphones by 85 percent and nearly halved their time spent watching on a traditional TV screen. According to the latest report from Ooyala [4], 46% of all video plays in the fourth quarter of 2015 were on mobile devices like tablets and smartphones. In fact, tablet and smartphone video consumption grew 35% in the year 2014 and have grown 170% since 2013. Actually, mobile video playback has experienced a significant growth of 2084% from 2011 to 2015. As some of the most energy-consuming tasks, encoding, decoding and presentation of video sequences are among the main subjects of research on power management in multimedia systems. In fact, the introduction of new standards, such as High-efficiency Video Coding (HEVC) [5], is increasing the energy requirement of video tasks with respect to the previous standards, such as H.264/AVC [6]. Given that the energy capacities of the batteries used in those small sets are not expected to

satisfy user needs [7], research on optimizing the energy consumption of the systems becomes imperative.

The research of this thesis is a continue branch of the energy-centric scheduling module [8], the energy-centric scheduling module is in charge of scheduling the battery energy to the applications to maximize the battery energy utilization. To achieve that, the energy consumption of hardware devices should be accurately modeled and correctly accounted to the corresponding application that causes the device activities. When the mobile devices are running different applications, the scheduler can allocate energy to different applications. With respect to video decoders, they are some of the most energy consuming applications and the research group has experience of researching in a number of decoders [14]. Thus, this thesis focuses on control algorithms for energy optimization in multimedia devices while running video-decoder applications, which can significantly mitigate the system energy consumption problems.

When the users run video-related applications on a battery based mobile device, they typically have different preferences for the applications and there is a need of how long the battery should last for their current applications. While the ability to ensure the target battery lifetime increases the confidence and security of the user using the mobile system [8], in many cases, the mobile devices can only provide a notice of the remaining battery capacity, but cannot guarantee the target battery lifetime against video applications, which consume most energy. For example, when the user is watching a football match, to reduce the power consumption and keep the device working until the end of the match is more desirable than to offer a good video quality; when the user is carrying out a video call, the quality of video decoding will help the user to be more comfortable to chat with friends and family; when the user is having a video meeting, a failure to achieve the expected battery lifetime will reduce the QoE of the system or even bring economic losses to the users. Therefore, how to dynamically achieve the user requirements in real time and under the battery energy restriction is one of the most fundamental requirements of mobile system users. This is also an essential element in the assessment of QoE. For the above objective, to equip the mobile devices with a high-capacity battery seems to be a simple and direct solution. Unfortunately, little industrial progress is achieved in the technology to enhance the battery capacity and density in the past few decades. On the contrary, nowadays, the continuous pursuit of designing mobile devices to be thinner and lighter poses further restriction on the battery size and capacity. For the above reasons, there is no doubt that breaking

the bottleneck of hardware limit from the software perspective is a smart choice.

To guarantee the expected battery lifetime, the power consumption of the multimedia applications should be known but the majority of current consumer mobile devices does not integrate any power measurement equipment. Adding power measurement sensors to the consumer mobile devices is not a good solution because extra cost of hardware will be incurred; besides, this solution is not applicable to those already sold devices and has a long time-to market. Therefore, monitoring the energy consumption in real time without the assistance of specific power measurement sensors becomes a new challenge. If this difficulty is overcome, online power measurement and optimization can be widely applied to the existing mobile devices with no extra hardware cost. Now, there are some APPs which can estimate the battery lifetime generally when the mobile devices are under different work situations. For example, some APPs can list the standby time, call time, or even online/offline video viewing time. However, those APPs can only play the role of notifying the users instead of controlling the mobile devices. Therefore, in order to conveniently monitor power consumption, one innovation of this thesis is to regulate the power consumption of embedded multimedia systems without the need of adding power monitors but relying on power estimations derived from commonly available resources in mobile devices. Besides, some mobile operating systems support different power saving modes, such that user can select power saving, balance mode or high performance mode. But there are not battery lifetime-oriented modes. How to estimate the energy consumption and control the battery lifetime in real time is worth to explore. Our research direction field can fill the gap between current solutions and user needs. The aim of this work is to guarantee the battery lifetime while running applications.

Against the above challenges, system designers have to explore the way to optimize energy consumption of mobile systems while maintaining a reasonable QoE under the battery storage limit. Under this background, a method to optimize energy consumption from the operating system can ease the urgent needs of video decoding in mobile devices. The way to solve the challenges can contribute to a new generation of mobile devices.

1.2 Motivation

Embedded and mobile multimedia systems require, like others, the optimization of the quality of experience (QoE) they offer to the user. However, their common battery dependency makes also necessary the optimization of their energy consumption. Indeed, for example, the

wide spectrum of usual available applications for current smartphones make them to have quite limited operating times, especially when they execute common video encoding, decoding and/or presentation applications. Therefore, there is an increasing effort into trying to reduce the energy consumption of this kind of systems from different points of view. Hardware and software are the two aspects that are considered to optimize energy consumption. In order to easily transplant the energy consumption optimization methods, software design is a very convenient way to investigate. There are two directions, one is from the applications in the user space, and other is through the operating system.

The first direction investigates how APPs can save energy consumption. Redundancy reduction and multiple modes switching are typical methods of power management. Redundancy reduction concerns basic solutions like simplifying the framework of the APPs and reducing unnecessary operations of computing or communication. Multiple modes switching can support more ways to save energy depending on the user requirements, which means, against the different states of battery, the APPs can choose high performance with high energy consumption or low performance with low energy consumption. One typical application of multiple modes switching in video decoding is: when there is enough battery, high-definition video is streamed to the mobile devices, and when the battery is low, low-definition and small-sized video is delivered to save energy and extend battery lifetime.

The second research direction is how to control the energy consumption in general through the operating-system level. The operating system plays an important role because it is aware of the power consumption status of the platform and the battery discharging rate, as well as, it can monitor the users' requirements and the applications performance through special interfaces. Therefore, research in this field is popular and there are numbers of technologies of power optimization through operating system. Unfortunately, the majority of them are not strong power-aware enough to provide a battery lifetime guarantee.

The work of this dissertation focuses on implementing control algorithms for energy optimization by controlling power consumption in multimedia mobile devices when they are decoding video, while also maintaining a reasonable quality of user experience (QoE). For this reason, a control system has been proposed for the platform under control (PuC), in which the microprocessor unit (MPU) executes a video decoder application.

1.3 Objectives

Currently, multimedia hand-held devices like smartphones have operating times less than a few hours and the QoE is a fundamental issue that determines the degree of use of a platform. Therefore, it is necessary that QoE and energy consumption are considered jointly in present multimedia embedded systems. Since it is not foreseeable that the density of energy stored in lithium batteries will increase considerably in coming years, the only improvement of batteries will not significantly increase the operating time of terminals.

And within these multimedia systems, the functionalities of video encoding, decoding and presentation consume a very important part of the energy available in the terminals. Moreover, the introduction of emerging standards as HEVC is increasing this balance.

In this thesis, the main objective is to apply the control theory to the optimization of power consumption in this kind of systems. In order to implement it in multimedia devices, the main objective can be separated into the following sub-objectives.

1. Activate the dynamic voltage and frequency scaling subsystem (DVFS): One of the features of these systems that can be used to act on their own power consumption is the processor DVFS subsystem. With DVFS, clock rates and voltages can be scaled by software based on the performance requirements of the application. For each operating performance point (OPP), a software module sends control signals to external regulators in order to set the minimum allowable voltage. DVFS is a method commonly employed to reduce energy consumption and extend battery lifetime for mobile devices. It provides an efficient energy saving mechanism for components that remain in active states. In this manuscript, DVFS is supported by processors designed for decoder applications such as hand-held devices, in which multiple voltage and frequency levels can be utilized by the system software in different conditions to save on energy consumption. For example, when an application does not need to be run at the highest performance, it may reduce the frequency and voltage so as to reduce the power consumption while remaining reasonable QoE.

2. Set-up of an estimation model: In order to have a feedback line with power consumption information and given that conventional devices do not offer it, certain power estimation methodology is required to estimate the power consumption of video decoders at each OPP. The methodology should be able to identify the most appropriate training data and power-related events to be counted, to cover the main application characteristics. It is needed to build a

dynamic power estimation model of a video decoder, considering the observation of a set of actual consumption experimental results. In the operating system, since it is hard to know the actual power consumption directly, the estimation model is required to calculate the power values, which can be fed to the control system back as well as to the user interface.

3. Implement a general power estimator: Once the estimation model has been built, it should be applied into the power estimator in order to avoid the need of a hardware power monitor subsystem, different approaches should be explored to estimate the power consumption. The power estimating approaches will be compared to select the suitable one that will work as the feedback. The estimator used in the control system should satisfy two requirements: the first one is that the estimator should accurately reflect the power consumption of the applications; the second one is that it should decouple the user multimedia application execution from the power control system.

4. Design and implement different control algorithms: once the PuC is provided with suitable input action and output feedback signals, different closed-loop control strategies will be applied. This implies the steps of system modeling, simulation and implementation. Through suitable experiments, the benefits of the each proposed control algorithm will be showed in maximizing the user experience in battery-limited multimedia mobile systems.

5. Implement the Power Control Governor (PCG): once the closed-loop control subsystem is verified, a battery discharge estimator that implemented into the operating system will estimate the battery state of charge (SoC) based on the feedback power estimation. Then, a power budget generator will complete the PCG to generate a suitable set-point for the closed-loop control subsystem. Therefore, the power budget generator will provide multiple personalized battery discharge mechanisms to guarantee a target battery lifetime while satisfying user requirements. The whole control system should be tested with stable workload and varying workload in order to be compared with other methods.

1.4 Contribution

This work investigates control algorithms for energy optimization by controlling power consumption in multimedia hand-held devices. Hand-held devices are battery based mobile devices that are under energy limit. This thesis focuses on saving energy by controlling power consumption to extend the battery lifetime in order to satisfy the users' requirements while

maintaining a reasonable QoE. Based on this starting point, this work explores the design of control algorithms for controlling power consumption in order to guarantee a certain battery lifetime in mobile systems. The main contributions of this dissertation are the following:

1. Energy optimization for one of the most energy-consuming multimedia applications. Among the various APPs, capture, encoding, decoding and presentation of video sequences are some of the most energy-consuming tasks for that type of equipment. This thesis focuses on energy optimization of decoder multimedia applications, which can significantly alleviate the energy consumption problems.

2. General and simple mathematical model of PuC. The design of the system controller is based on a suitable model of the PuC. In order to facilitate the application of the classic control theory, a simple and general model of the PuC has been obtained and validated in an application case. Although the proposed classic control algorithms have generated promising results, this model could also be refined and sophisticated for applying different advanced closed-loop control strategies.

3. Precise PMC-based power-awareness model. Power-awareness of mobile devices while running applications includes sensors monitoring, estimation and prediction. After selecting the high energy-related PMCs, PMC-based power model is built which is close to the actual power consumption of PuC while running video decoder and without needing specific power sensors.

4. Generally applicable estimation subsystems. To widely apply the estimation subsystem on various mobile devices, it was implemented within the operating system, such that it is able to calculate power estimation samples in real time, periodically and independently of the application in order to act as feedback for the control system.

5. Power control governor. The proposed PCG can calculate the set-point of the closed-loop subsystem based on the feedback of power-consumption estimation. Besides, the PCG supports personalized and multiple battery-discharge profiles to regulate video decoding power consumption, while maintaining a reasonable QoE.

6. Closed-loop power control subsystem. Under battery limit, the target battery lifetime can be achieved if the power consumption is controlled. Depending on the desired battery lifetime, the proposed closed-loop control subsystem is correspondingly given a suitable set-point by the PCG to guarantee the battery lifetime in real time. Besides, the closed-loop power control

subsystem can quickly respond to workload variations in order to keep stable the power consumption.

7. Multiple controllers. A set of classic linear controllers has been designed to check their effects on the power control. They imply different system behaviors, which is a heuristic exploration for adjusting the power consumption of mobile devices while executing applications.

8 The real-time control system. The proposed control system, which includes a PCG and a closed-loop subsystem, can dynamically adjust the battery lifetime depending on the users requirements while maintaining a reasonable QoE in real time. The battery lifetime management is based on users' current activities. Even in the face of disturbances, the control system can control power consumption of the video decoder application regardless of the complexity of the video sequences.

9. Linux-based implementation of the control system. The proposed control system is implemented in the Linux kernel. Experiments based on a concrete computing platform and different decoded sequences are tested to evaluate the Linux-based control system, the accuracy of the feedback estimator and the extended battery lifetime.

10. Experimental and analytical exploration of energy optimization for battery-based mobile systems. Through a comparative analysis of the experimental results under the PCG and the default Linux *cpufreq* governors, this work explores a method to extend battery lifetime based on a power control system, while maintaining a reasonable QoE.

1.5 Methodology and organization

To achieve the above objectives Figure 1-1 indicates the methodology and organization of this thesis. Through the block diagram, it can be seen the structure of the dissertation around the design, simulation and implementation of the control system.

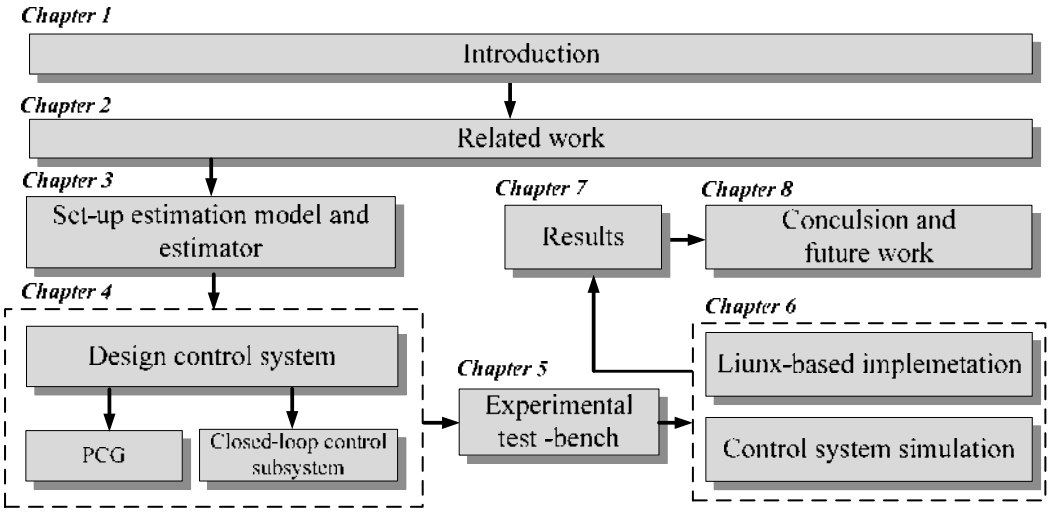


Figure 1- 1 Block diagram of the Methodology and Thesis Organization

Chapter 1 introduces the background and challenges of energy optimizing in battery-limited mobile devices. It lists the issues that need to be solved and also proposes the objectives of this thesis. Chapter 2 concerns the related research field, from the origin of the power management until the advanced techniques to optimize the energy consumption. Besides, comparisons of the related work with the control system are proposed in this dissertation. Chapter 3 presents the method of building estimation model and setting up the software estimator, which can be used as feedback so that no hardware sensors are required. Different approaches have been explored and their features have been compared in order to select the most suitable one for the next step. In Chapter 4 the control system is designed and the whole system is separated in two parts: closed-loop control subsystem and PCG. The former introduces the method of modeling the PuC and designing the controller. The latter presents the method to design the PCG, which can control the battery lifetime by using the feedback estimation and dynamically supports set-points for the closed-loop subsystem. Chapter 5 describes the test-bench and experimental methodology. The computing platform is introduced from both hardware and software point of view. The tools used to measure the power consumption, construct the applications and access performance monitor counters (PMC) are introduced in detail, besides, the system simulator is also explained. After that, Chapter 6 presents the implementation of the control system in a Linux-based test bench. The implementation consists of the two approaches of the estimator, classic controllers and the PCG. Considering the complexity and difficulty of the Linux implementation work, a simulation tool is employed for verifying the behaviors of

different controllers and the PCG. In Chapter 7, the results are analyzed and discussed. The accuracy of the estimator has been verified and, based on that, the behavior of controllers has been compared with the simulation results to check the correctness. What's more, the whole control system is tested under constant workload and variable workload and the results compared to other approaches. Finally, Chapter 8 concludes the thesis work and suggests directions for future research.

Chapter 2 Related work

Given the increasing concern on saving energy whenever is possible, a great number of research developments can be found related to energy consumption optimization in microprocessor-based systems, even with the application of control techniques [9][10]. The development of the power control system needs an investigation of the related works on both energy awareness and control algorithms. In this chapter, different power management mechanisms are firstly surveyed, with a focus on the power awareness of multimedia applications and battery lifetime management of the mobile devices. Then, the control algorithms against different embedded systems and applications are investigated and comparatively discussed. Besides, the investigations of generating energy consumption models of video decoder applications are introduced. After that, the related works on power estimation and control algorithms are summarized, and finally the possibility of applying control algorithms for energy optimization is discussed.

2.1 OS- and Application-level Power Management

2.1.1 Introduction

In recent years, there has been continuous consideration, research, and innovation of the energy management of mobile devices, not only in battery-operated systems [11][12] but also in wireless networks [13] and multimedia applications [14]-[17]. To reduce power consumption of embedded processors, a Power Management Unit (PMU) with Power management (PM) capability is often employed. PM observes the state of the system and the workload to control the power-performance tradeoff of the system by issuing a given PM policy. Most PM schemes fall into two categories: Dynamic Power Management (DPM) [18]-[20], which is designed to deliver peak performance of CPU and disk and then reset the system into the idle mode, and Dynamic Voltage and Frequency Scaling (DVFS) [21]-[24], which is a framework to change the frequency and/or operating voltage of the processor based on system performance requirements. The design principle of both DPM and DVFS is to allow the devices to perform needed tasks with the minimum amount of required power. To achieve this objective, these two PM schemes firstly allow the applications to be executed with the desired performance requirements, and after the

performance goal is achieved, it starts to save energy consumption. Therefore, these PM schemes can be considered as low power-aware which cannot adjust the energy consumption depending on the battery discharging status. In order to guarantee the battery lifetime and maintain the reasonable QoE, the power consumption of applications and the battery lifetime should be managed.

2.1.2 Power-aware schemes

To monitor the power consumption of APPs, a number of research works have been carried out from different aspects. One example of energy management for multimedia applications in battery-based devices is that of Kamat [11], which conserves the battery power by intelligently exploiting the features and redundancy that are specific to multimedia applications. Energy awareness is built into each application and depending on the battery state, which is monitored by a sensor. Mercati et *al.* [25] proposes the Applications-dependent Power states (AP-states) to monitor the frequency of Central Processing Unit (CPU) and Graphics Processor Unit (GPU) as well as the execution time of each APP. Then, the average power consumption of each APP can be calculated by the pairs of frequency and execution time. Hwang *et al.* [26] proposed a PMU design that is a hardware-based method of collecting and analyzing the pattern of Program Counter (PC) values to make predictions on when the next I/O device accesses will resume. There are also previous approaches to this type of estimation, such as Wang *et al.* [27], where the L2 cache power consumption is estimated by using the processor PMCs (Performance Monitor Counters), or Lively *et al.* [28], and Xiao *et al.* [29], where those PMCs are used in combination with the multivariate adaptive regression splines (MARS) method to model an energy consumption pattern.

There are many methods to access PMCs, for instant, in windows 2000 and later versions there are graphic tools, such as System Monitor, Performance Logs and Alerts, and Server Performance Advisor, that can indicate how the system performs by counting the data which are consumed by applications. Besides, Linux provides tools, such as *perf* [30] and *perfmonX* [31], which are performance monitor interfaces to access PMCs from user space. Red Hat Enterprise Linux 6 includes Performance Counter for Linux (PCL) which is a new kernel-based subsystem for collecting and analyzing performance data. The system-activity-related parameters, also known as PMC events, may vary based on the performance monitoring hardware and the

software configuration of the system. To sum up, in order to regulate the power consumption of commonly embedded multimedia systems without the need of adding power monitors, to estimate the power consumption based on PMCs is a smart choice.

2.1.3 Battery lifetime-aware management

Battery lifetime-aware schemes are aware of the battery discharging state and are able to adapt the application performance according to the user requirements while maintain a reasonable QoE [32]. Under battery lifetime-aware schemes, the target battery lifetime can be achieved if the applications can adapt their performance based on remaining battery energy [33]. Flinn *et al.* [34] firstly proposed the Odyssey platform [35][36], which achieves the battery lifetime by periodically measuring the residual battery energy, and predicting future energy demand based on historical power usage. The behaviors of applications will self-adapt based on the energy demands. Kamat [11] built the energy awareness into each application and the battery lifetime is extended by changing the operating point of the applications. Mercati *et al*. [25] presents a method to maximize performance of applications while letting the device battery to last at least for a certain required lifetime. In their research, the power consumption of frequently used APPs are measured to estimate the battery lifetime while running different APPs. Nemesis [37] requires applications to be energy-aware and cooperative, but introduces a model of Quality of Service (QoS) to provide feedback to the applications. Since there are millions of applications, battery-lifetime-aware management through operating system is more economic than through applications that have to be programmed as self-adaptive.

2.2 Control algorithms for energy optimization

Efforts towards research and innovation of control algorithms for saving energy have increased during the last years, and across a wide variety of microprocessor-based areas. A great number of these research lines are based on control systems. The application of closed-loop techniques appears in the literature of all these fields with widespread use of DVFS. However, where there is a broader variety of proposals is in how to feed back the closed-loop system, mainly because there is not a clear feedback signal available in conventional platforms, as mentioned above.

In [38] and [39] the controlled variable is the processor utilization factor (U), which is varied through the DVFS system by means of a PI controller. The energy savings increase as U

approaches 100%, meeting the task deadlines. Also based on targeting a suitable value of U, in [40] the feedback signal is the memory access rate (MAR), calculated from PMC values. Some examples based on DVFS are [41], in which a PID controller is used to minimize the energy-delay product by controlling the number of data/instructions stored in uniprocessor multiple-clock domain queues and threads in chip multiprocessor queues; and [42], where a nonlinear controller is used in queue-based streaming applications.

Other closed-loop approaches are those in which the controlled variable is a time for which a relationship with energy consumption can be found. For example, end-to-end delay in [43] or average slack time in [44] and [45], all of them are based again on DVFS. There are cases in which the control loop adapts the DVFS OPP to the rightly needed frequency by estimating the processor workload, like [46] where the clock cycles for each game frame are estimated by a PID controller; and [46] where a Kalman filter estimates the computation time needed by MPEG-2 decoded frames. Another example, presented in Ramakrishnan *et al.* [47], is a fuzzy-logic-based closed-loop control system whose feedback information is actual received-signal strength. In this system, a base station receiver detects the received power level from a mobile station through a reverse channel. Then from that power level, the base station makes an estimation of power control bits and transmits through forward channel control bits to the mobile station so as to adjust the transmitting power of mobile station to the desired level. Other examples are Wang *et al.* [48], and Mishra *et al.* [49], in which linear controllers are inserted in loops where the feedback signal is the processor utilization factor, related to its power consumption. In cases like Garg *et al.* [50], the feedback signal is the occupancy of some system queues, given that a constant occupancy would imply that the consumed energy is the optimal one. Other approaches relate the energy consumption with the processor workload, which acts as feedback signal, such as in Bang *et al.* [46] or in some Linux *cpufreq* governors [51][79][80]. When the actual power consumption is directly used as the feedback signal, as in Wang *et al.* [10] and Kamat [11], some specific power sensors are needed in the system, which is not always possible.

2.3 Decoder-specific schemes

As one of the most energy-consuming multimedia application of mobile devices, there are many researches on energy estimation models of video decoders. Herglotz [58] investigates the energy required by a CPU when decoding videos on mobile platforms. A model is derived that describes the energy consumption of the new HEVC decoder for intra-coded videos. Ren *et*

al. [14] proposed a platform-independent energy estimation methodology, which can estimate the energy consumption of reconfigurable video coding (RVC)-CAL video codec specifications. Monteiro et al. [59] presents analysis of energy consumption of software HEVC decoder, specifically to estimate the energy consumption in all levels of cache hierarchies. X. Li *et al.* [60] proposes an analytical power consumption model for H.264/AVC video decoding using hardware accelerator on popular mobile platforms and the mod*el* is expressed as the product of the power functions of video spatial resolution (i.e., frame size) and temporal resolution. Benmoussa *et al.* [61] developed a model, which describes the relationship between performance and the energy consumption of H.264/AVC video decoding on both Digital Signal Processors (DSP) and General Purpose Processors (GPP) in terms of video bit-rate, clock frequency and a set of comprehensive hardware and video related coefficients. The entire energy model included four sub-models: quantization parameters (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. Their model achieves a balance between an abstract high level model and a detailed lower level one while guaranteeing very good prediction properties for the tested videos. Mallikarachchi *et al.* [62] proposes an energy model whose parameters describe the relationship among energy requirements of decoder, the number of nonzero DCT coefficients (NNZ) and the QP. The proposed model determines the NNZ for a given Coding Tree Unit (CTU) and predicts the energy requirements of the decoder, thereby facilitating the encoder to determine the appropriate level of quantization required for a CTU to generate a bit stream that operates within the decoder's limited energy budget. As the above-mentioned schemes are quite specific to the decoder details, a more general approach will be explored at the system level in this dissertation, which is based on the power consumption of the whole decoder application rather than on some of its parameters.

2.4 Comparison and discussion

As it has been introduced in Chapter 1, this thesis aims to accurately calculate the remaining battery energy based on power estimation of a video decoder application. And the closed-loop control subsystem and PCG are integrated into the operating system, which is not affected by user space. The control system can adjust the power consumption depending on the user requirements in order to guarantee the battery lifetime. Some heuristic optimization algorithms are listed and compared as below.

One example of power saving mechanism in multimedia mobile devices is Kim *et al.* [52], which implemented a load-based processor *hotplug* algorithm. This algorithm periodically monitors the average load of online cores and turns off the surplus cores according average load. In turn, multi-core consists of several single cores, therefore power control of a single core is an energy-saving mechanism which can collaborate with multi-core *hotplug* algorithm to enhance overall power saving. And, instead of *hotplug* multi core, saving energy based on each independent core itself can fundamentally optimize energy consumption in lower level. The research of this thesis is based on energy optimization of the most basic unit to provide the most basic guarantee for more complex mobile devices.

Other recent example refers to the set-top box (STB) as a small multimedia device, which is widely used in smart homes. Jung *et al.* [17] indicate a power saving method by using bitmap-based activity logs to turn on/off some STB functions. What's more, the passive standby mode uses activity logs, which are represented in a bitmap form, to find a pattern and to predict the next user activity. Similarly, Lee *et al.* [53] also suppose a hybrid system model to perform future idle period prediction. But the work developed in this thesis is a real-time control system, which can dynamically adjust the power saving state depending on user current activity, which can accurately satisfy the users' requirements.

Besides, Choi *et al.* [54] investigated a Power-Saving Mode (PSM) for mobile Voice over IP (VoIP) devices in wireless networks. And they evaluated the performance of the VoIP PSM and derived a theoretical maximum bound of sleep interval that minimizes the total power consumption of mobile stations while still guaranteeing VoIP QoS. Lim *et al.* [55] proposed a solution for finding out the optimal checkpoint interval, which minimizes the energy expenditure of a mobile device in remote check pointing wireless environments. There are other researches about the development of 3G/4G networks, such as Huang *et al*. [56] and Fowler *et al*. [57]. Although the work of this thesis focuses on the video-decoding task as the main power-consuming application, it could also be considered to be conveniently transplanted to other systems with suitable adaptations. For example, it could be introduced into power saving mechanisms related to 3G/4G like those referenced above. I.e., when the wireless interface transfers the multimedia data packets, the proposed closed-loop subsystem could feed the power-consumption estimation back, which could help the wireless module to adjust the power saving model [56] while satisfying QoE. .

In addition, comparing with other related work, such as the work of Ren [14], in which it can be highlighted that power measurements are correlated off-line with counts of some suitable events by using the processor PMCs. From that correlation, static power estimations are obtained for the processor decoding video at a fixed OPP. In order to implement power optimization in mobile devices, the power estimation should react to variable OPPs. One further innovation of this thesis is an accurate PMC-based estimator, which can be applied with all 27 OPPs. What's more, the estimation subsystem was implemented within the operating system, such that it is able to calculate power estimation samples in real time, periodically and independently of the video-frame rate in order to act as feedback for the control system.

Besides, currently, operating systems also provide dynamic governors that support energy saving. For instance, in Linux operating system, there are two dynamic governors, ondemand and conservative [51][79][80], which can reduce energy consumption depending on the system workload. Although they can extend the battery lifetime to a certain degree, the battery lifetime varies depending on the workload. The work of this thesis not only can extend longer battery lifetime, but also guarantee the work time of mobile devices to satisfy user's requirements regardless of the video decoding workload. Considering that the original Linux dynamic governors are widely used to optimize energy consumption and that they are highly related to this work, more quantitative comparison details are explained in Chapter 7.

As a summary, comparing with other control algorithms, the work of this thesis can dynamically adjust the battery lifetime between the shortest battery lifetime (under the best performance) and the longest (under the lowest performance) while still maintaining a reasonable QoE, as well as guarantee the battery lifetime regardless of the workload variation. The work of this manuscript is based on the basic processing unit that can be integrated into other multi-core devices to optimize energy consumption. "In order to get a more meaningful comparison, the results obtained are finally contrasted with other highly related works which a coherent comparison can be set with, such as original Linux governors (see Section 7.4). Besides, the accuracy of the power estimation model is compared with Ren et al. [74] in Section 8.1.

2.5 Summary

In this chapter, the related works on OS- and application-level power management, control algorithms for energy optimization, and decoder-specific power management have been surveyed. In some specific cases, the target system includes a power monitor unit that is able to

feed actual consumption data back to the closed-loop controller. In order to generally apply the OS-level power management to common mobile devices, there is research using third party-tools, graphic tools and hardware tools to monitor the power consumption of the applications. To directly and conveniently achieve power management, the aim of this thesis is to reach a control system which can regulate the power consumption of an embedded multimedia system without the need of adding power monitors but relying on power estimations derived from commonly available information. Therefore the researches that access the PMCs to estimate the power consumption while executing applications give a good inspiration to this thesis. What's more, since a good control of the battery lifetime is pivotal to the user experience of mobile terminal, battery-lifetime-aware management schemes have also been investigated. The researches of battery-lifetime-aware usually have two focal points, one is to merge energy-awareness into the applications, and another one is battery lifetime management through the operating system. Considering that the inclusion of energy self-adaptation into applications would imply to modify lots of them, the later solution is more convenient and economical. Besides, a great number of research lines are related to control algorithms for energy optimization. Some of them apply closed-loop and DVFS techniques to control the energy consumption and there are various feedback signals, such as memory access rate, actual received-signal strength, occupancy of some system queues, processor utilization factor and so on. There is not a clear feedback signal available in all conventional platforms to promote the application of closed-loop control; therefore, the idea of using commonly available information, such as PMCs is a practical choice. Then, comparing with other control-based research lines, the idea of this thesis can complement and cooperate with other energy-saving control algorithms. Finally, video decoder application is one of the most energy-consuming multimedia applications. This thesis focuses on energy optimization of decoder multimedia application, which can significantly alleviate the energy consumption problems. Previous researches about decoder-specific power management give inspiration to us, but they are quite specific to the decoder details, a more general approach will be explored at the system level in this dissertation.

Chapter 3 Power estimator

Chapter 2 has introduced some researches that focus on energy optimization in mobile devices while running different multimedia applications. Among those researches, sometimes control techniques are applied in the operating system that can conveniently and effectively control the power consumption in order to guarantee the battery lifetime. The present work aims to base the control system in a feedback signal as close to the actual power consumption as possible, but without needing specific power monitoring sensors that are not available in many common consumer mobile platforms. The adopted solution is to estimate the power consumption from commonly available system event counters. The proposed power consumption estimation method used to feed power consumption information back to the controller is specifically based on Ren *et al*. [14], where energy measurements are correlated off-line with counts of some suitable events by using the processor PMCs. This chapter focus on the integration of the power estimator in both OS and decoder application, as a part of the whole control system. It is based on event counts taken from the PMCs of the MPU.

To simplify the work and focus on the power estimator, this chapter starts introducing PMCs and the method of filtering the PMCs that are highly related with power consumption. Then, two methods of accessing PMCs from user space and kernel space are indicated. After that, how to build power estimation model is explained.

3.1 PMC events selection

3.1.1 PMC introduction

PMCs are used as a valuable tool for measuring performance of a program that can be analyzed to identify the bottlenecks in the program. These counters are hardware registers attached within the processor that measure various programmable events occurring in the processor, such as instructions executed, cache misses or branches miss predicted. These counters are present in most modern processors such as Intel Core and ARM Cortex. They do not require any additional overhead and supports a wide range of events. Implementation of PMCs in different processors could differ from the quantity or the monitored types of events. In a broad sense, PMCs consist of three types: a cycle counter, event counters and counters controlling. The

cycle counter is programmed to increment on every clock cycle; an event counter can be configured to select one specific event and increments as this event occurs; counters controlling is used to control the according PMC to carry out various operations which include enable, reset, start, stop or enable interrupts on counter overflow.

3.1.2 PMC event redundancy

The main concept of the PMC-based estimator is to relate the energy behavior to the occurrence of several events [1], which depend on the hardware monitoring capabilities. The platforms support several PMC events and the available PMC events are different against different platforms.

Introducing as many PMC events as possible is a simple way to estimate power consumption, but it will cause high overhead. Therefore, suitable PMC events should be selected to build the estimation model in order to guarantee the accuracy of power estimator and low overhead. What's more, considering the system integrity and continuity, there are dependencies among PMC events. It means the information provided by one PMC event can be predicted or explained by others PMC events, so some of the PMC events are highly correlated. If two PMC events are perfectly correlated, they include the same content to build the estimator, which will increase unnecessarily the number of PMC events. Since multi-collinearity will lead to PMC event redundancy, a filter method is needed to be applied in order to reduce the PMC events redundancy.

3.1.3 PMCs filter Method

In order to reduce the PMC redundancy and maintain the accuracy of power estimator, the utilized PMCs filter method includes two parts: the spearman rank correlation coefficient is introduced to calculate the dependence relationship between different PMC events which is used to reduce the information redundancy; another part has to do with the method of selecting energy-related PMC events.

3.1.3.1 Spearman Rank Correlation Coefficient

Spearman's rank correlation coefficient (ρ) is a non-parametric statistic parameter that is used to describe the statistical dependence and the relationship between two variables. One

variable is a strictly monotone function of the other if the Spearman correlation coefficient is +1 or -1 when there are no repeated values of the sampling data. These two values, +1 and -1, are called perfect Spearman correlation.

For example, let X_i and Y_i be two variables. When there is no repeated value in the original data samples, the correlation coefficient r_s can be calculated by equation 3-1, otherwise r_ρ is calculated by equation 3-2.

$$r_s = 1 - \frac{6\sum d_i^2}{n(n^2-1)} \qquad 3\text{-}1$$

$$r_\rho = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 (y_i - \bar{y})^2}} \qquad 3\text{-}2$$

Where d_i denotes the difference between the ranks of each observation on the two variables, the original variables X_i and Y_i are converted into ranks x_i and y_i. Let $\bar{x}$ denote the average descending rank of x_i, $\bar{y}$ denote the average descending rank of y_i, n denote the size of a sample and *i* denote the paired score.

Judging the variable dependence with the correlation coefficient is mainly based on experience. For example, if the value of the correlation coefficient is between 0.8 and 1.0, it can be considered that the two variables are strongly related; and when the value belongs to the interval [0.6,0.8], the variables are highly related. If the value is larger than 0.4 and smaller than 0.6, that means there is a moderate relationship between them; otherwise, they only have weak relationship.

A more accurate method to interpret the correlation coefficient is to calculate the coefficient of determination (r^2). The coefficient of determination is the square of the correlation between predicted scores and actual scores and it ranges from 0 to 1. When r^2 equals to 0, it means the dependent variable cannot be predicted from the independent variable and when r^2 equals to 1, it means the dependent variable can be predicted without error from the independent variable. If r^2 is between 0 and 1, it indicates the extent to which the dependent variable is predictable. For example, if r^2=0.850, it means that 85% of the total variation in y_i can be

explained by the linear relationship between x_i *and* y_i. The other 15% of the total variation in y_i remains unexplained.

3.1.3.2 PMC Event Selection method

Not all the performance monitor events are relevant to power estimation; therefore, the events with higher correlation to power consumption should be selected. Assuming a linear correlation between PMCs and power consumption, equation 3-3 is employed to predict the system power consumption:

$$Power = \sum_{i=1}^{n} \alpha_i PMC_i + P_{idle} \qquad 3\text{-}3$$

Where α_i is the linear parameter of power weights, n is the number of selected PMCs and P_{idle} is a constant representing the idle processor power consumption. There are also some non-linear relationships, but this work shows high accuracy with a fully-linear model.

The filtering procedure can identify the set of events that are most significantly related to the power consumption. Then, the Spearman's rank correlation r_ρ is computed between each event and power consumption. After this step, a threshold α is established to eliminate any event below this threshold. On the other hand, to reduce event redundancy, correlations r_ρ *(i,j)* between each pair of events '*i*' and '*j*' are computed to identify the event relationship. The purpose is to eliminate those events whose *information* can also be obtained from other events. Hence, starting from the event '*a*' with the largest correlation r_ρ *a*, those events '*j*' whose correlation r_ρ *(a,j)* exceeds certain threshold β are eliminated. Then, the procedure continues with event '*b*', with correlation value r_ρ *b*, to eliminate the events '*j*' whose r_ρ *(b,j)* exceeds β. This process is repeated until there are no more events to eliminate. Finally, the remaining events are orthogonal to each other and highly related with power consumption. In this work, specifically, α is set to 0.5 and β is set to 0.9 because this leads to the best results.

3.2 Accessing PMCs

PMCs can be accessed through both user space and kernel space. If accessing PMCs from userspace, a suitable PMCs monitor tool should be used. The used tool not only supports the PuC

in this work, but also can be easily applied for other normal different platforms. If accessing PMCs from kernel space, corresponding source code should be added such that the PMCs can be periodically accessed.

3.2.1 Accessing PMCs form userspace

PMCs can be accessed from user space though third-party tools. In Windows operating system, there are some tools with graphical monitoring view of system working state by counting the data consumed by applications. In Linux operating system, performance counters for Linux (PCL) is a new kernel-based subsystem that provides a framework for collecting and analyzing performance-related data. The PMC events will vary based on the performance monitoring hardware and the software configuration of the system, such as perf_event [30] which is an application programming interface (API) of the Linux kernel and perfmonX [31] which is a hardware-based performance monitoring interface for reading the PMCs from user space. Since the hardware used in this work does not support perfmonX nor perf_event, another tool is considered to easily and effectively access PMCs. In Linux operating system, Performance Application Programming Interface (PAPI) is a widely used third-party tool which can easily access PMCs from the application level. Besides, the interface of PAPI is the same for all platforms. Therefore, in this work PAPI is used to monitor PMCs for preliminary tests. The implementation details will be explained in Chapter 6.

3.2.2 Accessing PMCs from kernel space

In order to completely decouple the user applications from the power control system, the power estimator has been included into the OS, which directly and periodically accesses the PMCs for carrying out the estimation task at kernel level. The need of including the estimator into the OS implies to know the processor low-level details and to develop the code to access the registers.

The work of this thesis focuses on control algorithms for energy optimization in multimedia devices; therefore a development board with a single-core is considered to be the smallest unit of study. Cortex A8 is the general-purpose processor included in the experimental test bench used in this work and it has four PMCs, which are accessed, in system control coprocessor (CP15) space. CP15 can control and provide status information for the functions

implemented in the processor. It has some main functions, such as overall system control and configuration, cache configuration and management, Memory Management Unit (MMU) configuration and management, preloading engine for L2 cache and system performance monitoring. Performance monitor registers are mapped into the CP15 register and the purpose of it is to monitor and count system events, such as cache misses, TLB misses, pipeline stalls and other related features. Performance monitor registers can help system developers to profile energy-related behaviors of the processor when it executes different applications. It can generate interruptions when the number of events reaches a given value.

Table 3-1 shows a summary of the register allocation and reset values of the performance monitor registers C9, which is used to control CP15 and reserved encodings for implementation-defined performance monitors. In the table, CRn is the register number of CP15, Op1 is the Opcode_1 value for the register, CRm is the operational register, Op2 is the Opcode_2 valued for the register, and security state can be either secure (S) or non-secure (NS). R/W means read/write access in privileged modes only and X indicates the register access depends on another register or an external signal.

Table 3-1 CP15 Performance Monitors in Cortex A8 processor

CRn	Op1	CRm	Op2	Register or Operation	Security State NS	S
c9	0	c12	0	Performance Monitor Control(PMNC)	R/W,X	R/W,X
			1	Count Enable Set (CNTENS)	R/W,X	R/W,X
			2	Count Enable Clear (CNTENC)	R/W,X	R/W,X
			3	Overflow Flag Status (FLAG)	R/W,X	R/W,X
			4	Software Increment (SWINCR)	R/W,X	R/W,X
			5	Performance Counter Selection (PMNXSEL)	R/W,X	R/W,X
		c13	0	Cycle Count (CCNT)	R/W,X	R/W,X
			1	Event Selection (EVTSEL)	R/W,X	R/W,X
			2	Performance Monitor Count (PMCNT)	R/W,X	R/W,X
		c14	0	User Enable (USEREN)	R/W	R/W
			1	Interrupt Enable Set (INTENS)	R/W	R/W

The purpose of the performance monitor control (PMNC) register is to control the operation of the four performance monitor counts registers and the cycle count register, as well as to inform about the hardware processor and the number of PMCs available in the hardware. Besides, in order to enable or disable any PMCs, the count enable set (CNTENS) or count enable clear (CNTENC) register is needed. When reading these two registers, any enable that reads as 0

indicates the counter is disabled and any enable that reads as 1 indicates the counter is enabled. When writing to the enable-bit of CNTENS with the value of 0 is ignored, and when writing with the value of 1 indicates to enable the counter. Similarly, writing into CNTENC with the value of 0 cannot update the counter state while writing with the value of 1 clears the enable-bit to 0 to disable the counter. The overflow flag status (FLAG) register can enable or disable any of the PMCs to produce an overflow flag. When reading this register, overflow flag that is read as 0 indicates the counter has not overflowed, and once there is any overflow that reads as 1 indicates the counter has overflowed. And, if the interrupt overflow enable bit is written with a value of 0 it is ignored, while any overflow flag written with a value of 1 clears the counter overflow. The purpose of the software increment (SWINCR) register is to increment the count of PMC register. When writing to this register, the value of 1 increment the counter and the value of 0 does nothing. The performance counter selection (PMNXSEL) register can select a PMC register through writing the corresponding bit value into the SEL field. The purpose of the cycle count (CCNT) register is to count the number of clock cycles since the register was reset. It should be disabled before any software can write into it; otherwise, any attempt to write to this register when it is enabled will lead to unpredictable result. Moreover, the event selection (EVTSEL) register can select the events which PMC registers are needed for counting. In Cortex A-8 processor, the four PMCNT registers (PMCNT0-PMCNT3) are selected by the PMNXSEL register and each of them counts instances of an event that is selected by the EVTSEL register. Bits [31:0] of each PMCNT register contain an event count. Accessing the PMCNTs in user space needs to enable the user mode of the PMCNTs, and user enable (USEREN) register is used to control this configuration. The purpose of the interrupt enables set or interrupt enable clear (INTENS) register is to determine if any of the PMCNTs, PMCNT0-PMCNT3 or CCNT, generates an interrupt on overflow. When reading this register, any interrupt overflow enable bit read as 0 indicates the interrupt overflow flag is disabled, and when any interrupt overflow enable bit is read as 1, it indicates the interrupt overflow flag is enabled. Writing to this register is similar, when any interrupt overflow enable bit is written with a value of 0, it will be ignored. Any interrupt overflow enable bit written with a value of 1 sets the interrupt overflow enable bit.

PMCs can be accessed by reading or writing CP15 with the ARM assembly MRC and MCR instructions, respectively.

The instruction MRC can transfer an internal co-processor register to an ARM-processor register. It takes the form as below:

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

The register <co-pro reg> is written to <ARM reg>, by using operation <op>, while the register <co-pro reg2> is written to <ARM reg> by using operation <op2>.

The form of instruction MCR is the same as MRC and MCR instruction also can transfer a co-processor register to an ARM-processor register. It takes the form as below:

MCR <co-pro>, <op>, <ARM reg>, <co-pro reg>, <co-pro reg2>, <op2>;

The contents of the ARM register are written to the co-processor register using the given operation code, which may be further modified by the second co-processor register and/or the second operation code.

The process of accessing PMCs from kernel space can be described as below:

The PMNC can control the operation of the PMCs, with one register used to set up each counter. The four PMCNT registers contain the event counts for the selected events being counted. The MRC instruction can be used by programs or procedure running at any privilege level to read these counters. After setting CNTENS and EVTSEL, PMCs start to work until FLAG or CNTENC are enabled. Besides, the counters can be stopped by clearing the enable counters flag or by clearing all the bits in the CNTENC. The PMCs are periodically recorded, and the data can be used to analyze the performance of the applications.

3.3 Modeling power estimator

Modeling power estimator is to find the relationship between PMC events and real power consumption. Chapter 2 introduced some methods to estimate power consumption by using PMCs. Based on Ren et al. [1], in this work the mathematic method is MARS which can accurately estimate the power consumption.

3.3.1 MARS method

Multivariate Adaptive Regression Splines (MARS) is an implementation of techniques to predict the values of a continuous dependent or outcome variable from a set of independent or predictor variables. MARS constructs the relationship between the dependent and independent variables from a set of coefficients and basis functions that are entirely driven from the regression data.

MARS partitions the input variable space by the tensor product of interval sets on each of the *n* axes and the tensor product of spline functions are defined as the basis functions. Each input axis is partitioned into *K+1* intervals delineated by *K* points ("knots") and the regions in the n-dimensional space are taken to be the $(K+1)^n$ intersections of all such intervals. Each divided space corresponds to a coefficient and an input variable x_i. MARS model obtains its prediction value by combining all basis functions. In the system, the input set contains the independent variables $x = (x_1, \dots, x_q)$, and the output set contains the dependent variables $y = (y_1, \dots, y_q)$. The method that generates the data is described by equation 3-4.

$$y = f(x_1,...,x_q) + \varepsilon \tag{3-4}$$

Where f is a single valued deterministic function of its n-dimensional argument, q is the number of independent variables, ε is the additive stochastic component whose expected value is defined to be zero.

MARS can obtain an approximated function $\hat{f}$ to analyze and calculate the system response through a series of training data, and it can be represented by summing up a set of basis functions as indicated in equation 3-5:

$$\hat{f}(x_1,...,x_p) = \sum_{m=1}^{M} c_m B_m(x) \tag{3-5}$$

c_m are the coefficients of the expansion and *M* is the number of basis functions.

The basis function $B_m(x)$ is indicated in equation 3-6:

$$B_m(x) = \prod_{k=1}^{K_m} b_{km}(x_{(k,m)} \left| P_{(k,m)} \right.)_+ \tag{3-6}$$

K_m is the number of factors in the tensor product; $x_{(k,m)}$ is a subset of independent variables; $P_{(k,m)}$ is a set of functional parameters, i.e. $P_{(k,m)} = (s_{km}, t_{km})$. b_{km} is a constant or a hinge function expression in 3-7.

$$b_{km}(x|s,t)=[s\cdot(x-t)]_{+} \qquad (-\infty\le t\le+\infty) \tag{3-7}$$

Where s is the truncated direction $s=\pm 1$; t is the knot position of the basis function. The subscript "+" of 3-6 indicates a positive part, i.e.:

$$z_{+}=\begin{cases} z & z>0 \\ 0 & z\le 0 \end{cases} \tag{3-8}$$

To achieve the desired accuracy of objective function $\hat{f}$, MARS algorithm obtains the set of basis functions through a forward and backward iterative process. Forward pass iteratively divides the training data and fits the estimation models, which will produce a large number of basis functions. The backward pass will selectively remove some basis functions with the premise to ensure the highest suitability to fit the final model.

In order to improve estimation accuracy and save computing time, a reasonable number of counts should be selected. Therefore it is unnecessary to test whether each point is suitable for a new basis function. There are a great number of data for knot calculation; therefore, a minimum step size L for variable selection is introduced, which can reduce the selection of data. The step L is calculated as in equation 3-9:

$$L(\alpha)=\frac{-\log_2\left[-\frac{1}{nN_m}\ln(1-\alpha)\right]}{2.5} \tag{3-9}$$

Where α locates in a closed interval $[0.01, 0.05]$, which is a reasonable range for narrowing the selection of candidate nodes, n is the number of predictors or input variables; the quantity N_m is the number of observations.

In the forward procedure, the entire iterative process will continue until the number of basis functions reaches the maximum number of basis functions $M_{\max}$ or the minimal lack of fit

(*LoF*) is achieved, being LoF the difference between the real function f and the model function $\hat{f}$.

As mentioned above, since MARS algorithm only allows building new basis functions based on those previous generated basis functions, this will cause too many basis functions constructed by the forward procedure. Besides, the original generated functions used to produce subsequent basis functions have little influence on the final model. Therefore, in order to improve the generalization ability, MARS backward procedure will remove the basis functions which have small effect on the final model, until it finds the best model. The performances of models are evaluated by using generalized cross validation (GCV), which is calculated as in equation 3-10.

$$GCV(M) = \frac{\frac{1}{N}\sum_{i=1}^{N}\left[y_i - \hat{f}_M(x_i)\right]^2}{\left[1 - \frac{C(M)}{N}\right]^2} \qquad 3\text{-}10$$

Where N is the number of observations; $C(M)$ is the effective number of parameters and M is the number of hinge-function knots.

The final model obtained by the MARS algorithm is expressed in equation 3-11:

$$\hat{f}(x) = c_0 + \sum_{m=1}^{M} c_m \prod_{k=1}^{K_m}\left[s_{km} \cdot (x_{v(k,m)} - t_{km})\right]_+ \qquad 3\text{-}11$$

Where c_0 is a constant basis function, M is the number of basis functions, c_m is the constant coefficient of every basis function and $s_{km} = \pm 1$. In this thesis K_m is set as 1 in order to simplify the model, which also simplifies equation 3-11 to 3-12:

$$\hat{f}(x) = c_0 + \sum_{m=1}^{M} c_m \left[s_m \cdot (x_{v(m)} - t_m)\right]_+ \qquad 3\text{-}12$$

In order to make the model be continuous and have continuous first derivative, the hinge function can be replaced by its corresponding cubic truncated form as equations 3-13 and 3-14:

$$C(x|s=+1,t\_,t,t_+)=\begin{cases}0 & x\le t\_,\\ P_+(x-t\_)^2+r_+(x-t\_)^3 & t\_<x<t_+,\\ x-t & x\ge t_+\end{cases} \quad 3\text{-}13$$

$$C(x|s=-1,t\_,t,t_+)=\begin{cases}-(x-t) & x\le t\_,\\ P_-(x-t_+)^2+r_-(x-t_+)^3 & t\_<x<t_+,\\ 0 & x\ge t_+\end{cases} \quad 3\text{-}14$$

with $t_- < t < t_+, t$， t_+ and t_- are the knots of cubic function. $P_\pm$ and $r_\pm$ are as 3-15 to 3-18.

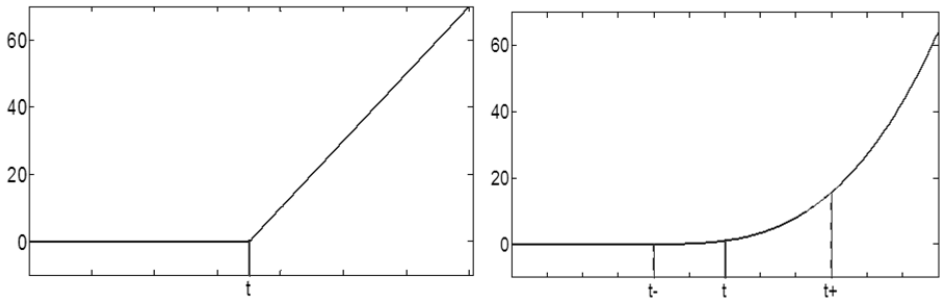
$$P_+=\frac{(2t_++t_--3t)}{(t_+-t_-)^2} \quad 3\text{-}15$$

$$r_+=\frac{(2t+t_+-t_-)}{(t_+-t_-)^3} \quad 3\text{-}16$$

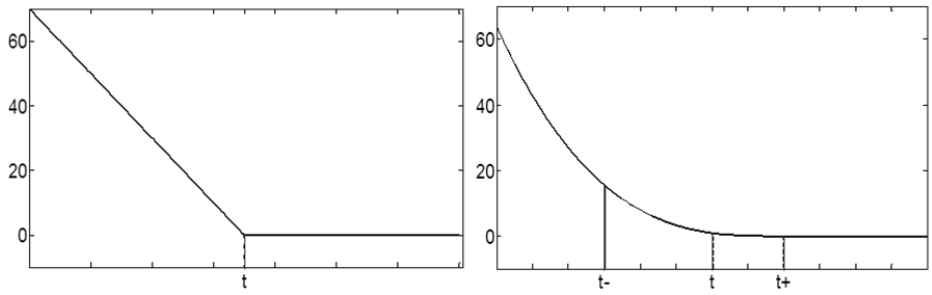
$$P_-=\frac{(3t-2t_--t_+)}{(t_--t_+)^2} \quad 3\text{-}17$$

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

$C(x|s,t_-,t,t_+)$ is first order differentiable, but its second derivative is not continuous at $x=t_\pm$. Each knot t can define a linear truncated function, while a cubic function needs three knots: t， t_+， t_-. Figure 3-1 shows an example of linear and cubic hinge functions [14]. The cubic functions are smoother and accurate than linear functions, which do not introduce many computable complexities.



(a) Linear and cubic basis functions when s=1.



(b) Linear and cubic basis functions when s=-1.

Figure 3 -1 Linear and cubic basis functions

3.3.2 Building the power estimation model

Since system activities can be quantified with PMCs, the power estimation model builds the relationship between power consumption and a set of selected PMC events. Figure 3-2 describes the power estimation modeling procedure, in which the MARS method is employed to fit the selected PMC events and power consumption due to its simplicity and high efficiency. In order to estimate the power consumption of PuC, PMCs are recorded when executing the application under different states of PuC. The current consumption of the PuC and supply voltage have been measured and their multiplication is used as the power consumption input. Section 3.3.1 explains the MARS modeling procedure and once the coefficients of independent variables are set, the model can estimate the power consumption. Then, by comparing the estimation with power consumption, the accuracy rate is calculated. When the accuracy is not good enough, it means the estimation value cannot suitably reflect the power consumption. In this case, some adjustments should be considered, such as re-select PMCs, remove spikes of PMCs data, re-calculate average power consumption and so on. If the accuracy of the model is adequately good, the estimation model can be used to estimate the power consumption of PuC.

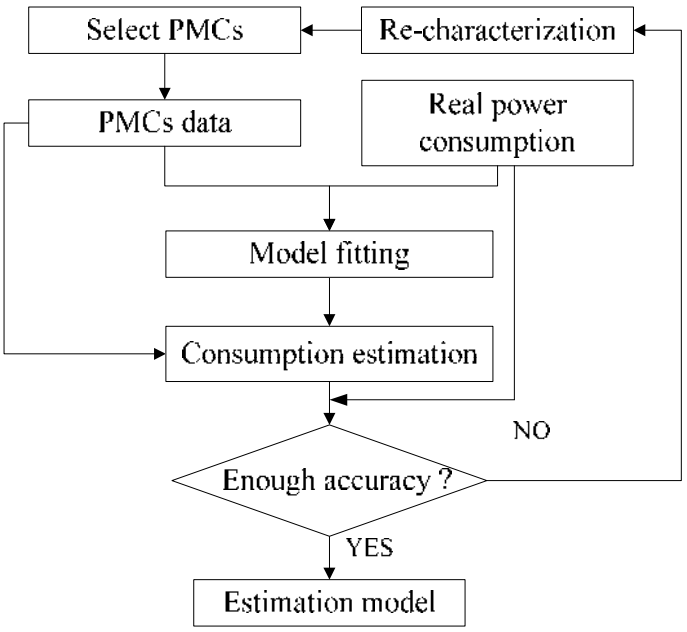


Figure 3-2 Structure diagram of power estimation modeling procedure.

3.4 Summary

In this chapter, the power estimation modeling method has been presented. The model based on PMCs that can reflect the power consumption while running the video decoder has been introduced. To graduate the accuracy of the power estimation and do not cause too much overhead, the PMCs strongly related with power consumption should be selected. Correlation coefficient is a common method to evaluate the degree of relationship between two variables. Based on Spearman's rank correlation coefficient and coefficient of determination, suitable PMC events have been selected. PMCs can be accessed through user space and kernel space. In this dissertation, PAPI has been merged into the decoder to monitor the PMCs from userspace. Moreover, PMCs can also be accessed from kernel space. Once the list of significant events has been obtained, MARS method is applied to build the power estimation model. Based on selected PMC events and power consumption of PuC, MARS function can calculate the parameters of models. After checking and adjusting the accuracy of power estimation, the power estimator can be refined and fixed.

Chapter 4 Real-time control system

4.1 Introduction

This section focuses on control algorithms for energy optimization in multimedia mobile devices when they are decoding video, while also keeping a reasonable quality of user experience (QoE). For this reason, a control system has been built in the PuC, in which the microprocessor unit (MPU) executes a video decoder application. Figure 4-1 shows the general topology of the control system, which includes a power-control governor (PCG) and a closed-loop control subsystem. With respect to the closed-loop subsystem, depending on the power-consumption set-point, the controller will operate on the multimedia platform (plant) input and keep the plant output close to the set-point. For the plant input, the dynamic voltage and frequency scaling (DVFS) mechanism of the MPU is employed because it is present in many commercial platforms. DVFS enables to adjust the MPU dynamic power consumption by putting it to work under different operating performance points (OPPs), i.e., voltage and frequency pairs. Meanwhile, the plant output provides the feedback signal, which should be its own power consumption. The feedback information not only works in the closed-loop subsystem, but also feeds the PCG. The PCG consists of a battery-discharge estimator, which estimates the remaining battery energy, i.e., the battery state of charge (SoC), and a power-budget generator, which selects in real time a suitable power budget for the MPU to guarantee a certain battery lifetime. The generated power budget is passed to the closed-loop control subsystem as its *set point.* Due to its feedback feature, the closed-loop control subsystem is able to regulate the MPU power consumption, and, hence, the battery discharge rate, according to the set-point, regardless of possible fluctuations in the decoder power-consumption demand (which act as disturbances over the closed-loop subsystem). One more interesting innovation is that PCG supports multiple and personalized power-budget profiles to meet user requirements. Then, different control algorithms have been implemented in the control system to verify the effectiveness and stability of the system.

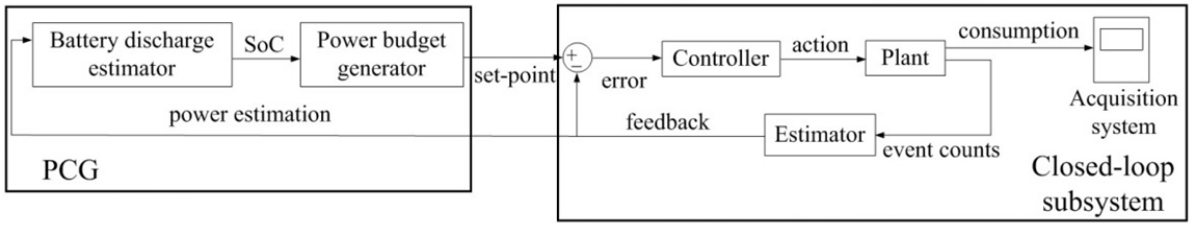


Figure 4-1 General topology of the control system.

In order to generally apply the control system in common multimedia mobile devices, instead of using any special power sensor, the precise PMC-based OS-level real-time power-consumption estimator has been used as the feedback-signal generator (see Figure 4-1). The proposed estimator has been introduced in chapter 3.

4.2 Theoretical model of the closed-loop control subsystem

The general topology of the proposed closed-loop control subsystem is depicted in Figure 4-2. The idea is to regulate the power consumption according to a *set point* (target), i.e., the system keeps the power consumption close to the *set point*, which will depend on the power needs at each moment, regardless of possible consumption fluctuations (disturbances) in the multimedia plant, i.e., the MPU executing the video decoder. For this purpose, a *controller* acts on the *plant* depending on the system *error* between the target and the estimation-based *feedback*. Both the feedback from, and the *action* to the plant are proposed to be based on features generally available in common mobile processors, i.e., PMCs and DVFS, respectively. Anyway, as also depicted in Figure4-2, an acquisition system has been used during the test phases to confirm that the control system is working properly. Additionally, as depicted in Figure 4-1, based on power-estimation feedback, the battery-discharge estimator will calculate the battery state of charge (SoC). In that way, the power-budget generator can correspondingly select a suitable power budget for the plant in order to guarantee battery lifetime. Therefore, the set-point will dynamically change based on power budget.

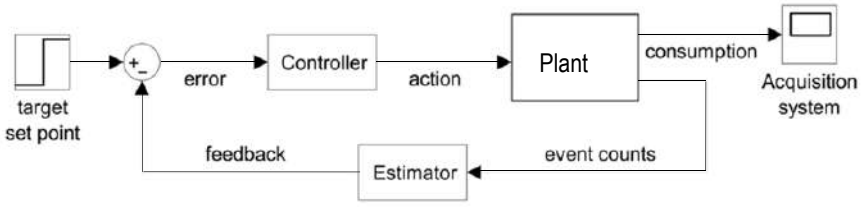


Figure 4-2 General topology of the proposed closed-loop consumption control system based on estimation feedback.

4.2.1 Plant model

One mean of designing the system controller is to base it on a suitable model of the plant. As a first approach to the problem, working with actual consumption data, a simplified model is used to facilitate the application of the classic control theory. Later, this model could be refined and sophisticated and different advanced closed-loop control strategies could be applied.

For modelling purposes, the actual power consumption of the open-loop plant has to be captured when it is decoding video under different OPPs. As exposed in more detail in Chapter 5, this has been done for the experimental test bench by measuring the board current consumption.

As an example, Figure 4-3 shows the real measured current of the board while continuously decoding video and with OPPs increasing. The repetition of the capture experiment for always the same video sequence indicates that there is a certain basis of (average) consumption for each OPP, which is nearly constant for all repetitions, plus a number of big consumption spikes that appear in different moments in each repetition. For this reason, those spikes are not considered to be due to the video task executed in the CPU but to other "irrelevant" sinks in the board. Moreover, it can be observed from the graph that, apart from the biggest "random" consumption spikes, there is a second level of pseudo-periodic consumption peaks, whose period decreases as the OPP frequency increases. These are due to accesses to the SD card to get the video file data packets but not to decoding activities. Hence, these consumption peaks should not be taken into account to model the plant, given that neither the power estimation procedure will reflect them.

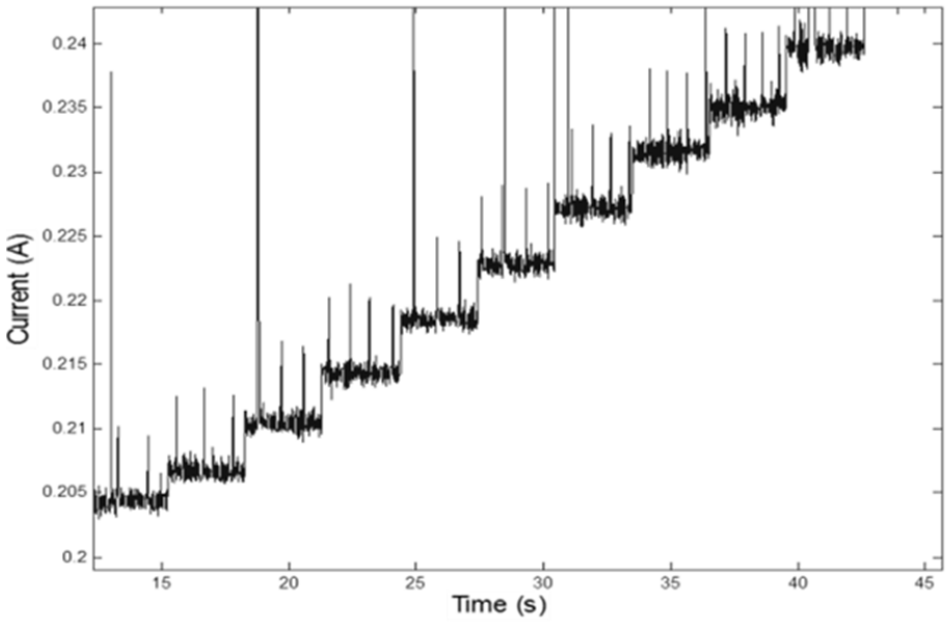


Figure 4-3 Detail of the real board consumption profile for increasing OPPs

If the zoom of Figure 4-3 is focused on how the consumption changes from one OPP to another, the dynamics of this change can be analysed. From this analysis, a mathematical model of the system plant can be obtained. Thus, for example, choosing a simple linear first-order Laplace-based model, a transfer function of the plant, $G(s)$, can be obtained [63]. Equation 4-1 shows *G(s)* for the experimental test bench:

$$G(s) = \frac{1}{2.75\times10^{-3}s+1} = \frac{363.63}{s+363.63} \qquad 4\text{-}1$$

G(s) in Equation 4-1 relates the current consumption with input OPP average current level[3]. Figure 4-4 shows the comparison between the time response of this theoretical model and the actual consumption for an input step from OPP26 to OPP27 levels in the experimental test bench. The time response of *G(s)* has been compared also with the rest of steps of the OPP sequence described in Figure 4-3 and its average validity has been verified.

[3] The consideration of the measured OPP average current level as the input for the plant transfer function is for normalization purposes, i.e., unitary steady-state gain.

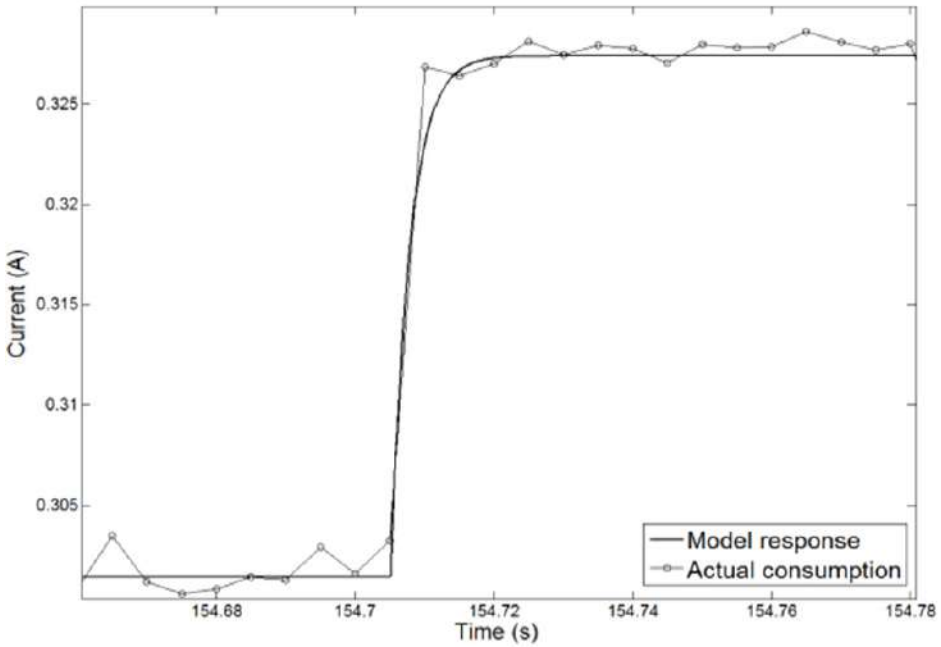


Figure 4-4 Actual consumption and model response for OPP26 to OPP27 step

The proposed model *G(s)* is a continuous one, which has to be discretized depending on the action period *T* to be considered. If a digital-to-analog converter (zero-order hold) + continuous process + analog-to-digital converter (sample & hold) scheme model is considered for the discretization, a Z transfer function can be derived from the continuous model as equation 4-2 [64]:

$$G(z)=\frac{1-e^{pT}}{z-e^{pT}} \qquad 4\text{-}2$$

Where *p* is the pole of *G(s)*. As it can be deduced from Figure 4-4, the time needed by the hardware to adapt to a new OPP to change the power consumption is quite short. Mainly, this time should be always much shorter than the action period *T* in order to avoid unnecessary overhead. Taking this into account, the value of the pole of *G(z)* in Equation 4-2 should tend to zero. Then, the discrete-time transfer function of the plant can be simplified to the one shown in equation 4-3 [65]. If the results finally prove the validity of this model, it will have the additional advantage of its simplicity and generality.

$$G(z)=\frac{C(z)}{A(z)}\approx\frac{1}{z} \qquad 4\text{-}3$$

The z^{-1} term in $G(z)$ represents that, for a reasonable low-overhead action period T, the system adapts its power consumption after an OPP step in less than a sample time. On the other hand, the unitary static gain of $G(z)$, i.e., $G(1)$=1, represents a normalized model in which the input, modelled with $A(z)$ in 4-3, is the average power consumption value corresponding to the active OPP and the output, modelled with $C(z)$ in equation 4-3, is the power consumed by the system for that OPP. As introduced above, this simple model is valid while the system sample period is much longer than the settling time of the analogue power consumption process, otherwise the system overhead would be unbearable.

The records of consumption estimation used to validate the estimator module are also useful for modeling purposes. For example, if the power estimator is considered to be included into the plant itself, the analysis of how it responds to a change in the OPP enables the plant dynamics modelling. Thus, for example, Figure 4-5 shows the estimator output for the OPP changing from number 26 to number 27 during a certain video sequence decoding in the experimental test bench. Since the estimation period is long enough as to allow the estimator to detect completely the new consumption level due to the OPP switch from one sample to the next, as it can be seen at t=260s in Figure 4-5, the model of Equation 4-3 is also valid in terms of power estimation apart from power consumption.

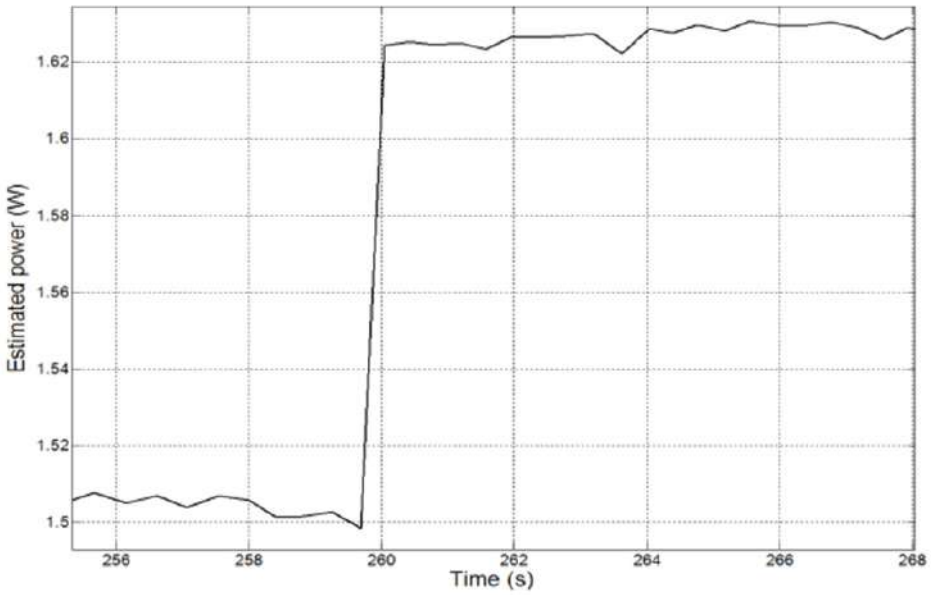


Figure 4-5 (Open-loop) estimated consumption for an OPP26 to OPP27 step

4.2.2 System transfer-function calculation

In the ideal linear closed-loop control system model, the action signal to the plant input comes from the controller output, as depicted in Figure 4-6. Hence, if the controller Z transfer function is called *F(z)*, then it can be expressed as *F(z)=A(z)/E(z)*, where *E(z)*, i.e., the controller input, models the input-output error of the system (see Figure. 4-6).

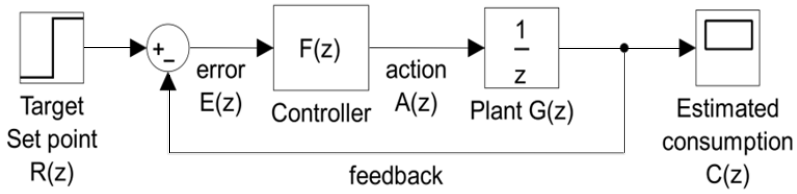


Figure 4-6 Conceptual and mathematical block diagram of the system model.

Because the output of the system model is the plant output, i.e., the power consumption estimation, modelled with *C(z)* in equation 4-3, then the feedback transfer function is unitary; therefore, the Z transform of the error sequence can be calculated as *E(z)=R(z)-C(z),* where the system input, modelled with *R(z)*, is the desired power consumption, i.e., the set point. All these relationships can be better understood with the block diagram of the system model shown in Figure 4-6. Taking into account the previous definitions, the Z transfer function of the whole closed-loop system, *M(z)*, can be calculated as indicated in equation 4-4.

$$M(z)=\frac{C(z)}{R(z)}=\frac{F(z)\cdot G(z)}{1+F(z)\cdot G(z)} \qquad 4\text{-}4$$

Once a mathematical model of the system is obtained, different techniques can be applied to design controllers and to foresee the corresponding system behaviour, as explained below. Further simulation and implementation details are included in Chapter 6.

4.2.3 Controller design

Apart from other technological issues, the stability of the closed-loop system is one of the characteristics that must be ensured. As a first approach, the simplest controller that can be used in closed loop is a proportional (P) one [2]. It would lead to a system transfer function $M_P(z)=K_P/(z+K_P)$ from 4-3 and 4-4 and where K_P is the P-controller gain. With the system pole in $p_{MP}=-K_P$, the (highest positive) critical gain in the limit of the system stability is $K_{Pc}=1$.

Because the closed-loop steady-state error, e_{ss}, is calculated with equation 4-5 as the percentage of a stepped set point:

$$e_{ss} = \frac{100}{1+F(1)\cdot G(1)} \qquad 4\text{-}5$$

For the P controller we have $e_{ssP}=100/(1+K_P)$. Therefore, the lower bound for the closed-loop system error in steady state is $min(e_{ss})=100/(1+max(K_p))=50\%$ [1], which is too high. In order to avoid this limitation and still keeping a classic linear controller, an integral action (I) should be added to it. As stated above for the plant model, if the results finally prove the validity of these simple controllers, they will have the additional advantage of its simplicity, which is a desirable feature for an algorithm that has to be implemented in an embedded system with limited resources.

Three types of I controllers are proposed, based on a forward rectangular rule (FRR), a backward rectangular rule (BRR) and a trapezoid or Tustin's bilinear rule (TR), respectively. Their corresponding Z transfer functions, $F_F(z)$ for the FRR-I, $F_B(z)$ for the BRR-I and $F_T(z)$ for the TR-I one, are shown in equation 4-6:

$$F_F(z) = \frac{K_F \cdot T}{z-1}; F_B(z) = \frac{K_B \cdot T \cdot z}{z-1}; F_T(z) = \frac{K_T \cdot T \cdot (z+1)}{2(z-1)} \qquad 4\text{-}6$$

where K_F, K_B and K_T are the respective integral gains and T is the system sample period, as previously defined. In all cases, the controller pole in $z=1$ ensures a null value of e_{ss} in equation 4-5.

To calculate $M(z)$ for the I cases, i.e., $M_F(z)$, $M_B(z)$ and $M_T(z)$, equation 4-3 and 4-6 should be applied to equation 4-4, resulting in the three cases of equation 4-7, with the closed-loop system poles indicated in equation 4-8, i.e., p_{MF}, p_{MB} and p_{MT}.

$$M_F(z) = \frac{K_F \cdot T}{z^2 - z + K_F \cdot T};\ M_B(z) = \frac{K_B \cdot T}{z - 1 + K_B \cdot T}$$
$$M_T(z) = \frac{K_T \cdot T/2 \cdot (z+1)}{z^2 + (K_T \cdot T/2 - 1)z + K_T \cdot T/2} \qquad 4\text{-}7$$

$$p_{MF} = \frac{1 \pm \sqrt{1 - 4 \cdot K_F \cdot T}}{2};\ p_{MB} = 1 - K_B \cdot T$$
$$p_{MT} = \frac{2 - K_T \cdot T \pm \sqrt{(K_T \cdot T)^2 - 12 K_T \cdot T + 4}}{4} \qquad 4\text{-}8$$

The difference of Z transfer functions between FRR and BRR is only a zero in z=0. The zero-pole cancellation in a series of a BRR I and G(z) will enable shorter settling times than with a FRR I when closing the control loop because the system dominant pole can be closer to z=0. This can be deduced from the Z-plane root loci shown in Figure 4-7. In the FRR case, the modulus of the system dominant pole is always greater than or equal to 0.5, whereas in the BRR case the system dominant pole can reach the minimum value of 0 thus enabling settling times shorter than the sample period. Thus, considering the BRR option, the closed-loop pole of the system for a long enough sample period is p_{MB} =1- K_BT, being again K_B the controller gain. Let us consider a sample period T of 100ms, which seems to be a good trade-off value for keeping reasonable relative overhead, immunity to jitter effects and frequency of control actions. For this period, the critical gain which leads the system to instability (p_{MB} =-1) is K_B =20, whereas the gain for the shortest settling time (p_{MB} =0) is K_B =10.

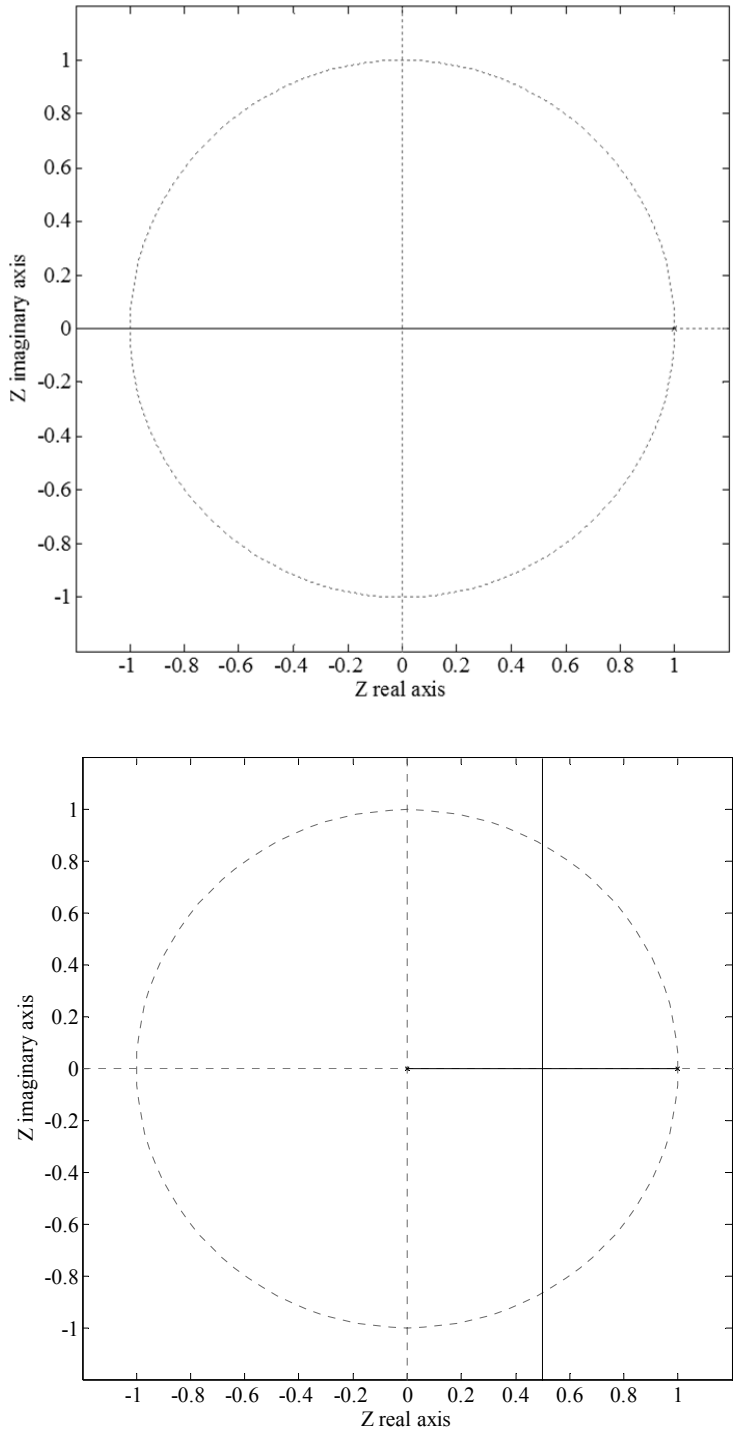


Figure 4-7 System root locus with BRR-I (up) and FRR-I (down) controllers

The root loci of TR I controller is shown in Figure 4-8, with two real/complex branches which, starting from within the unit circle, allow the system stability.

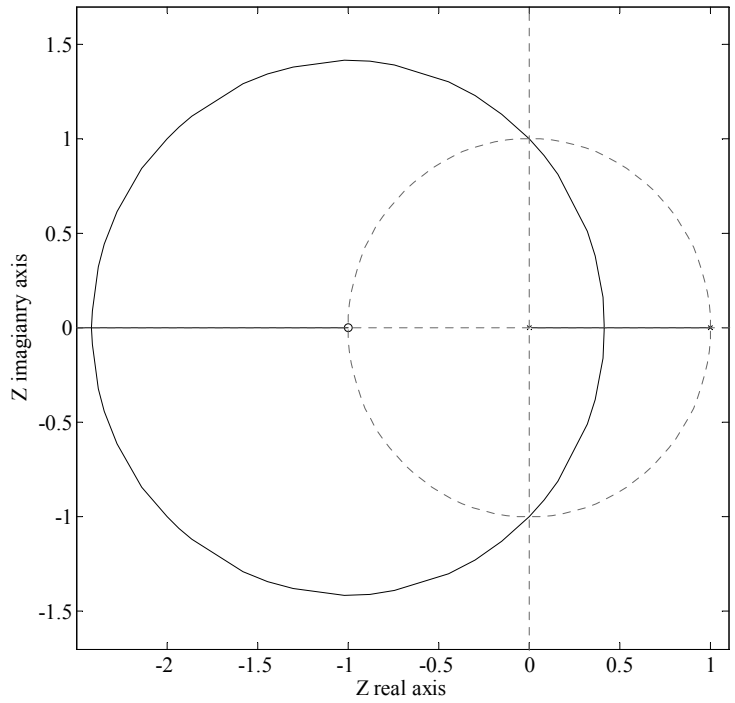


Figure 4-8 System root locus with TR-I controller

Some combinations of the previous linear controllers can also be used. Examples like proportional-integral (PI) or even proportional-integral-derivative (PID) are well known in the control domain. Their transfer functions, $F_{PI}(z)$ and $F_{PID}(z)$, are shown in 4-9, where K_{PI} and K_{PID} are the respective gains, and c, c_1 and c_2 are zeros of the transfer functions. The inclusion of the integral action in both cases keeps a controller pole in z=1, which still ensures a null value of e_{ss} in 4-5.

$$F_{PI}(z)=\frac{K_{PI}(z-c)}{z-1};\ F_{PID}(z)=\frac{K_{PID}(z-c_1)\cdot(z-c_2)}{z(z-1)} \qquad 4\text{-}9$$

As it can be deduced from 4-9, once the period T is fixed, the PI combination offers two degrees of freedom, i.e., K_{PI} and c, whereas the PID combination offers three degrees of freedom, i.e., K_{PID}, c_1 and c_2. In order to test one case of each of these two combinations, the values of the

zeros are fixed to achieve a system root locus that, at least, allows the closed-loop system stability. Thus, for the PI combination, if a value of c=0 were chosen, then $F_{PI}(z)$ would be analogous to $F_B(z)$, as it can be deduced from 4-6 and 4-9. In order to have a different behaviour, a value of c=0.5 has been chosen, which leads to a root locus like the one shown in Figure 4-9, with two real branches, one positive and finite and another negative and infinite. The coexistence of both branches within the unit circle allows the system stability.

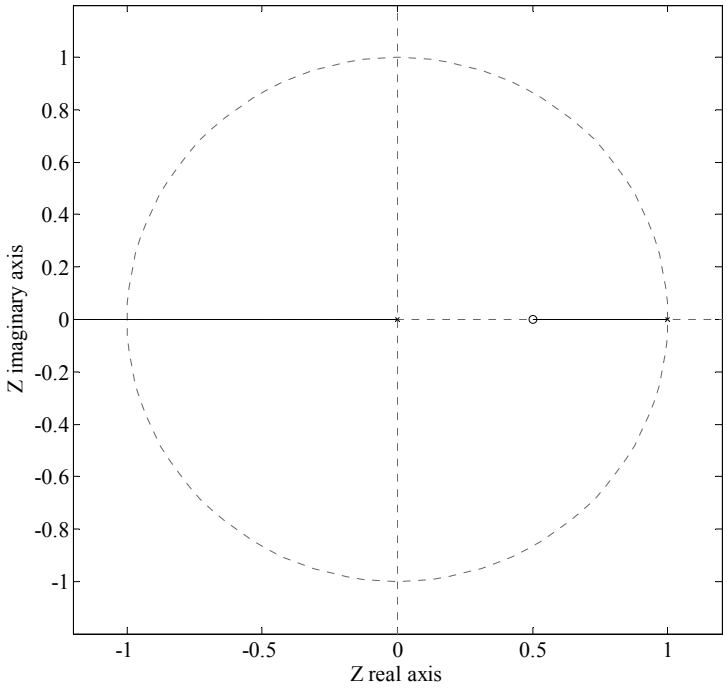


Figure 4-9 System root locus with PI controller and c=0.5

For the PID combination, a common value of c1=c2=-1 leads to a system root locus like the one shown in Figure 4-10, with two real/complex branches and another real, negative and finite. Again, the coexistence of the three branches within the unit circle allows the system stability.

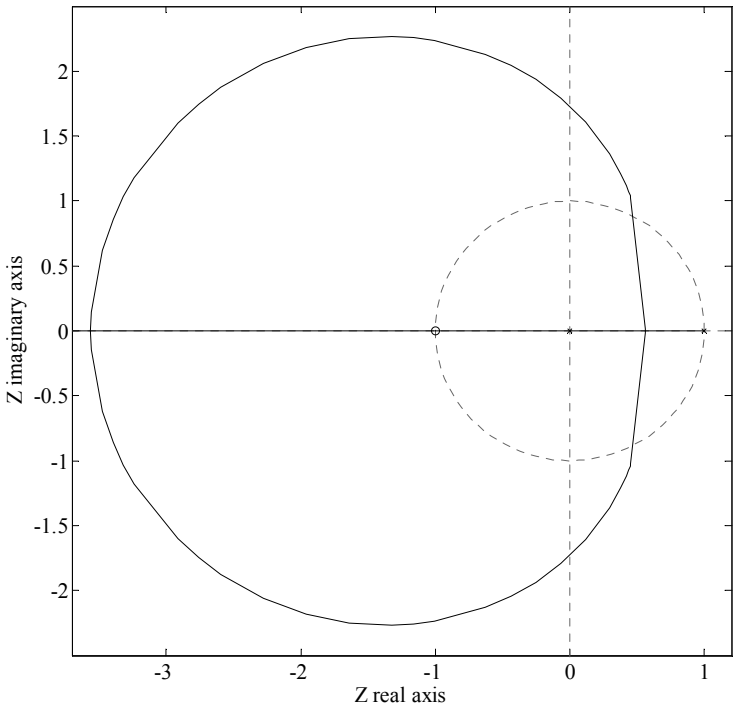


Figure 4-10 System root locus with PID controller and c_1=c_2=-1

Using the chosen values for the zeros and applying 4-3 and 4-9 to 4-4, the closed-loop transfer function *M(z)* for both combinations appears in the two cases of 4-10, i.e., $M_{PI}(z)$ and $M_{PID}(z)$. The poles of these two transfer functions are represented in Figure 4-11 and Figure 4-12 as functions of K_{PI} and K_{PID}, respectively.

$$M_{PI}(z) = \frac{K_{PI} \cdot (z - 0.5)}{z^2 + (K_{PI} - 1)z - 0.5K_{PI}}$$
$$M_{PID}(z) = \frac{K_{PID} \cdot (z + 1)^2}{z^3 + (K_{PID} - 1)z^2 + 2K_{PID}z + K_{PID}} \qquad 4\text{-}10$$

4.3 PCG

In order to guarantee the battery lifetime of mobile devices under a predefined performance requirement, a dynamic PCG is creatively added in the control system in order to adjust a suitable set-point of the closed-loop control subsystem. The PCG consists of a battery discharge estimator and a budget calculator, which depends on the current remaining battery. Different

from the Linux original governors, PCG bases its work on the estimation power consumption instead of the CPU usage. Another interesting innovation is that the PCG supports multiple and personalized power budget profiles to meet user requirements.

4.3.1 Battery discharge estimator

For those cases in which the battery SoC cannot be measured directly or indirectly, this work proposes a SoC estimator based on the value of power-consumption estimation (Power) of the closed-loop subsystem (see Figure 4-1). Current SoC can be estimated in units of A·s at every sample instant from previous-instant SoC (SoCpre), working voltage V and sample time T, as indicated in 4-11.

$$SoC = SoC_{pre} - \frac{Power}{V} \cdot T \qquad 4\text{-}11$$

The power consumption of a real system can be abstracted to a model in which the system power consumption is a piecewise constant function of the execution time. Their relationship is illustrated with the example of Figure 4-11 during different intervals. In the proposed system, the power consumption depends on the video-decoding task and the active OPP caused by the controller.

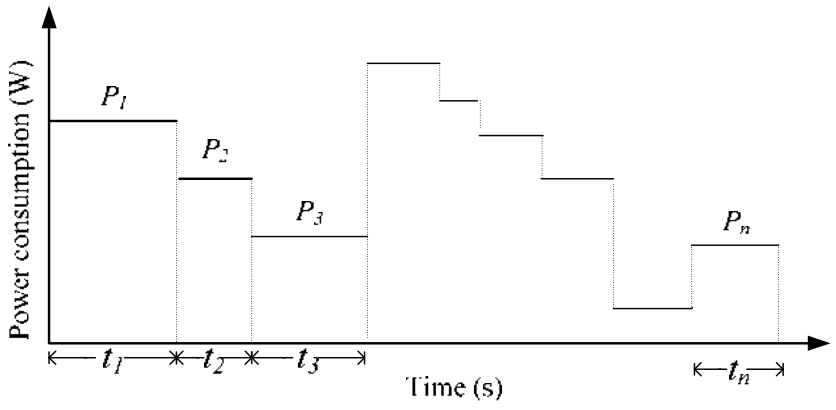


Figure 4-11 Example of the relationship between the system power consumption and execution time

From the power-consumption model illustrated in Figure 4-11, the energy E(n) removed from the battery between t_1 and t_n could be calculated as in 4-12.

$$E(n) = \sum_{i=1}^{n} P_i \cdot t_i \qquad 4\text{-}12$$

Where P_i denotes the system power consumption during interval i, t_i denotes time interval i, and n is the number of elapsed intervals.

4.3.2 Budget calculator

Corresponding to the value of SoC, the power-budget generator calculates a suitable power budget in order to extend the battery lifetime while keeping a reasonable QoE. A power-budget profile can describe the relationship between SoC and power budget, i.e., the set-point for the closed-loop subsystem. Although even a quasi-continuous function could be used to calculate the power budget from the SoC, given the quantized nature of the DVFS-based plant input of the closed-loop subsystem, it is more suitable to use a discrete function with no more steps than the number of available OPPs. Hence, Table 4-1 shows the general structure of the preset discrete power-budget profiler, which can be particularized for different user requirements. A number n of SoC thresholds, i.e., $SoC_{Th(i)}$ with i from 1 to n, can be defined such that the power budget is changed each time the SoC crosses a threshold. In order to have a good tradeoff between QoE and battery lifetime, the power budget should be decreased as the SoC crosses thresholds down, like proposed in Table 4-1. It means that, for example after a battery recharge, when the battery energy is higher than $SoC_{Th(n)}$, the power budget for the video decoder can be the highest one in order to support very good QoE. With the energy state changing, the set-point is dynamically changed following the power-budget profile. Thus, for example, when the SoC is in the interval [$SoC_{Th(1)}$, $SoC_{Th(2)}$), a suitable medium-low set-point should be selected. Finally, when the SoC is lower than $SoC_{Th(1)}$, the mobile devices should consume as less power as possible in order to guarantee battery lifetime. The threshold and set-point values can be adjusted by requirements of users. Therefore, depending on feedback power consumption, PCG can generate a power-budget profile which supports the set-point of the closed-loop subsystem.

Table 4-1 SoC and General Power-Budget Profiler

SoC	Energy state	Power budget
$SoC \geq SoC_{Th(n)}$	High energy	Set-point *n* (highest)
$SoC_{Th(n)} > SoC \geq SoC_{Th(n-1)}$	Medium-high energy	Set-point *n*-1
…	…	…
$SoC_{Th(2)} > SoC \geq SoC_{Th(1)}$	Medium-low energy	Set-point 1
$SoC < SoC_{Th(1)}$	Low energy	Set-point 0 (lowest)

The ondemand and conservative governors are dynamic Linux *cpufreq* governors which can change the OPPs following the different *cpufreq* policies. Ondemand and conservative can increase or decrease the frequency depending on the workload and some parameters can be set in order to decide when changing to another OPP. This is one of the methods to support power optimization. However, PCG can dynamically extend the battery lifetime depending on the user requirements and maintain it regardless of the complexity of the video sequences, whereas the Linux dynamic governors vary the battery lifetime depending on the workload, as it will be shown in Chapter 7.

4.4 Summary

In this chapter, a control system has been introduced, which includes a closed-loop control subsystem and a PCG. Firstly, before designing the system controller, a suitable model of the plant has been built. The model of the plant is fixed by analyzing the power consumption behavior in OPP switching situations. Once the plant model has been obtained, a system model is fixed so that classic controllers can be explored before having a system implementation. Proportional (P) and integral (I) controllers, as well as some classic combinations of linear controllers, are designed to verify the behavior of the closed-loop control subsystem in subsequent chapters. The feedback of the closed-loop control subsystem is based on an OS-level estimator which can accurately calculate the estimation power consumption in real time. The feedback information is subtracted to the set-point in order to adjust the output close to the desired power consumption by means of the controller. At the same time, the estimation value is also taken by the PCG to estimate the battery SoC and then generate the power budget to lead the set-point. On the other hand, the original Linux dynamic *cpufreq* governors adjust the OPPs

depending on the CPU usage, and they can be configured when to change the OPPs. Their *cpufreq* policies can save energy consumption without considering remaining battery. The proposed PCG can estimate the remaining battery based on the feedback power estimation and it supports multiple and personalized power budget profiles to meet user requirements.

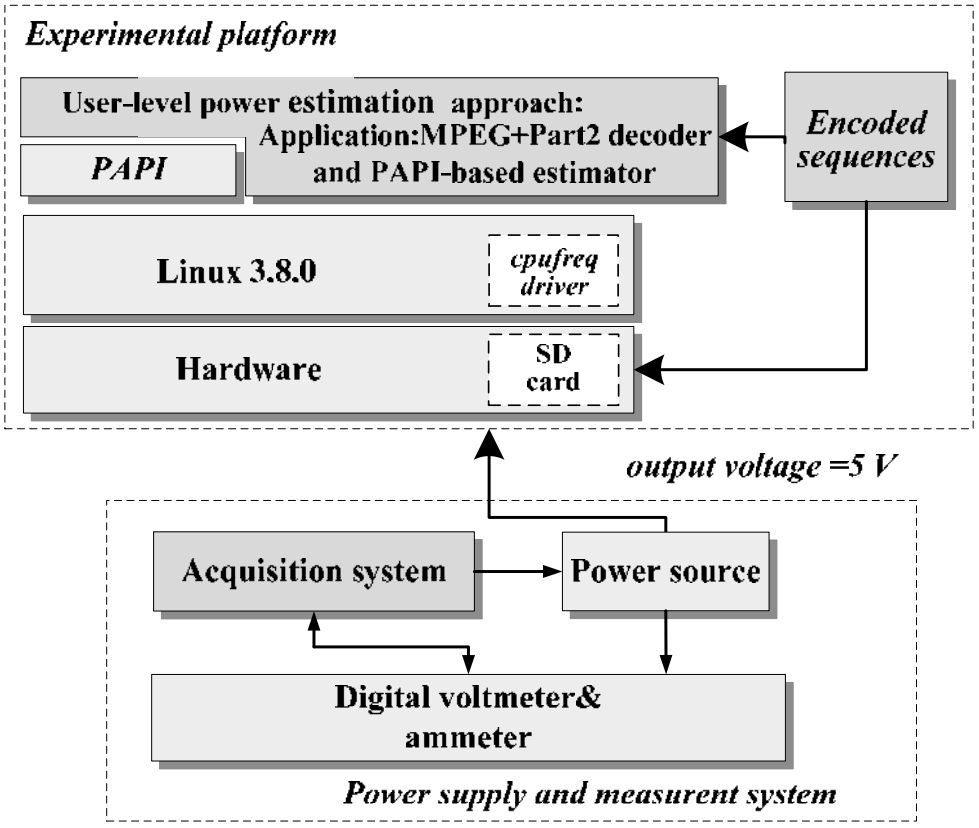
Chapter 5 Test bench

The system proposed in Chapters 3 and 4 is implemented and tested as described in next chapters. This chapter presents the test-bench design of the experiments. It starts with an overview of the test-bench architecture, which includes an experimental platform and a power supply and measurement system, as well as the experimental method. Then, it continues with an introduction to the experimental platform from both hardware and software environments, and to the power supply and measurement system. Besides, the tool used to access PMCs is presented. To integrate the tool interface into the decoder application, the decoder and its reconfigurable development environment are also described.

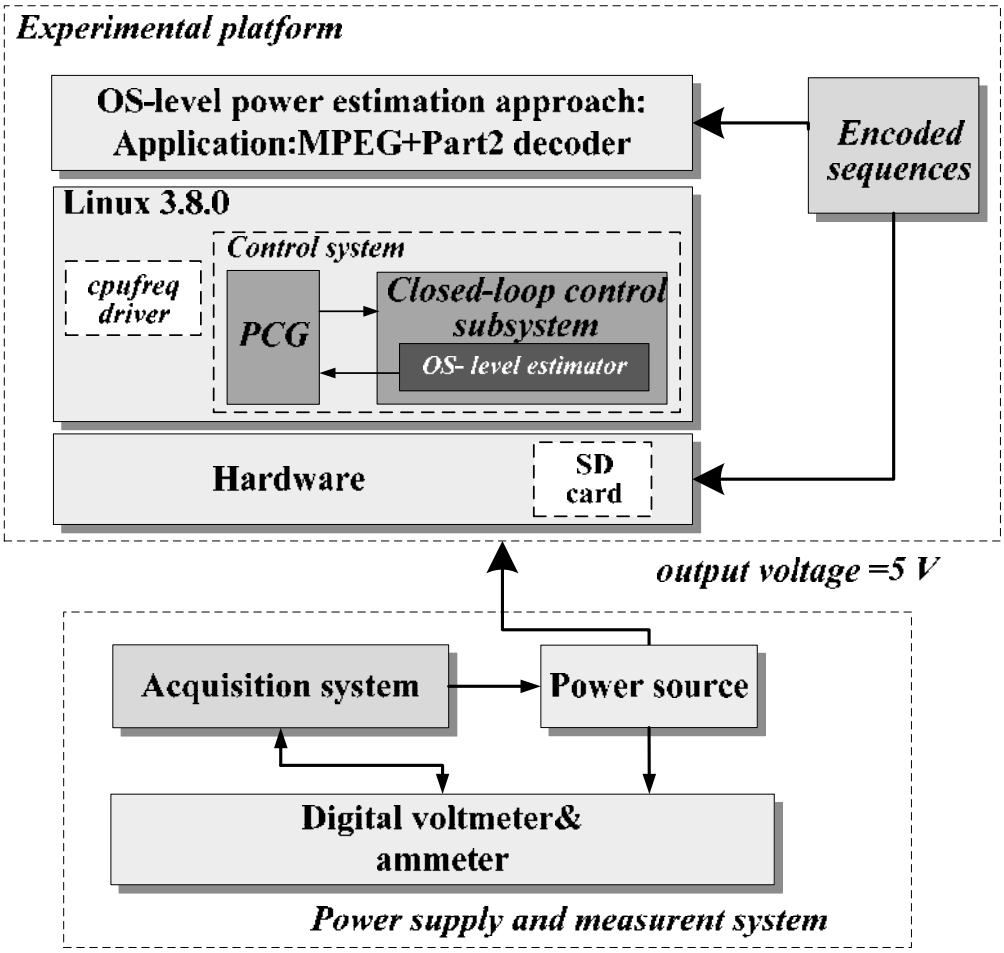
5.1 Test-bench and methodology overview

5.1.1 Test-bench architecture

The architecture of the test bench for the control system is shown in Fig. 5-1.



(a) The test bench of User-level power estiamtion approach



(b) The test bench of OS-level power estimation approach

Figure 5-1 Block diagram of the test bench

As it can be observed, the test bench consists of two parts, the experimental platform and the power supply and measurement system. The experimental platform includes a multimedia application, encoded sequences, a Linux-based operating system, an ARM-based hardware and a SD card. As explained in Chapter 3, there are two approaches of power estimation. Figure 5-1 (a) shows the experimental platform when using the user-level power estimation approach. If the PAPI-based power estimator is considered, the decoder application includes the PAPI interface to access the PMCs, as well as the MARS model, which calculates the consumption estimation based on PMCs. If the OS-level power estimator is used (Figure 5-1 (b)), the decoder application does not need any modifications. The PAPI-based power estimation approach is used only as the initial study of accessing PMCs to feedback power estimation, therefore, when the work applies

to the real-time control system, the OS-level power estimator is used for the feedback information instead of the PAPI-based power estimator. The detail of comparing the two estimators will be explained in Chapter 6. In addition, the control system includes a PCG and a closed-loop control subsystem where the OS-level power estimator has been included. The PCG can emulate the battery discharging and in simulation part it is employed to replace a real Lithium-ion battery. And different video sequences are employed to test the control system. In the following sections, an expanded description of the experimental platform through hardware and software is given.

The power supply and measurement system is composed of a power source, an acquisition system and a digital voltmeter & ammeter. The acquisition system can set the voltage in 5 V and control the sampling frequency of digital voltmeter & ammeter. The digital voltmeter & ammeter measures the voltage and current supplied by the power source to the experimental platform. Then, a data-acquisition system takes samples of the measured voltage and current and sends them to a PC-based software, through which the measuring sample time can be configured, as well as the output voltage level. The digital voltmeter & ammeter is used to profile the power consumption of the whole experimental platform. A detailed description of the power supply and measurement system is provided in Section 5.3.

5.1.2 Experimental methodology

The control system relies on the online real-time power estimation which feeds the power consumption of the application back. Therefore, the ideal way to address the control system is to build a complete power-centric system with the power accounting module properly implemented.

The experimental methodology is explained in Figure 5-2. Two different approaches have been considered to get power consumption estimations prior to the the final control-system implementation. The first approach (Figure 5-2①) is based on a user-level implementation through a third-party library, PAPI, which is used as the PMC driver and is included into the user-level decoding application to read PMC event counts after decoding every frame. Besides, a second alternative approach (Figure 5-2②) is added: in order to decouple the user application from the power control system, the power estimator is moved from the user level to the OS, which accesses directly and periodically the PMCs for carrying out the estimation task at kernel level. The OS-level estimator is used to feedback the power consumption estimation.

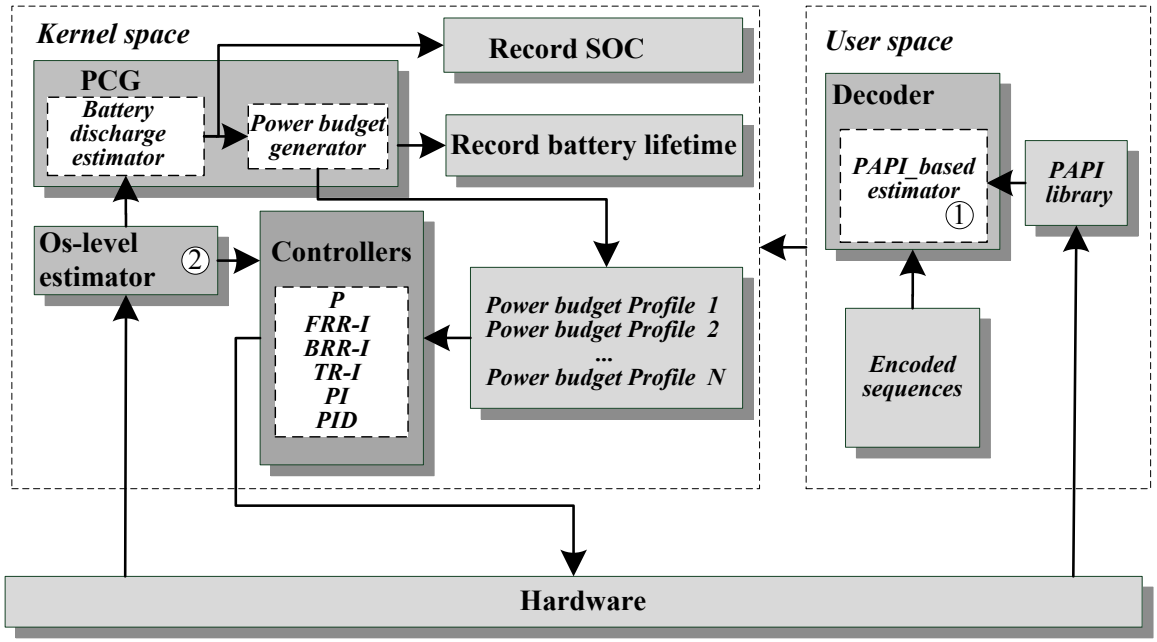


Figure 5-2 Overview of the Experimental Methodology

The methodology includes three parts. The first part focuses on the power estimation of decoder application while decoding different video sequences; the second part is to design and test different controllers within the closed-loop control subsystem; and the third part is to add the PCG, which will generate different power budget profiles in order to lead the set-point of the closed-loop control system.

In the first part, each sequence is individually decoded on the experimental platform; and in the meantime, the power consumption is profiled via the power measurement unit that periodically samples the voltage and current of the experimental platform. After repeating this process for those sequences, the power consumption of all sequences can be obtained. Besides, the PMCs of PuC when decoding every sequence are also recorded. Based on PMCs and power consumption, the generally applicable power estimation model was built at the end of the first part. In the second part, different classic controllers have been introduced to the closed-loop control subsystem, and it has been tested on the experimental platform to validate the results of the controllers. In the third phase, the PCG has been added to the system in order to generate suitable power budget profiles, which will generate the set-point for the closed-loop controllers.

5.2 Experimental platform

To implement and analyze control algorithms for energy optimization, a simple and low-cost single-core multimedia platform was chosen in a first approach. After knowing how the control system works in the single-core hardware, the control algorithms can be enhanced to be tested in multi-core platforms. Besides, in order to focus on energy optimization of the single-core devices, less peripheral circuits and functions will be able to interfere into the energy consumption. What's more, the hardware should support the DVFS and PMC mechanisms. Hence, the hardware for the experiments is a commercial board named BeagleBoard [66], which runs the Ubuntu Linux operating system with the control system implemented in the kernel. In the remaining of this section, the experimental platform will be introduced from two aspects: the hardware and the software environments.

5.2.1 The hardware environment

As introduced above, the test bench is based on a single-core hardware development platform for multimedia embedded systems: BeagleBoard. Its features are listed in Table 5-1.

Table 5-1 Features of BeagleBoard

Component	Features	
Processors	One ARM Cortex-A8 core Digital Signal Processor (DSP) Image and Video accelerator Image Signal Processor (ISP) 2D/3D graphic accelerator	
Memory	512 MB LPDDR RAM ,SD/MMC Card Cage	
Indicators	Power	2-User Controllable
	PMU	
Connector	Video	Audio
	DVI-D S-Video Connector LCD Expansion Connector	L+R Stereo out, L+R Stereo In
	10/100 Ethernet	
USB Port	USB 2.0 OTG Port, USB Host Ports	
Expansion	General Purpose Expansion (I2C, USB, MMC,DSS...) Camera Expansion Connector	
Debug	14 Pin JTAG, UART/RS-232 Port, GPIO Pins	
User Interface	Switches, Reset Button	

A general description of the BeagleBoard architecture is given in Figure 5-3. As shown in Figure 5-3, the BeagleBoard features an OMAP 3530 system-on-chip, which includes a 720 MHz ARM Cortex-A8 CPU for general purpose computation and a TMS320C64x+ DSP for accelerated multimedia applications. Built-in storage and memory is provided for the OMAP 3530 through a Package-On-Package (POP) chip that includes 256MB of NAND flash and 256MB of SDRAM. Additional memory can be added to the BeagleBoard by installing a SD or MMC card in the SD/MMC slot, or driving a USB thumb drive or hard drive through the USB OTG port and the EHCI USB port. The TPS65950 is a power management chip (PMIC) that provides different power domains and clock frequencies to the BeagleBoard, its 5V power source can come from the USB OTG port connected to a PC powered USB HUB, or a 5V DC supply. Besides, TPS65950 also provides stereo audio in and out. The video output of the BeagleBoard is provided through a separate S-Video connector and a DVI-D connector that can partially support High-Definition Multimedia Interface (HDMI). In addition, BeagleBoard provides a RS-232 serial connector, a Join Test Action Group (JTAG) connector, and an expansion connector.

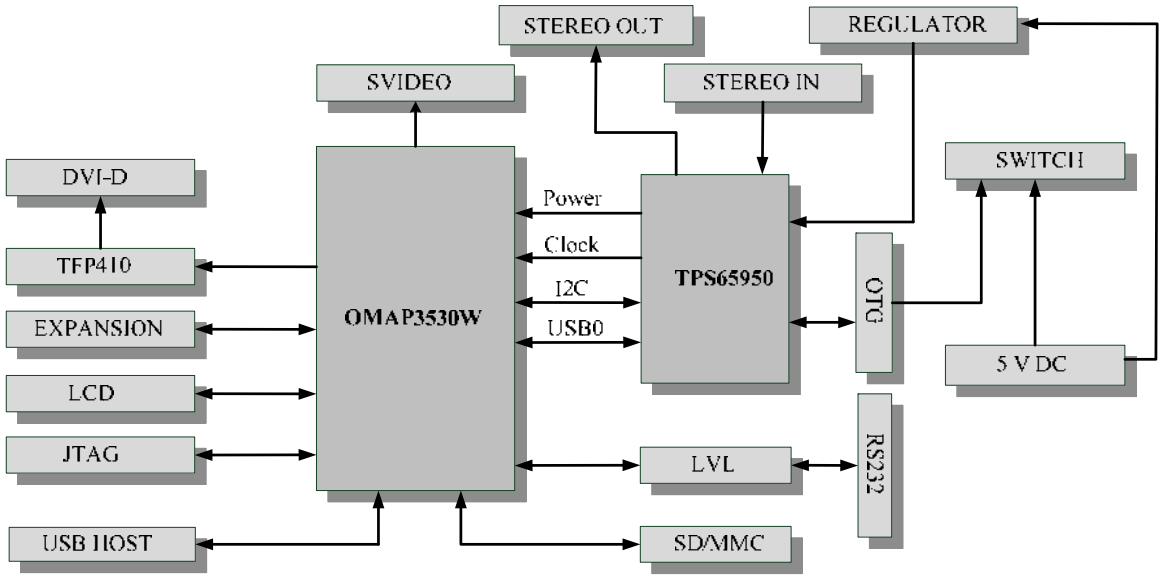


Figure 5-3 Block Diagram of BeagleBoard

The functions of the BeagleBoard can be divided into four categories: computation, storage, I/O, and communication. Note that the communication unit and the I/O unit are combined together due to the lack of specific physical interface for network communication. But

there are several USB to network adapters on the market that can add Ethernet, Wi-Fi or Bluetooth connectivity to the BeagleBoard by using the EHCI USB port or the USB OTG port in the host mode.

In this thesis work, not all functions and devices of the BeagleBoard are employed for the experiments. To simplify the work and focus on the energy consumptions caused by the ARM Cortex-A8 CPU, the memory subsystem, and the related I/O buses, the BeagleBoard has been configured as a minimal system that disables the unnecessary components such as the display and network subsystems.

5.2.2 The software environment

The Ubuntu Linux is a special Linux distribution that is tailored for embedded systems and shipped with the BeagleBoard. A full package of the Ubuntu distribution images includes an X-loader (MLO), a U-boot (u-boot.bin), a Linux kernel image (uImage), and a Linux root filesystem. To boot the Ubuntu Linux on the BeagleBoard from SD card, the SD card has been formatted into two partitions, with the X-loader, U-boot and uImage held in the first partition and the Linux root filesystem held in the second partition. The procedure of Linux booting is as follows: when the BeagleBorad is powered on, the ROM program loads and executes the X-loader, which further loads the U-boot and executes it; the U-boot reads its commands and loads the Linux OS kernel image with the U-boot commands as arguments; once the kernel image is fully loaded to the memory, it is uncompressed and begins the initialization procedure; at certain point of the kernel initialization, the kernel mounts the root filesystem partition based on the U-boot commands; after the Linux OS is fully booted, a login interface appears and the system is ready for use.

In this work, a Linux 3.8.0 kernel, patched to support the platform DVFS mechanism, is running in the processor. The DVFS subsystem is managed through the *cpufreq* Linux driver. As already addressed in the previous chapter, this driver includes four predefined governors to fix the MPU OPP, two static and two dynamic, which react to the system load. This is achieved by a function called *cpufreq_driver_target*, one of whose input parameters is the target frequency of the desired OPP to switch to. This function searches the target frequency among the ones of the OPPs defined in an internal table and selects the appropriate one by applying a ceil- or a floor-rounding algorithm, depending on another input parameter. The function then sets the frequency

and the voltage corresponding to the selected OPP. The default *cpufreq* definitions for the BeagleBoard only consider 6 OPPs. In order to decrease this strong nonlinearity in the DVFS-based plant input, additional valid OPPs were searched. The BeagleBoard supports voltage scaling from 0.6V to 1.45 V with a step of 12.5 mV. After verified all possible pairs of frequency and voltage, there are 27 valid OPPs. The characteristics of these 27 OPPs are shown in Table 5-2. The first column is the OPP number, whereas the frequency and voltage of each OPP are indicated in the second and third column, respectively. Besides, running the decoder application to decode the 78 sequences under each OPPs while measureing the power consumption of the board. The measured average current consumption for every OPP is listed in the fourth column.

Table 5-2 OPP data

No.	MHz	V	A	No.	MHz	V	A
1	125	0.981	0.186	15	430	1.156	0.245
2	200	0.933	0.197	16	500	1.168	0.259
3	210	1.006	0.199	17	510	1.181	0.263
4	220	1.018	0.201	18	520	1.193	0.266
5	240	1.031	0.205	19	530	1.206	0.270
6	250	1.043	0.207	20	540	1.218	0.274
7	270	1.056	0.211	21	550	1.230	0.278
8	290	1.068	0.215	22	560	1.243	0.282
9	310	1.081	0.219	23	570	1.256	0.288
10	330	1.093	0.224	24	580	1.280	0.290
11	350	1.106	0.228	25	590	1.293	0.297
12	370	1.118	0.233	26	600	1.306	0.301
13	390	1.131	0.236	27	720	1.306	0.327
14	410	1.143	0.241				

5.2.3 Cpufreq governors

As a special part of the Linux kernel, the DVFS mechanism, which acts on the plant, is managed through the *cpufreq* Linux driver. This driver includes four predefined governors to fix the MPU OPP, two static and two dynamic, which react to the system load. In the initial version of Linux kernel, the *cpufreq* driver only offered the CPU to be set to a fixed frequency. Only the two static governors “powersave” and “performance” were patched in, which set the frequency statically to the lowest or highest frequency, respectively. Since kernel 2.6.9, “ondemand” and “conservative” governors have been added, such that they can dynamically scale the CPU frequency up or down in order to save power. The procedure of *cpufreq* scaling is shown in

Figure 5-4 [51]. In order to achieve dynamic frequency scaling, the *cpufreq* core should be able to tell the *cpufreq* driver a “target frequency”. So the *cpufreq* driver was transformed by *cpufreq* core to offer a "->target/target_index" call instead of the existing "->set policy" call. Through *cpufreq* governors the frequency within the *cpufreq* policy can be decided.

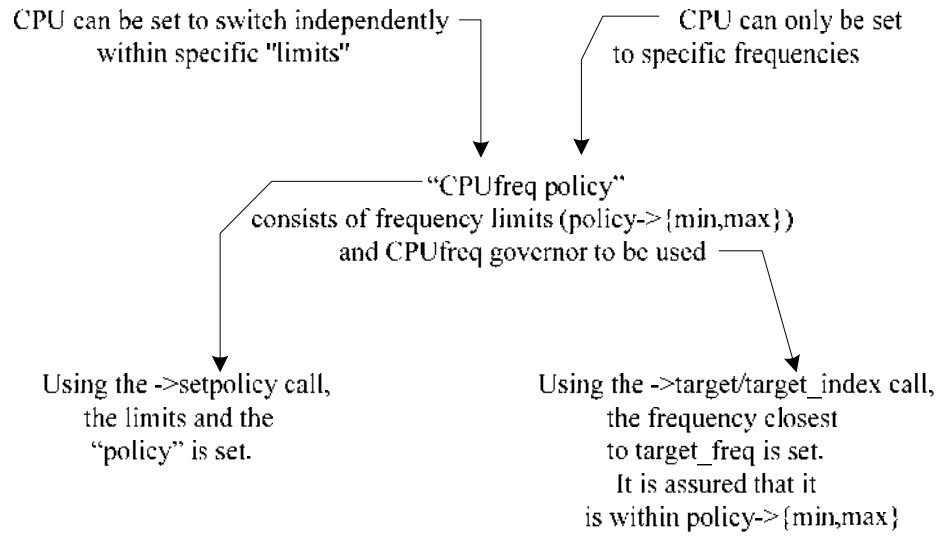


Figure 5-4 Procedure of *cpufreq* scaling [51]

5.2.3.1 Ondemand governor

The *cpufreq* ondemand governor sets the CPU depending on the current usage and CPU must have the capability to switch the frequency very quickly. Below are the main accessible parameters of ondemand governor:

sampling_rate: The sampling rate is measured in microseconds, which indicates how often the kernel monitors the CPU usage and makes decisions on what to do. Sampling rate is adjusted by considering the frequency transition latency, which default value is transition_latency * 1000.

Sampling_rate_min: The sampling rate is limited by the hardware transition latency and its default value is transition_latency * 100.

up_threshold: The parameter defines the threshold value, which means when the average CPU usage is higher than the defined value during the sample time, the frequency will be increased. For example, when the up_threshold is set as 95, this means that within a checking interval, if the average usage of CPU is larger than 95%, the CPU frequency will be increased.

ignore_nice_load: This parameter takes 0 or 1. When it is set as the default value 0, all processes are counted towards the CPU utilisation value. Otherwise, when it is set as 1, the processes that are executed with a *nice* value will not be counted in the overall usage.

sampling_down_factor: This parameter controls the rate at which the kernel makes a decision on when to decrease the frequency while running at top speed. When it is set as the default value 1 decisions to revaluate load are made during the same interval regardless the current clock speed. When it is set greater than 1, it acts as a multiplier for the scheduling interval for revaluating load when the CPU is at its top speed. This improves performance by reducing the overhead of load evaluation and helps the CPU to stay at its top speed when the workload is heavy, rather than shifting back and forth in speed. This parameter has no effect on behavior at lower speeds/lower CPU loads.

powersave_bias: This parameter takes a value between 0 and 100 which defines the percentage (times 10) value of the target frequency that will be saved off of the target. For example, when it set to 100, instead of setting the target frequency to 1000Mhz, ondemand governor will target 1000MHz-(10%* 1000MHz)=900MHz. This is set to 0 (disabled) by default.

5.2.3.2 Conservative governor

The *cpufreq* conservative governor works similar to ondemand governor, which sets the CPU frequency depending on the current usage. However, the behaviour of conservative governor increases and decreases more gracefully the CPU speed rather than jumping to the highest speed while the workload is variable. Below are the main accessible parameters of conservative governor:

freq_step: This parameter describes what percentage steps the *cpufreq* should be smoothly increased and decreased. The default value is 5 which means CPU frequency will increase in 5% chunks of the maximum CPU frequency. The value of freq_step can be set between 0 and 100, in which 0 will effectively lock the CPU at a fixed speed regardless of its load, while 100 means the conservative governor will work the same as ondemand governor.

down_threshold: The parameter defines the threshold value, which means when the average CPU usage is lower than the defined value during the sample time, the frequency will be decreased. For example, the default value is 20, which means if the CPU usage is less than 20% during every sample time, the frequency will be decreased.

sampling_down_factor: This parameter controls the rate at which the kernel makes a decision on when to decrease the frequency while running at any speed.

5.4 Power supply and measurement system

The power supply and measurement system is based on commercial equipment from Agilent, i.e., Agilent 66321D [67]. As it has been mentioned in the architecture overview of the test-bench, the power supply and measurement system consists of three functional modules: a power source, a digital voltmeter & ammeter and a PC-based acquisition system. The block diagram of the power supply and measurement system is shown in Figure 5-4.

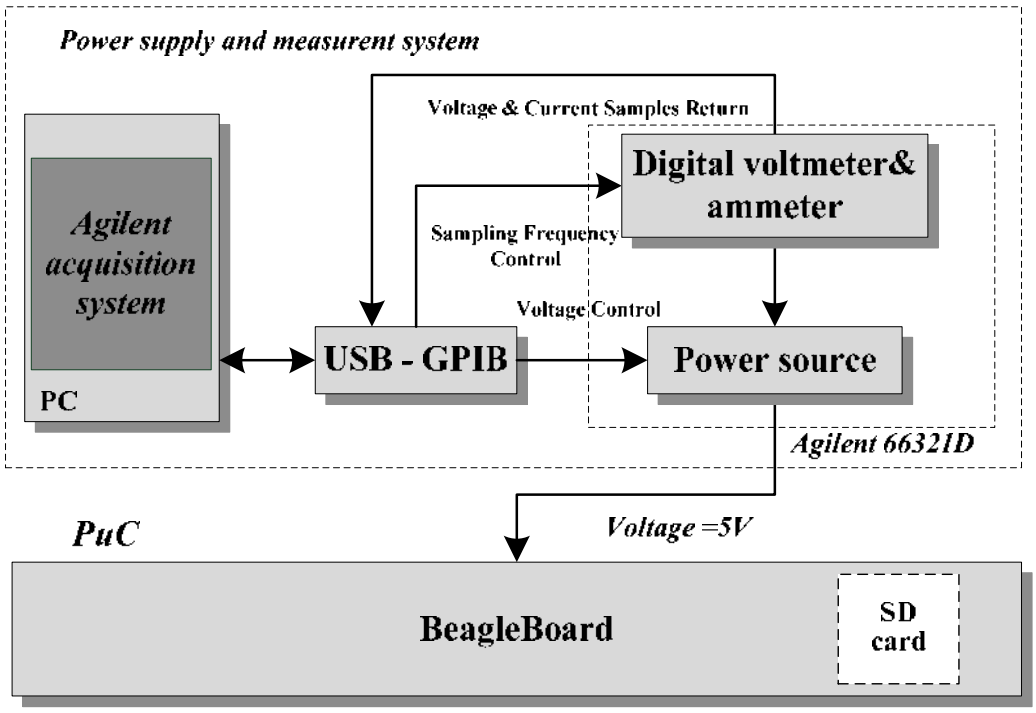


Figure 5-5 Block diagram of the power supply and measurement system

Agilent 66321D internally includes a digital voltmeter & ammeter and a power source. The power source supplies power to PuC while the digital voltmeter & ammeter measures the supplied current and voltage. Besides, the acquisition system controls the sampling frequency and the voltage output, which is fixed in a level of 5V to power on the BeagleBoard. To build the power measurement system, the PC and the Agilent 66321D are connected through a USB to General-Purpose Interface Bus (USB-GPIB). The voltage value chosen from the acquisition

system is programmed into the power supply via the USB-GPIB. In the meantime, digital voltmeter & ammeter measure the voltage and current values of the PuC. Again, through the USB-GPIO interface, the acquisition system can configure the measuring frequency of the digital voltmeter & ammeter and obtain the voltage and current samples from it.

The software user interface of the Agilent acquisition system is shown in Figure 5-5. On the left, the output voltage option is to set the voltage via the USB-GPIB interface (see in Figure 5-5①). The integration time option is used to set the sample time of digital ammeter, and in this thesis work, the sampling period is set as 5 milliseconds (see in Figure 5-5②). At the middle of the window, measured current is shown in real time and the historical data can be saved (see in Figure 5-5③).

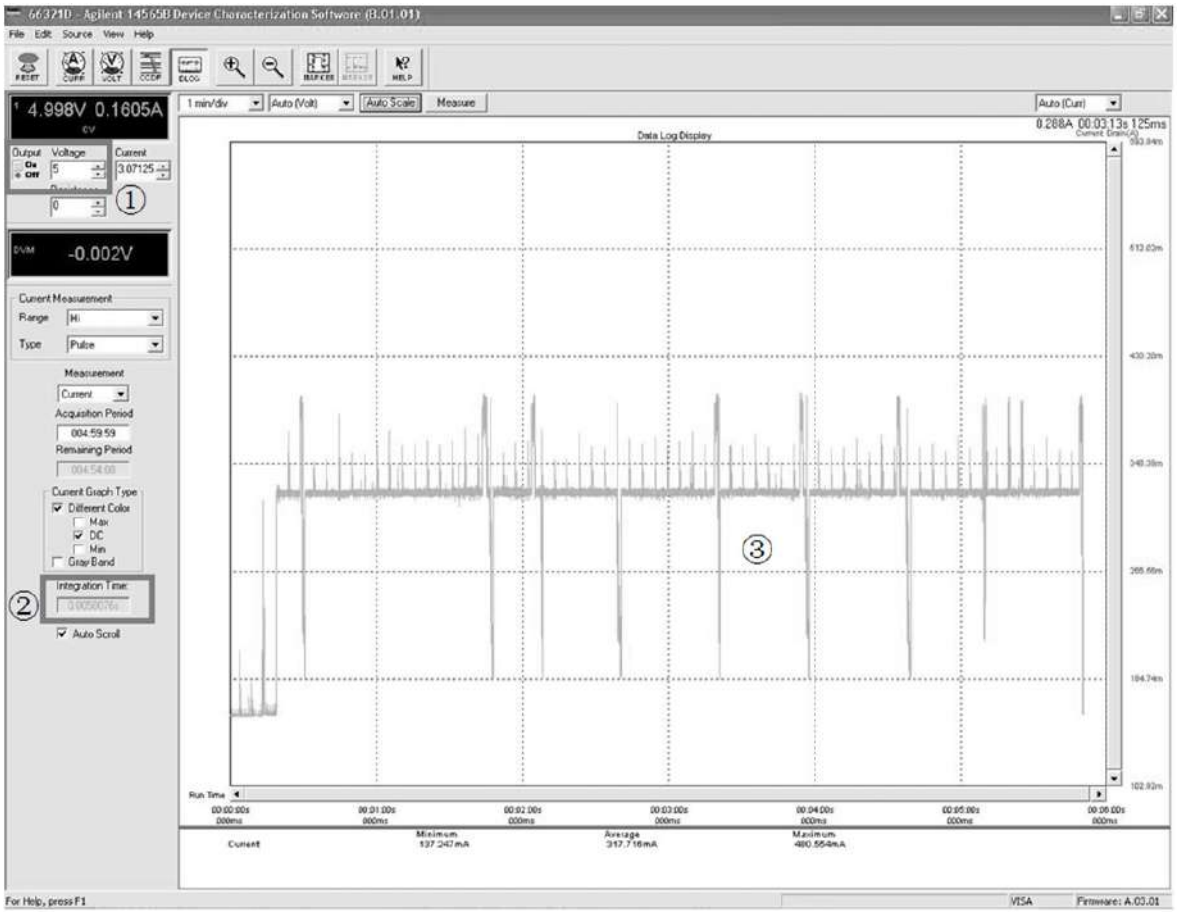


Figure 5-6 Software user interface of Agilent acquisition system

5.5 PMC Programming Tool

As section 3.2.1 has introduced, the tool used to access PMC in this thesis work is PAPI. PAPI can easily and directly access PMCs from the application level. Besides, the interface of PAPI is the same for all platforms so that it is widely used. PMCs. Based on Ren *et al*. [14], PAPI is used as the initial approach to access PMCs. As it can be seen in Figure 5-6, PAPI can be divided into two layers of software.

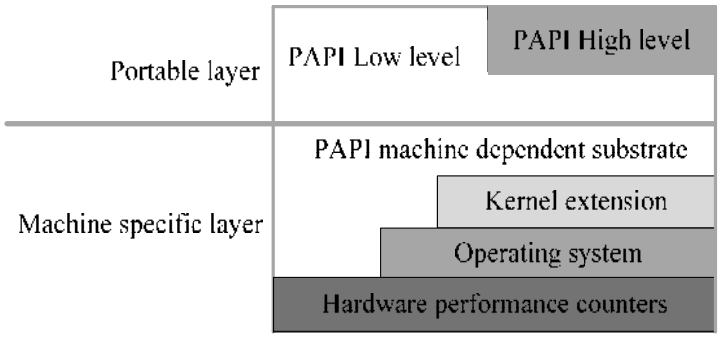


Figure 5-7 PAPI structure

The upper layer is a portable layer that consists of the API in low level and high level, as well as machine independent support functions. High-level interface is used for collecting simple measurements and it simply provides the ability to start, stop and read specific events, one at a time. Low-level interface is directed towards users with more sophisticated needs, it deals with hardware events in groups called EventSets. EventSets reflect how the counters are most frequently used, such as taking simultaneous measurements of different hardware events and relating them to one another. For example, relating cycles to memory references or flops to level 1 cache miss can indicate poor locality and memory management. In addition, EventSets allow a highly efficient implementation, which translates to more detailed and accurate measurements. EventSets have features such as guaranteed thread safety, writing of counter values, multiplexing and notification on threshold crossing, as well as processor specific features. Both a high-level and a low-level interface and are implemented on a number of Linux platforms and the latest release now provides support for ARM Cortex A8, which is the platform used for this work. The lower layer is a machine specific layer that defines and exports a machine independent interface to machine dependent functions and data structures. These functions access the substrate, which

may consist of the operating system, a kernel extension or assembly functions to directly access the processor registers.

PAPI only monitors the hardware events that are occurrences of specific signals related to a processor's function, such as cache misses and floating point operations while the program executes on the processor. Each processor has a number of events that are native to that architecture. PAPI provides a software abstraction of these architecture-dependent native events into a collection of preset events that are accessible through the PAPI interface. Preset events are a common set of events deemed relevant and useful for application performance tuning. They 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 approximately 100 preset events for CPUs, but some of them may be unavailable on certain platforms. For a given platform, some preset events can be counted through both the high- and low-level interfaces of the portable layer. PAPI provides access to native events on all supported platforms through the low-level interface. Even if there is no preset event available, native events can still be accessed directly. Table 5-3 lists the native events which are directly accessed from a single event in Cortex A8 processor, the one employed in this work. Derived events that use more than one event at the same time could intensify the limitation of the simultaneous PMCs number. In order to avoid too much overhead, only native events are used in this dissertation.

Table 5-3 Common Preset Events of Cortex A8 processor

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
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

5.6 Decoder application

5.6.1 MPEG-4 part2

Due to the hardware limitations of BeagleBoard, too complex video decoder standards cannot be executed on it, therefore MPEG-4 part2 standard [68]-[70] is employed. All the test video 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.

MPEG-4 is a video-coding standard designed for rich multimedia. It provides various codec tools with good compression capability. MPEG-4 uses a number of technologies such as shape encoding and adaptive discrete cosine transform (DCT) to improve the coding efficiency. MPEG-4 Part2 is a DCT based standard defined to provide high compression efficiency with some compression tools such as combination of motion-compensated prediction and scalar-quantized DCT coefficient coding [68]. Video applications are ranged from low-quality and low-resolution requirements to high definition preference; thus, video standards are structured in profiles 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 the video decoder source code. SP is designed for applications that are constrained by low bit rate and low-resolution conditions.

5.6.2 Decoder development environment

Since the PAPI interface needs to be included into the decoder, it should be reconfigured. Table 5-4 lists the tools and libraries used for building the decoder development environment.

Table 5-4 Tools and packages used for building the decoder application

Tools and Libraries	Functionalities
ORCC	A plugin for Programming languages translation
SDL	An open source library to facilitate multimedia implementation

Cmake	An advanced platform-crossed compilation tool for source code management and compilation
Eclipse	An integrated development environment
Java-JRE and Java-JDK	Support for Java running environment and development environment

Orcc is an open-source Integrated Development Environment based on Eclipse and dedicated to dataflow programming. The primary purpose of Orcc is to provide developers with a compiler infrastructure to allow software/hardware code to be generated from dataflow descriptions. Orcc does not generate assembly or executable code directly; rather it generates source code that must be compiled by another tool.

SDL is a simple open source cross-platform library designed to provide a common abstract layer to hardware components via OpenGL and Direct3D [71]. SDL officially supports Windows, Mac OS X, Linux, iOS, and Android. SDL is designed in C language and provides several low level controls on images, audio, and I/O peripherals and currently is widely used for developing games, simulators, media player, and other multimedia applications.

CMake is an open-source, cross-platform family of tools designed to build, test and package software. It is also used to control the software compilation process using simple platform and compiler independent configuration files, and generate native makefiles and workspaces that can be used in the compiler environment. 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. Instead of directly building the final executable file, it can generate the standard build files, 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 [72]. The main goal to use Cmake in this thesis is to compile and install the decoder in the target environment. What's more, Cmake is more like a tool to facilitate source code management and compilation rather than a compiler. Cmake is OS-dependent and the calling of a real compiler is embedded into the configuration file of Cmake. For Linux-based platforms, GCC-based method is the most widely used tool to obtain the executable files.

Eclipse IDE is an open source community of tools, projects and collaborative working groups. ORCC is implemented in Java as an Eclipse plugin. In this work, depending on user needs, Eclipse IDE packages either 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 [73].

5.7 Summary

This chapter presents the experimental test bench and experimental methodology that is employed to implement and verify the proposed control system. The experimental test bench is composed of two parts: the experimental platform and the power supply and measurement system. The hardware of the experimental platform is a single-core multimedia device: BeagleBoard. The main features of the board have been listed and since the work of this thesis focuses on controlling the power consumption caused by the ARM Cortex-A8 CPU, the BeagleBoard has been configured as a minimal system that disables the unnecessary components, such as the display and network subsystems. The BeagleBoard runs Linux 3.8.0 kernel, patched to support the platform DVFS mechanism, in which additional valid OPPs were added.

The power supply and measurement system is used to supply the experimental platform and measure its power consumption. It is composed of three functional modules: a power source, a digital voltmeter & ammeter and a PC-based acquisition system. The acquisition system can configure the output voltage and control the sampling frequency, thus, the software user interface of Agilent acquisition system has been shown to explain how to configure these options. The digital voltmeter & ammeter can measure the voltage and current of the whole PuC and return the values to the acquisition software. PC and power supply are connected through a USB-GPIB interface, which is a communication bridge.

In this thesis work, there are two approaches of power estimator. The first approach is a PAPI-based power estimator, for which the PAPI interface should be integrated into the decoder to access the PMCs from userspace. Besides, the decoder application also includes the MARS model that calculates the consumption estimation based on PMCs. Apart from introducing the MPEG4 part2 decoder application, those tools and packages used for compiling the decoder application have also been introduced. The second approach is the OS-level power estimator, for which the decoder application does not need any modifications. The PAPI-based power estimator approach is used as the initial way to address the PMC access. Comparing with OS-level estimator, PAPI-based estimator has some limitations that make it not suitable to be applied into the control system. The comparing process is explained in next chapter. Then, the OS-level

power estimator is decided to be used as the final feedback source instead of the PAPI-based power estimator.

Chapter 6 Simulation and implementation

In this chapter the method to simulate and implement the control system is presented. The implementation of the control system consists of three main parts: the power estimator, the closed-loop control subsystem and the PCG. To manage the estimator to feedback power consumption, an estimation model has been explained in Chapter 3, which is based on PMCs. Besides, two approaches of building power estimators have also been introduced in order to describe the process of implementation. The process of simulation and implementation of different controllers in the closed-loop control subsystem will be described, as well as the features of the control system. Finally, the PCG, which includes a battery SoC estimator and a power budget profile, will be explained in detail.

6.1 Platform PMC and DVFS enabling

The particular details of the embedded hardware/software platform used in this thesis were described in Chapter 5. As already mentioned in Chapter 5, a low-cost development board was adopted for the test-bench application platform. Since the focus of the work is firstly on the control system, the decision on the chosen board was more based on aspects like easy DVFS and PMC accessibility or open hardware/software, than on others like computational performance. Hence, some of the main features of the platform are the following.

The OS is based on Linux kernel 3.8.0 patched to support the DVFS mechanism, which is managed through the cpufreq Linux driver. In order to enable the PMC, the below configurations should be followed. One is enabling performance events and counters, and other is enabling OMAP 3 debugging peripheral to enable the according hardware. Besides, the option of generic dynamic voltage and frequency scaling support should be selected to enable DVFS.

The Linux performance event subsystem provides a framework for collecting and analyzing performance data. These events will vary based on the performance monitoring hardware and software configuration of the system. Besides, it provides per task and per CPU counters, and it provides event capabilities on top of those. The Linux performance events and counters are selected to enable the kernel support for various performance events.

6.2 PMCs access

PMC is widely implemented in majority modern processors. In this work, two different approaches have been considered to obtain PMCs prior to the estimator implementation. The first approach is based on a user-level implementation through a third-party library (Performance Application Programming Interface – PAPI), which is used as the PMC driver and is integrated into the user-level decoding application to read PMC event counts after decoding every frame. The second approach: in order to decouple the user application from the power control system, accessing PMCs is moved from the user level to the OS, which accesses directly and periodically the PMCs for carrying out the estimation task at kernel level.

6.2.1 PMC implementation based on PAPI

PAPI functions should be integrated in decoder to take PMC event samples when decoder is being executed. Figure 6-1 shows the decoder whith PAPI function calls to monitor PMCs have been inserted in. As it can be seen, after initializing the decoder, PAPI starts to access PMCs, and after a number of frames are decoded, a stop signal sent from the display actor stops accessing PMCs. The number of decoded frames can be set depending on the users requirements, which mean the period of monitoring PMCs is fixed on demand. Then the PMCs data are stored in an array for next work.

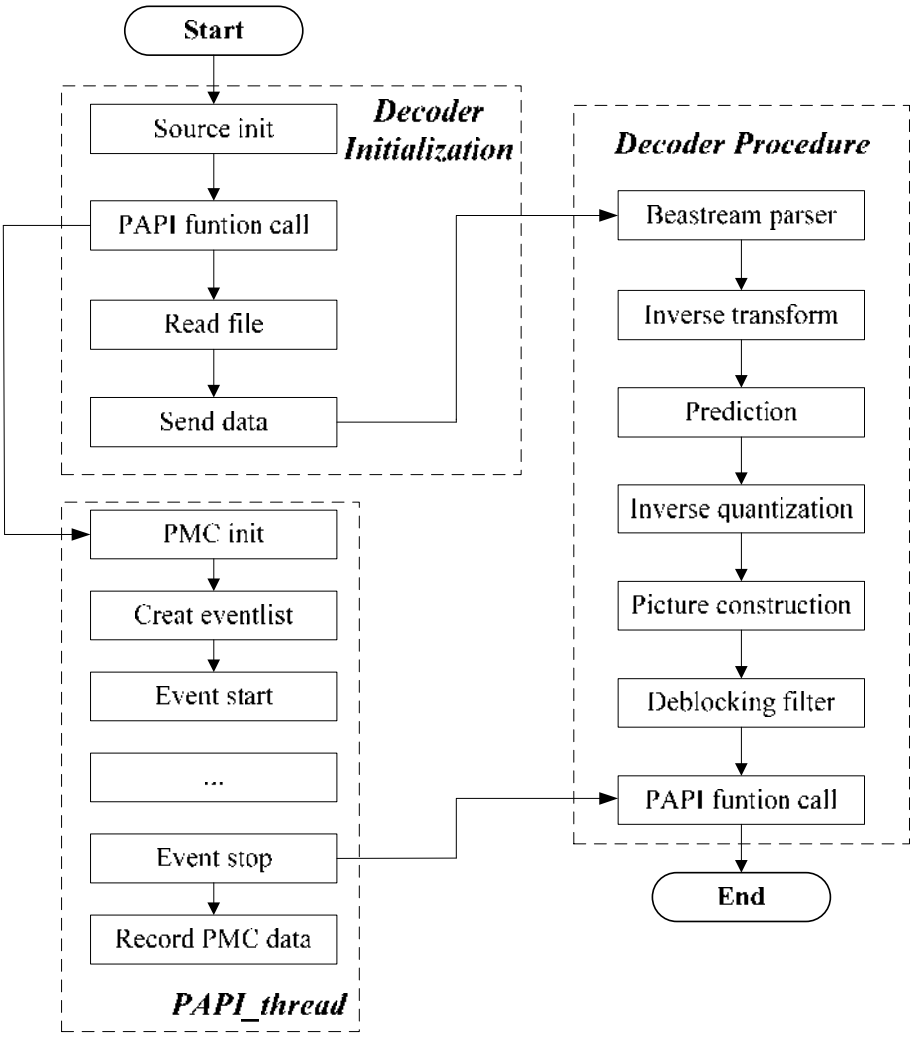


Figure 6-1 PAPI Tool Integration

As the initial approach, PAPI is used to count events by reading the PMCs before and after a performance-critical region of code. The application is a multimedia decoder, which can decode videos frame by frame. PAPI functions are inserted before and after the frame-decoding code. Figure 6-2 shows the flow chart of using PAPI.

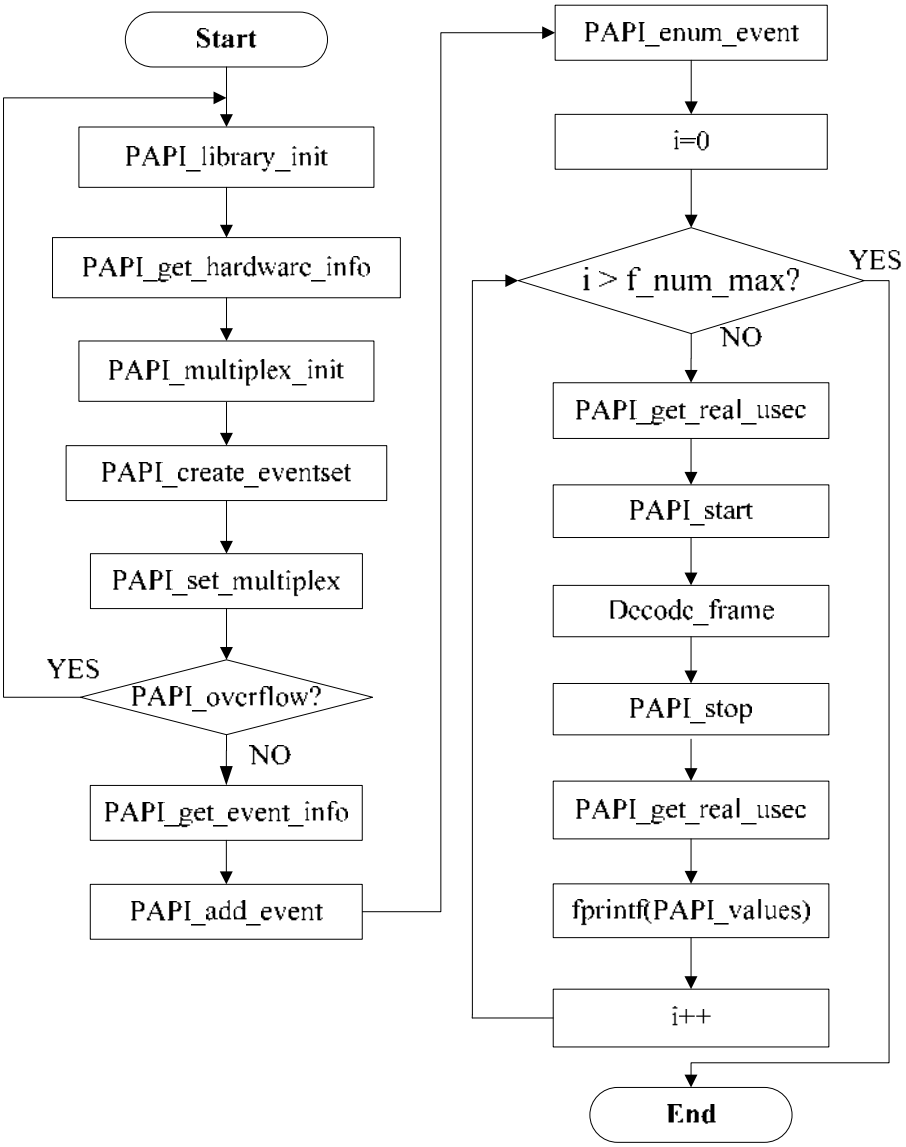


Figure 6-2 Flow chart of using PAPI

In order to initialize PAPI, the functions *PAPI_library_init*, which initializes the PAPI library, and *PAPI_get_hardware_info*, which gets information of the system hardware, are inserted into the decoder. Then, to configure PAPI, several functions are needed, such as *PAPI_multiplex_init*, which initializes multiplex support in the PAPI library; *PAPI_create_eventset*, to create a new empty PAPI event set; *PAPI_set_multiplex*, which converts the created event set to a multiplexed event set. Then it is needed to check the overflow of PAPI counters, which can set up an event to begin registering overflows.

PAPI_get_event_info can get the name and description for a given preset or native event code. Chapter 3 has introduced the method of selecting power-related events, which can be added though *PAPI_add_event* function. *PAPI_add_event* (int EventSet, int Event) only adds single PAPI events to an event set, then, *PAPI_enum_event* will return the event code for the next available present or native event. After the selected events are all added in the set, *PAPI_get_real_usec* is inserted before and after PAPI counting in order to return the total number of microseconds since the starting point. PAPI only supports thread monitoring, which means PAPI will not inherit the counting information or values from the parent threads that can distinguish individual threads; it will not confuse the parent thread with other child thread. *PAPI_start* starts counting hardware events in an event set, while *PAPI_stop* will stop counting hardware events in the event set. In this work, PAPI counts the events during the decoding of a frame, then, it will continue counting for the next frame and recording the data in a file until all the frames have been decoded, i.e., the frame number i exceeds *f_num_max*, which is the maximum frame number of a sequence.

PAPI monitors the PMC counters between frames, and the frame time is variable. The recorded data of PMCs and frame time can be used to estimate power consumption. However, the sampling time of the real-time control system has to be fixed and, therefore, PAPI is only used as an initial access to PMCs from userspace and it can help in validating the final implementation.

6.2.2 PMC driver in kernel space

In order to synchronize the PMC access time with the sampling time of the control system, a PMC driver has been inserted into the Linux *cpufreq* driver as part of a governor, Figure 6-3 shows the flow chart of PMC driver. The real-time sampling period is synchronously set to T=100 ms in all the control system modules in order to meet the time requirements of the mathematical model, as explained in Chapter 4.

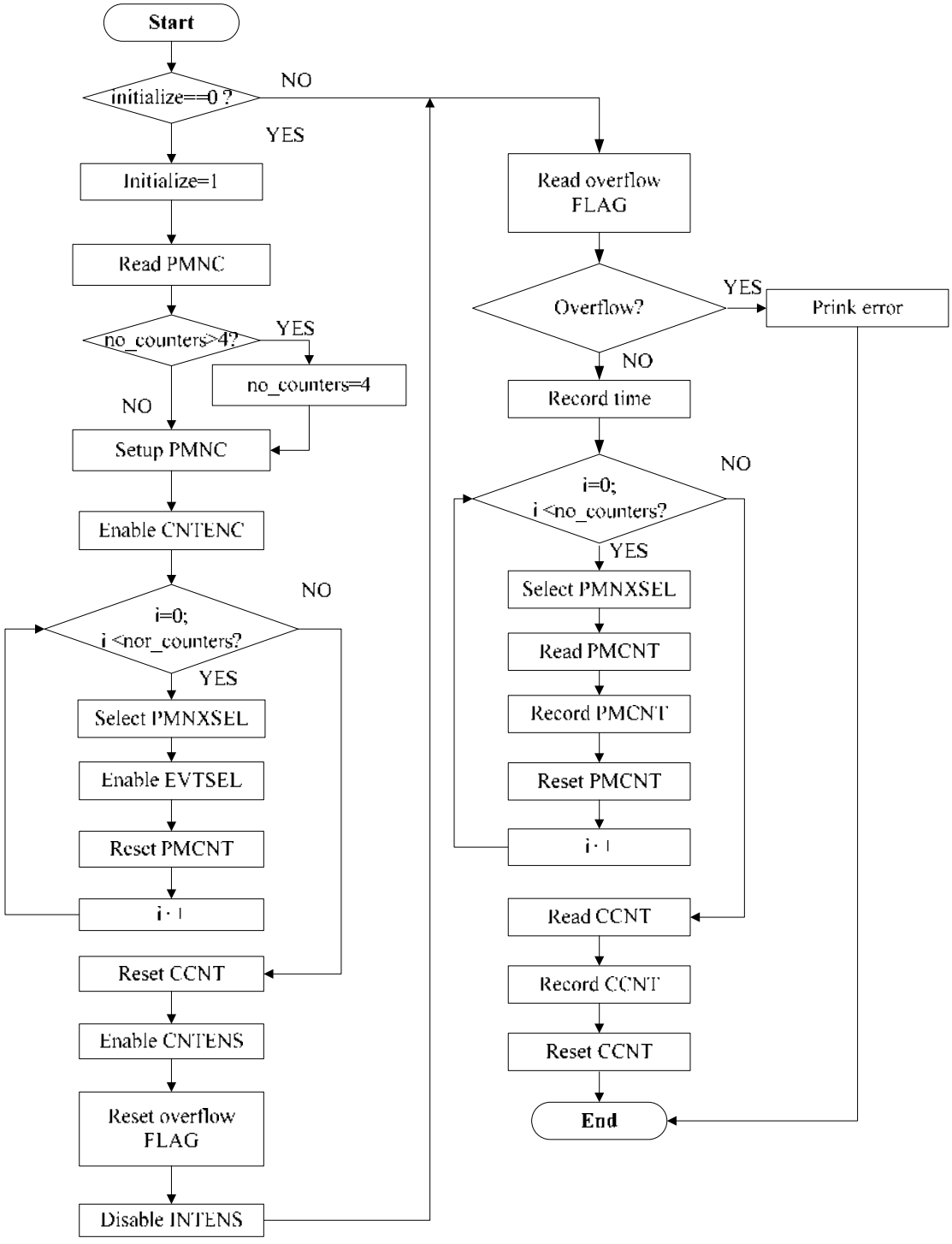


Figure 6-3 Flow chart of using PMC driver

PMC driver includes two main parts, one is initializing the related registers, and another is to monitor and record PMC counters. In the initialization part, PMNC register is read and the number of PMC registers is checked whether it is equal to 4 (ARM Cortex-A8 in Beagleboard

features 4 PMC registers), if not, there is an error, otherwise, the PMNC register is configured to control the operation of the four PMC registers and the cycle counter register. Figure 6-4 shows the bit arrangement of the PMNC register. The bits [31:24] are preset to 0x41 which means the processor is ARM; [15:11] are preset to 0x4 in order to fix the number of available PMC counters; D bit is set to 1 to count every 64th processor clock cycle; C bit is set to 1 in order to reset cycle counter (CCNT); P bit configured to 1 means reset all performance counters to zero; E bit also set to 1 enables all counters including CCNT.

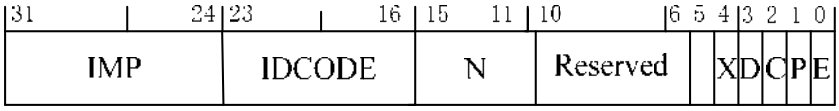


Figure 6-4 Bit arrangement of the PMNC register

Then, the procedure enters a checking where the number of counters is issued, if for any reason it is larger than 4, the number of counters is reset as 4. Otherwise, enable CNTENS so that PMC is enabled. After that, the procedure enters a loop to select performance counter (select PMNXSEL) and event (enable EVTSEL), then it resets the performance counter (reset PMCNT) until the 4 PMCs are all accessed. After the previous loop, the cycle count is reset (reset CCNT), CNTENS is set to enable PMCs, and the overflow flag is reset. Then INTENS is disabled to make sure there is not overflow. Once the initialization part is finished, the next part is executed periodically. First, the overflow flag is read; if there is not an overflow, a time stamp starts to record the period time of accessing PMCs. Then the loop enters in a loop for accessing PMCs, the counters are selected and PMCNT is read to know its value. The data of counters is recorded in an array for the further work, and PMCNT is reset after previous work. After the four PMCs are recorded, the cycle count will be read, recorded and reset.

6.3 Power consumption estimator

The first phase of the estimator implementation consists on identifying the set of events, which are most significant with respect to the power estimation. This is achieved by a filtering procedure that has been explained in Chapter 3. Table 6-1 lists the events resulting from the filter procedure.

Table 6-1 Selected Events and Functionality [74]

Event names	Description
L2_TCM	Level2 total cache misses
TLB_IM	Instruction translation look aside buffer misses
BR_TKN	Conditional branch instruction taken
SR_INS	Store instructions executed
TOT_CYC	Total cycles

In the Beagleboard environment, five events, L2_TCM, TLB_IM, BR_TKN, SR_INS and TOT_CYC are finally selected as the events that are highly related with power consumption.

L2_TCM: A significant percentage of stall cycles might lead to cache misses [75] which means it costs more power consumption than when obtaining useful data, therefore, L2_TCM should be included in the selected events.

TLB_IM: Level 2 cache can indicate the affections of instruction and data misses. TLB misses have greater influences on power consumption due to the processor needs to handle memory page table, therefore, TLB_IM is related with power consumption.

BR_TKN: If branch prediction fails, the pipeline will no longer wait for new instructions filling, which is important to CPU stalls and effects power consumption.

SR_INS: The store instruction can monitor the data write/read operations in any layer, which should be taken into account in affecting power consumption.

TOT_CYC: The number of cycles indicates a basic principle that is the application power tendency is depending on its execution time. Although the predication is not quite precise, it also descripts the high relation with power consumption.

Once the list of significant events is obtained, the MARS method is applied to estimate power consumption from PMC event counts. The power models have been adjusted and tested with all the available test sequences against 27 OPPs.

6.3.1 Estimation model

The model is build based on MARS method that was already introduced in Chapter 3. In order to calculate the coefficients between PMCs and power consumption, 78 video sequences have been used to model the power consumption of the decoding system. They are a subset of the conformance-test bit streams of the MPEG-4 part 2 simple profiles as part of ISO/IEC 14496-4 standard and which can be downloaded from [76]. The power consumption and PMCs of the 78 sequences are used to build 78 models, which are then used to estimate the power

consumption of other sequences. The average absolute percentage error (AAPE), which is the percentage of the difference between the estimated consumption and the measured power consumption, is calculated to show the accuracy of models as equation 7-1.

$$AAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{x_t - \hat{x}_t}{x_t}\right| \times 100\% \qquad 7\text{-}1$$

Where x_t is the measured value, $\hat{x}_t$ is the estimated value, and n is the number of the fitted points, i.e., the number of frames of this sequence or the number of samples.

The AAPE between estimate power consumption and real power consumption of those 78 models can be identified in Figure 6-5.

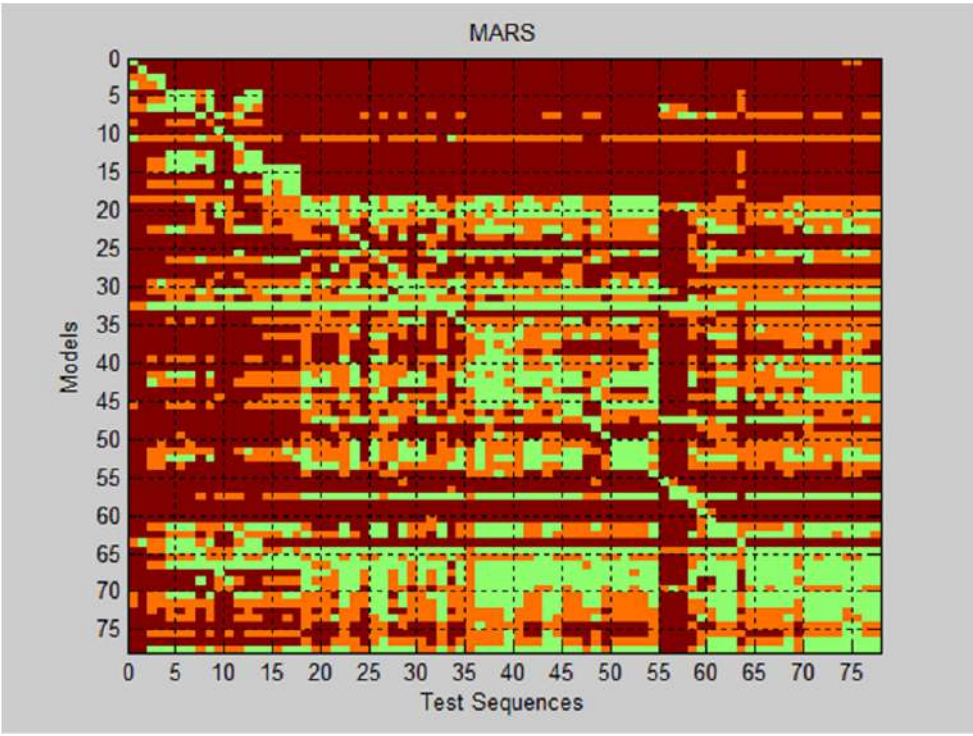


Figure 6-5 Errors of 78 models

The brown color means the error is higher than 40%, the orange part means the error is between 10% and 40% and the green one indicates the error is less than 10%. From the initial approach to build the model, it can be seen that the brown color occupies a great part of the figure, which means models results are not good enough.

Therefore, a new way to rebuild the model should be explored in order to reduce the error. The 78 different test sequences belong to 5 resolution groups and their resolutions are: 16x16,

144x80, 288x176, 176x144 and 352x288. The number of sequences belonging to each of the 5 resolution groups is listed in Table 6-2.

Table 6-2 Resolution distribution

Resolution	Number of sequences
16x16	2
144x80	2
288x176	4
176x144	10
352x288	60

Another approach is to mix the 78 sequences by resolutions, which means to classify the 78 sequences in 5 group arrays corresponding to their resolution, and the PMC values and power estimation value of each sequence can be assumed as an element. Taking out one element from each of the 5 arrays and integrating the corresponding 5 elements as a new element that can also be considered as a new mix sequence, the number of mixed sequences is $2\times2\times4\times10\times60=9600$. Then the new mixed sequences are used to rebuild the model. Figure 6-6 shows the AAPE among those mixed sequences.

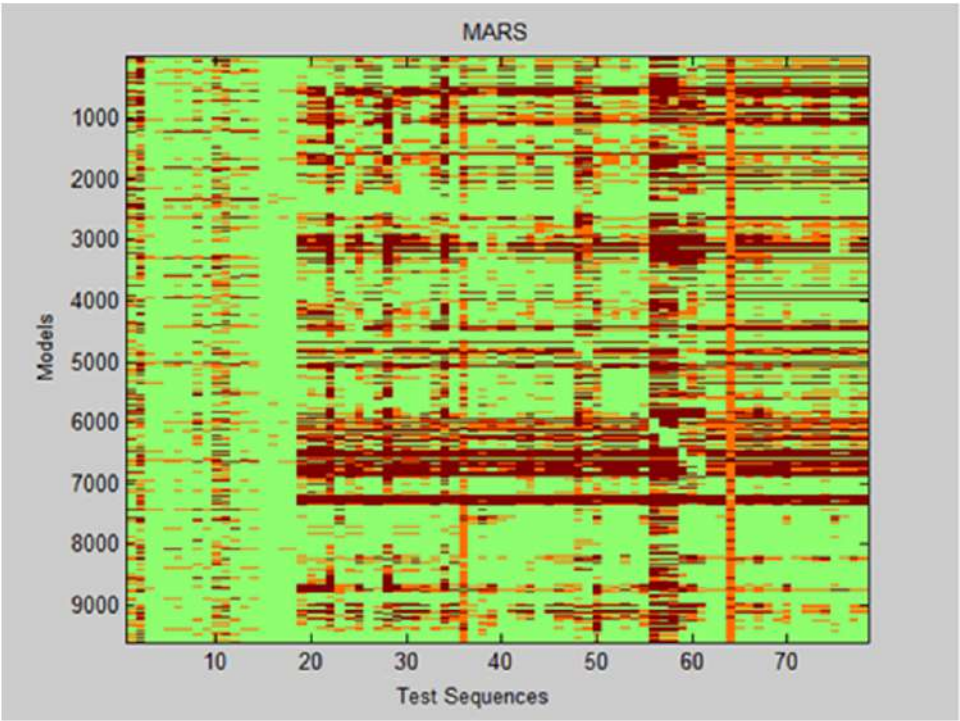


Figure 6-6 Model errors of mixed sequences

Since it has a larger green surface, Figure 6-6 shows that the results are better than with previous models. Line 4752 corresponds to the selected model which has the smallest average error, that is, 2.36%.

6.3.2 PAPI-based estimator

PAPI functions have been integrated into the decoder source code in order to periodically record PMCs counters, and estimation functions have been also inserted into the decoder to calculate the power consumption estimation by using the MARS model parameters. Then, the source code is recompiled to build the decoder. While the decoder is executing, the sequences will be decoded and the PMCs data and the power consumption estimation will be recorded. Since the control system is applied into the kernel space, the estimation value, which is produced in user space, is sent to the kernel space. That means if PAPI-based estimator cooperates with control system, communication between user space and kernel space is necessary. To send the estimation values, the system call interfaces should be added into the operating system and decoder application.

6.3.3 OS-level estimator

Now, the MARS-based estimator is implemented inside the Linux kernel, specifically in one of the *cpufreq* governors in order to have also an easy access to the DVFS facilities. The governor code has been modified to get real-time power estimations of the PUC. The migration of the user-level implementation to this new kernel-level one has implied two main challenges, which have been successfully overcome: the first is the lack of PAPI support within the kernel, whereas the second is the need of using integer numbers instead of floating-point ones.

In the final implementation, a kernel thread has been created and the performance monitoring unit (PMU) of the MPU has been configured by means of specific assembler instructions to read the selected event counts from PMCs. Besides, an endless loop was included into the kernel thread such that it repeats periodically the estimation procedure with a delay period of 100ms. With this period, the estimation and DVFS overhead has been measured to be less than 3%. In each loop iteration, the PMC values are sent to the MARS module to calculate a power estimation sample. The estimation samples are written into a file for off-line validation purposes, and also are used as real-time feedback samples in the final closed-loop control

system. It is worth noting that this OS-level implementation runs transparently in the *cpufreq* governor while the video decoder is executed at the user level.

6.3.4 Comparison of both estimators

PAPI is easily used to directly access PMCs and it can easily be applied to different devices, therefore, PAPI-based estimator was the initial approach for estimating power consumption. Besides, as another approach, OS-level estimator allows the OS to obtain accurate power consumption estimations of a video decoding task in a multimedia mobile device. Comparing the two estimators, the OS-level estimator has some advantages: the OS-level estimator can run without interfering with the user-level decoding application; related to this, whereas PAPI-based estimations have to be calculated on a video-frame time basis, the sampling frequency of the OS-level estimator can be freely fixed; furthermore, the OS-level infrastructure avoids the need of user signals to the OS for DVFS commands. Besides, the OS-level average estimation error is lower than with PAPI-based estimator, which is presented in Chapter 7. For these reasons, OS-level estimator is a good approach to be used as the feedback unit of the real-time closed-loop control system which is aimed to be implemented to regulate the power consumption of OS-based multimedia mobile devices.

6.4 Control system simulator

A simulator has been developed for the system by means of a commercial tool of dynamic system simulation [77]. This tool allows users to develop, configure and simulate graphic block diagrams. The simulator enables to anticipate the behavior of the power control system before implementing it, even considering the effects that the nonlinear OPP-based plant interface implies on the system response expected from the theoretical linear model. Furthermore, it is also helpful for validating the results obtained from the real system, as far as they match the simulation results. Figure 6-7 shows the general diagram of the control system simulator. It has two parts: a closed-loop control subsystem simulator and a PCG. The closed-loop control subsystem simulator is implemented to simulate the behavior of different controllers. Besides, based on the consumption estimation, the PCG can emulate the battery discharging and is employed to provide a suitable power budget for the video decoder depending on the battery SoC. Next sections introduce more details of the control system simulator.

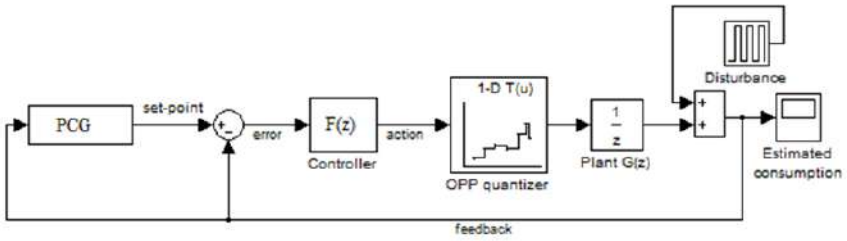


Figure 6-7 Diagram of the control system simulator.

6.4.1 Closed-loop control subsystem simulator

In section 4.2.2, the linear model of the closed-loop control subsystem was obtained. Now, that initial linear model is being enhanced in the simulator with more real system details. For example, one of the main advantages of closed-loop control systems is their capability to react to disturbances on the controlled output. Therefore, the simulation model should be tested with a disturbance input, as shown in Figure 6-8, to analyse its influence. The disturbance input would simulate the effect of a consumption variation when the system is following the set point, due, for example, to a variation in the processor load.

What's more, the clearest nonlinearity of the system is that the DVFS interface to the plant only admits a discrete number of different levels, i.e., the OPPs. This implies a strong quantization process previous to the plant, whose steps can even be irregular. Hence, the closed-loop control subsystem simulator should include a block, previous to the plant, implementing this quantization (shown in Figure 6-8).

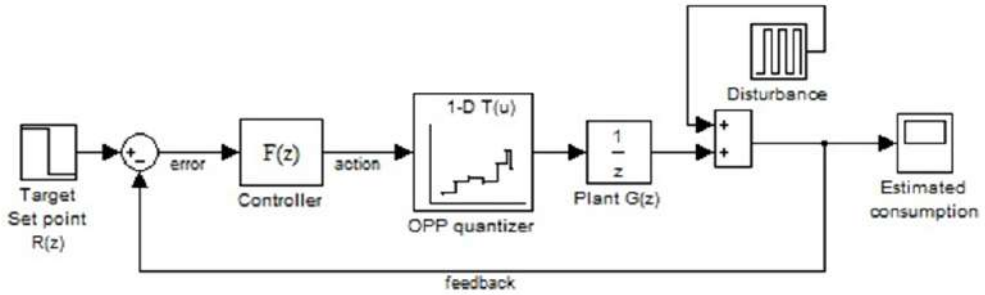


Figure 6-8 Simulation model of the nonlinear closed-loop subsystem with OPP quantization block and disturbance input block.

The quantization also includes implicitly the nonlinear effect of saturation beyond the limits of the extreme OPPs. The transfer function of this quantization block has to be obtained from the OPP average consumption values. In the case of the experimental test bench, it has the aspect represented with the stepped line of 27 irregular steps shown in Figure 6-9. The diagonal line of that figure is a reference to identify how the input breakpoints should be fixed in the middle of the step values in order to limit the maximum quantization error to ±step/2. This is a feature added to the final implementation in the experimental test bench because the default DVFS interface offers both ceil and floor functionality but not rounding to the nearest valid OPP value.

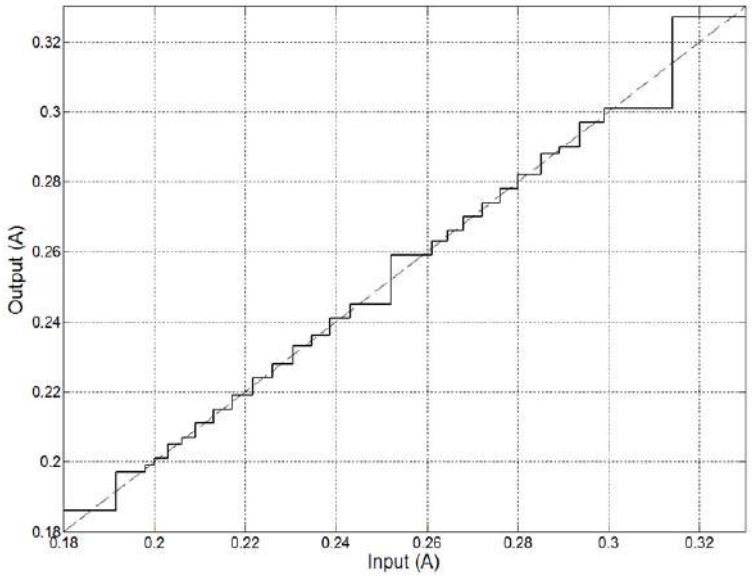


Figure 6-9 Transfer function of the discrete OPP quantization effect.

6.4.2 PCG simulator

Figure 6-10 shows the structure of the PCG which is simulated and implemented in the system. The PCG consists of a battery discharge estimator that can emulate the battery discharge, and a power budget generator that produces power budget profiles. The input of PCG is the power estimation that is calculated by OS-level estimator, and depending on the estimation value, the remaining battery can be estimated, so that a SoC percentage can be calculated. Then, the power budget generator will generate the power budgets corresponding to the battery SoC. The

power budget profiles contain the changing policies of the set-point of the closed-loop control subsystem. Next subsections introduce the PCG components.

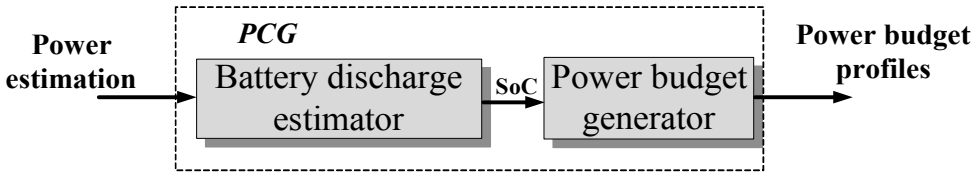


Figure 6-10 Block diagram of the PCG

6.4.2.1 Battery discharge estimator

The battery discharge estimator in PCG subsystem is used to estimate the SoC. Although the capacity of lithium-ion batteries is currently greater than 2000 mAh in most mobile devices, for testing purposes, the capacity of the battery is set to only 15 mAh (54 A·s) in order to easily and quickly monitor battery lifetime. Figure 6-11 shows the block diagram of the battery discharge estimator into the simulator.

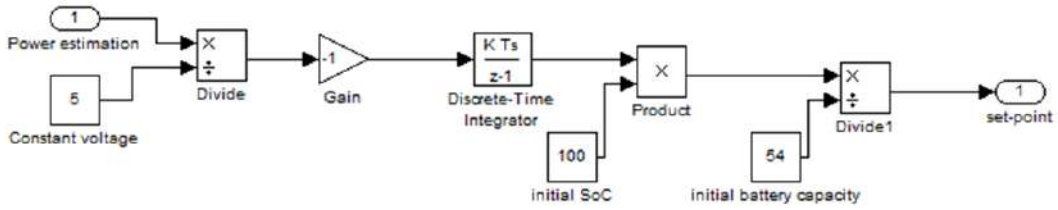


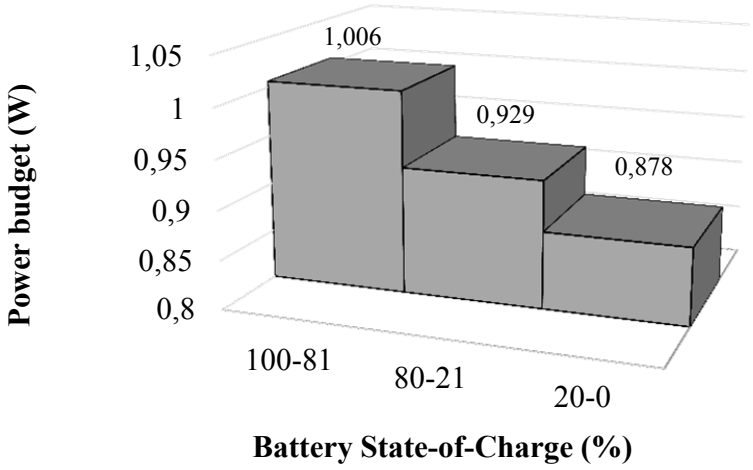
Figure 6-11 Block diagram of battery discharge estimator into the simulator

The input (1) of the battery discharge estimator is the power estimation, which divided by the voltage (5 V) is the estimated current. The estimated current is multiplied by -1 to decrease charge and then is integrated to calculate the remaining charge of the battery. The integrator block in Figure 6-11 has three parameters, the initial value of the integrator (54 A·s), i.e., the battery capacity, the gain (K=1), and the system sampling period (Ts =100 ms). Finally, the battery SoC can be monitored through outputs 1.

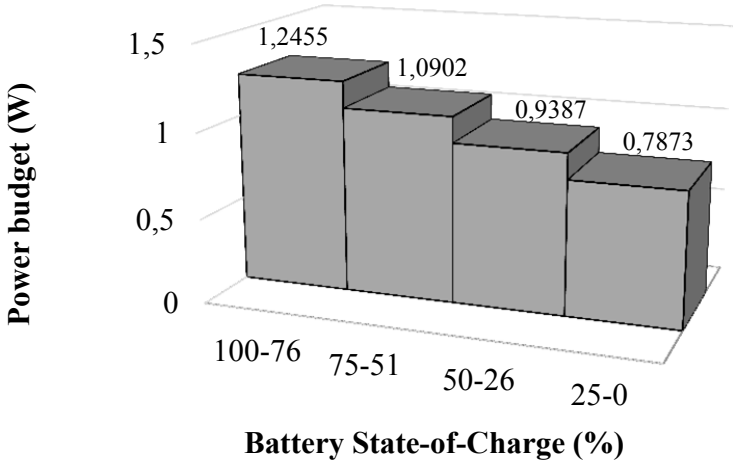
6.4.2.2 Power budget generator

The power budget generator can produce different power budgets depending on the SoC of battery. Figure 6-12 shows three examples of power budget profiles. In Figure 6-1 2(a), when the

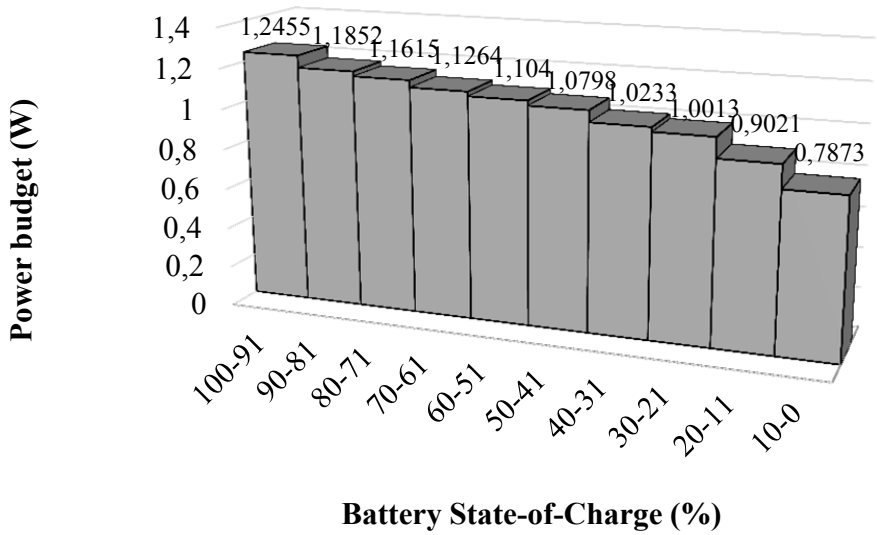
battery SoC is larger than 80%, the power budget is set as 1.006 W; when the SoC is between 80% and 20%, the power budget remains at 0.929 W; and when the battery SoC is less than 20%, the power budget is 0.878 W. In Figure 6-12(b), every 25% of the SoC, the power budget follows this sequence: 1.2455W, 1.0902W, 0.9387W and 0.7873, respectively. In Figure 6-1 2(c), every 10% of SoC decrease, the power budget changes.



(a) Example 1



(b) Example 2



(c) Example 3

Figure 6-12 Power budget profile examples

6.5 Choice of controller gains

The model proposed in chapter 4.2.2 should be tested in the control system. For this purpose and once the sample period has been fixed to T=100 ms in the implementation, the mathematical transfer functions of the controllers have to be particularized for a suitable value of gain. Looking for maximizing the relative stability of the control system, a key issue in closed-loop schemes, the modulus of the closed-loop system dominant pole should be theoretically as small as possible, i.e., the relative stability of the system improves as the dominant pole is deeper included into the unit circle. In this sense, Figure 6-13 represents that modulus, from 4-6, versus the integral gain for the aforementioned value of T and for the three integral controllers. From Figure 6-13, it is clear that the smallest modulus of the system dominant pole is achieved with the BRR-I controller: $\min(|p_{MB}|)=0$ for a gain of K_B=10. Next, the TR-I controller achieves a $\min(|p_{MT}|)=0.41$ for a gain of K_T=3.43. And finally, the FRR-I controller achieves a $\min(|p_{MF}|)=0.5$ when K_F=2.5. These are, then, the integral gain values chosen for each integral controller.

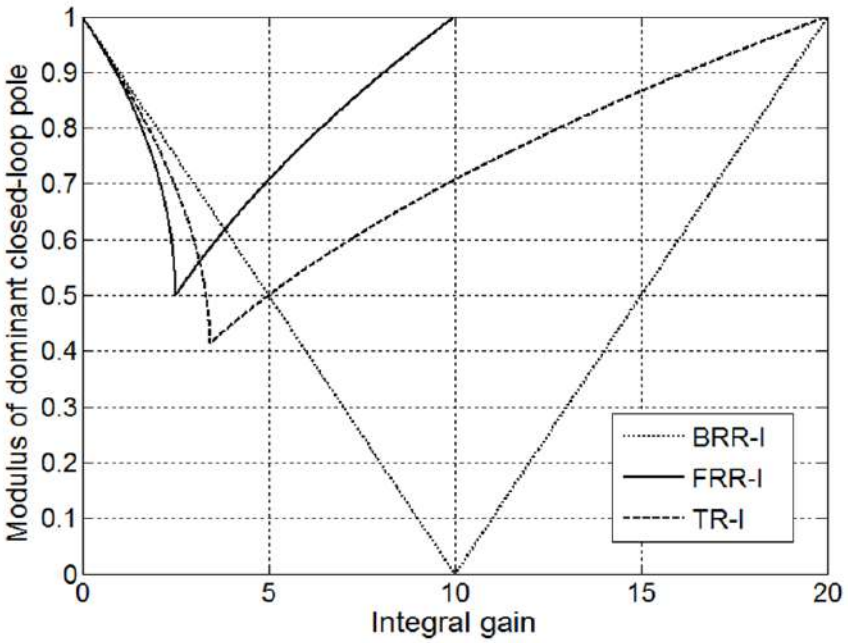


Figure 6-13 Modulus of dominant closed-loop system pole vs integral gain for the I controllers

With respect to the PI combination, Figure 6-14 shows the modulus of the two poles of M_{PI} versus K_{PI}. In order to avoid undesired oscillations in the system response, the positive pole should be kept as the dominant one, i.e., its modulus should be greater than that of the negative pole [65]. Taking this into account, a value of K_{PI}=0.75 has been chosen, which leads to a dominant closed-loop pole p_{MPI}=0.75.

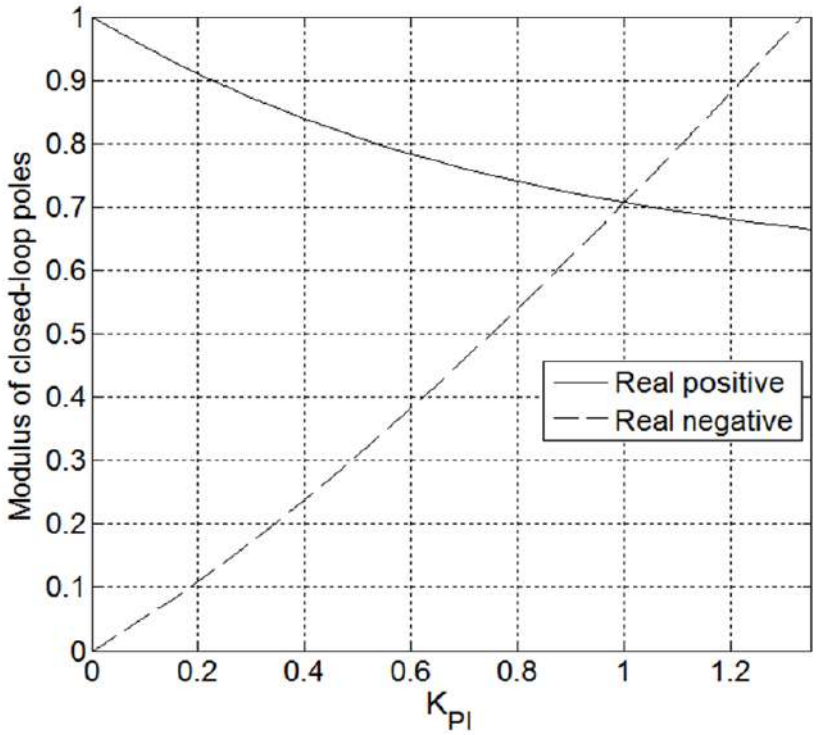


Figure 6-14 Modulus of closed-loop system poles vs K_{PI} for the PI controller

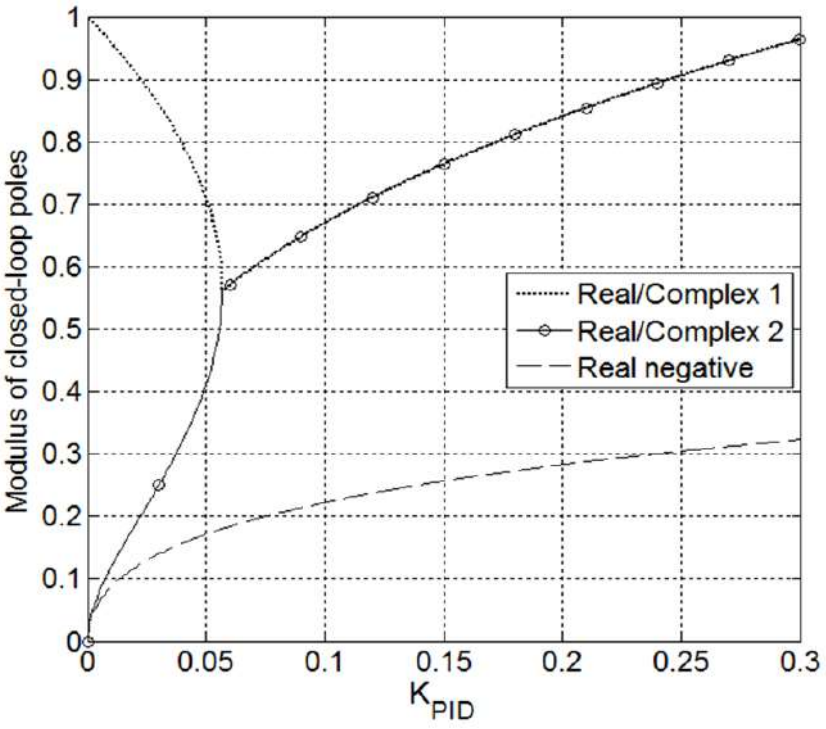


Figure 6-15 Modulus of closed-loop system poles vs K_{PID} for the PID controller

Finally, for comparison and initial checking purposes, the P controller was also tested with a value of K_P=0.5, which has been chosen as a mid-point within the system stability margin. Hence, it puts the closed-loop pole in p_{MP}=–0.5.

As a summary, Table 6-3 includes the chosen value for each of the controller gains, as well as the corresponding closed-loop dominant pole.

Table 6-3 Gain and system dominant pole for each controller

Controller	**Gain**	**Dominant Pole**
P	*KP=0.5*	*pMP=-0.5*
FRR-I	*KF=2.5*	*pMF=0.5*
BRR-I	*KB=10*	*pMB=0*
TR-I	*KT=3.43*	*pMT=0.41*
PI	*KPI=0.75*	*pMPI=0.75*
PID	*KPID=0.056*	*pMPID=0.56*

6.6 Linux-based control system implementation

After the control system has been simulated, the next step is to integrate the control system into the C-language kernel code. Thus, it can be analysed if the behaviour of the control system matches the simulation results. In this section, the method of implementation is presented and the flowchart of the implementation code is indicated in Figure 6-16,

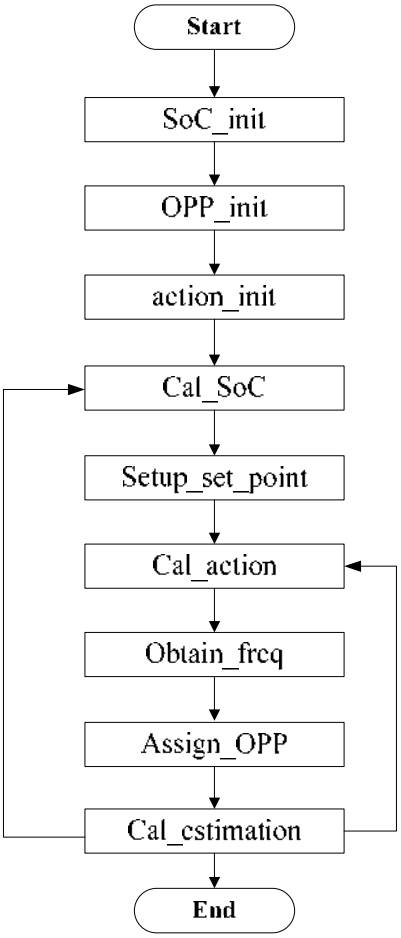


Figure 6 -16 Flow chart of control system implementation code

At the beginning, *SoC_init* sets the initial battery SoC to 100, which means the battery is full of charge, *OPP_init* sets the medium OPP 14 as the initial OPP and, correspondingly, *action_init* sets the initial value of action to 1.059 W, which is the average consumption estimation of OPP14. Within the working loop, *Cal_SoC* is used to calculate the current battery SoC. It is calculated from the battery SoC of previous sample time and the estimation of the power consumed during the current sample time. The calculation method is shown in equation 4-11. Then, depending on the current SoC, *Setup_set_point* produces the set-point of the closed-loop control subsystem following a certain preset power-budget profile. The set-point changing policy is defined through considering the battery lifetime while maintaining a reasonable QoE. Besides, *Setup_set_point* supports multiple and personalized power budget profiles to meet user requirement, Figure 6-12 shows three examples. The set-point is the input of the closed-loop

control subsystem, whose first stage in Figure 6-16 is *Cal_action* (observe the feedback line from the *Cal-estimation* stage). *Cal_action* calculates the action value of the control system in two steps: first, it obtains the closed-loop error between the set-point and the feedback (power estimation); and second, it obtains the action value by processing that error through the controller. Several controllers have been implemented in the control system through the difference equations derived from the transfer functions proposed in Chapter 4 and their results are shown in Chapter 7. Based on the action value, *Obtain_freq* finds in a lookup table the processor frequency that corresponds to the power consumption suggested by the controller action. The processor frequency is needed as the input parameter of the *cpufreq* function used to set the OPP in *Assign_OPP*. *Cal_estimation* works as OS-level estimator, accessing PMCs and estimating power consumption during every sample period. *Cal_estimation* sends finally the estimation value to *Cal_SoC* and *Cal_action* to close the functional loop.

6.7 Summary

In this chapter both simulation and implementation of the control system have been presented. DVFS mechanism of the MPU is used because it is present in many consumer-electronics platforms. This mechanism enables the MPU to work in different OPPs. Besides, the feedback information of PuC is the power consumption estimation that is based on PMCs. Therefore, both DVFS and PMC should be enabling in the platform. There are two approaches to access PMC: one is accessing PMC through PAPI and another one is accessing PMC through kernel space. After the estimation model is gotten in Chapter 3, PAPI-based estimator and OS-level estimator have been built. Furthermore, the system call interface of Linux is extended to allow kernel-space interact with the user-space threads or processes. Through comparing the advantages and disadvantages of these two estimators, OS-level estimator has been selected to be used as the feedback of closed-loop control subsystem. After that, the control system simulator has been built, which consists of a closed-loop control subsystem simulator and a PCG. After the initial linear model of the system is obtained in Chapter 4, more real system details are added to enhance the simulator. Therefore, OPP quantization block and disturbance input block were added into the simulator. Besides, a PCG is developed to simulate the behavior of battery discharge and provides power budget for the video decoder. Finally, different controllers have been researched in Chapter 4, previous to their implementation in the operating system.

Chapter 7 Experiments and Results

To meet the requirements of control algorithms for energy optimization, the control system has been implemented in Linux operating system, which can extend the battery lifetime of multimedia mobile devices depending on the user requirements while maintaining a reasonable QoE. This mechanism includes an OS-level estimator that works as the feedback of the control system. PAPI-based estimator as the first approach to estimate power consumption is used to compare with the OS-level one in order to accurately calculate estimation values. In this chapter, the experimental results, including the validation and evaluation of the two estimators, the controllers implementation, the battery life time extension and the test of disturbance will be given in four parts: for the first part, the accuracy of PAPI-based and OS-level estimators have been compared and their features for the control system are stated, the overhead of the OS-level estimator will be given to show its real performance. Then, different classic controllers have been implemented in both system simulator and real system; their behaviors also are compared in order to verify the correction. Thirdly, the potential battery life extension achieved by PCG will be shown, one example of power budget profiles will be listed in order to compare its features with other Linux original governors. Finally, the effect of power consumption variations has been tested in both simulation and implementation.

All the experiments are carried out on the BeagleBoard platform, running a Linux 3.8.0 kernel patched with a DVFS mechanism. As Chapter 5 has introduced, the simple profile of the MPEG4 Part 2 decoder has been considered as the decoder application. 78 conformance sequences were used to test the control system and they have been configured with the common test conditions such as different spatial resolutions, frame combinations, slice types, quantization parameters, frame rates, and entropy coding methods.

7.1 Estimators validation and evaluation

Rather than using any sensor, to correctly feedback the power consumption for the control system, an accurate estimator is necessary. As described in Chapter 3, before building the estimator, firstly, suitable PMCs that are high related with power consumption should be filtered; then, MARS regression is used to build the power estimation model. Once the model is built, estimation results and measurement results are compared to test the accuracy of the model.

As an example, Figure 7-1 compares real power consumption of the board with PAPI-based estimations and OS-level estimations, while decoding the foreman sequence through a set of consecutive OPPs, which change every 10s. As it can be seen in Figure 7-1, the power consumption and its estimations increase with the OPP. In order to understand the differences between the shape of the real consumption and that of the estimations, note that consumption records are acquired from the whole board, whereas the estimations focus on the power consumption core mainly due to the decoding task in the processor, without the sporadic consumption spikes not due to the decoding activity. Note also in Figure 7-1 that the time resolution (sampling frequency) of OS-level estimations is higher than PAPI-based ones. Moreover, Table 7-1 lists maximum, minimum and average error between mean real power consumption core and the two power estimation approaches.

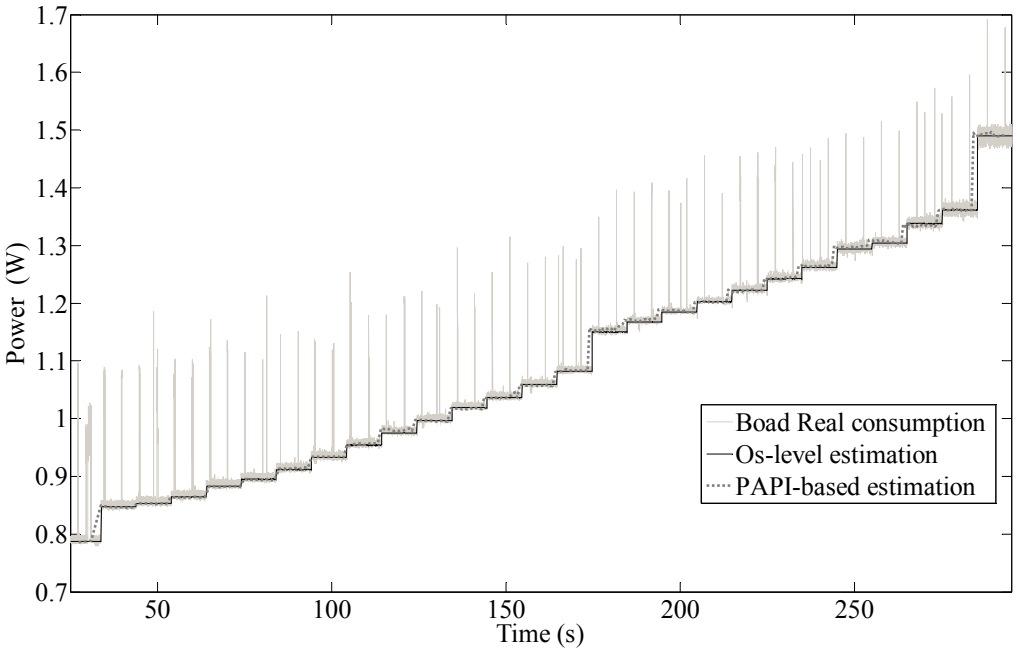


Figure 7-1 Power estimations and real power consumption

Table 7-1 Estimation Error

Error (%)	**Max**	**Min**	**Avg**
PAPI-based estimation	4.92	0.12	3.63
OS-level estimation	3.14	0.00	2.36

7.2 Test of closed-loop subsystem

The general model proposed in Chapter 4 has been implemented and tested in the system presented in Chapter 5. The sampling period has been fixed to T=100 ms in both simulation and implementation, whose details are given in Chapter 6. Next subsections show the experimental test and its results.

7.2.1 Test case

The system has been tested initially against steps in the set-point input. A step-shaped set-point would indicate that a different (constant) power is desired for the system consumption from a certain point in time. Thus, the results show how the system behaves in a situation in which a constant power consumption is required during a video decoding activity, for example to lead to a regulated battery-discharge rate regardless of other energy- or QoE-related issues. In further experiments, the set point will have to be dynamically adjusted to satisfy the power needs at each moment. These needs depend mainly on two aspects: the user expectations, in terms of battery recharge cycle and/or QoE, as well as the video decoder requirements, in terms of, for example, decoding complexity. Recent developments in digital-video standards are proposing the integration of metadata in the bit stream for signaling the video decoding complexity, which could also be used to tune the control-subsystem set point.

Suppose the system is working in its mid-OPP (the default state), which implies an average consumption estimation of 1.059 W. Suppose also that, at a certain moment, it is needed that the system reduces its power consumption. For the sake of clarity in the analysis of the closed-loop subsystem response, the target reduced consumption is made to coincide with the average consumption level of an OPP. If the target consumption were chosen between two consecutive OPP levels, the integral action of the proposed controllers (except P controller) would generate oscillations in the steady-state system response due to the nonlinear quantized plant input. Thus, the proposed test decreases the set point from 1.059 W to the average consumption level of OPP8, i.e. 933 mW, at a time t=1 s. This implies an input step of -126 mW of amplitude.

7.2.2 Results of closed-loop subsystem and their discussion

The following figures show how the control system responds to the test-case step. For

validation purposes, each figure represents both the simulation and the real time response for a controller, i.e., the output of the systems depicted in Figure 4-2 and Figure 4-6, respectively. Thus, as it can be observed in the figures, the power consumption of the real system follows the profile predicted by simulation for all controllers, which validates the implementation of the control subsystem. On the other hand, the comparison between figures enables the identification of the effects of the different controllers on the system response. In this sense, perhaps the most especial case is the one corresponding to the P controller, in Figure 7-2, because it is the only one which does not reach the desired final average consumption of 933 mW. It is due to the inherent steady-state error of the P controller. Moreover, the P controller response shows a one-sample overshoot due to the negative value of the system pole (see P_{MP} in Table 6-3). Besides, although the settling time could be thought to be longer for a value of $|P_{MP}|$=0.5 (comparable to the FRR-I or PID cases, as it can be seen in Table 6-3), the combined effect of both the system error (avoided in the integral cases) and the nonlinear quantized plant input, similar to a dead-zone effect, prevents that the system response can reach its linearly expected final value, thus shortening the transient response.

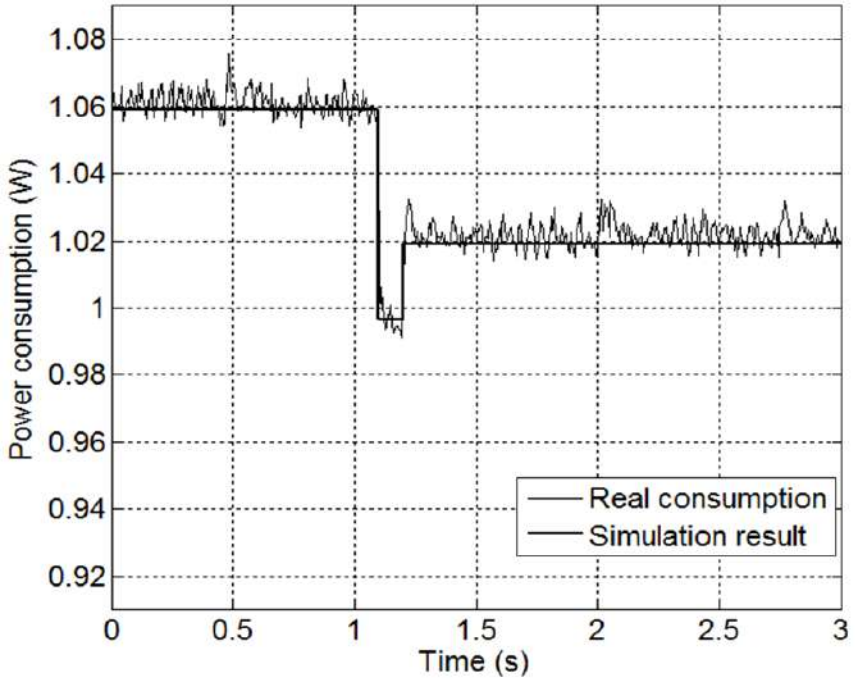


Figure 7-2 System time response for the P controller

The rest of controllers, which include an integral component, reach the desired final value because they imply a null steady-state error to a step-shaped set point (e_{ss} in 4-3). Hence, the differences between their time responses are confined to the transient part. In this sense, an interesting case is the one corresponding to the BRR-I controller, because it gives rise to the shortest settling time in the system response, i.e., a single sample interval, as it can be seen in Figure 7-3. This is because the modulus of the system pole is the smallest one for this controller, i.e., p_{MB}=0 (see Table 6-3).

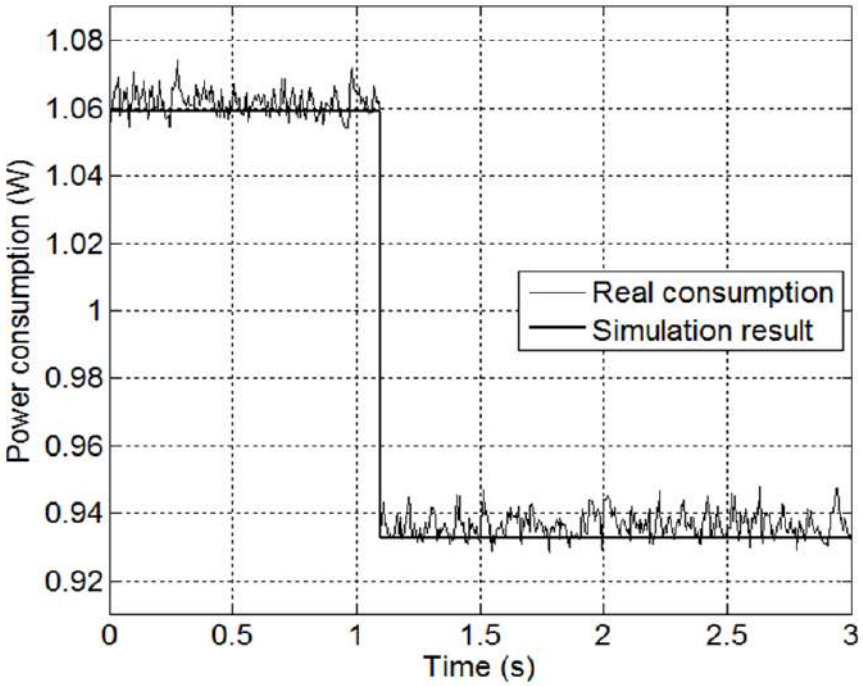


Figure 7-3 System time response for the BRR-I controller

The TR-I controller places the two poles of the system in z=0.41 (see P_{MT} in Table 6-3) and leads to a system response like that of Figure 7-4. In this case, the settling time of the step response is 500 ms and it approaches monotonically to the final value along the 5 samples of this interval, as corresponds to positive real poles.

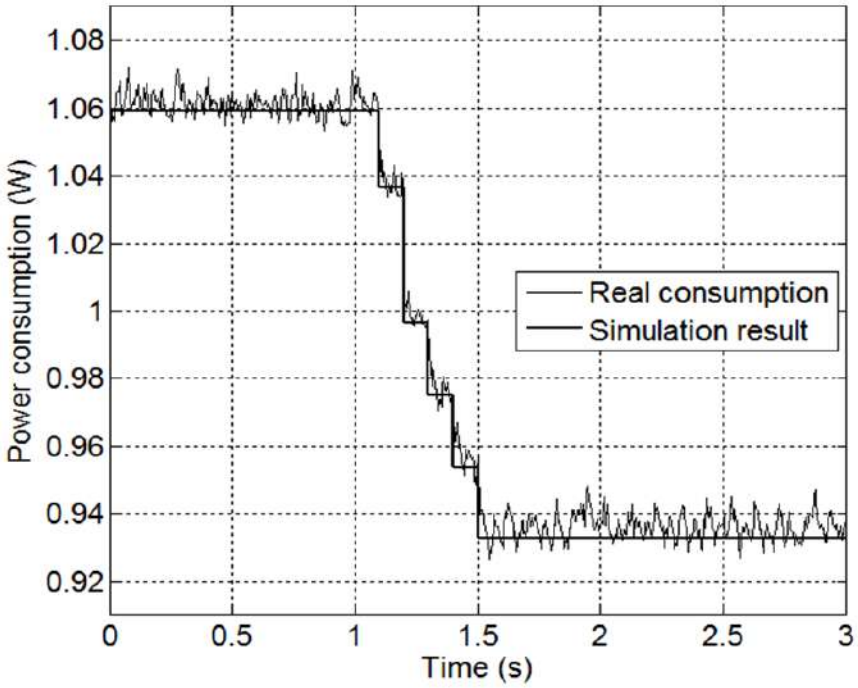


Figure 7-4 System time response for the TR-I controller

In turn, the FRR-I controller places the two system poles in z=0.5 (see P_{MF} in Table 6-3) and gives rise to a time response like the one shown in Figure 7-5. With a pole modulus slightly greater than that of the TR-I case, the settling time of the system response goes a bit further (700 ms). Besides, since the transfer function of the FRR-I controller has no zeros, it adds a sample delay to the one of the plant itself and, therefore, the first change in the system response appears two samples after the input step, i.e., at t=1.2 (see Figure 7-5). Here again, the positive real poles avoid oscillations in the time response.

With respect to the two combined controllers, the PI places the dominant system pole in z=0.75. Since it has the greatest modulus (see p_{MPI} in Table 6-3), the settling time is also the longest, i.e., almost 1 s (see Figure 7-6). Furthermore, since the other system pole, although non-dominant, is real negative, its effect appears in the form of an oscillation at the end of the transient response.

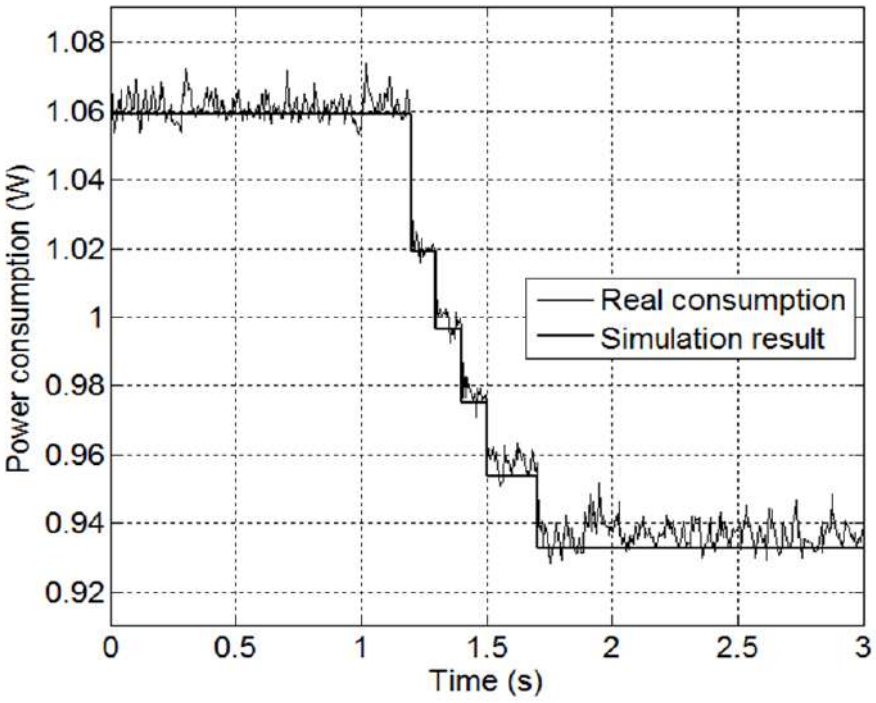


Figure 7-5 System time response for the FRR-I controller

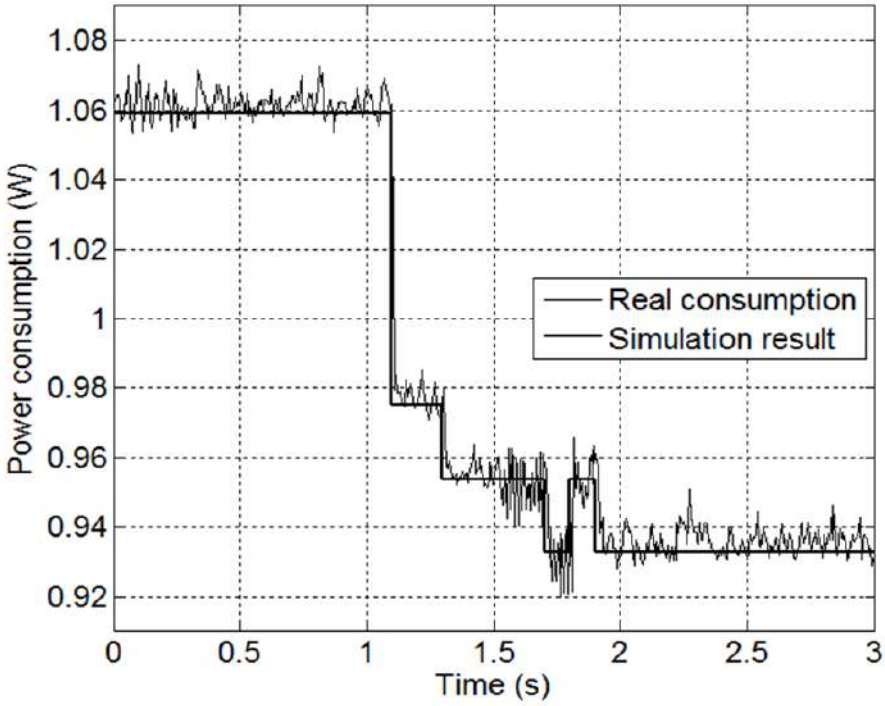


Figure 7-6 System time response for the PI controller

Finally, the PID combination, with a dominant double pole of the system in z=0.56 (see p_{MPID} in Table 6-3), leads to a settling time similar to that of the FRR-I controller, with p_{MF}=0.5 (see Figure 7-7). Also as in the case of the FRR-I controller, the time response shows a two-sample input-output delay, i.e., 200 ms are needed to react to the input step. In this case, the reason is simply the quantization process at the plant input.

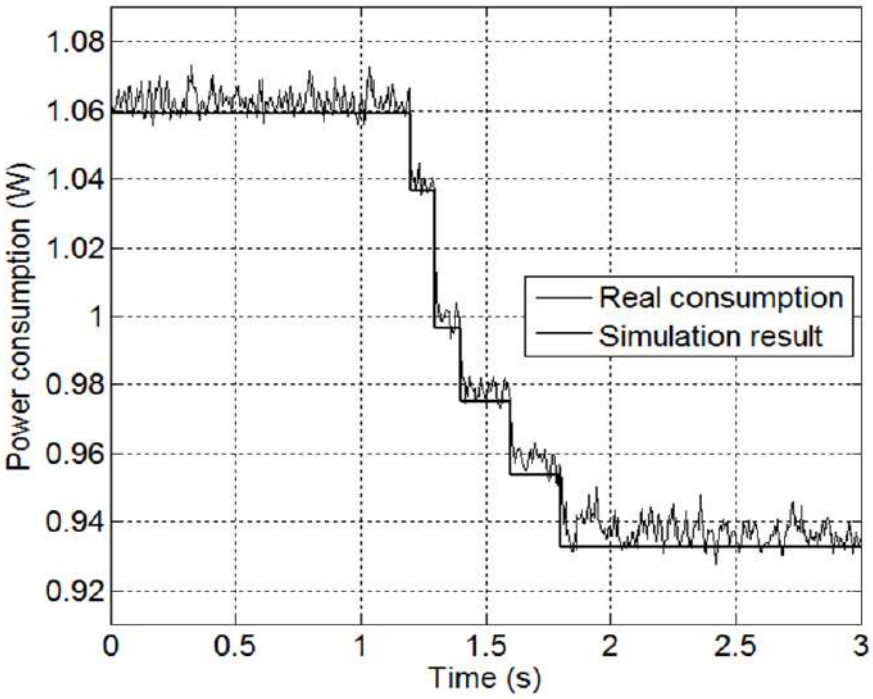


Figure 7-7 System time response for the PID controller

The results show, on one hand, how the real system consumption matches the simulated response, which validates the mathematical model proposed as well as the system implementation. On the other hand, the system is stable in all cases and is able to follow the set point in steady state, with clear advantages of I-based controllers over the simplest P controller. Furthermore, the combined PI and PID controllers do not achieve better results than some of the simpler integral controllers. In fact, the BRR-I controller leads to the fastest response, with only one sample of settling interval. However, due to the DVFS nonlinearity, this responsiveness might cause undesired and frequent oscillations when there is no OPP that can cancel the loop error. Among the tested controllers, the best trade-off between settling time and responsiveness is achieved with the TR-I controller, which leads to a smooth and quick enough response.

Therefore, TR-I is chosen as a valid simple controller to be applied into the control system for next system tests. These results, along with the low overhead implied by the control system, validate it as a means of keeping the system consumption close to the desired value (set point), regardless of the dynamic consumption demand of the video decoder as proven in next section. The overhead is equal to the execution time of running TR-I controller is divided by the sample time and the value of overhead is 4.1%.

7.3 Test of Disturbance

To test the effect of power consumption variations due to variations in the video processing load, two video sequences of different complexity are decoded in a never-ending succession (carousel), for 60 seconds each. Table 7-2 lists the MPU workloads under different OPPs when decoding the two sequences, i.e., Sequence_s (simpler) and Sequence_c (more complex). The workload is expressed as the average percentage of time the MPU needs to process a video frame for each frame interval of 40 ms.

Table 7-2 MPU Workload for different Sequences and OPPs

OPP	Sequence_s	Sequence_c	OPP	Sequence_s	Sequence_c
1	76.67%	100%	15	22.59%	44.4%
2	45.2%	99.64%	16	19.61%	37.77%
3	43.18%	95.75%	17	19.46%	37.46%
4	41.2%	87.82%	18	19%	36.62%
5	37.54%	81.33%	19	18.92%	36.12%
6	36.85%	75.15%	20	18.46%	34.49%
7	33.88%	68.36%	21	18.31%	33.42%
8	31.36%	61.19%	22	18.3%	32.96%
9	30.98%	59.66%	23	17.78%	32.04%
10	29.07%	56.84%	24	17.47%	31.82%
11	28.61%	53.18%	25	17.32%	31.28%
12	25.18%	51.12%	26	17.24%	29.98%
13	23.73%	49.18%	27	11.6%	26.17%
14	22.97%	46.62%			

Figure 7-8 is generated from Table 7-2 data, which intuitively clarifies with the growing

of OPP number, the workload of MPU is decreasing. Besides, the higher the video complexity is, the higher the workload.

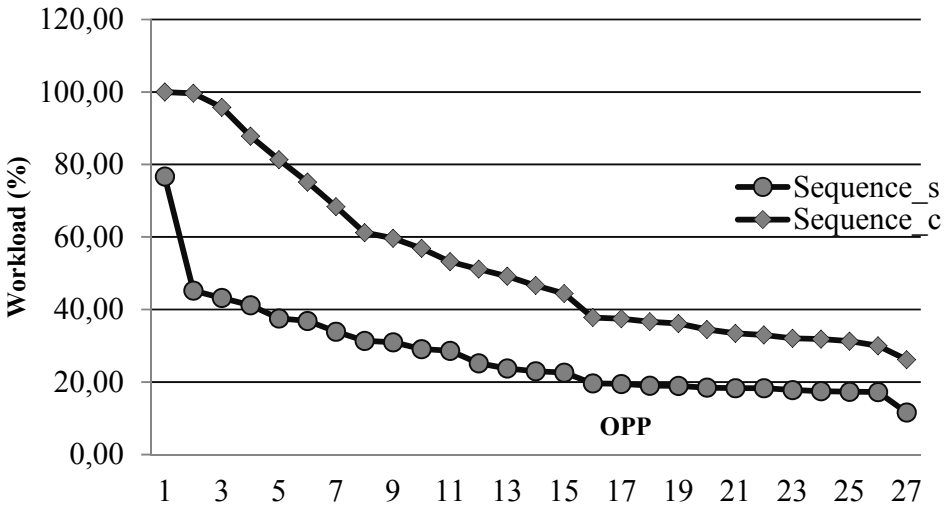


Figure 7-8 MPU Workload for different complexity sequences and OPPs

Table 7-3 lists the average power estimation, average power consumption and average absolute error of those two sequences when running them under different OPPs, while maintaining a reasonable QoE, which means the decoder can decode at least 25 frames per second. The general average absolute estimation error of the Sequence_c and Sequence_s in all OPPs are 1.99% and 1.931%, respectively, which can indicate the estimation model has high accuracy.

Table 7-3 Average Estimation and consumption of different complexity sequences

OPP	Sequence_s			Sequence_c		
	Avg. Est (W)	Avg. Con (W)	Avg.error (%)	Avg. Est (W)	Avg. Con(W)	Avg.error (%)
OPP1	0.7789	0.7850	0.5240	0.8101	0.7913	1.5196
OPP2	0.8002	0.8215	1.8297	0.8643	0.8455	1.5196
OPP3	0.8080	0.8253	1.4861	0.8776	0.8492	2.2955
OPP4	0.8099	0.8316	1.8641	0.8781	0.8581	1.6166
OPP5	0.8253	0.8442	1.6236	0.9021	0.8719	2.4410
OPP6	0.8285	0.8505	1.8899	0.9053	0.8807	1.9884
OPP7	0.8308	0.8618	2.6630	0.9128	0.8921	1.6731

OPP8	0.8429	0.8732	2.6029	0.9291	0.9072	1.7701
OPP9	0.8529	0.8832	2.6029	0.9387	0.9223	1.3256
OPP10	0.8661	0.8933	2.3366	0.9560	0.9349	1.7055
OPP11	0.8861	0.9059	1.7009	0.9769	0.9500	2.1743
OPP12	0.8958	0.9223	2.2764	0.9916	0.9651	2.1419
OPP13	0.9056	0.9261	1.7610	1.0013	0.9727	2.3117
OPP14	0.9198	0.9399	1.7267	1.0058	0.9878	1.4549
OPP15	0.9433	0.9550	1.0051	1.0293	1.0042	2.0288
OPP16	0.9553	0.9916	3.1183	1.0630	1.0419	1.7055
OPP17	0.9707	1.0004	2.5513	1.0798	1.0520	2.2470
OPP18	0.9803	1.0092	2.4826	1.0902	1.0659	1.9641
OPP19	0.9923	1.0193	2.3194	1.1040	1.0759	2.2713
OPP20	1.0076	1.0281	1.7610	1.1174	1.0860	2.5380
OPP21	1.0170	1.0394	1.9242	1.1264	1.0986	2.2470
OPP22	1.0396	1.0495	0.8504	1.1454	1.1112	2.7643
OPP23	1.0516	1.0684	1.4432	1.1615	1.1301	2.5380
OPP24	1.0598	1.0722	1.0652	1.1722	1.1377	2.7886
OPP25	1.0696	1.0923	1.9500	1.1852	1.1616	1.9075
OPP26	1.0887	1.1062	1.5033	1.2040	1.1754	2.3117
OPP27	1.1275	1.1641	3.1441	1.2455	1.2372	0.6709

From Figure 7-9, it can be seen that the average power consumption increases with the growth of OPP. On the other hand, the higher the video complexity is, the higher the power consumption.

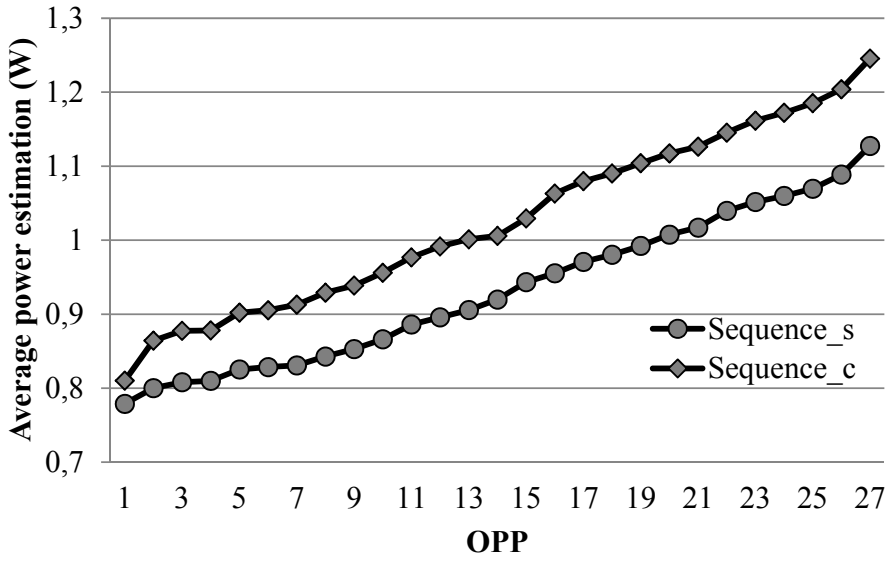
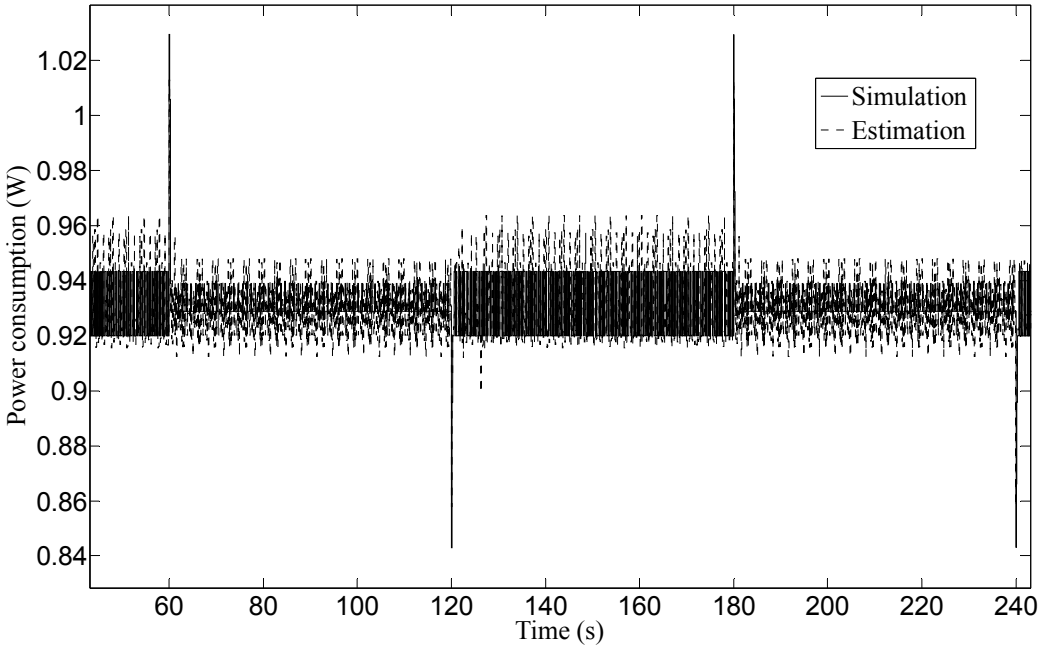


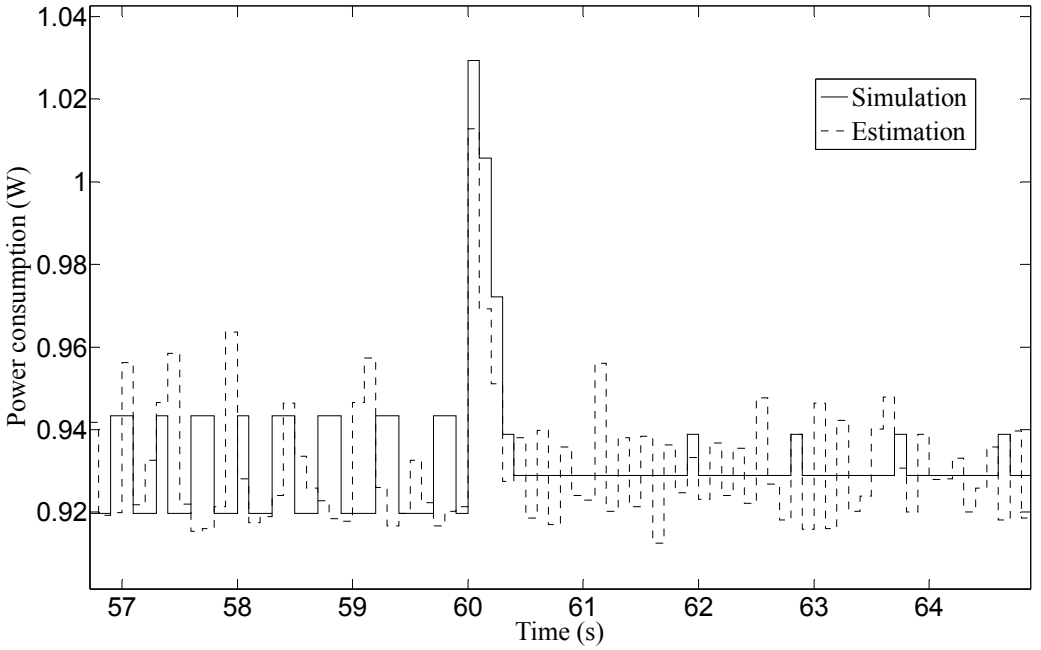
Figure 7-9 Average Estimation and consumption of different complexity sequences and OPPs

Accordingly, the disturbance block of the simulator simulates the differential average power consumption of one sequence relative to the other in open loop, which has been measured as 86 mW, i.e., the average difference between the two lines of Figure 7-9 is 86 mW. Therefore, the disturbance block is a +86-mW-amplitude step-shaped input that is active while the higher-consumption video sequence (Sequence_c) decoding is being simulated and inactive while the lower-consumption video sequence (Sequence_s) decoding is being simulated.

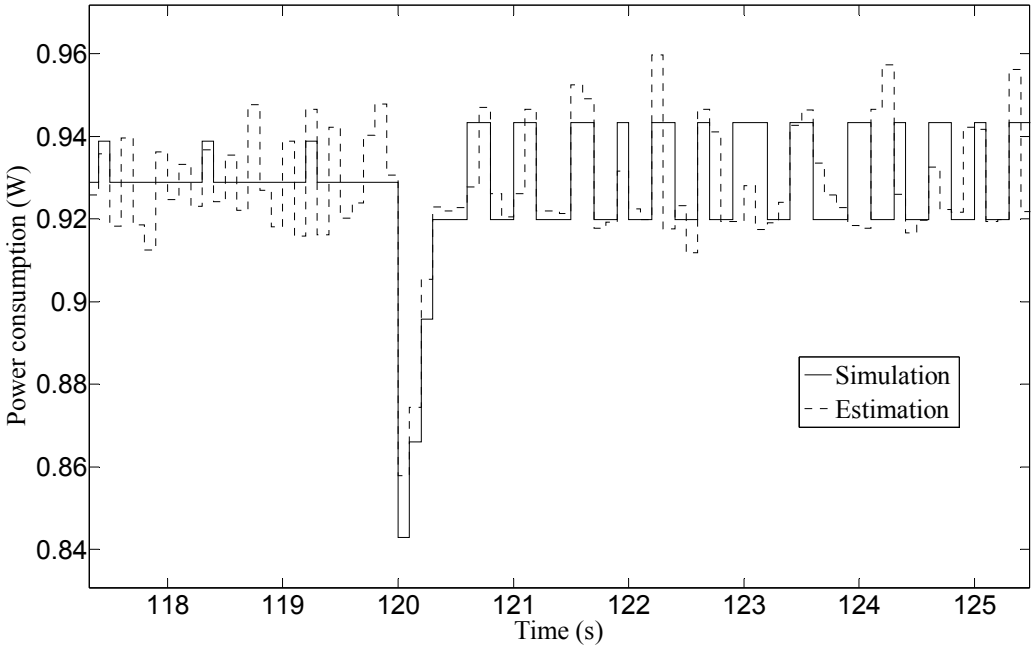
Figure 7-10 shows the behavior of the closed-loop subsystem output in the presence of a consumption disturbance with the chosen TR-I controller. Thus, for a set point of 933 mW, Figure 7-10(a) shows how the average power consumptions, both estimated by the real system and simulated, stay near the set point regardless of the video sequence complexity. Some spikes (overshoots) can be seen at the instants in which the decoded video sequence switches within the carousel, i.e., at t=60 s, t=120 s, t=180 s and t=240 s. In Figure 7-10 (b), the details of what occurs when the decoding of Sequence_c begins can be seen, which results in a transient rise in power consumption at t=60 s of approximately 86 mW. In Figure 7-10 (c), the details of what occurs when Sequence_s returns to be decoded can be seen, which leads to a transient decrease at t=120 s corresponding again to the difference in power consumption between both sequences, i.e., approximately 86 mW. In any case, the TR-I controller returns the system consumption to the desired value after a settling time of approximately 500 ms, which matches the results obtained from the step response experiment described in previous section (see Figure 7-4).


Simulation
Estimation
Power consumption (W)
Time (s)
1.02
1
0.98
0.96
0.94
0.92
0.9
0.88
0.86
0.84
60
80
100
120
140
160
180
200
220
240

(a)


Simulation
Estimation
Power consumption (W)
Time (s)
1.04
1.02
1
0.98
0.96
0.94
0.92
57
58
59
60
61
62
63
64

(b)



(c)

Figure 7-10 Closed-loop subsystem response to disturbance

Clearly, this is achieved by changing the OPP and, correspondingly, the MPU working frequency, while maintaining a reasonable QoE. Since the closed-loop subsystem is tasked with maintaining the power budget for the video decoder, Figure 7-11 illustrates how when the consumption demand of the decoding task is lower (until t=60 s), the decoder can make use of more system resources, i.e., a higher OPP, whereas when the consumption demand of the decoding task is higher (from t=60 s), the decoder has to manage with fewer resources, i.e., the OPP decreases from OPP14/15 to OPP 8/9, in order to respect its power budget.

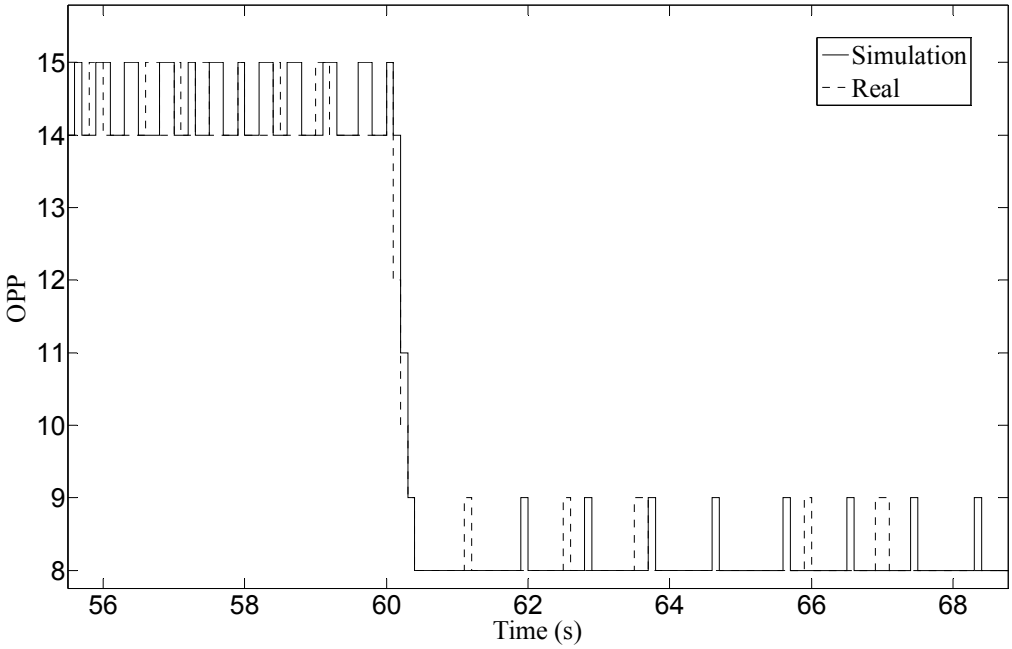


Figure 7-11 Detail of the active OPP when the consumption demand of the decoding task increases

7.4 Test of PCG

The PCG has been implemented in the Linux *cpufreq* driver such that, depending on the estimated SoC, a power budget is assigned to the decoder in real time and the closed-loop control subsystem takes charge of controlling it. That means the power budget for the decoder is the set point for the control subsystem. A number of profiles were implemented and tested for the following comparative tests, and Figure 6-12(a) is one representative example of them. In this profile, when the battery SoC is larger than 80%, the power budget is set as 1.006 W; when the SoC is between 80% and 20%, the power budget remains at 0.929 W; and when the battery SoC is less than 20%, the power budget is 0.878 W.

In order to have a reference of the performance of the proposed system implemented in the experimental platform, a coherent comparison has been done with original Linux governors. Performance, conservative, ondemand and powersave are the original Linux *cpufreq* governors. The performance governor sets statically the highest OPP, whereas the powersave governor sets statically the lowest OPP. The other two *cpufreq* governors, ondemand and conservative, are dynamic governors that can set the OPP in real time depending on the current workload. The difference between them is that the conservative governor gradually increases and decreases the

CPU speed rather than jumping to the maximum speed when there is any load increase on the CPU. It is well known that, since the performance governor sets the highest OPP, it leads to the shortest battery lifetime, whereas the powersave governor works under the lowest OPP in order to guarantee the longest possible battery lifetime. However, since they are static, they cannot adapt to the system status. Figure 7-12 shows a comparison of the battery lifetimes achieved by the two dynamic *cpufreq* governors for the example profile of PCG proposed in Figure 6-12(a) when decoding the same video sequence for a battery capacity of 15 mAh. In that figure, it can be seen that conservative and ondemand governors can make the battery last up to 257.1 s and 262.6 s, respectively, whereas the example profile of PCG leads to a battery lifetime of 289.7 s (12.6% and 10.3% improvement, respectively) in the real system, with a similar simulation result.

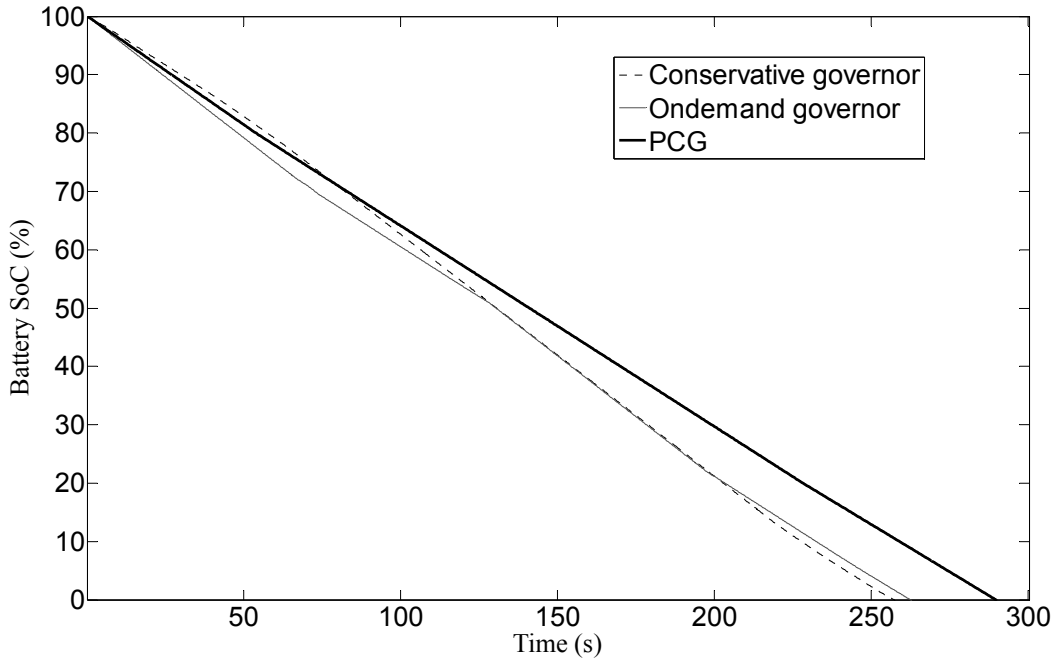


Figure 7-12 Battery lifetime under dynamic governors when decoding the simpler sequence

On the other hand, when there are variations in the video-processing load, forced in the tests by decoding two video sequences of different complexity in carousel for 60 s each, as explained above, the corresponding battery lifetimes are those indicated in Figure 7-13. In this case, with conservative and ondemand governors the battery lifetime decreases to 250.7 s and 256.9 s, respectively, because there are some intervals during which the processing load

increases with respect to the experiment of Figure 7-12. However, the proposed example profile of PCG keeps a battery lifetime of 289.7 s (15.5% and 12.8% improvement, respectively) in the real system, with a similar simulation result. This is due to the robustness of the I-based closed-loop control subsystem to the disturbances.

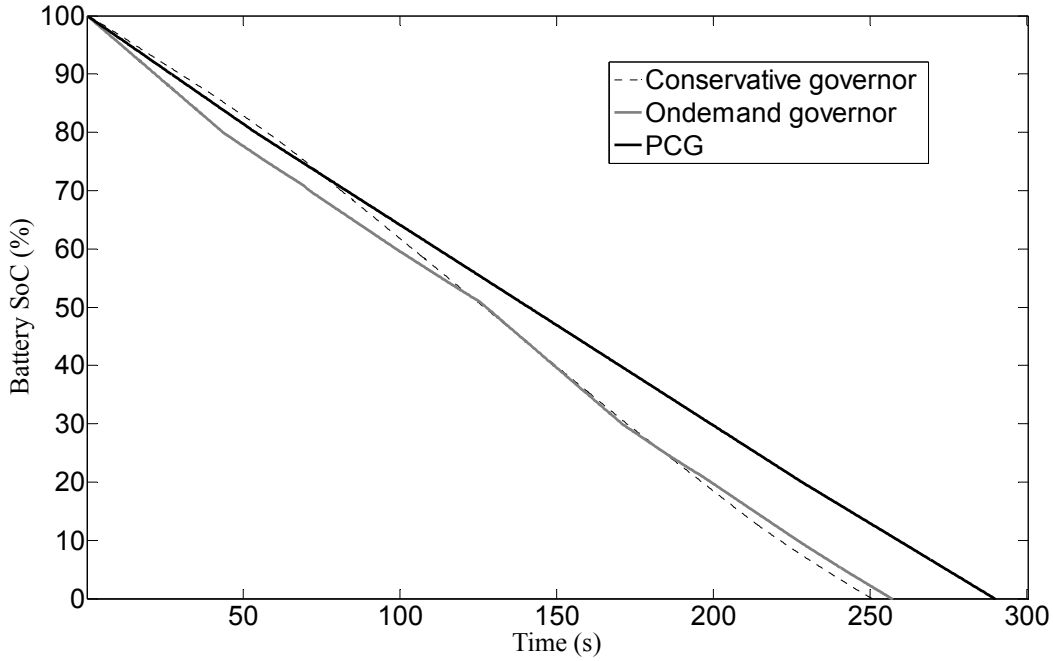


Figure 7-13 Battery lifetime under dynamic governors when decoding sequences of different complexity

7.5 Summary

In this chapter, the two proposed estimators are validated and evaluated, their AAPE have been listed in a table which indicates that the accuracy of OS-level estimator is slightly higher than PAPI-based estimator. Then, the effects on the system behaviour of those classic linear controllers that were designed in Chapter 4 have been checked. The results show the real system consumption matches the simulated response. Besides, the system is stable in all proposed cases and is able to follow the set point in steady state. However, the DVFS nonlinearity might cause undesired and frequent oscillations when there is no OPP that can cancel the loop error. Therefore, TR-I controller, which leads to a smooth and quick enough response, is the best trade-off between settling time and responsiveness, with enough simplicity as to not imply a

significant overhead on the system implementation. Therefore, TR-I controller is selected to be applied in the control system.

After that, a disturbance block is added to the simulator to test the capability of the control system to react to disturbances in its controlled output. This is because changes in the video sequence complexity imply changes in the decoder power consumption, which can be seen as disturbances over the control-system plant. For testing purposes, the disturbance has been considered from two levels of workload: the decoding of a default simpler (low power consumption) video sequence with a periodic swap to the decoding of a more complex (high power consumption) sequence. The corresponding disturbance simulation block switches periodically between two constant levels. One of them is 0, as the default no-disturbance status for the initial reference sequence, and the other one is the change (increase) in the open-loop power consumption estimation while decoding the more complex sequence in the real system. The simulation and implementation results match well.

Finally, PCG was tested with an example profile and compared with other original Linux governors. PCG can dynamically extend the battery lifetime depending on the user requirements and maintain it regardless of the complexity of the video sequences, whereas the Linux dynamic governors vary the battery lifetime depending on the workload. Using the proposed example of power budget profile, PCG can extend the battery lifetime further than with the conservative and ondemand Linux governors, up to a 15.5% and 12.8% of improvement, respectively, when there are variations in the video-processing load.

Chapter 8 Conclusion and future work

The microelectronics industry has been boosting the capabilities of multimedia mobile devices, but the battery, which is the only power source of most mobile devices, is experiencing relatively slow development. Therefore, determining how to optimize the energy consumption of mobile devices under a predefined performance requirement has become a critical issue. The video decoder, as one of the main energy-consuming multimedia applications, is the target application of this work/thesis. This chapter is divided into three sections, the first one summarizes the work of this dissertation and states its contributions; the second section analyses the limitations and depicts the future work that can be explored in a next step. The final section highlights again the contribution of this work to the research of applying control algorithms to energy optimization in multimedia hand-held devices.

8.1 Summary

The user experience of modern mobile systems is greatly affected by the battery lifetime, which should be addressed by energy optimization to provide a strong guarantee on the battery lifetime. In addition, as a popular multimedia task, video decoding is one of the main energy-consuming applications. Therefore, this thesis proposes a power-control system that can effectively save energy and extend the battery lifetime while maintaining a reasonable QoE. This power-control system consists of a closed-loop subsystem and a PCG. In the forward part of the closed loop, the common DVFS mechanism is selected to act on the power consumption, whereas in the feedback part, the use of a PMC-based power estimation method is proposed instead of using specific power/energy sensors. This design makes our proposed solution highly applicable because both DVFS and PMCs are widely available in common consumer-electronic mobile devices. Two approaches to estimate the power consumption have been considered: PAPI-based estimations (*1st approach*) and OS-level estimations (*2nd approach*). PAPI is a widely used third-party tool which can easily access PMCs from the application level. Besides, the interface of PAPI is the same for all platforms. Therefore, in this work PAPI is used to monitor PMCs for preliminary tests, because it is a more generally and easily applicable tool than the more specific OS-based solution. Comparing the two approaches, however, the OS-level estimator has some other advantages: it can run without interfering with the user-level decoding

application; related to this, whereas PAPI-based estimations have to be calculated on a video-frame time basis, the sampling frequency of the OS-level estimator can be freely fixed; furthermore, the OS-level infrastructure avoids the need of user signals to the OS for DVFS commands. Besides, the OS-level average estimation error is only 0.15%, even slightly lower than that of the PAPI-based estimator. For these reasons, OS-level estimator is used into the control system as the feedback unit of the real-time closed-loop control subsystem, which is implemented to regulate the power consumption of OS-based multimedia mobile devices. Besides, comparing with the work of Ren [74], whose energy estimation model has an AAPE about 5% and can only be applied with one fixed OPP, the OS-based estimator proposed in this dissertation has less AAPE, about 2.46%, while it can be applied to all OPPs.

The proposed simplified linear system model is general and simple enough as to be applicable to most common multimedia mobile platforms. To validate the proposal, a commercial low-cost open-source software application platform has been used to implement the video decoder and power-control system for the set of designed controllers. As a first approach, classic linear controllers have been tested and the result obtained from them has been considered good enough as to avoid the need to explore other more complex ones. Also a simulator including the OPP-based nonlinear DVFS interface has been used to validate the control system.

The results show, on one hand, how the real system estimation matches the simulated response. On the other hand, the system is stable for all the proposed controllers and is able to follow the set point in steady state, with clear advantages of I-based controllers over the simplest P controller. Furthermore, the combined PI and PID controllers do not achieve better results than some of the simpler integral controllers. In fact, the BRR-I controller leads to the fastest response, with only one sample of settling interval. However, due to the nonlinearity of DVFS, this responsiveness might cause undesired and frequent oscillations when there is no OPP that can cancel the loop error. Therefore, the best trade-off between settling time and responsiveness is achieved with the TR-I controller, which leads to a smooth and quick enough response. Additionally, a small-capacity battery module is simulated to test the battery lifetime. Disturbances in the decoder power demand have been both generated in the real system and simulated as a disturbance generator block in the simulator. Both real and simulation results indicate that the closed-loop subsystem is able to react to disturbances and regulate video decoding power consumption in steady state with quite short settling times, regardless of the

complexity of the video sequences. These results, along with the low overhead implied by the control system, validate it as a means of keeping the system consumption close to the desired value (set point), regardless of the dynamic consumption demand of the video decoder.

The original Linux dynamic *cpufreq* governors change the processor frequency depending on the current CPU workload, which tends to reduce overall power consumption, regardless of the battery SoC and user requirements. PCG can dynamically extend the battery lifetime depending on the user requirements and maintain it regardless of the complexity of the video sequences, whereas the Linux dynamic governors vary the battery lifetime depending only on that decoding complexity. Using a proposed example of power budget profile, PCG can extend the battery lifetime 15.5% more than the conservative Linux governor and 12.8% more than the ondemand Linux governor when the decoding workload is variable.

Therefore, with respect to other energy-saving algorithms, the algorithm proposed in this dissertation has a number of advantages. Firstly, it relies on a PMC-based power-consumption estimator that can accurately feed the current power consumption back in real time, which is applicable even to many consumer devices that lack specific power measurement sensors. Secondly, it focuses on one of the main high power-consuming multimedia applications, i.e., the video decoder, and is thus expected to lead to substantial energy savings for multimedia applications. Thirdly, it supports personalized and multiple battery discharge profiles while maintaining a reasonable QoE. Finally, it can guarantee the battery lifetime to avoid unexpected power outages.

8.2 Limitations and future work

Further developments can be addressed, such as the real-time adaptation of the set point to other system variables like QoE, performance parameters or video-decoding complexity. This will pave the way to the comparison of the proposed approach with other kinds of energy saving algorithms. Besides, other types of control techniques (predictive, adaptive, robust, fuzzy, etc.), not necessarily linear, can be applied to evaluate if the increase in complexity compensates the possible improvement of results. In addition, a cooperation with other advanced control methodologies could be set, for example, an adaptive predictive expert control methodology using adaptive predictive control (APC) to predict the process evolution and preempt potential deviations from the desired set-point could be applied [78]. Moreover, the OS-level estimator can be more universally used, which can contribute to estimate the power consumption of more

types of applications, not only video decoders. Besides, the governor can include more functionality such as thermal management, in such a way that the set-point could also consider the temperature of the hardware or certain component. Other further work can lead to extend the proposal to more complex and realistic scenarios in which, for example, both the CPU and GPU offer more than one processor and the software is partitioned among different processing cores. Then, the closed-loop control of power consumption of several CPU cores or coprocessors in a platform will be worth researching.

8.3 Final words

This dissertation presents control algorithms for power regulation under the limited battery capacity of multimedia hand-held devices while executing a video decoder application and maintaining a reasonable quality of user experience. The proposed control system, which includes a real-time closed-loop control subsystem and a power-control governor (PCG), has been implemented in the operating system of a low-cost development board to validate the control algorithms. Instead of using any specific power sensor, a PMC-based estimator is used as the feedback signal in the closed-loop subsystem. After a theoretical system model has been obtained and verified, classic controllers have been implemented in the development board for validation purposes. The control system is able to regulate the system power consumption and the battery discharge rate in the presence of fluctuations in the decoder power-consumption demand. The proposed PCG has better performance in extending the battery lifetime than the conservative and ondemand governors of Linux in the presence of disturbances.

8.4 Publications

International conference papers

[1] Q. Tang, A. M. Groba, E. Juárez, and C. Sanz, "Modeling, analysis and design of a closed-loop power regulation system for multimedia embedded devices," *in Proc. International Conference on Pervasive and Embedded Computing and Communication Systems*, Angers, France, pp. 363-372, Feb. 2015.

In this paper, the plant modeling as well as the theoretical analysis and design and simulation of a closed-loop control system for the power consumption of a hand-held multimedia embedded device are presented. This is a first validation step for a target system in which the power consumption will be regulated based on estimation feedback. Prior to the availability of power estimation data, actual power consumption measurements are used to obtain a mathematical model of the controlled plant. Then, classic control-theory methods are applied to get a closed-loop integral controller able to regulate the power consumption of a video decoder running in an embedded development platform.

[2] Q. Tang, A. M. Groba, E. Juárez, and C. Sanz, "On the estimation-based closed-loop power consumption control in multimedia mobile devices," *in Proc. International Conference on Advances in Multimedia*, Barcelona, Spain, pp. 61-66, Apr. 2015.

In this paper, a closed-loop approach to control the power consumption of multimedia mobile devices is presented, such that the feedback signal is an estimation based on monitored system events.

[3] Q. Tang, A. M. Groba, E. Blázquez, and E. Juárez, "OS-level power consumption estimator for multimedia mobile devices," *in Proc. IEEE International Symposium on Consumer Electronics*, Madrid, Spain, pp. 1-2, June 2015.

In this paper, an OS-level power estimator based on monitored system events for multimedia mobile devices is presented. The OS level power estimator periodically obtains significant-events count values and calculates power-consumption estimations through mathematical models.

[4] Q. Tang, A. M. Groba, E. Juárez, and C. Sanz, "Real-time power consumption control system for multimedia mobile devices," *in Proc. IEEE International Conference on Consumer Electronics*, Las Vegas, USA, pp. 385-386, Jan. 2016.

This paper presents a real-time closed-loop system to regulate the power consumption of multimedia mobile devices. The system feedback is an OS-level power estimator based on monitored events of the target system, i.e., an embedded platform executing a video decoder.

JCR papers

[1] Q. Tang, A. M. Groba, E. Juárez, C. Sanz and F. Pescador, "Real-time power-consumption control system for multimedia mobile devices," IEEE Trans. Consum. Electron, vol. 62, no. 4, pp. 362-370, Nov. 2016.

This paper presents the proposal, implementation and test of a real-time closed-loop control system applicable to the power-consumption regulation of multimedia mobile devices.

[2] Q. Tang, A. M. Groba, E. Juárez, and C. Sanz, "Closed-loop Power-control Governor for Multimedia Mobile Devices," IEEE Trans. Consum. Electron, submitted.

This paper presents control algorithms for power regulation under the limited battery capacities of multimedia hand-held devices while executing a decoder application and maintaining a reasonable quality of user experience. A control system, which includes a real-time closed-loop control subsystem and a power-control governor (PCG), has been implemented in the operating system of a low-cost development board and its results in regulating the battery lifetime are shown.

References

[1] U. Reiter, “Perceived quality in consumer electronics – from quality of service to quality of experience,” 13th IEEE International Symposium on Consumer Electronics, Kyoto, Japan, May. 2009.

[2] 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.

[3] Ericsson Mobility Report. https://www.ericsson.com/res/docs/2016/ericsson-mobility-report-2016.pdf , (Last access: May 2017).

[4] Global video index Q4 2015. http://go.ooyala.com/rs/447-EQK-225/images/Ooyala-Global-Video-Index-Q4-2015.pdf, (Last access: May 2017).

[5] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand, “Overview of the high efficiency video coding (HEVC) standard,” IEEE Trans. Circuits Syst. Video Technol., vol. 22, no. 12, pp. 1649-1668, Dec. 2012.

[6] ITU-T Rec. H.264 & ISO/IEC 14496-10, Advanced video coding for generic audiovisual services, 2012.

[7] T. Coughlin, “A Moore’s law for mobile energy: improving upon conventional batteries and energy sources for mobile devices,” IEEE Consum. Electron. Mag., vol. 4, no. 1, pp. 74-82, 2015

[8] 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 Transactions on Consumer Electronics, Volume. 60, Issue. 2, pp. 267-275, May. 2014.

[9] D. Le and H. Wang, “An effective feedback-driven approach for energy saving in battery powered systems,” Procs. of IEEE Int. Workshop on Quality of Service, 2010, pp. 1-9.

[10] Y. Wang, K. Ma, and X. Wang, “Temperature-constrained power control for chip multiprocessors with online model estimation,” Procs. of Int. Symposium on Computer Architecture, 2009, pp. 314-324.

[11] S. P. Kamat, “Energy management architecture for multimedia applications in battery powered devices,” IEEE Trans. Consum. Electron., vol. 55, no. 2, pp. 763-767, May. 2009.

[12] D. Le and H. Wang, “An effective feedback-driven approach for energy saving in battery powered system,” in Proc. International Workshop on Quality of Service, Beijing, China, pp. 1-9, June 2010.

[13] V. Petrucci, E. V. Carrera, O. Loques, J. C. B. Leite, and D. Mossé, “Optimized management of power and performance for virtualized heterogeneous server clusters,” in Proc. International Symposium on Cluster, Cloud and Grid Computing, Newport Beach, CA, USA, pp. 23-32, May 2011.

[14] 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. Consum. Electron., vol. 60, no. 3, pp. 499-507, Aug. 2014.

[15] E. Juárez, F. Pescador, P. Lobo, A. Groba, and C. Sanz, “Distortion-energy analysis of an OMAP-based H.264/SVC decoder,” in Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Vol. 77: Mobile Multimedia Communications, Springer Berlin Heidelberg, 2012, pp 544-559.

[16] G. Cao, A. Ravindran, S. Kamalasadan, B. Joshi, and A. Mukherjee, “A cross-stack predictive control framework for multimedia applications,” in Proc. International Symposium on Multimedia, Anaheim, CA, USA, pp. 403-404, Dec. 2013.

[17] E. S. Jung, J. Bang, Y. T. Lee and W. Ryu, “Power saving with passive standby mode using bitmap-based activity logs for energy-efficient set-top box,” IEEE Trans. Consum. Electron, vol. 62, no. 1, pp. 62-68, Feb. 2016.

[18] J. Lorch and A. Smith. “Software Strategies for Portable Computer Energy Management,” IEEE Personal Commun., vol. 5, pp. 60–73, June 1998.

[19] L. Benini and G. De Micheli. “Dynamic Power Management: Design Techniques and CAD Tools,” Norwell, MA: Kluwer, 1998.

[20] L. Benini, A. Bogliolo, S. Cavallucci, and B. Riccó. “Monitoring System Activity for OS-directed Dynamic Power Management,” In Proceedings of the International Symposium on Low Power Electronics and Design, pp. 185–190, Aug. 1998.

[21] T. Pering, T. D. Burd, and R. W. Brodersen. “The Simulation and Evaluation of Dynamic Scaling Algorithms,” In Proceedings of the International Symposium on Low Power Electronics and Design, August 1998.

[22] W. Yuan and K. Nahrstedt. “Energy-efficient Soft Real-time CPU Scheduling for Mobile Multimedia Systems,” In Proc. 19th ACM Symp. Operating Systems Principles (SOSP 03), ACM Press, pp. 149–163, 2003.

[23] J. Lorch and A. Smith. “Improving Dynamic Voltage Scaling Algorithms with PACE,” In Proc. of ACM SIGMETRICS Conference, June 2001.

[24] P. Pillai and K. G. Shin. “Real-time Dynamic Voltage Scaling for Low-power Embedded Operating Systems,” In Proc. of 18th Symposium on Operating Systems Principles, Oct. 2001.

[25] P. Mercati, V. Hanumaiah, J. Kulkarni, S. Bloch, and T. Rosing. “BLAST: Battery Lifetime-constrained Adaptation with Selected Target in Mobile Devices,” In Proc. of 12th EAI International Conference on Mobile and Ubiquitous Systems, July 2015.

[26] Y. S. Hwang, S. K. Ku and K. S. Chung, “A predictive dynamic power management technique for embedded mobile devices,” IEEE Trans. Consum. Electron., vol. 56, no. 2, pp. 713-719, May. 2011.

[27] X. Wang, K. Ma, and Y. Wang, “Achieving fair or differentiated cache sharing in power-constrained chip multiprocessors,” in Proc. International Conference on Parallel Processing, San Diego, CA, USA, pp. 1-10, Sept. 2010.

[28] C. Lively, X. Wu, V. Taylor, S. Moore, H. C. , et al., “Power-aware predictive models of hybrid (MPI/OpenMP) scientific applications on multicore system,” Comput. Sci.-Res. & Dev., vol. 27, no. 4, pp. 245-253, Nov. 2012.

[29] Y. Xiao, R. Bhaumik, Z. Yang, M. Siekkinen, P. Savolainen, et al., “A system-level model for runtime power estimation on mobile devices,” in Proc. International Conference on Green Computing and Communications, Hangzhou, China, pp. 27-34, Dec. 2010.

[30] V.Weaver,“perf_eventprogrammingguide”, http://web.eece.maine.edu/~vweaver/projects/perf_events/programming.html, (Last access: May 2017).

[31] Perfmon2, http://perfmon2.sourceforge.net/docs_v4.html, (Last access: May 2017).

[32] R. Neugebauer and D. McAuley, “Energy is just another resource: energy accounting and energy pricing in the Nemesis OS,” In Proceedings of the 8th IEEE Workshop on Hot Topics in Operating Systems, May. 2001.

[33] J Wei, E. Juarez, M. J. Garrido and F. Pescador. " Maximzing the user experience with energy-based fair sharing in battery limited mobile systems", IEEE Trans. Consum. Electro., vol. 59. no.3, pp. 690-698, 2013.

[34] J. Flinn, M. Satyanarayanan, "Managing battery lifetime with energy-aware adaptation," ACM Transactions on Computer Systems (TOCS), v.22 n.2, p.137-179, May.2004.

[35] C. S. Ellis. "The Case for Higher-Level Power Management," In Proceedings of the 7th Workshop on Hot Topics in Operating Systems, Rio Rico, AZ, March.1999.

[36] B. D. Noble, M. Satyanarayanan, D. Narayanan, J. E. Tilton, J. Flinn et al., "Agile application-aware adaptation for mobility,"In Proceedings of the 16th ACM Symposium on Operating Systems and Principles, pages 276–287, Saint-Malo, France, October.1997.

[37] I. Leslie, D. McAuley, R. Black, T. Roscoe, P. Barham, D. Evers, R. Fairbairns, and E. Hyden. "The Design and Implementation of an Operating System to Support Distributed Multimedia Applications," IEEE Journal on Selected Areas In Communications, 14(7): 1280-1297, Sept.1996.

[38] F. Xia, Y. C. Tian, Y. Sun. and, J. Dong. "Control-theoretic dynamic voltage scaling for embedded controllers," IET Computers & Digital Techniques, vol. 2, no. 5, pp. 377-385.

[39] A. S. Ahmadian, M. Hosseingholi, and A. Ejlali, "A control-theoretic energy management for fault-tolerant hard real-time systems," IEEE International Conference on Computer Design, Nov.2010.

[40] C. Poellabauer, L. Singleton, and K. Schwan, " Feedback-based dynamic voltage and frequency scaling for memory-bound real-time applications," 11th IEEE Real Time and Embedded Technology and Applications Symposium , March.2005.

[41] Q. Wu, P. Juang, M. Martonosi, L. S. Peh, and D. W. Clark, " Formal control techniques for power-performance management," IEEE Micro, vol. 25, no. 5, pp. 52-62, 2005.

[42] A. Alimonda, S. Carta, A. Acquaviva, A. Pisano, and L. Benini, "A feedback-based approach to DVFS in data-flow applications," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 28, no. 11, pp. 1691-1704, 2009.

[43] T. Horvath, T. Abdelzaher, K. Skadron, and X. Liu, "Dynamic voltage scaling in multitier web servers with end-to-end delay control," IEEE Transactions on Computers, vol.56, no. 4, pp. 444-458, 2007.

[44] Z. Lu, J. Lach, M. Stan, and K. Skadron, “Design and implementation of an energy efficient multimedia playback system,” Asilomar Conference on Signals, Systems and Computers, Nov.2006.

[45] Y. Zhu, and F. Mueller, “Exploiting synchronous and asynchronous DVS for feedback EDF scheduling on an embedded platform,” ACM Transactions on Embedded Computing Systems, vol. 7, no. 1, pp. 3:1-3:26, 2007.

[46] S. Y. Bang, K. Bang, S. Yoon, and E. Y. Chung, “Run-time adaptive workload estimation for dynamic voltage scaling,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 28, no. 9, pp. 1334-1347, Sept. 2009.

[47] S. Ramakrishnan ; B. T. Krishna, “Fuzzy logic based closed loop adaptive power control for minimization of near-far interference in CDMA systems,” in Proc. Computer Communication and Systems, Chennai, India, pp. 20-21, Feb. 2014.

[48] X. Wang, X. Fu, X. Liu, and Z. Gu, “PAUC: power-aware utilization control in distributed real-time systems,” IEEE Trans. Ind. Informat., vol. 6, no. 3, pp. 302-315, Aug. 2010.

[49] A. K. Mishra, S. Srikantaiah, M. Kandemir, and C. R. Das, “CPM in CMPs: coordinated power management in chip-multiprocessors,” in Proc. International Conference for High Performance Computing, Networking, Storage and Analysis, New Orleans, LA, USA, pp. 1-12, Nov. 2010.

[50] S. Garg, D. Marculescu, and R. Marculescu, “Custom feedback control: enabling truly scalable on-chip power management for MPSoCs,” in Proc. International Symposium on Low-Power Electronics and Design, Austin, TX, USA, pp. 425-430, Aug. 2010.

[51] https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt, (Last access: May 2017).

[52] S. Kim, H. Kim, J. Hwang, J. Lee and E. Seo, “An event-driven power management schem for mobile consumer electronics,” IEEE Trans. Consum. Electron., vol. 59, no. 4, pp. 259-266, Feb. 2013

[53] W. K. Lee, S. W. Lee and W. O. S, “Hybrid model for dynamic power management,” IEEE Trans. Consum. Electron., vol. 55, no. 2, pp. 656-664, 2009.

[54] H. H. Choi, J. R. Lee and D. H. Cho, “On the use of a power-saving mode for mobile VoIP devices and its performance evaluation,” IEEE Trans. Consum. Electron., vol. 55, no. 3, pp. 1537-1545, Aug. 2009.

[55] S. H. Lim, S. W. Lee, B. H. Lee and S. Lee, “Power-aware optimal checkpoint intervals for mobile consumer devices,” IEEE Trans. Consum. Electron., vol. 57, no. 4, pp. 1637-1645, Nov. 2011.

[56] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen and O. Spatscheck, “A Close Examination of Performance and Power Characteristics of 4G LTE Networks,” in Proc. International Conference on mobile systems, applications and services, New York, USA, pp. 225-238, June 2012.

[57] S. Fowler, R. S. Bhamber and A. Mellouk, “Analysis of Adjustable and Fixed DRX Mechanism for Power Saving in LTE/LTE-Advanced,” in Proc. IEEE International Conference on Communications, pp. 1964-1969, June 2012.

[58] C. Herglotz , D. Springer , A. Eichenseer , and A. Kaup, “Modeling the energy consumption of HEVC intra decoding,” in Proc. 20th International Conference on Systems, Signals and Image Processing, Oct. 2013.

[59] E. Monteiro, M. Grellert, S. Bampi, B. Zatt, “Energy-aware cache assessment of HEVC decoding,” in Proc. IEEE International Symposium on Circuits and Systems, May 2016.

[60] 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.

[61] 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.

[62] T. Mallikarachchi , H. K. Arachchi , D. S. Talagala and A. Fernando, “CTU level decoder energy consumption modelling for decoder energy-aware HEVC encoding” in Proc. IEEE International Conference on CE, Jan. 2016.

[63] Ogata, K., 2010. Modern control engineering, Prentice Hall, 5th edition.

[64] Phillips, C. L. and Parr, J. M., 2010. Feedback Control Systems, Prentice Hall, 5th edition.

[65] Franklin, G. F. and Powell, J. D., 1997. Digital Control of Dynamic Systems. Addison Wesley, 3rd edition.

[66] BeagleBoard System Reference Manual Rev. C4, December 2009.

[67] USER'S GUIDE, Agilent Technologies, Model 66319B/D, 66321B/D, Mobile Communications DC Source. http://cp.literature.agilent.com/litweb/pdf/5964-8184.pdf, (Last access: May 2017).

[68] ISO/IEC 14496-2 (MPEG-4), "Information Technology-Coding of Audio-Visual Objects-Part 2: Visual, 2002

[69] MPEG-4 Visual, http://mpeg.chiariglione.org/standards/mpeg-4/video

[70] 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.

[71] Simple DirectMedia Layer, http://wiki.libsdl.org/FrontPage, (Last access: May 2017).

[72] Cross Platform Make, http://www.cmake.org/overview/, (Last access: May 2017).

[73] Orcc,http://orcc.sourceforge.net/getting-started/install-orcc/, (Last access: May 2017).

[74] R. Ren, "Energy/Power Consumption Model for an Embedded Processor Board", Máster en Ingeniería de Sistemas y Servicios para la Sociedad de la Información, Trabajo Fin de Máster, ETSIST-UPM, 2012.

[75] H. Jin, R. Hood, J. Chang, J. Djomehri, D. Jespersen, K. Taylor, R. Biswas, and P. Mehrotra, "Characterizing application performance sensitivity to resource contention in multicore architectures," Technical Report NAS-09-002, NASA Ames Research Center, 2009.

[76] RVC-CAL-Sequences. Available: http://sourceforge.net/projects/orcc/files/Sequences, (Last access: May 2017).

[77] Q. Tang, A. M. Groba, E. Juárez, and C. Sanz, "On the estimation-based closed-loop power consumption control in multimedia mobile devices," in Proc. International Conference on Advances in Multimedia, Barcelona, Spain, pp. 61-66, Apr. 2015.

[78] J.M. Martín Sánchez, J. Rodellar, "ADEX Optimized Adaptive Controllers and Systems," Springer International Publishing, ISBN 978319097930, 2015.

[79] V. Pallipadi, A. Starikovskiy, "The ondemand governor: past, present, and future", Intel open source technology center, Linux Symposium., vol. 2, pp. 215-230, July, 2006.

[80] L. Brown, A. Keshavamurthy, D. S. Li, R. Moore, V. Pallipadi, L. Yu, “ACPI in Linux Architecture, advances, and challenges”, Intel open source technology center, Linux Symposium., vol. 1, pp. 51-67, July, 2005.