Performance and Energy Consumption Characterization and Modeling of Video Decoding on Multi-core Heterogenous SoC and their Applications

Yahia Benmoussa

THESE / UNIVERSITE DE BRETAGNE OCCIDENTALE
sous le sceau de l'Université Européenne de Bretagne
pour obtenir le titre de
DOCTEUR DE L'UNIVERSITE DE BRETAGNE OCCIDENTALE
Mention : Informatique
École doctorale SICMA

Présentée par
Yahia BENMOUSSA

Préparée au Lab-STICC - Laboratoire des Sciences et Techniques de l'Information, de la Communication et de la Connaissance. Brest/Lorient

Performance and Energy Consumption Characterization and Modeling of Video Decoding on Multi-core Heterogenous SoC and their Applications

Thèse soutenue le 16 Juin 2015
devant le jury composé de :

Frank Singhoff
Professeur, UBO Brest / Président du jury

Daniel Menard
Maitre de conférences, HDR, Université de Rennes 1 / Rapporteur

Daniel Chillet
Maitre de conférences, HDR, Enssat / Rapporteur

Smail Niar
Professeur, Université de Valencienne / Examnateur

Mouloud Koudil
Professeur, École Supérieure d'Informatique/ Examinateur

Eric SENN
Maitre de conférences, HDR, UBS Lorient / Directeur de thèse

Jalil Boukhobza
Maitre de conférences, UBO Brest / co-encadrant

Djamel Benazzouz
Professeur, Université de Boumerdes / Co-directeur de thèse

To my mother, my father and all my family.

Foremost, I would like to express my gratitude to Jalil Boukhobza for his valuable help and feedback during this thesis. My sincere thanks also goes to Eric Senn and Djamel Benazzouz for their supports.

Many thanks to my thesis committee for accepting to evaluate this work.

I must also acknowledge the Lab-STICC, the computer science department of the University of Western Brittany in Brest and the M'Hamed Bougara University of Boumerdes for their financial support.

Special thanks to Yassine Hadjadj-Aoul from University of Rennes for his numerous comments and suggestions, Michael Lanoe for his help in using Open-PEOPLE platform, and Richard Jouet from Laudren Electronics for his technical help in instrumenting embedded boards used in my thesis experimentations.

Finally, I would like to thank to all those who helped me to achieve this work.

Abstract

To meet the increasing complexity of mobile multimedia applications, the System on Chip (SoC) equipping modern mobile devices integrate powerful heterogeneous processing elements among which General Purpose Processors (GPP), Digital Signal Processors (DSP), hardware accelerator are the most common ones.

Due to the ever-growing gap between battery lifetime and hardware/software complexity in addition to application computing power needs, the energy saving issue becomes crucial in the design of such systems. In this context, we propose a study aiming to enhance the understanding of the energy consumption behavior of video decoding on these kinds of systems.

Accordingly, an end-to-end methodology for characterizing and modeling the performance and the energy consumption of video decoding on GPP and DSP is proposed. The characterization step is based on an exhaustive experimental methodology for evaluating, at different abstraction levels, the performance and the energy consumption of video decoding. It was achieved on embedded platforms on which were executed a wide range of video decoding configurations. This step highlighted the importance to consider different parameters which may pertain to different abstraction levels in evaluating the overall energy efficiency of a given system.

The measurements obtained in this step were used to build empirically performance and energy models for video decoding on both GPP and DSP. The proposed models gave very accurate estimation ($R^2 = 97\%$) of both the performance and the energy consumption of video decoding in terms of a rich set of parameters including the video quality and the processor frequency. Moreover, based on a multi-level characterization and sub-model decomposition approaches, we show how the developed models, unlike classic empirical models, are easily and rapidly generalizable to other platforms.

Some possible applications using the developed models, in the context of adaptive video decoding, were proposed. In general, it consists to use the capability of the proposed performance model to predict the decoding time of a given video quality in dimensioning/scheduling the processing resources.

Due to the increasing demand on High Definition (HD), the characterization methodology was extended to consider HD video decoding on both parallel multi-cores and hardware video accelerator. This part highlighted the potential of parallelism video decoding to increase the energy efficiency of video decoding and point out some open issues in this domain.

Résumé

Pour répondre à la complexité croissante des applications multimédia mobiles, les systèmes sur puce équipant les appareils mobiles modernes intègrent des unités de calcul puissantes et hétérogène. Parmi ces units de calcul, on peut trouver des processeurs à usage général, des processeur de traitement de signal et des accélérateurs matériels.

En raison de l'écart toujours croissant entre la durée de vie des batteries et la demande de plus en plus importante en puissance de calcul, l'économie d'énergie devient un enjeu crucial dans la conception des systèmes mobiles. Cette problématique est accentuée par l'augmentation de la complexité des logiciels et architectures matériels utilisés. Dans ce contexte, nous proposons une étude visant à améliorer la compréhension des considérations énergétiques du décodage vidéo sur ce genre de systèmes.

Nous proposerons ainsi une méthodologie pour la caractérisation et la modélisation des performances et de la consommation d'énergie du décodage vidéo, aussi bien sur des processeurs à usage général de type ARM que sur un processeur de traitement de signal. L'étape de caractérisation est basée sur une méthodologie expérimentale pour évaluer de faon exhaustive et à différents niveaux d'abstraction, les performances et la consommation d'énergie du décodage vidéo. Cette caractérisation a été réalisée sur des plates-formes embarquées sur lesquels ont été exécutés un large éventail de configurations du décodage vidéo. Cette étape a souligné l'importance d'examiner différents paramètres qui peuvent se rapporter à différents niveaux d'abstraction dans l'évaluation de l'efficacité énergétique globale d'un système donné.

Les mesures obtenues dans cette étape ont été utilisées pour construire empiriquement des modèles de performance et de consommation d'énergie pour le décodage vidéo à la fois sur des processeurs à usage général type ARM et sur un processeur de traitement de signal. Les modèles proposés peuvent estimer avec une grande précision ($R^2 = 97\%$) la performance et la consommation d'énergie de décodage vidéo en fonction d'un nombre de paramètres comprenant la qualité de la vidéo et la fréquence du processeur. En plus, en se basant sur une caractérisation multi-niveaux et une approches de modélisation par décomposition en sous-modèles, nous montrons comment les modèles développés, contrairement aux modèles empiriques classiques, sont facilement et rapidement généralisables à d'autres plates-formes.

Nous proposerons également certaines applications possibles des modèles développés, dans le cadre du décodage vidéo adaptatif. En général, cela consiste à exploiter la capacité du modèle de performance proposé pour prédire le temps de décodage d'une qualité vidéo donnée afin de mieux dimensionner les ressources de calculs dans un but de réduire leur consommation d'énergie.

En raison de la croissance de l'utilisation des vidéos de haute définition (HD), la méthodologie de caractérisation a été étendu pour considérer le décodage vidéo HD aussi bien sur les architectures multi-coeurs parallèles que sur les accélérateurs vidéo matériels. Cette partie a souligné le potentiel de décodage vidéo parallèle pour augmenter l'efficacité énergétique du décodage vidéo et met en exergue les principaux défis rencontrés dans cet axe.

Contents

List of Figures **xi**

List of Tables **xiv**

List of Abbreviations & Notations **xiv**

List of Publications **xviii**

1 Introduction **1**

1.1 Context 2

1.1.1 Increase in mobile devices power consumption 2

1.1.2 New trends in mobile video applications usage : Implications on power consumption 3

1.1.2.1 Increase in mobile video traffic 3

1.1.2.2 Ubiquitous video applications 4

1.1.3 Complex multimedia mobile devices 5

1.1.3.1 Multi-core and heterogeneous processing 6

1.1.3.2 Sophisticated embedded operating systems 7

1.2 Problem statement : Energy consumption modeling of processor-based video systems 8

1.2.1 Low level modeling 9

1.2.2 High level modeling 10

1.3 Thesis scope and approach 10

1.4 Thesis contributions 11

1.4.1 Experimental methodology 11

1.4.2 Energy characterization methodology 12

1.4.3 Energy modeling methodology 13

1.4.4 Applications 14
1.5 Outline 14

2 Background and related works **16**
2.1 Introduction 18
2.2 Background 18
2.2.1 Concepts on video encoding and decoding 18
2.2.1.1 Principles of MPEG standards 18
2.2.1.2 Video quality assessment metrics 21
2.2.1.3 Video playback QoS assessment metrics 21
Deadline miss rate (DMR) 22
Decoded frames per second (FPS) 22
Latency 22
2.2.2 Energy consumption in electronic circuits 23
2.2.2.1 Static *vs* dynamic energy consumption 23
2.2.2.2 Dynamic voltage and frequency scaling (DVFS) 24
2.2.2.3 Dynamic Power management (DPM) 24
2.2.3 Discussion 25
2.3 Principles of energy saving in video decoding 25
2.3.1 Frequency scaling: Performance vs energy consumption 27
2.3.1.1 Frame-by-frame based DVFS 27
2.3.1.2 Average workload based DVFS 27
2.3.1.3 Video-aware DVFS : Challenges and issues 28
2.3.2 Parallel multi-core video decoding 29
2.3.3 Specialized processing 30
2.3.3.1 Hardware video codecs 30
2.3.3.2 Graphical processing unit 31
2.3.3.3 Digital signal processor 33
2.3.4 Discussion 33
2.4 Performances and energy consumption characterization of video decoding 34
2.4.1 Video decoding performances characterization 34
2.4.1.1 Application level 34
2.4.1.2 System level 35

2.4.1.3 Architecture level 35
2.4.2 Video decoding energy consumption characterization 36
2.4.2.1 Application level 36
2.4.2.2 System level 37
2.4.2.3 Architecture level 37
2.4.3 Discussion 38
2.5 Performances and energy consumption modeling of video decoding . . . 40
2.5.1 Video decoding performances modeling 40
2.5.1.1 Frame based models 40
Empirical models 40
Statistical models 41
Metadata-based models 42
2.5.1.2 Interval-based models 42
2.5.1.3 Memory-aware performance models 44
2.5.2 Video decoding energy consumption modeling 45
2.5.2.1 Application level 45
2.5.2.2 System level 46
2.5.2.3 Architecture level 47
2.5.3 Discussion 47
2.6 Conclusions 49

3 Methodology **50**
3.1 Introduction 51
3.2 Characterization methodology 52
3.2.1 Video complexity characterization 53
3.2.2 Operating-system level characterization 54
3.2.3 Video-frame level characterization 56
3.2.4 Video sequence level characterization 56
3.3 Modeling methodology 57
3.3.1 Video rate Sub-model 58
3.3.2 Power sub-model 59
3.3.3 Decoding-time sub-model 59
3.3.4 Models validation 60

3.4 Experimental methodology . . . 60
3.4.1 Hardware setup . . . 61
3.4.1.1 MistralEVM3530 . . . 61
3.4.1.2 PandaBoard . . . 62
3.4.2 Power consumption measurement . . . 62
3.4.2.1 Open-PEOPLE platform . . . 62
3.4.2.2 Power consumption measurement methodology . . . 65
3.4.2.3 Boards instrumentation . . . 66
Mistral OMAP3530EVM . . . 66
PandaBoard . . . 67
3.4.3 Software setup . . . 67
3.4.3.1 Video encoder . . . 67
3.4.3.2 Operating system . . . 68
Dynamic power management . . . 69
Frequency scaling . . . 69
3.4.3.3 Video decoder framework . . . 70
Elimination of the I/O interference . . . 71
Overhead calculation . . . 72
3.5 Conclusion . . . 73

4 Performance and Energy Consumption Characterization 74

4.1 Introduction . . . 75
4.2 Video complexity characterization . . . 75
4.3 Video decoding performance and energy characterization . . . 76
4.3.1 Operating-system level . . . 77
4.3.1.1 ARM processor . . . 78
4.3.1.2 DSP processor . . . 80
4.3.1.3 C-states transition overhead . . . 80
Discussion . . . 82
4.3.2 Video-frame level . . . 83
4.3.2.1 Inter-processor communication time overhead . . . 85
4.3.2.2 Inter-processor communication energy overhead . . . 87
4.3.2.3 Discussion . . . 88

4.3.3 Video-sequence level . . . 88
4.3.3.1 Decoding time . . . 88
4.3.3.2 Power consumption . . . 90
4.3.3.3 Energy consumption . . . 92
4.3.3.4 Discussion . . . 92
4.4 Conclusion . . . 94

5 Performance and Energy Consumption Modeling of Video Decoding 96
5.1 Introduction . . . 97
5.2 Video rate sub-model . . . 98
5.2.1 Parameters discussion . . . 99
5.3 Power sub-model . . . 99
5.3.1 Static power sub-model . . . 100
5.3.2 Dynamic power sub-model . . . 100
5.3.2.1 ARM video decoding . . . 100
5.3.2.2 DSP video decoding . . . 100
5.3.2.3 Dynamic power modeling . . . 102
5.3.3 Parameters discussion . . . 102
5.4 Decoding time sub-model . . . 103
5.4.1 Parameters discussion . . . 105
5.5 Energy model . . . 106
5.6 Models validation . . . 106
5.6.1 Models accuracy on OMAP3530 . . . 107
5.6.1.1 Decoding time model . . . 107
5.6.1.2 Energy model . . . 108
5.6.2 Models generalization: OMAP4460 SoC case study . . . 110
5.6.2.1 Decoding time model . . . 110
5.6.2.2 Energy model . . . 111
5.7 Conclusion . . . 113

6 Applications and open issues 114
6.1 Introduction . . . 115
6.2 Energy-aware video decoding in adaptive streaming . . . 115
6.2.1 Motivational example . . . 115

6.2.2 Energy aware scheduling of video decoding on heterogeneous multi-core *SoCs* 116
6.2.2.1 Principles 117
6.2.2.2 Implementation 117
6.2.2.3 Evaluation 119
6.2.3 Video-quality aware DVFS 120
6.2.3.1 Problem description 120
6.2.3.2 Proposed solution 121
6.3 Energy efficiency of high definition video decoding 123
6.3.1 Motivation 123
6.3.2 Experimental evaluation 124
6.3.2.1 Hardware and software setup 124
6.3.2.2 Performance measurement 125
6.3.2.3 Energy consumption measurement 126
6.3.3 Experimental results 126
6.3.3.1 Video decoding performances 126
6.3.3.2 Video decoding energy consumption 128
6.4 Discussion 132
6.4.1 Per-core frequency scaling 132
6.4.2 Processing migration on asymmetric multi-cores 133
6.5 Conclusion 134

7 Conclusions and future works **135**

Bibliography **153**

List of Figures

1.1 SoC consumer portable power consumption trends [1] 2
1.2 Evolution of mobile video traffic 3
1.3 Energy consumption in a mobile device (Video playback) 4
1.4 Ubiquitous wireless video streaming to different mobile devices over diverse wireless access networks 4
1.5 Evolution of possessor frequencies 5
1.6 Heterogeneous cores in mobile SoC 6
1.7 Energy efficiency *vs* processor type [2] 7
1.8 Energy modeling methodologies and levels 10
1.9 Thesis contributions 12

2.1 Structure of a video sequence 19
2.2 Principles of video encoding/decoding 20
2.3 Video decoding performance metrics 22
2.4 Frequency scaling in video decoding 26
2.5 Just-in-time DVFS 27
2.6 Workload averaging DVFS 27
2.7 Energy model convexity 28
2.8 Energy efficiency of parallel video decoding 29
2.9 GPP vs specialized processors energy efficiency 32
2.10 Video decoding complexity *vs* frame size (MPEG2 vs H.264/AVC) [3] . 41
2.11 Impact of memory latency on performance scaling [4] 44

3.1 Overview of the modeling methodology 51
3.2 Characterization and modeling methodology 55
3.3 Mistral EVM3530 Board (left) and PandaBoard ES (right) 62
3.4 A view on Open-PEOPLE rack 63

3.5 Open-PEOPLE platform architecture 64
3.6 Power consumption measurement using a shunt resistor 65
3.7 Sampling rate of the energy consumption measurement 66
3.8 PandaBoard-ES Instrumentation 67
3.9 Used video test sequences 68
3.10 GStreamer DSP video decoding plug-in 70
3.11 GStreamer GPP and DSP video decoding pipes 72

4.1 Characterization methodology 75
4.2 Mapping between the video qp_{avg} and the bit-rates 76
4.3 Cortex A8 power consumption 78
4.4 Power consumption of standby mode 79
4.5 TMS320C64x DSP power consumption 79
4.6 ARM *idle* mode transition latency 80
4.7 ARM standby mode transition latency 81
4.8 DSP standby mode transition latency 81
4.9 OMAP3530 power consumption 82
4.10 ARM and DSP video frames decoding 84
4.11 Profiling result of ARM and DSP video decoding 86
4.12 ARM and DSP video decoding performance 89
4.13 ARM and DSP video decoding power consumption 91
4.14 ARM and DSP video decoding energy consumption 93

5.1 Modeling methodology 97
5.2 Rate model fitting 99
5.3 OMAP3530 static power 100
5.4 ARM and DSP video decoding time in terms of f and qp 103
5.5 Performance scaling behavior 106
5.6 Measured vs predicted video decoding time (OMAP3530) 108
5.7 Measured vs predicted video decoding energy consumption (OMAP3530) 109
5.8 Difference between ARM and DSP energy consumption (OMAP3530) . 109
5.9 Multi-linear regression of the video decoding time (OMAP4460) 110
5.10 Measured vs predicted video decoding time (OMAP4460) 111
5.11 Cortex A9 power consumption model 112

5.12 Measured vs predicted video decoding energy consumption (OMAP4460) 112

6.1 Adaptive streaming 116
6.2 Video-quality and energy aware video decoding on heterogeneous *SoCs* 117
6.3 Embedding video chunks in MP4 file 118
6.4 Dynamic processor switching solution design using GStreamer 119
6.5 Impact of dynamic processor switching on video decoding power consumption 120
6.6 Video-quality aware DVFS 120
6.7 Performance and energy consumption models building 122
6.8 i.MX6 SoC power domains 125
6.9 Performance of HD video decoding (i.MX6) 127
6.10 Processor usage of HD video decoding 127
6.11 Energy consumption of HD video decoding (i.MX6 SoC) 128
6.12 VPU HD Video decoding power consumption 129
6.13 Parallel multi-core HD video decoding energy consumption 130
6.14 Power consumption of little and big core in Exynos 5422 SoC 133

List of Tables

2.1 Video frame complexities 26
2.2 Summary of studies on performance and energy characterization of video decoding 39
2.3 Summary of studies on performance and energy modeling of video decoding 48

3.1 Mobile devices using OMAP SoCs 61
3.2 Power measurement materiel 63
3.3 Hardware and software setup summary 71

4.1 Mapping between *qp* and the bit-rate 77
4.2 Summary of OMAP3530 measured power consumption 82
4.3 ARM and DSP decoding time overhead 85
4.4 Results of video decoding profiling 86
4.5 ARM and DSP decoding energy overhead 87
4.6 Energy efficiency *vs* Processor type *vs* video quality 94

5.1 Model fitting results of the bit-rate model 98
5.2 Model fitting results of the dynamic power model 101
5.3 Multi-linear regression of 1/t in terms of f and *qp* 104
5.4 Summary of the model constant parameters 107
5.5 Performance model R^2 108
5.6 Energy model R^2 108

6.1 HD video decoding performances (fps) (i.MX6) 126
6.2 HD video decoding energy consumption (mJ/Frame) (i.MX6) 128

List of Abbreviations & Notations

Abbreviations

ARM	Advanced RISC Machines
ASIC	Application Specific Integrated Circuit
CBR	Constant Bit Rate
CISC	Complex Instruction Set Computing
CMOS	Complementary Metal-Oxide Semiconductor
DASH	Dynamic Adaptive Streaming over HTTP
DCT	Discrete cosine transform
DMA	Direct Memory Access
DMR	Deadline Miss Rate
DPM	Dynamic Power Management
DSP	Digital Signal Processor
DVFS	Dynamic Voltage and Frequency Scaling
EOS	Embedded Operating System
ES	Embedded System
FPS	Frames Per Second
GPP	General Purpose Processor
GPU	Graphical Processing unit
HD	High Definition
HEVC	High Efficiency Video Coding
IDCT	Inverse Discrete cosine transform
IPC	Inter Process Communication
IPTV	Internet Protocol television
MB	Macroblock
MPEG	Moving Picture Experts Group

OMAP	Open Multimedia Applications Platform
OS	Operating System
PSNR	Peak Signal-to-Noise Ratio
QoS	Quality of Service
qp	Quantization parameter
RISC	Reduced Instruction Set Computing
SD	Standard Definition
SIMD	Single Instruction Multiple Data
SoC	System on Chip
VBR	Variable Bit Rate
VPU	Video Processing Unit
WFI	Wait For Interrupt

Notations

a	Rate model exponent
C_{eff}	Effective capacitance
E_{dyn}	Dynamic energy
E_{static}	Static energy
E, E_{tot}	Total energy (static + dynamic)
$E(\ldots)$	Energy model
f_{arm}	Frequency of the ARM processor
f_{dsp}	Frequency of the DSP processor
f	Processor frequency
I	Current intensity
P_{dyn}	Dynamic power
P_{idle}	*Idle* state power
P_{static}	Static power
P, P_{tot}	Total power (static + dynamic)
$P(\ldots)$	Power model
q_{min}	Minimum step-size
qp_{avg}	Average quantization parameter
qp	Quantization parameter
q	Step-size

$Q(\ldots)$	Rate model
$R_{display}$	Video display rate
r_{max}	Maximum bit-rate
R_{shunt}	Shunt resistor
r	Bite-rate
t	Time
$T(\ldots)$	Time model
V, v_{dd}	Voltage
V_{shunt}	Voltage across a shunt resistor

List of publications

Journals

- **Y. Benmoussa**, J. Boukhobza, E. Senn, Y. Hadjadj-Aoul, D. Benazzouz. A Methodology for Performance/Energy Consumption Characterization and Modeling of Video Decoding on Heterogeneous SoC and its Applications. Journal of Systems Architecture. Volume 61, issue 1. January 2015.
- **Y. Benmoussa**, J. Boukhobza, E. Senn, D. Benazzouz. On the Energy Efficiency of Parallel Multi-core vs Hardware Accelerated HD Video Decoding. ACM SIGBED Review, Volume 11 Issue 4 (Special issue on Ewili'14 Workshop). December 2014.
- **Y. Benmoussa**, J. Boukhobza, E. Senn, D. Benazzouz, Y. Hadjadj-Aoul. DyPS : Dynamic Processor Switching for Energy-Aware Video Decoding on Multi-core SoCs. ACM SIGBED Review, Volume 11 Issue 1 (Special issue on Ewili'13 Workshop). February 2014.

International conferences

- **Y. Benmoussa**, E. Senn, J. Boukhobza, Michael Lanoe, D. Benazzouz. OpenPEOPLE, A Collaborative Platform for Remote & Accurate Measurement And Evaluation of Embedded Systems Power Consumption. MASCOTS : IEEE 22th International Symposium On Modeling, Analysis and Simulation of Computer and Telecommunication Systems. Paris, France. September 2014.
- **Y. Benmoussa**, J. Boukhobza, E. Senn, D. Benazzouz. DSP vs GPP : A Performance/Energy Characterization and Evaluation of Video Decoding. MASCOTS : IEEE 21st International Symposium On Modeling, Analysis and Simulation of Computer and Telecommunication Systems. San Fransisco, USA. August 2013.

- **Y. Benmoussa**, J. Boukhobza, E. Senn, D. Benazzouz. Energy Consumption Modeling of H.264/AVC Video Decoding for GPP and DSP. 16th Euromicro Conference on Digital System Design. Santander, Spain. . September 2013.

National conferences

- **Y. Benmoussa**, J. Boukhobza, E. Senn, D. Benazzouz. Evaluation of the Performance/Energy Overhead in DSP Video Decoding and its Implications. Colloque du GDR SoC SiP. Lyon, France. June 2013.

- **Y. Benmoussa**, J. Boukhobza, L. Lagadec, D. Benazzouz, Y. Hadjadj-Aoul. Behavioral System Level Power Consumption Modeling of Mobile Video Streaming Applications. Colloque du GDR SoC SiP. Paris, France. June 2012.

CHAPTER 1

Introduction

Contents

1.1 Context 2

1.1.1 Increase in mobile devices power consumption 2

1.1.2 New trends in mobile video applications usage : Implications on power consumption 3

1.1.3 Complex multimedia mobile devices 5

1.2 Problem statement : Energy consumption modeling of processor-based video systems 8

1.2.1 Low level modeling 9

1.2.2 High level modeling 10

1.3 Thesis scope and approach 10

1.4 Thesis contributions 11

1.4.1 Experimental methodology 11

1.4.2 Energy characterization methodology 12

1.4.3 Energy modeling methodology 13

1.4.4 Applications 14

1.5 Outline 14

This thesis addresses the important issue of the energy consumption of video decoding on low power mobile System on Chip (SoC). It aims to enhance the understanding of the energy saving consideration and implications of video decoding on modern SoC equipping mobile devices.

In this chapter, we present the context which has motivated us to address this issue. Then, we define the scope we are targeting and the position of our work compared to other studies. Finally, we point out various important results obtained in this thesis.

1.1 Context

1.1.1 Increase in mobile devices power consumption

Mobile devices such as smart-phones and tablets are more and more used in everyday life. One of the most important issue faced by the hardware and software designers of these mobile devices is the drastic increase of their energy consumption.

In fact, the increase in mobile applications usage combined with the explosion of the power consumption of the hardware make the energy consumption issue very critical. As illustrated in Fig. 1.1, the International Technology Road-map for Semiconductors (ITRS) forecasts that the power consumption of the SoC equipping mobile devices will increase with a factor of 2.5 during the next decade [1]. In a context where lithium battery technologies are not evolving fast enough to absorb the ever-growing energy requirements of such mobile architectures [7], the autonomy of the mobile devices may be drastically impacted.

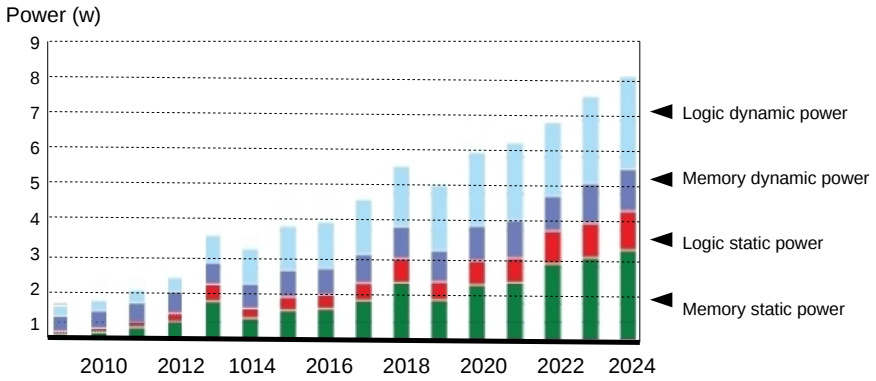

Figure 1.1: SoC consumer portable power consumption trends [1]

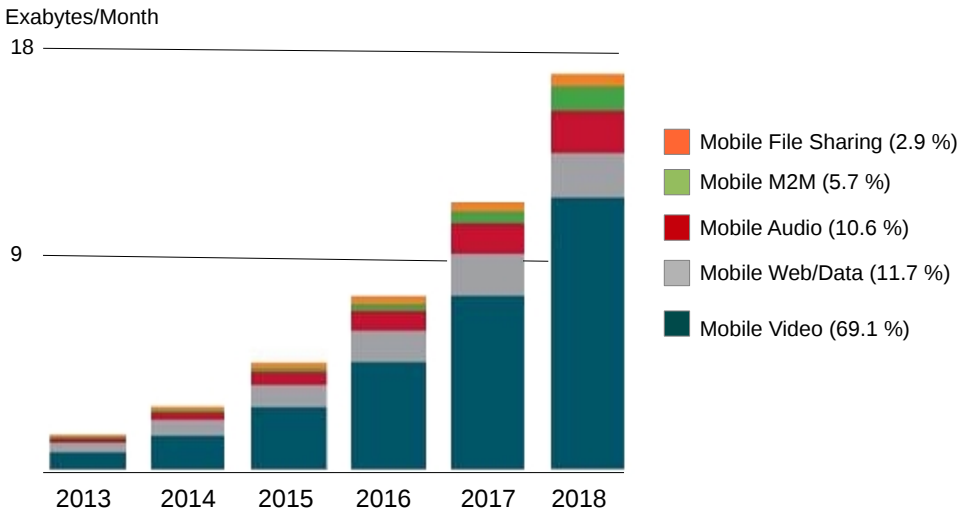

Figure 1.2: Evolution of mobile video traffic

1.1.2 New trends in mobile video applications usage : Implications on power consumption

The following section presents the new trends in mobile video application usage and their relation with the energy consumption issue.

1.1.2.1 Increase in mobile video traffic

One of the most popular applications running on mobile devices is video playback. This is due to the growing use of video-sharing platforms (e.g. YouTube, Netflix, Dailymotion), social networks (e.g. Facebook, Twitter), mobile IPTV and video-conferencing. As illustrated in Fig. 1.2, it is expected that the video data will represent 70% of the overall Internet mobile traffic in the next few years [8]. Moreover, according to a recent study [9] achieved on 200 millions of mobile users, the average video watching time is 52 minutes per day.

This new trend in using video content further accentuates the energy consumption issue. In fact, modern video codecs use more and more complex and aggressive compression algorithms to fit the ever growing demand on video. While they allow to achieve high compression ratios, they increase the demand on processing resources and thus on the energy consumption at the decoder side. For example, according to [10, 11], the processing resources are responsible of more than 60% of the power consumption during the video playback of a H.264/AVC video as shown in Fig. 1.3.

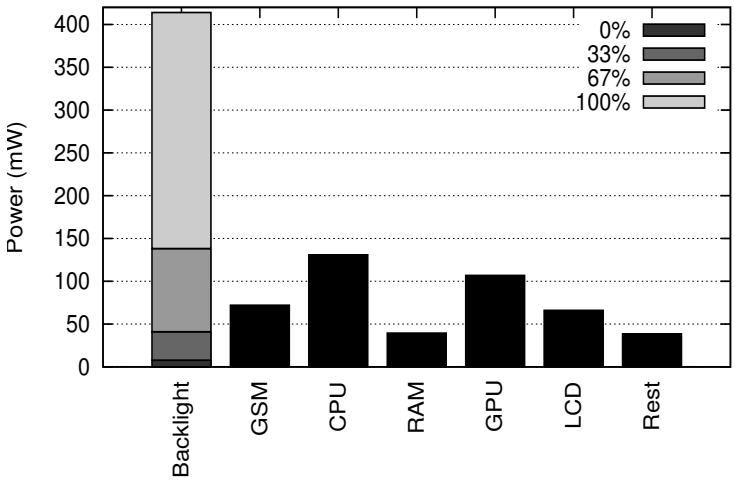

Figure 1.3: Energy consumption in a mobile device (Video playback)

1.1.2.2 Ubiquitous video applications

The increase in mobile traffic highlighted in the previous section is boosted by the ubiquitous video applications. These applications are executed in a heterogeneous environment. As illustrated in Fig. 1.4, the video content may be accessed from various mobile devices with different processing and displaying video capabilities. Moreover, the network technologies for transporting the video content may have different bandwidth capacities which range from tens of Kbits to tens of Mbits per second.

To cope with these different capabilities of the mobile devices, the most important video content providers (ex. Youtube and Netflix) support the dynamic quality adap-

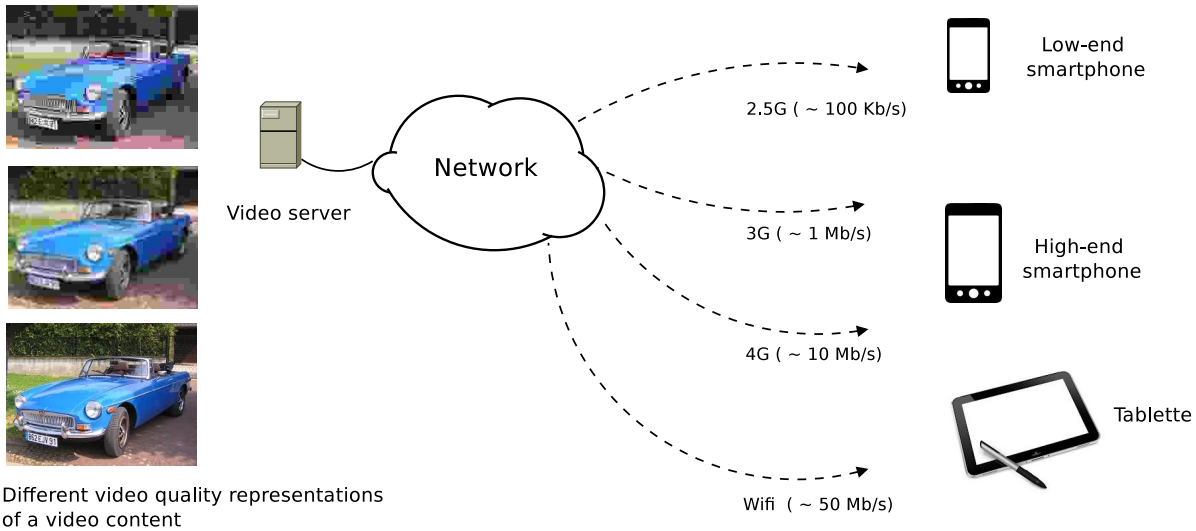

Figure 1.4: Ubiquitous wireless video streaming to different mobile devices over diverse wireless access networks

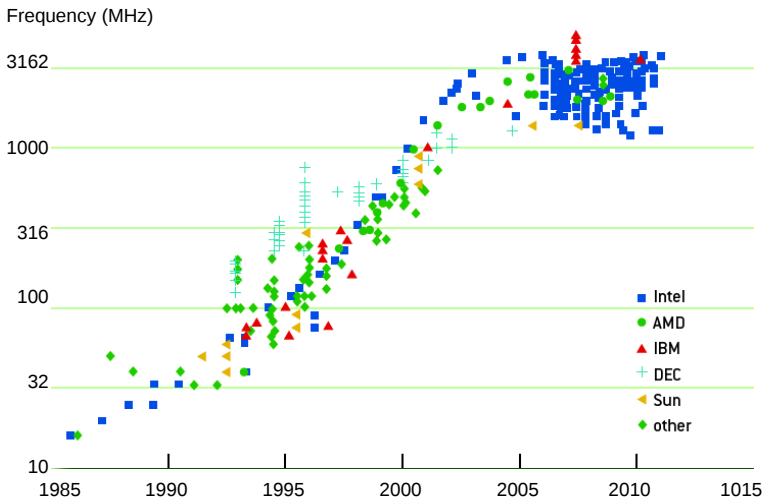

Figure 1.5: Evolution of possessor frequencies

tation of video decoding [12], a technique allowing the video decoder to adjust the video quality at run time. The video quality is no longer a fixed parameter defined statically by the video content provider. It is up to the video decoder to select it depending on its capabilities.

In addition to network bandwidth and displaying capabilities of the mobile device, the energy budget constraint start to be considered as one of the criteria determining the video quality to retrieve from the network [13]. For example, the video decoder may consider the remaining energy budget (battery level) when selecting the video playback quality to increase its autonomy. Switching to a lower video quality playback may allow thus to extend the autonomy.

1.1.3 Complex multimedia mobile devices

The above discussed new trends of mobile users make the mobile device manufacturers competing for providing products designed toward multimedia applications and energy efficiency. In fact, multimedia capabilities available on modern mobile devices are close to those of personal computers while consuming much less energy. To make this possible, modern smart-phones and tablets integrates sophisticated embedded systems (ES). These ES includes multi-cores and heterogeneous processing resources running complex embedded operating systems (EOS).

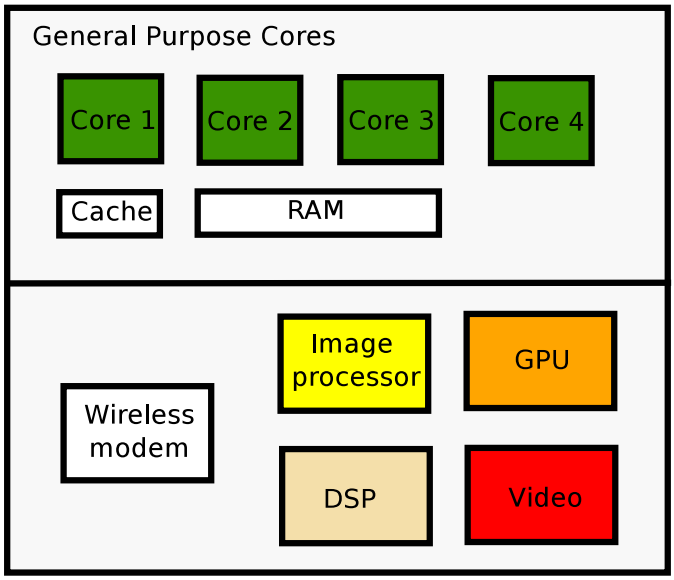

Figure 1.6: Heterogeneous cores in mobile SoC

1.1.3.1 Multi-core and heterogeneous processing

The processing capabilities of embedded processors equipping mobile devices have been growing considerably. The clock frequency of current mobile processors exceeds 1 GHz in almost all standard mobile devices. However, the performance of microprocessors tends to stall due to power and frequency wall limitation [14], it is no longer possible to continuously increase processor frequencies. This can be illustrated in Fig. 1.5 showing the processor frequency stall starting from the beginning of last decade [14][1].

To continue to scale performance without drastic power dissipation, manufacturers began to include more processor cores within a System on Chip (SoC). These cores may be general purpose processors (GPP) or specialized processing units. Figure 1.6 shows the components of a typical SoC equipping a modern mobile device. It includes multiple GPP cores in addition to Digital Signal Processor (DSP), a Graphical Processing Unit (GPU) and Application Specific Integrated Circuit (ASIC).

In general, the more a processor is specialized, the more it is energy efficient. Indeed, the use of parallelism in these processors in addition to optimized execution flows increase their performance without requiring higher voltages and frequencies [15]. This makes them an energy-efficient choice in energy constrained devices [16, 17]. This is illustrated in Fig. 1.7, showing the energy consumption (Million of OPeration per

[1]The data are extracted from the CPUDB project (http://cpudb.stanford.edu) maintaining a database of hundreds of processor characteristics. The project is mainly focusing on CISC processors; however, the trends are the same for RISC processors equipping mobile devices

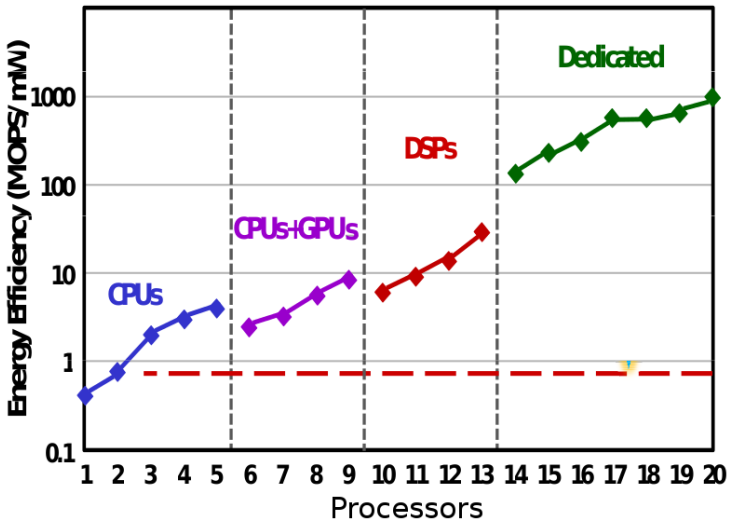

Figure 1.7: Energy efficiency *vs* processor type [2]

Second / mW) of different type of processors including CPU (GPP), GPU, DSP and dedicated circuits (ASIC).

Video decoding application can be implemented on GPP, DSP, GPU, or hardware accelerated video codecs. Each of these processing resources has advantages an drawbacks. For example, hardware video codecs are very energy-efficient [2], however they are not flexible and require a long time to market for new video standards chips design. On the other hand, GPP are not energy efficient while they are easy to program. GPU and DSP provide a balance between energy efficiency and flexibility.

Thus, within a single hardware platform, several heterogeneous processing configuration choices are available including GPP, DSP and ASIC. Each of these elements has different processing capabilities and energy consumption levels.

1.1.3.2 Sophisticated embedded operating systems

The increasing complexity of mobile device architectures imposes the use of sophisticated embedded operating systems (EOS) comparable in complexity and functionality to desktop or server ones. In fact, they provide an abstraction mechanism for sharing and managing hardware resources such as processors, storage, multimedia devices and implement almost all standard OS functionalities such as process scheduling, memory management and Input/Output (I/O) support. This central role of the OS in managing mobile devices makes it a very important component to consider when analyzing the energy consumption properties.

From the power consumption viewpoint, the OS is both a source of energy con-

sumption and an energy saving enabler. Indeed, like all the applications running on a mobile device, OS tasks use some part of processing resources and thus contribute in consuming the energy budget [18]. On the other hand, the OS is the component which has the best knowledge of hardware resources utilization which makes it ideal for implementing energy saving policies.

For example, in case of a video application, the OS may be highly involved in the video decoding process to manage the I/O with external video specialized processor (hardware video codec or DSP) or to schedule the decoding over multiple processor cores. These tasks are sources of additional energy consumption. On the other hand, the OS is able to save energy by *idling* the processors or lowering its frequency during low activity periods in the video decoding process.

1.2 Problem statement : Energy consumption modeling of processor-based video systems

In general, an energy model allows understanding and predicting the energy consumption in terms of well identified factors or parameters. Understanding how much a given parameter impacts the energy consumption can help in tuning it to reduce the consumed energy. This is especially true when the effects of interactions between the different parameters are understood. On the other hand, the prediction of the amount of consumed energy is extremely useful to dimension the energy budget and estimate the autonomy.

As highlighted previously, the processing resources are a major source of energy consumption in the context of video decoding applications. However, in case of complex processor-based system, well understanding the energy consumption properties should consider in addition to the processing resources, the executed operating system and applications. For example, the above sections show that the energy consumption considerations of video applications are present across different mobile device components.

Ideally, an energy model for the above described systems should consider all the relevant parameters and should estimate accordingly the consumed energy accurately. However, in practice, this is hard to achieve for complex systems and these objectives may be fulfilled partially. Actually, realistic energy consumption models for complex system can consider only a subset of parameters. On the other hand, they may induce

some errors in their predictions as compared to real energy consumption values. The questions which can raise is how to select the most relevant parameters and how to make the developed model as accurate as possible?

Answering these questions depend mainly on the considered abstraction level of the targeted system. As we will discuss hereafter, there exist two main approaches: 1) Low level modeling and 2) High level modeling.

1.2.1 Low level modeling

The energy models at the lower levels (ex. Register Transfer Level (RTL), cycle, Instruction) can provide very accurate estimation because they are able to represent low level details such as fabrication technology, pipelining, parallelism or memory hierarchy.

At these levels, the energy modeling is achieved at hardware design phase to allow hardware architects to explore the energy efficiency of processor architectures early by testing the impact of different hardware configurations. For this purpose, simulation tools are usually used to estimate the energy consumption of these complex systems including hardware and software.

These simulation tools are very flexible and allow representing a wide range of configuration and parameters at different levels of details. As illustrated in Fig. 1.8, the lower are the represented architecture details, the higher is the accuracy of energy modeling. In general, they are based on architecture simulators which feed low level and fine grained analytical power models with timing information to estimate the overall consumed energy [19].

To model application and/or operating system level parameters on these simulation frameworks, one should execute them on the simulators and estimate accordingly the consumed energy.

The drawback of energy simulators is that they are hard to build and require a very deep knowledge of the targeted microprocessors micro-architecture. Moreover, they are far from representing a complete modern low power SoC. For example, for estimating the energy consumption of video application, there is no energy simulation framework supporting a complete heterogeneous SoC including a GPP, DSP, GPU and hardware video codec.

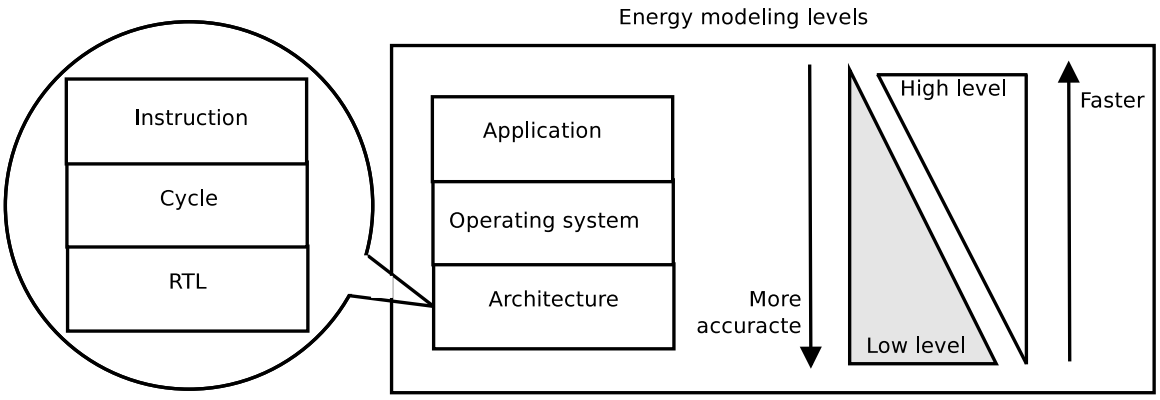

Figure 1.8: Energy modeling methodologies and levels

1.2.2 High level modeling

Models at a higher levels do not rely on detailed microarchitectural knowledge of a particular processor. They may consider the targeted processors as a black-box which sacrifices some accuracy in order to avoid relying on detailed knowledge of the hardware implementation [20].

Since these models do not consider low level details, they may be built rapidly upon real platforms based on coarse grained experimental measurements. This lets the model developer focuses on high level parameters related to the executed applications and/or operating system.

For example, modeling the impact of scaling the processor frequency on the energy consumption of video decoding may be easily derived from energy measurement data achieved on a real platform. However, the use of energy simulator to estimate the impact of frequency scaling is not straightforward and may need a considerable integration effort [21].

The drawback of high level models is that they are decoupled from low level details which limits their generality and portability. In fact, an energy model built for a given hardware is difficult to be generalized to other platforms because there is not explicit mapping between the developed model and low level parameters.

1.3 Thesis scope and approach

In this thesis, we aim to model the energy consumption of video decoding executed on complex embedded systems including embedded operating systems and heteroge-

neous processing elements. Accordingly, we propose a high level modeling approach considering a set of parameters at application and operating system and architecture levels.

At architecture level, we study the impact on the energy consumption of different processing configuration available on heterogeneous SoC. We particularly focus on modeling the energy consumption of video decoding on two widely used microprocessor architectures on mobile SoC : GPP (ARM) and DSP. We explore also the energy efficiency of video decoding using parallel core and hardware accelerated codecs. At operating system level, we focus on studying the impact on the energy efficiency of the processor clock frequency and the inter-processor communication mechanism implemented by the OS for scheduling video decoding task on heterogeneous processors. At application level, we consider the impact of the video quality (bit-rate and resolution) and the scene complexity on the energy consumption of video decoding. The considered video codec is H.264/AVC, a widely used compression video standard.

To model the energy consumption at the considered levels, we chose to use a high level methodology based on extensive experimental power measurement achieved on real embedded platforms. This is motivated by our desire to build fast energy model which represents very accurately real life scenarios.

As discussed previously, high level models may be hard to be generalized to other architecture since they do not consider low level details. However, we believe that one can find the middle ground and achieve a balance between the advantages of high level approaches and simulation based ones. In our opinion, we could make the experimental based energy models more portable using a deep characterization methodology at all the considered levels to map the developed model with comprehensive relevant parameters.

1.4 Thesis contributions

Figure 1.9 illustrates the main steps executed in this thesis. The different proposed contributions we will list below are represented by the dashed-rectangles.

1.4.1 Experimental methodology

An experimental methodology for energy consumption measurement of embedded systems was implemented. Open-PEOPLE (Open-Power and Energy Optimization PLatform and Estimator), a high accuracy power measurement platform, was used to mea-

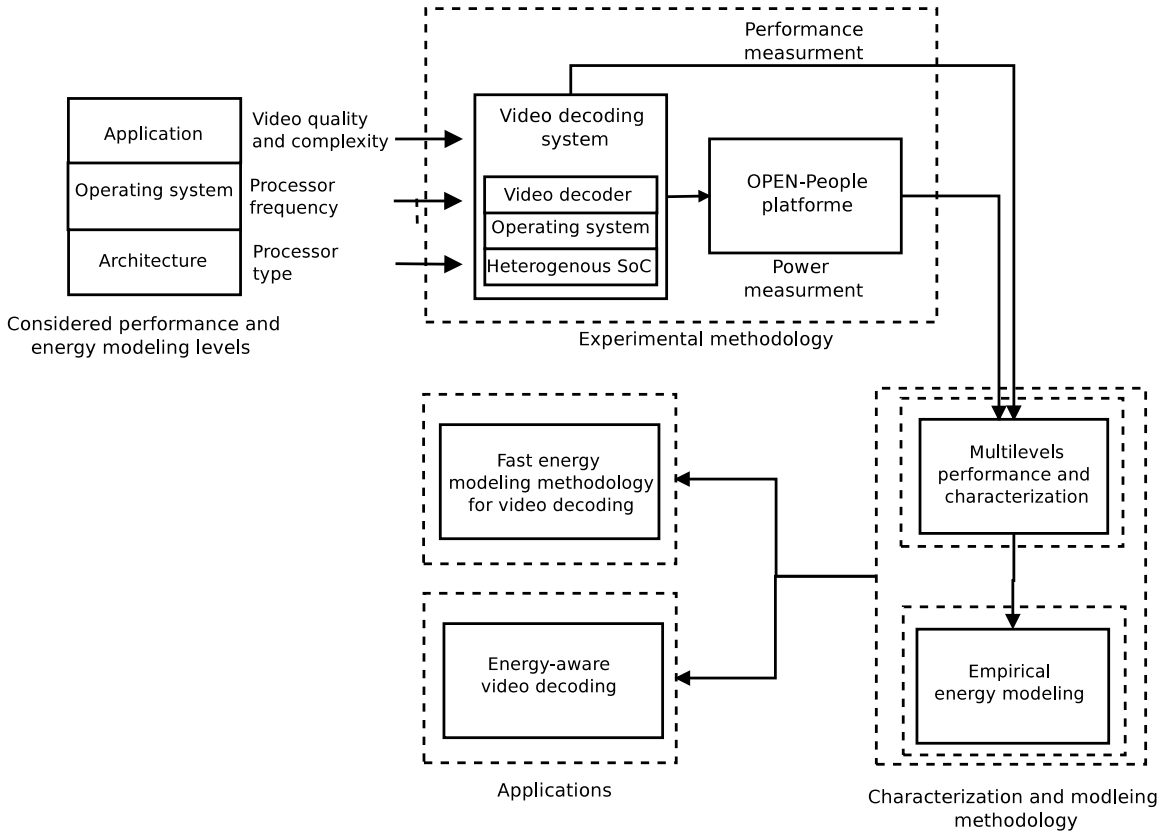

Figure 1.9: Thesis contributions

sure the energy consumption of a set of embedded boards containing SoC similar to those used in mobile devices.

The advantage of the proposed experimental methodology is that it uses a common framework (measurement tools + embedded operating system + video decoder) for evaluating the performance and the energy consumption of video decoding. This allows accurate energy evaluation and objective comparison between the different targeted architectures including GPP, DSP, multi-cores and hardware codecs. This contribution is published in [22, 23].

1.4.2 Energy characterization methodology

A performance and energy characterization of video decoding was achieved based on an extensive experimental measurement methodology. In these experimentations , single core and multi-core ARM processors, DSP and hardware video codecs architectures were considered. On these different processor architectures, the processor frequency and video quality (Standard Definition (SD) and High Definition (HD) quality) parameters were considered.

An extensive characterization achieved on GPP and DSP processors for decoding SD video quality revealed that the performance-energy trade-off highly depends on the decoded video quality and the type of processor architecture. It was highlighted that the scheduling overhead over heterogeneous processor impacts the energy efficiency of video decoding when achieved on external specialized processor. Thus, depending on the video quality, it may be more energy efficient to decode a video on a GPP rather than a DSP. This contribution is published in [22, 5].

On the other hand, a performance and an energy characterization of HD video decoding on various processing configuration including of mono-core, multi-core GPP and hardware codec was achieved. It was shown that parallel HD video decoding on multi-core processors reduces considerably the gap between the energy consumption of hardware accelerated decoder and software-based ones. It is thus an interesting solution achieving a balance between the software flexibility and hardware codec energy efficiency. This contribution is published in [24].

1.4.3 Energy modeling methodology

Based on the results of the performance and the energy characterization of video decoding on GPP (ARM) and DSP, it was proposed:

- A performance analytical model for video decoding which considers both clock frequency and video quality parameters. This model describes also the impact of the off-chip memory access latency on performance variation of video decoding when varying the processor clock frequency. This contribution is published in [25]

- An energy consumption model for video decoding which estimates analytically the consumed energy as a function of the processor clock frequency, the video bit-rate and a set of comprehensive architecture, system and video related parameters. The developed model has a very good energy consumption prediction properties ($R^2 = 97\%$) for the two type of processors. This contribution is published in [25]

- A methodology to generalize and port the proposed energy model to other ARM processor architectures. This contribution is published in [26].

1.4.4 Applications

Based on the conclusions emerged from the characterization and the modeling methodologies, a set of applications are proposed:

- A set of guidelines for online performance and energy models building methodology where the model parameters calculation roles are identified within a video systems including the video encoder, the video decoder and the execution platform. We explain how to use such a model to build a proactive DVFS algorithm for energy aware adaptive video decoding in a context of adaptive video decoding. This contribution is published in [26].

- An energy-aware video decoding scheduling technique on heterogeneous SoC which was implemented on top of a video decoder. This scheduling technique consists in selecting the best energy-efficient processor in the context of video quality adaptive video decoding. This contribution is published in [27].

1.5 Outline

This thesis report is organized as follows:

Chapter 2, presents an overview of the most important state-of-the-art works studying the impact of the processor architectures on energy efficiency of video decoding. Then, the most common approaches and tools for high level energy modeling and estimation of video decoding are presented.

Chapter 3, is dedicated to explain the used methodology. It contains details about the characterization, the modeling steps in addition to a description of the experimental hardware/software setups used to achieve the experiments.

Chapter 4 contains the detailed description of the measured results of the performance and energy consumption characterization methodology of video decoding according to the execution of the above methodology. Conclusion on the energy efficiency of video decoding on GPP/DSP architecture can be found also in this section.

In chapter 5, the previous characterization results are used to build a performance and energy models for video decoding. This chapter includes also a model validation and generalization discussions.

Chapter 6 proposes some applications of the results emerged from the characterization and the modeling phases and discusses some open issues related to parallel and

hardware accelerated High Definition video decoding.

Chapter 7 draws the conclusions on this study and leaves space for discussion on some future works.

CHAPTER 2

Background and related works

Contents

2.1 Introduction 18

2.2 Background 18

2.2.1 Concepts on video encoding and decoding 18

2.2.2 Energy consumption in electronic circuits 23

2.2.3 Discussion 25

2.3 Principles of energy saving in video decoding 25

2.3.1 Frequency scaling: Performance vs energy consumption . . . 27

2.3.2 Parallel multi-core video decoding 29

2.3.3 Specialized processing 30

2.3.4 Discussion 33

2.4 Performances and energy consumption characterization of video decoding 34

2.4.1 Video decoding performances characterization 34

2.4.2 Video decoding energy consumption characterization 36

2.4.3 Discussion 38

2.5 Performances and energy consumption modeling of video decoding 40

2.5.1 Video decoding performances modeling 40

2.5.2 Video decoding energy consumption modeling 45

2.5.3 Discussion 47

2.6 Conclusions 49

2.1 Introduction

This chapter is divided into three parts: first, a background on video decoding and basic concepts on energy consumption are presented. Then, some principles and techniques for saving energy of video decoding are introduced. Finally, the most important studies related to performance and energy characterization and modeling of video decoding are described.

2.2 Background

2.2.1 Concepts on video encoding and decoding

In this section, we describe the principles of video codecs and show how they allow reducing video data size at a cost of a drop in the video quality. Then, we discuss some metrics for evaluating the video quality as well as the quality of service (QoS) of the video playback. Then we introduce from a high level point of view some principles of saving the energy consumption of video decoding. We will show that it is usually a question of balancing the energy efficiency with the video QoS.

2.2.1.1 Principles of MPEG standards

MPEG Video coding standards are a set of techniques for compressing video data to ease their transportation and storage. All of these standards (MPEG2, H.264/AVC, HEVC) make use of temporal (inter-frame) and spatial (intra-frame) redundancy to compress video data. Hereafter, we describe a general concepts shared by all MPEG video standards.

An MPEG video sequence is composed of a set of frames (see Fig. 2.1). Each frame may contain several slices and each slice contains several macroblocks (MB = 16 x 16 pixels). There exists three main types of slices: I, P, and B. In a I slice, the MB are predicted based on other intra-frame MB (intra-prediction). An I slice is thus independent from the slices in other frames. In a P-slice, the MB are predicted based on an intra-frame MB or an inter-frame MB in a past frame. Finally, B-slices use bidirectional inter-prediction where a MB may be predicted based on another MB in a previous or future frame.

As illustrated in Fig. 2.2, while encoding a video, the first step is the prediction phase which aims to find a correlation between the MB to be encoded (current MB)

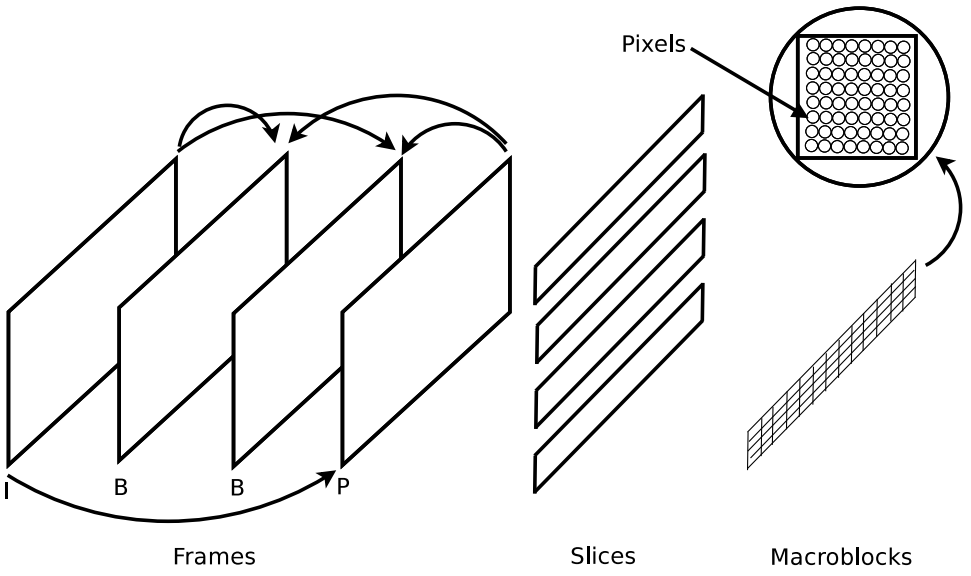

Figure 2.1: Structure of a video sequence

and a reference MB. As explained before, the reference MB may be in the same frame (intra-prediction) or in a past or future frame (inter-prediction). The operation of searching the reference MB (the "best matched" MB) is called motion estimation. The coordinates of the reference MB are stored in a motion vector (MV). The data obtained from subtracting the current MB from the reference MB is called a residual MB. The MV and the residual MB allows reconstructing the current MB. Matrix (a) in Fig. 2.2 is an example of (8x8) block extracted from of a residual MB.

In the second step, the residual MB are then transformed into another domain in which they are represented by transform coefficients. After the transformation, the data should be decorrelated and separated into compact group of data with minimal interdependence where most of the information should be concentrated into a small number of values. The transform operation can be achieved, for example, using Discrete Cosine Transform (DCT) or Wavelet transform.

The coefficients obtained from the transform operation (see matrix (b)) are quantized to remove insignificant values, leaving a small number of significant coefficients that provide a more compact representation of the residual data. The quantization reduces the precision of the transform coefficients according to a quantization parameter (qp). For example, the original coefficient values in matrix (b) are divided by a qp in the matrix (c) and rounded to the nearest integer. Typically, the result is a block in which most or all of the coefficients are zero (see matrix (d)), with a few non-zero coefficients. Setting qp to a high value means that more coefficients are set to zero,

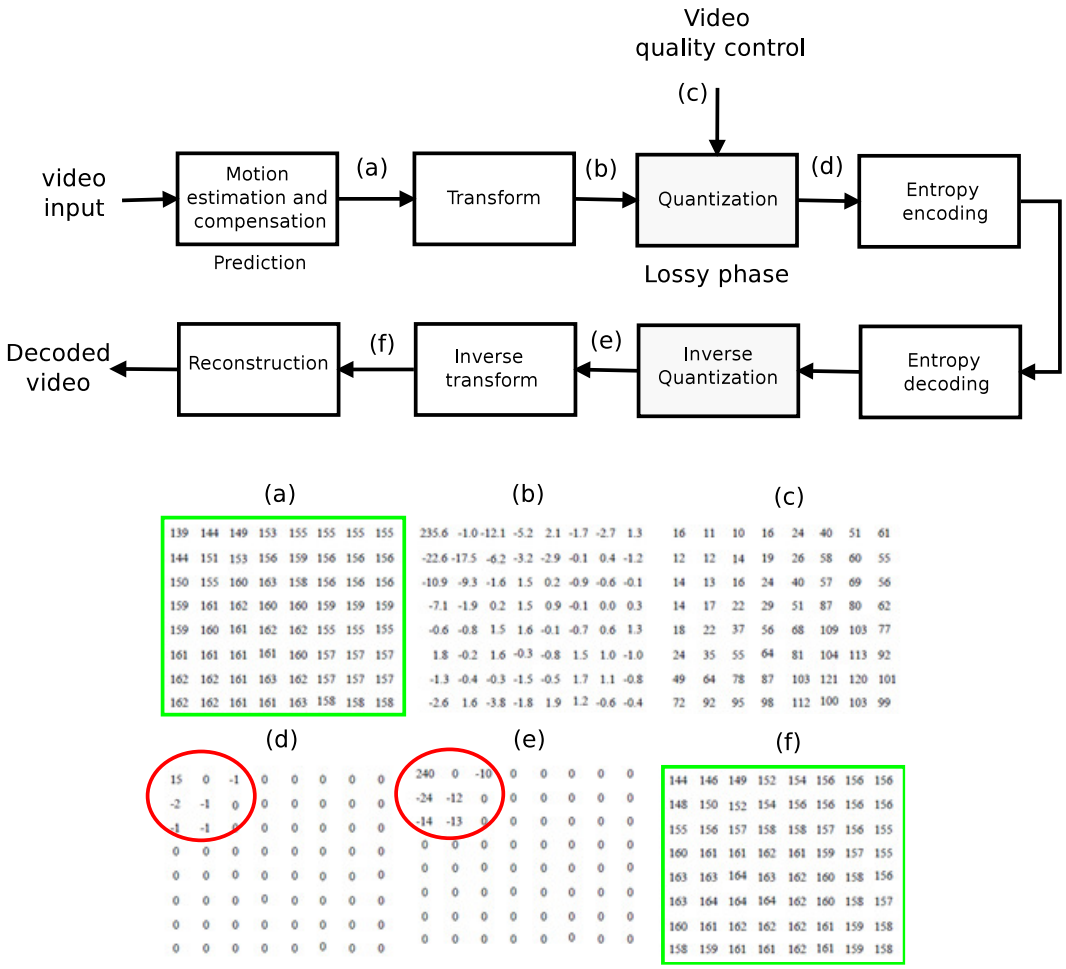

Figure 2.2: Principles of video encoding/decoding

resulting in high compression at the expense of poor decoded image quality. Setting qp to a low value means that more non-zero coefficients remain after quantization, resulting in better image quality at the decoder but also in lower compression.

All the obtained data from these steps (transform coefficient, MV, qp) are then compressed using an entropy encoder. This operation is reversible and no data is lost. At this step, the compressed data are ready to be sent to the decoder.

At the video decoder, firstly, the entropy decoding is executed to extract the MV, the residual data and the qp. The inverse quantization is then executed to obtain the transform coefficients. Notice that, only the coefficients concentrating the most relevant information are rescaled. The information associated to the null coefficient are thus lost (see matrix (e)). Consequently, the inverse transform based on the rescaled coefficient is not identical to the original video date (compare matrix (a) and (f)).

2.2.1.2 Video quality assessment metrics

The video encoding/decoding is a lossely process inducing a drop in the video visual quality. In general, the higher is the compression ratio, the lower is the obtained video quality. As explained in the previous section, the quality of the encoded video can be selected by tuning the quantization parameters. The higher is the value of qp, the lower is the video quality and vice versa.

In the case of a constant bit rate encoding mode[1] (CBR), the video bit-rate may provide indication on the video quality. For example, a video encoded at a 1024 Kb/s has a higher quality than one encoded at 512 Kb/s. In fact, the lower is the bit-rate constraint, the more aggressive is the quantization phase which results in a higher quality drop.

There exists other metrics which provide more accurate estimation of the video quality. For example, PSNR (Peak Signal to Noise Ratio) [28] is used to measure the quality of reconstruction of lossy compression codecs. The signal in this case is the original data, and the noise is the error introduced by compression. When comparing compression codecs, PSNR is an approximation to human perception of reconstruction quality.

2.2.1.3 Video playback QoS assessment metrics

In addition to the drop in video quality related to data loss in video encoding algorithm, the visual perception of the video content may be impacted by factors related to the quality of service (QoS) of the video playback process [29].

Figure 2.3 illustrates a typical video playback process. First, the video frames are retrieved from a source (network, file system, etc). Then, the video frames may be buffered in an input buffer to decouple the video decoding process from the fluctuation in the network bandwidth or the I/O system. The video decoder processes each frame and transmits it to the displaying process. The decoding time may vary considerably from a frame to another while the displaying process should display the decoded frames at a constant speed corresponding to the video sequence displaying rate $R_{display}$. To decouple the constant displaying speed from the fluctuation in the decoding time, an output buffer may be used between the decoder and the displaying device.

[1]Constant bit rate encoding means that the rate at which the encoder output data should be produced is constant

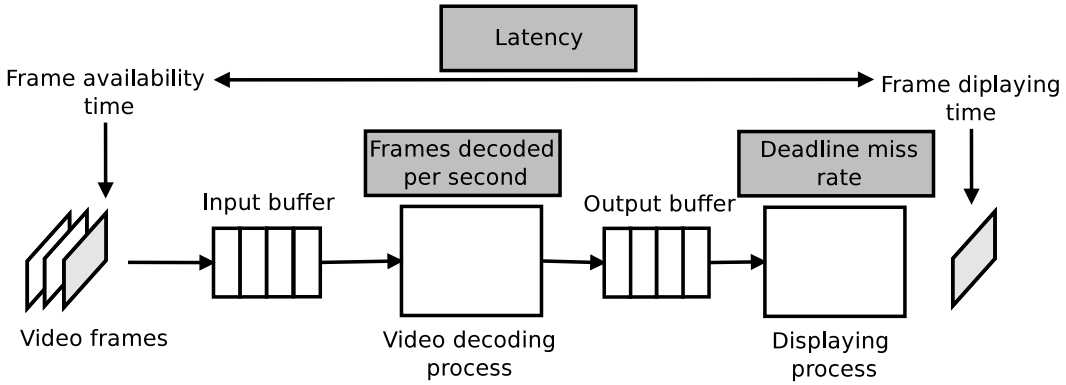

Figure 2.3: Video decoding performance metrics

Hereafter, we present some metrics for evaluating the QoS of the above described video decoding process.

Deadline miss rate (DMR)

Video decoding is a soft real time application. During the decoding process, each frame should be displayed before a deadline, otherwise, a *deadline miss* occurs. The higher is the *deadline miss rate* (DMR), the lower is the perceived quality by the end-user.

The DMR is an important parameter to be considered to assess the quality of the video playback quality. It may occur due to insufficient processing resources for decoding the video frames or to insufficient network resources in case the video content is retrieved from the network.

Decoded frames per second (FPS)

The average number of decoded frames per second (FPS) is another metric for evaluating the quality of video playback. A FPS higher or equal than the displaying rate is a necessary condition (but not sufficient) to avoid the deadline miss.

If the FPS is at least equal to $R_{display}$, a video decoding with zero DMR can be guaranteed if an output buffer may be inserted between the decoder and the displaying process to decouple variations in the decoding time from the constant displaying rate to avoid deadline miss [30].

Latency

The use of buffers in the decoding process increases the video quality playback, however it may induce an additional latency which represents the difference between the frame

availability time and the frame displaying time. A long latency (few seconds) may be allowed in case of non -live video (Youtube, movie playback, etc). On the other hand, the latency should be kept as minimal as possible in case of live application such as video conferencing.

2.2.2 Energy consumption in electronic circuits

We introduce in this section some background on the energy consumption of video decoding. We start by describing the different source of power consumption in electronic circuits. Then we explain the performance-energy trade-off in video decoding using dynamic frequency scaling and point out the impact of the processor architecture on the performance energy efficiency of video decoding.

2.2.2.1 Static *vs* dynamic energy consumption

The energy consumption (in Joule) of an electronic CMOS circuit is the amount of the power P consumed during a time t.

$$E = P.t \tag{2.1}$$

P, the total power consumption (in Watt), is the sum of the static power P_{static} and dynamic power P_{dyn} given by Eq. 2.2 and 2.3 respectively :

$$\begin{aligned} P_{static} &= L_g(V_{dd}I_{sub} + |V_{bs}|I_j + V_{dd}I_g), \\ I_{sub} &= K_3 e^{K_4} V_{dd} e^{K_5} V_{bs}, \\ I_g &= K_6 e^{K_7} V_{dd} \end{aligned} \tag{2.2}$$

I_{sub} is the sub-threshold current, V_{dd} is the supply voltage and L_g , V_{bs} , I_j , K_3 , K_4 , K_5 , K_6 and K_7 are constants which depend on the circuit fabrication technology [31].

$$P_{dyn} = C_{eff}.V^2.f \tag{2.3}$$

f is the clock frequency and C_{eff} is the circuit effective capacitance [32].

The static power does not depend on the executed program. It relies on the circuit fabrication technology and area. Below 65-nm circuits feature size, it becomes significant and poses new low-power design challenges [33].

On the other hand, the dynamic power relies on $C_{eff} = A.C$, where C is the circuit capacitance and A is the activity factor[2]. In a microprocessor, the C_{eff} parameter represents the average capacitance of all the processor blocks (control unit, cache, inter-connect) which depends on the type of instructions executed and on the data accessed [34].

We introduce hereafter two system level techniques to save the dynamic and static power consumption: dynamic voltage and frequency scaling (DVFS) and Dynamic Power management (DPM) respectively.

2.2.2.2 Dynamic voltage and frequency scaling (DVFS)

DVFS is a technique which consists in adapting dynamically the frequency of the processor according to the executed workload to save the dynamic energy. To each frequency value corresponds a voltage level. The power state represented by voltage-frequency pairs is called a P-state.

For the sake of simplicity, we consider that the processor frequency is proportional to the voltage (i.e. $V \propto f$) as assumed in [35, 36] then the dynamic energy is :

$$E_{dyn} = K.f^3.t \tag{2.4}$$

where K is a constant parameter . If the frequency is divided by 2, the execution time may be doubled but the power consumption is divided by 8, which explains the energy reduction. As we will discuss in next sections, DVFS strategies are based on a trade-off between energy consumption and performance.

2.2.2.3 Dynamic Power management (DPM)

In contrast to P-states, which are execution power saving states (During a P-state, the processor is still executing instructions), a processor can save more energy by disabling almost all clocks or shutting down some blocks during inactivity periods. Such kind of power states are called C-states. The technique consisting to use processor C-states to save power is called Dynamic Power Management (DPM) [37].

[2]The activity factor is a constant parameter representing the average switching activity in the circuit. Its values range from 0 to 1

2.2.3 Discussion

In the above sections, we have introduced some general principles of video encoding and decoding and explained all the steps of these processes. Then, we have presented some metrics for evaluating the perceived visual quality of the video. These quality metrics may be related to the lossy nature of encoder/decoder application or to the underlying video delivery and decoding system.

In the models developed in this thesis, we use the bit-rate (combined with the above presented quantization parameters[3]) to express the video quality drop due to data loss. The use of the bit-rate is motivated by the fact that it is easy to extract from a video content[4]. This is useful for online estimation of the performance and/or the energy of video decoding. In what follows, we use the terms "video quality" and "bit-rate" interchangeably.

On the other hand, the average decoded FPS is used to express the capacity of the underlying system for video decoding. As highlighted previously, the FPS allows to evaluate if the necessary condition for decoding a video is met or not. In what follows, we denote the FPS property as "video decoding performance". One can highlight that video performance may be modeled at per-frame basis in case of low latency applications. This may need a low level characterization of different decoding steps presented previously. This is out of the scope of this work.

Finally, we have discussed the basis of the energy consumptions in electronic circuits and introduced DVFS and DPM, two system level techniques for saving the static and the dynamic energy.

The advantage of DPM is that it allows to save both the dynamic and static power consumption. However, it may induce a non-negligible latency due to the entering/exiting C-state modes (see section 4.3.1). Thus, in what follows, we will focus on the impact on the frequency scaling (DVFS) on both the performance and the energy consumption of video decoding.

2.3 Principles of energy saving in video decoding

We will discuss in this section the energy saving techniques of video decoding at both system and architectural levels. We will show that, usually, energy saving of video

[3]See section 3.2.1 for more details.

[4]Bit-rate information is usually present in almost all video containers such as MP4 file format

F_i	$C_i(MC)$
F_1	12.5
F_2	50
F_3	37.5
F_4	25

Table 2.1: Video frame complexities

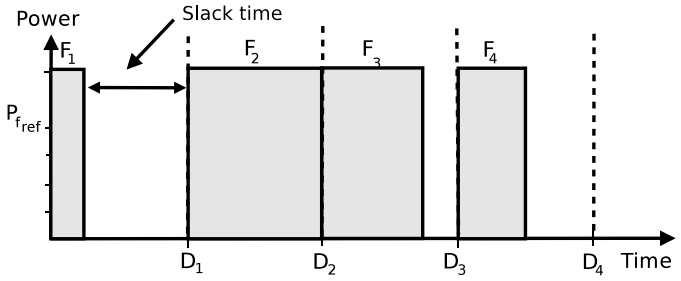

Figure 2.4: Frequency scaling in video decoding

decoding consists in achieving a balance between the video QoS (see section 2.2.1.3) and the energy consumption.

To explain this, we consider an example where we suppose a video decoding application executed by a processor supporting variable voltage and frequency scaling. The frequency values belong to the interval $]0, f_{max}]$ and are expressed as $\alpha.f_{ref}$ where $f_{ref} = 1GHz$ is considered as the frequency reference value and α a scaling factor.

A video sequence to be decoded is composed of four sequential and independent frames F_i $(1 \leq i \leq 4)$. Each frame F_i should be decoded before the deadline $D_i = i.D$ where $D = \frac{1}{25}s = 40ms$ is the period corresponding to 25 frames/s displaying rate. The frame complexities C_i, listed in the Table 2.1, are expressed in terms of the required number of processor mega-cycles (MC) to be decoded. We denote the total number of cycles required to decode all the frames $C_{total} = \sum_{i=1}^{4} C_i$.

The time to execute C processor cycles at $\alpha.f_{ref}$ frequency is assumed as follows[5] :

$$T_\alpha(C) = \frac{C}{\alpha.f_{ref}} \tag{2.5}$$

According to the Eq. 2.5 and the energy consumption model defined in Eq. 2.4, the energy consumption of decoding C cycles using the frequency αf_{ref} is thus :

$$E_{dyn_\alpha}(C) = K.(\alpha.f_{ref})^2.C \tag{2.6}$$

[5]Actually, the execution time does not scale linearly with the frequency due to the latency of the off-chip memory access. Refer to section 2.3.1.3 for more details regarding this issue.

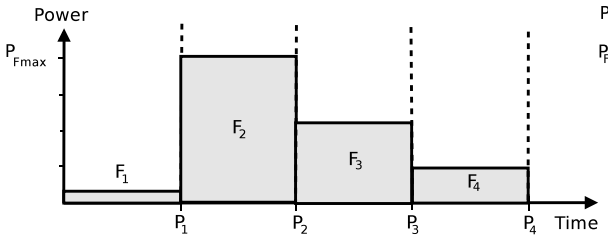

Figure 2.5: Just-in-time DVFS

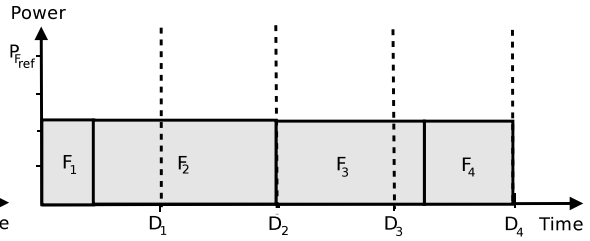

Figure 2.6: Workload averaging DVFS

We will use this simplified dynamic power model to explain the principle of energy saving of video decoding using DVFS.

2.3.1 Frequency scaling: Performance vs energy consumption

According to the frame decoding complexities given in Table 2.1, and the Eq. 2.5, the clock frequency allowing all the frames to be decoded before their deadline is $f = \frac{5}{4} f_{ref}$. The energy consumption in this case is $E = K.f^2.(\frac{5}{4})^2(C_1 + C_2 + C_3 + C_4) \simeq 195K.f^2$. As illustrated in Fig. 2.4, running constantly at this clock frequency results in early frame decoding (F_1, F_3 and F_4) leading to slack times. These slack times may be used to reduce the energy if the frequency is reduced using DVFS. We explain hereafter two DVFS strategies to achieve this objective.

2.3.1.1 Frame-by-frame based DVFS

According to Eq. 2.5, if the frames F_1, F_2, F_3 and F_4 are decoded using the frequencies $\frac{12.5}{40} f_{ref}$, $\frac{50}{40} f_{ref}$, $\frac{37.5}{40} f_{ref}$ and $\frac{25}{40} f_{ref}$, they will be ready to be displayed just before their deadline as illustrated in Fig. 2.5. Lowering the frequency allows to decrease the total energy to $E = K.f_{ref}^2.C_{total}.((\frac{12.5}{40})^2 C_1 + (\frac{50}{40})^2 C_2 + (\frac{37.5}{40})^2 C_3 + (\frac{25}{40})^2 C_4) \simeq 122K.f_{ref}^2$. This represents 38% energy saving as compared to running at constant frequency.

In order to use a frame-by-frame DVFS, the decoder needs to have a prior knowledge of the video frame complexities. We will discuss the frame-based performance models in section 2.5.1.1.

2.3.1.2 Average workload based DVFS

The averaging DVFS energy saving is based on the convexity of the $E(f)$ model (see $E(f)$ graph in Fig. 2.7) and Jensen's inequality [38]. In fact, applying this inequality

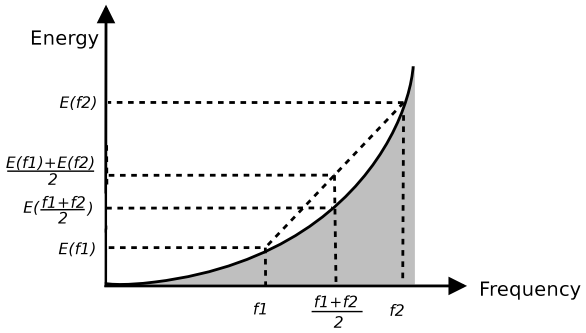

Figure 2.7: Energy model convexity

on a convex dynamic energy model results in :

$$E_{dyn}(\overline{f}) \leq \overline{E_{dyn}(f)} \tag{2.7}$$

This inequality means that the processing at the mean frequency is more energy efficient than processing at small number of discrete processing rate levels [39].

To explain this approach, we consider in the previous example that the frequency is set to the constant average frequency $f_{avg} = \frac{1}{4}(\frac{12.5}{40}f_{ref} + \frac{50}{40}f_{ref} + \frac{37.5}{40}f_{ref} + \frac{25}{40}f_{ref}) = \frac{25}{32}f_{ref}$.

The total energy consumption is $E_{averaging} = K.(\frac{25}{32})^2 f_{ref}^2(C_1 + C_2 + C_3 + C_4) = 76K.f_{ref}^2$. As compared with the "just in time" optimal policy ($122K.f_{ref}.D$), the averaging DVFS allows more energy saving. However, as illustrated in Fig. 2.6, although the four frame are decoded withing $4D$ time, the deadline of the frame F_3 is missed. This is explained by the fact that the frequency is set based on the average performance, not at a frame-by-frame basis. The deadline miss can be avoided if a buffer is inserted between the decoder and the displaying device but at a cost of an additional latency [40].

As we will discuss in section 2.5.1.2, the advantage of averaging DVFS is that it needs an average performance model rather than an accurate frame-based one.

2.3.1.3 Video-aware DVFS : Challenges and issues

Saving the energy consumption of video decoding using DVFS supposes that the video decoder has knowledge of the upcoming workload complexity. However, the video workload may vary considerably depending on various parameters such as the video quality and scene complexity. This makes the workload prediction one of the most challenging issues in video-aware DVFS.

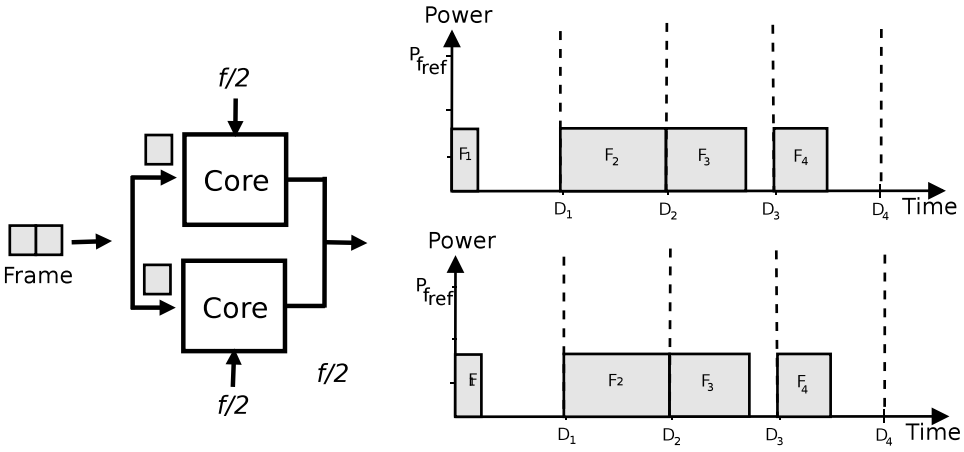

Figure 2.8: Energy efficiency of parallel video decoding

On the other hand, assuming that the upcoming video workload can be predicted using a given performance model, the video decoder has to select the appropriate frequency allowing to save the energy without impacting the video QoS. One issue which can be faced at this step is considering the impact of the off-chip memory access latency. In fact, video decoding is a memory-bound application which means that it makes use of lot of instructions accessing to the external memory. However, in processor architectures, the bus used for accessing the external memory is clocked at a frequency which is independent from that of the processor. Thus, the impact of scaling the processor frequency on the performance may depend considerably on the rate of the external memory access made by the decoder [41]. Consequently, given a number of a processor cycles, the execution time cannot be calculated simply using the Eq. 2.5 used for estimating the expected frame decoding times in the above examples. This adds an additional complexity to implement video-aware DVFS.

2.3.2 Parallel multi-core video decoding

With the raise of multi-core SoCs, processor performances have increased without the need to use high clock frequencies allowing thus to save energy. This is particularly true for video decoding. To explain this principle, we consider the previous example in case of a multi-core processor as illustrated in Fig. 2.8. We suppose that each frame can be decomposed into two independent parts which are decoded in parallel using two identical processors. This allows to achieve the same performances at $\frac{5}{8}f_{ref}$, which is the half of the required frequency when using one processor.

The energy consumption in this case is $E_{arch} = 2.K.f^2.(\frac{5}{8})^2(\frac{C_1}{2}+\frac{C_2}{2}+\frac{C_3}{2}+\frac{C_4}{2}) = \frac{1}{4}E$.

This means 75% energy saving but without a loss in the performances. In this case, the cost in term of energy consumption is an increasing static power consumption due to the use of two processors.

The parallelism of video decoding on multi-core processors can be achieved at a frame, slice or macro-block levels [42]. At a frame level, the frames may be decoded in parallel on different processing units. The drawback of such an approach is that it does not scale very well because the number of independent frames (ex. B frames) is limited at a given time. On the other hand, a higher scalability is possible at a slice level. However, this depends on the encoder setting to enable multi-slice frames. At a macro-block level, there is greater opportunity of parallelism but it is inefficient to execute on parallel multi-core processor due to the communication overhead. As we will discuss hereafter, this level of parallelism can be implemented more efficiently on specialized processor[6].

2.3.3 Specialized processing

In this section, we will discuss from a high level point of view, different ways to use specialized processors for video decoding. We will particularity discuss hardware codecs, Graphical Processing Unit and Digital Signal Processors.

2.3.3.1 Hardware video codecs

In general, a specialized processor is more energy efficient than a GPP [43]. In fact, in [44, 2], the authors show that the energy inefficiency is intrinsic to the programmable nature of general purpose processors. This is due to the control and the communication overheads in executing an instruction in a GPP. For example, according to these studies, only few pJ are needed to execute an addition operation on 45nm processor while 70pJ is needed to execute the entire instruction. This makes the specialized processors two orders of magnitude more energy efficient than GPP [2].

In case of video codecs, thanks to customized architecture design, hardware-codecs can provide better energy consumption properties than general purpose processors. In fact, a considerable energy saving can be achieved by eliminating instruction fetching characterizing the programmable nature of GPP's. In [44, 45], the authors show that

[6]The term specialized processor refers to non general purpose such as hardware codec, GPU or DSP.

specialized processing units achieve most of their efficiency gains by tuning data storage and compute structures and their connectivity to the data-flow and data-locality patterns in the codec.

Practically, hardware video codecs architecture designs are based on hard-wired functional block for executing different video codec modules. Each functional block makes use of intensive parallelism in data processing. This allows to decode HD video at very low clock frequency (few MHz) while consuming around tens of mW [46, 47, 48].

From the GPP point of view, the hardware codec is considered as an external device handled by a driver in the operating system. Decoding a video sequence needs a minimum control from the GPP side to manage the I/O from and to the hardware codec. For example, sending the frame location address to the codec, executing cache maintenance operations[7] or handling hardware interrupt (see Fig. 2.9-a).

A frame decoding is considered, from the GPP point of view, as an I/O operation generating a system latency caused by handling the I/O control. Since all the video decoding process is implemented in the hardware codec, this control is executed at a coarse granularity (it occurs at the beginning and at the end of the frame decoding). This reduce the communication overhead and off-loads the GPP during all frame decoding phases.

The drawback of hardware video codecs is that they are not flexible and cannot be adapted to the evolution in video standards [49]. For example, hardware accelerators for the new MPEG HEVC (High Efficiency Video Coding) standards are still not widely used on mobile devices at the time of writing this thesis.

2.3.3.2 Graphical processing unit

Graphical Processing Units (GPU) are processing units specialized in graphic computing. They support instruction set for accelerating geometric calculations such as the rotation and translation of vertices into different coordinate systems.

Because most of these computations involve matrix and vector operations, GPUs become more and more used for executing non graphical processing such as simulation, high performance computing [50] and especially video decoding [51, 52, 53].

[7]The fact that both the specialized processor and the GPP have their proper cache memory and communicate using a shared memory imposes to manage cache coherency each time data are shared between the hardware codec and the GPP.

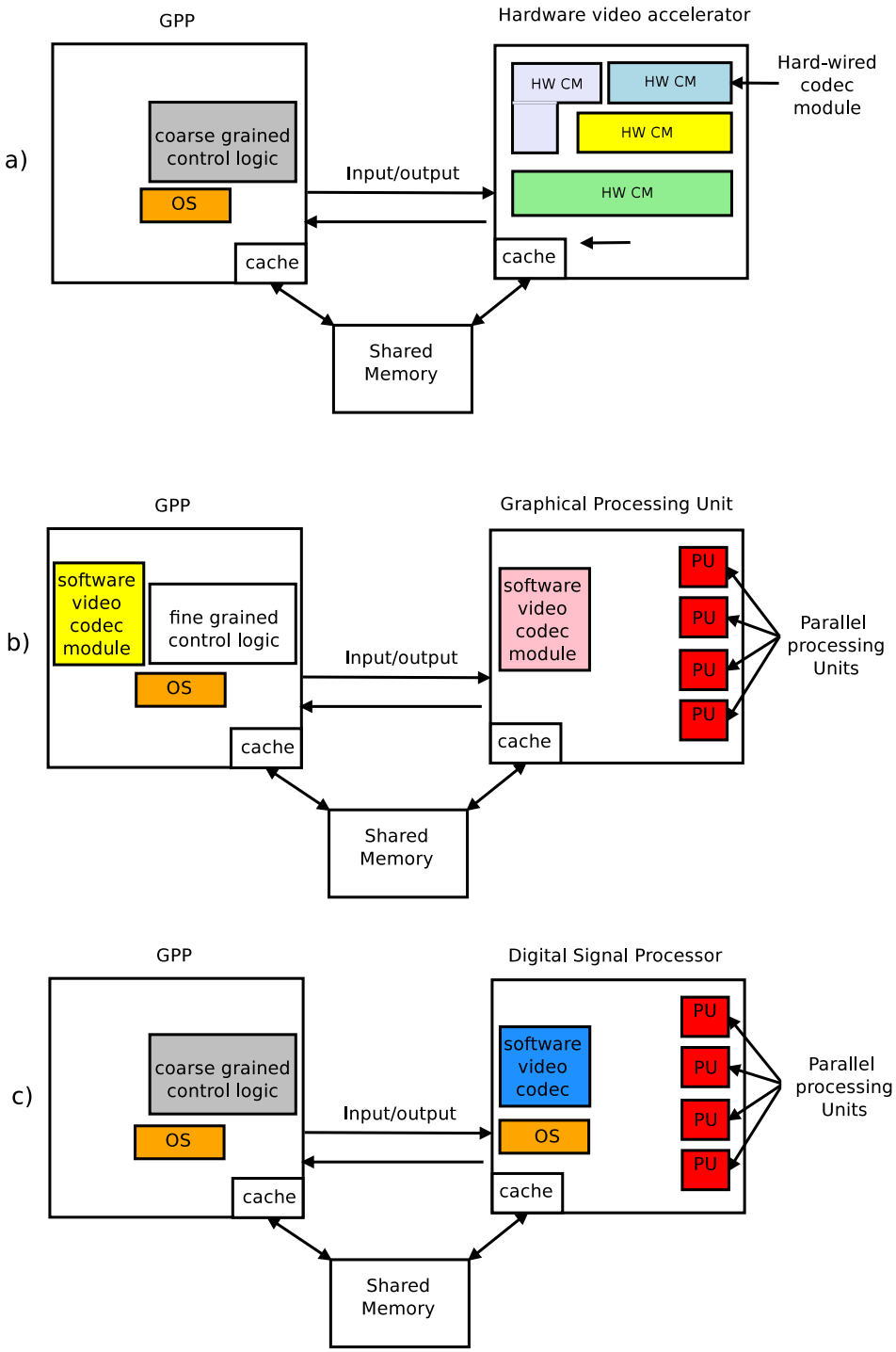

Figure 2.9: GPP vs specialized processors energy efficiency

In general, using a GPU for decoding a video consists in off-loading the GPP from some codec module by executing them on the GPU. The remaining modules are still executed on the GPP (see Fig. 2.9-b). In fact, the GPU has a limited instruction set which does not allow to execute efficiently the full codec [51]. Usually, motion compensation and color space conversion are better handled by the GPU where the inverse quantization, the inverse DCT and the entropy decoding are handled better by

the GPP [51].

Since a video decoding process cannot be handled entirely by the GPU, many I/O operations are required between the GPP and the GPU within each frame phase. Indeed, the control logic is executed at a fine granularity in the GPP to synchronize between the modules executed in the GPU and those executed on the GPP. This may have an impact on both performance scaling and energy consumption [45]. For this reason, almost all GPU manufacturers do not propose pure GPU video decoding in their SoC. They, instead, propose video decoding solutions relying on hardware accelerator integrated in their GPU. We can cite as an example Intel HD Graphics, Nvidia Pure Video and AMD Unified Video Decoder.

2.3.3.3 Digital signal processor

Digital signal processors (DSP) are specialized in signal processing operations. They support specialized instruction set such as MultiplyAccumulate (MAC) and Fused MultiplyAdd (FMA) operations, which are used extensively in all kinds of matrix operations. They also make use of parallelism by supporting Single Instruction Multiple Data (SIMD), Very Long Instruction Word (VLIW) and superscalar architecture.

Unlike the GPU, the DSPs have an advanced instruction set and are able to run complex programs. For example, a DSP is even able to run its own operating system [54]. As like the hardware codec, a DSP is able to implement a full video codec leading to a limited coarse grained control from the GPP side (see Fig. 2.9-c).

In [55, 15], the authors explain, from an architecture point of view, the source of energy efficiency of DSPs. The benefit of using them in energy constrained mobile devices highlighted in [16], especially for video decoding [56]. In fact, in addition to performance speed-up, DSP-based video codecs allows to enhance the energy efficiency of video decoding.

2.3.4 Discussion

The above sections highlighted two important points: first, the performance and the energy consumption properties should be considered together to evaluate the energy efficiency of video decoding systems. Second, the parameters which impact the performance and the energy consumption are numerous and may be localized at different levels: application, system and architecture.

Accordingly, we will survey, in the next sections, the studies focusing on characterizing and modeling both the performance and the energy consumption of video decoding at these levels. Thus, at application and system levels, we consider mainly the impact of the video quality, the processor frequency and the operating system overhead on the performance and the energy consumption of video decoding.

On the other hand, at architecture level, we consider the video decoding on different processor architectures: GPP, muti-core GPP and specialized processors including hardware codec and DSP. Low level design of these architectures will not be considered. However, we will focus on the impact of off-chip memory access latency on performance scaling (as highlighted in section 2.3.1.3) and show how this is important to consider in DVFS policies for video decoding. We highlight that GPUs will not be considered in the remaining related works survey as well as in our study. In fact, as explained previously, video decoding on this kind of processors requires a fine grained a low-level GPP/GPU partitioning of the video codec modules. This represents too low level details as compared to the scope of this thesis.

2.4 Performances and energy consumption characterization of video decoding

2.4.1 Video decoding performances characterization

The performance characterization of video decoding has been addressed by several studies. The objective was to identify the most processing-intensive part in the decoding process and understand the source of performance drop. We consider hereafter, application, system and architectural levels.

2.4.1.1 Application level

In [57], the video complexities of different video qualities are analyzed. For each video quality, different video sequences were used. The results show how the performance varies when increasing the quality of the video. Moreover, for the same video quality, it is shown that the decoding performance may vary considerably depending on the scene complexity of the video. A per-frame performance analysis shows also that the variation in the decoding time depends on the frame type (I frames are more complex to decode than B and P frames).

In [58], the characterization is executed at a finer granularity. The authors measured

the execution time of the basic H.264/AVC decoding modules. They show that motion compensation is the most time-consuming module, taking over 40% of the total CPU time. The entropy decoding takes about 22% of the total CPU time, whereas the inverse quantization and transform takes only 7%. The deblocking filter requires a large amount of computation as well, taking the remaining 20% of processing time. Moreover, they analyzed the performance breakdown of the different decoding modules when decoding higher video quality.

2.4.1.2 System level

In [59], the authors analyzed the impact on the performance of the complete flow of the communication between a GPP and a specialized processor (DSP). They measured the Inter-processor communication (IPC) overhead due to handling hardware interrupt and data transfer. They propose accordingly a technique to estimate the IPC performance at run-time and dynamically adjust the IPC strategies under environmental parameters and system resource constraints.

On the other hand, in [60] and [61], performance consideration of DSP decoding are analyzed according to cache coherency maintenance and DMA transfers. The authors show that significant performance increase can be achieved by modifying the decoder design so as to minimize the communication between the GPP and the DSP.

2.4.1.3 Architecture level

In [62], the authors used Simplescaler simulator [63] to characterize H.264/AVC video decoding workload using different architecture configurations. They focused mainly on the cache miss and instruction level parallelism (ILP) behaviors. They highlighted that there is a direct relation between the ILP parallelism and the cache performance. In fact, they showed that during video decoding, an important time is spent in a stall status waiting for data to be fetched from the main memory. This increases the number of cycles per instruction and consequently decreases the IPC.

In [64, 65], the authors analyzed the performance of the memory access while decoding the video. They focused mainly on the ratio of cache miss while decoding various type of multimedia workload. One observation they have highlighted is the increase of the number of memory access instructions while increasing the video quality.

In the same way, in [58], the authors focused on characterizing the memory bound

instruction but considered in addition the impact of performance scaling when using DVFS. Based on the cache miss statistics, they showed that entropy decoding and inverse transform/quantization are entirely computation-bound in H.264 decoding. On the other hand, motion compensation is memory intensive. The obtained data are used to explain the performance scaling when using DVFS by pointing out the relative performance scaling of the different decoding modules of video decoding. In fact, they showed that the speed-up of the motion estimation is lower than one of the inverse transform or entropy decoding. The reason is that in memory bound instructions, most of the CPU time is spent on waiting for the memory accesses which are independent from the clock frequency.

In [66], the authors focused on performance scaling of parallel video decoding on muticore processors and have discussed various parallelization possibilities. They showed that slice-level parallelism has two main limitations. First, using many slices increases the bit-rate and, second, not all sequences contain many slices since the encoder determines the number of slices per frame. In this study was also analyzed frame-level parallelism, which uses the fact that some frames (B frames) are not used as reference frames and can therefore be processed in parallel. It was shown that this approach is not very scalable because usually there are no more than three B frames between consecutive P frames. On the other hand, they show that MB level parallelism is very scalable without needing any requirements from the encoder side. However, the authors highlighted the need to reduce communication and synchronization overhead to avoid a performance drop.

2.4.2 Video decoding energy consumption characterization

2.4.2.1 Application level

In [67], the authors analyzed the energy consumption of different video codecs including H.264/AVC. Various experiments have been performed to investigate the effects of codec parameters such as the bit-rate and the resolution on the consumed energy. Experimental results show that increasing the resolution increases considerably the energy consumption. Whereas, increasing bit rate gives a better picture quality without inducing too much energy consumption.

In [68], the authors analyzed how video quality scalability impacts the energy con-

sumption. In addition to the video resolution (spatial scalability) and the bit-rate (PSNR scalability) studied in the above cited work, they consider the impact of the frame rate (temporal scalability). Their results match the previous conclusion: the video resolution is the most important video parameters that impact the energy consumption. Based on these results, the authors proposed a strategy for rescaling video quality settings on the decoder to save the energy.

2.4.2.2 System level

In [48], the authors analyzed the cost of multimedia framework and the operating system overhead in software and hardware based video decoder. The authors showed that, the interfacing overheads of the operating systems and software frameworks that hide the implementation details can be significant regardless of the implementation of video decoding.

In a more recent study [69], the authors analyzed the energy consumption of multimedia processing on heterogeneous SoC including an ARM processor, a DSP and a GPU. One observation they highlighted is that the power consumption of two or more cores running concurrently is lower than the sum of the power when each core is running alone. This is due to cores idling when completing the assigned task. As a result, the average power consumption might become lower (we have made the same observation in our experimentation. See section 4.3.3.2).

2.4.2.3 Architecture level

In [69], the authors evaluated the performance and energy benefits of utilizing the integrated GPU and DSP cores to off-load or share CPU compute-intensive tasks. The evaluation is conducted on three representative mobile platforms, TI's OMAP3530, Qualcomm's Snapdragon S2, and Nvidia's Tegra2, using common computation tasks in mobile applications including video decoding. The authors show that when off-loading a processing on specialized processors, the execution time reduction is greater than the increased power consumption. As a result, the overall energy consumption is reduced. Moreover, they highlighted that compute-intensive algorithms usually have a mixture of subtasks which exhibit a wide range of characteristics. Each type of cores would be more efficient for some, but not all, of the subtasks. By proportionally assigning the subtasks based on the cores characterization results, the overall performance and

energy of a mobile application can be optimized.

In [48], the energy efficiency of software and hardware video decoder is analyzed. The authors showed the energy efficiency of hardware accelerated video decoding as compared to software based ones especially for high resolutions. However, they highlighted that monolithic hardware codecs suffer from increasing complexity and less flexibly for multi-standard support. As a result, they suggest finer-grained video accelerators where basic codec module are hard-wired and the control/scheduling is executed by a GPP with a small energy footprints.

On the other hand, in [70], the authors focused on the energy efficiency of parallel video decoding on multi-core processor. They have evaluated the energy saving as compared to mono-core decoding and showed that using four ARM cores allows to reduce the energy consumption of HD video decoding up to 63 % as compared to using a single core.

2.4.3 Discussion

Table 2.2 summarizes the above cited related works on performance and energy characterization of video decoding. For each study, we have specified the parameters which were considered and the level to which they correspond.

Although these works cover a wide range of parameters corresponding to different abstraction levels of a video decoder, it is difficult to extract a clear and a single view point on the performance and energy consumption properties of video decoding in terms of video quality on various type of architectures. In fact, each study scope is restricted to a subset of parameters and/or levels. For example, in [58], application, system and architecture parameters are considered for performance characterization, but only for a x86 processor. On the other hand, in [48], different architectures were considered but the processor frequency parameter was not covered.

In addition, comparing the results of these studies is not possible since they may use in their experimentation hardware architecture with different technologies and heterogeneous software stacks.

In the performance and energy characterization part of this thesis, we propose a unified methodology based on an accurate performance and energy consumption measurement. In this methodology, we cover a wide range of application and system parameters (representative video qualities, frequency scaling, IPC). Moreover, different

	Performance characterization level		
	Application	System	Architecture
[57]	Video quality, frames type		
[58]	Per-module complexity		
[59]		Inter-processor communication	
[60],[61]		Maintenance of cache coherency, DMA transfers	
[62]			Instruction level parallelism, cache miss
[64, 65]	Video quality		cache miss rate
[58]	Video quality	processor frequency	memory-bound instruction rate
[66]	Level of parallelism (MB, Slice, Frame)		Multi-core processors
[This study]	Video quality	processor frequency, IPC	ARM, SDP, multi-cores processors.
	Energy characterization level		
	Application	System	Architecture
[67]	Video quality (bit-rate, resolution)		
[68]	Video quality (bit-rate, resolution, frame rate)		
[48]	Video quality	Multimedia framework and system overhead	Hardware vs software codec
[69]		Scheduling, Processor idling	Processor type (ARM, GPU, DSP)
[70]			Energy breakdown over homogenous multi-cores processors
[This study]	Video quality	Processor frequency, IPC	ARM, DSP, hardware codec, Multi-core processors.

Table 2.2: Summary of studies on performance and energy characterization of video decoding

low-power and mobile processor architectures were evaluated including GPP (ARM), DSP, multi-core and hardware accelerator. The execution of video decoding on these processor architectures is achieved using a single multimedia framework which allows to fully compare the different obtained results. As far as we know, there is no equivalent study in the literature which considers at the same time all the considered parameters and architectures.

2.5 Performances and energy consumption modeling of video decoding

2.5.1 Video decoding performances modeling

As discussed previously in section 2.3, video decoding energy saving should consider both the performance and the energy consumption aspects. In fact, the decoder should be able to estimate the upcoming workload to dimension the processing resources and thus, to save energy. In this section, we first start describing some proposed models for video decoding at both a video frame and an interval basis. We list then some works focusing on studying the impact of off-chip memory access on the performance scaling in the context of the use of DVFS.

2.5.1.1 Frame based models

The frame-based performance models aim to predict the video decoding complexity on a frame basis. As discussed in section 2.3.1.1, those models are useful for energy saving techniques in the case of video applications requiring low latency. We distinguish in the literature different approaches for frame-based performance modeling.

Empirical models

In [71, 72], authors used the linear relationship between an MPEG2 frame size and its decoding time as observed in [73] (see Fig. 2.10-a). In order to predict the frame decoding time, they proposed to maintain performance statistics per frame type (I, P, B) then the decoding time and frame length are correlated online to guide the selection of the appropriate processor frequency.

However, the used linear complexity model is no longer valid in the case of new video compression standards which use aggressive techniques to reach high compression ratios. For example, starting from MPEG4 standards, a distinction of frame types does

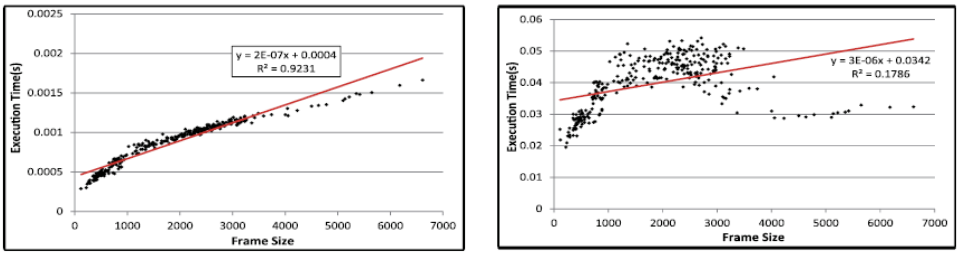

Figure 2.10: Video decoding complexity *vs* frame size (MPEG2 vs H.264/AVC) [3]

not even exist. Instead, every frame type can include different types of (I, B, or P) MBs, and each MB requires different amount of processing. It is thus difficult to achieve accurate estimation for the current frame decoding complexity merely from its size as illustrated in Fig. 2.10-b.

To overcome this issue, authors in [74] proposed an enhanced complexity model which take into account, in addition to the frame size, the number of MBs of a given type (I, P or B). These parameters are assigned weights to fit the frames complexity model. The developed model was used in a frame-by-frame DVFS strategy for MPEG4 video decoding.

Statistical models

To predict video decoding workload, some studies proposed to use sophisticated statistical adaptive filter tools to deal with the high variability of the decoding complexity from one frame to another.

In [75], authors proposed a frame-based prediction DVFS using *Kalman* filter. The proposed solution captures time-varying workload characteristics by adaptively reducing the prediction error via feedback control. On the other hand, in [3], they used *Particle Filter* which is known to be more powerful in dealing with even nonlinear/non-Gaussian time-varying workloads [3]. In the proposed solution, a function which correlates the frame size and its decoding time for each frame is fed to the particle filter to refine the estimates using its error-covariance feature. The proposed method was implemented in both MPEG2 and H.264 players and was validated on an ARM-based real testbed.

Metadata-based models

Video complexity metadata is information sent by the encoder to assist the decoder to predict the video workload for better processing resource allocation. This information is the result of an offline complexity analysis when encoding the video [76].

In [77], authors analyzed the computational complexity of a software-based H.264/AVC baseline profile decoder. Their approach is based on determining the number of basic computational operations required by a decoder to perform the decoding module (inverse transform, reconstruction, entropy decoding, etc). The frequency of use of each of the required decoding subfunctions is empirically derived using bitstreams generated from two different encoders for a variety of contents, resolutions and bit rates. Using the measured frequencies sent by the encoder as a metadata, estimates of the decoder time complexity for various hardware platforms can be determined.

In [78], a complexity model for H.264/AVC video decoding is derived by decomposing the entire decoder into several decoding modules (DM), and identifying the fundamental operation unit (termed complexity unit or CU) in each DM. The complexity of each DM is modeled by the product of the average complexity of one CU and the number of CUs required. The model is shown to be highly accurate for software video decoding both on Intel Pentium mobile 1.6-GHz and ARM Cortex A8 600-MHz processors, over a variety of video contents at different spatial and temporal resolutions and bit rates. Assuming the complexity information are sent by the encoder, the authors further show how to use this model to predict the required clock frequency and hence perform dynamic voltage and frequency scaling (DVFS) for energy efficient video decoding

2.5.1.2 Interval-based models

At the interval based granularity, the performance model aims to predict the average performance of decoding a set of frames. As explained in section 2.3.1.2, this suites applications accepting latency in the decoding process.

In [79], the authors proposed an algorithm (PAST) implemented on a Unix workstation which consists in monitoring the workload for a past interval and assumes the next one will be like the previous. In this study, this performance model is used to set the frequency based on the amount of time spent in idle state.

In [80], the authors propose to use a weighted moving average (WMA) filter to

predict the future interval workload. They consider N previous intervals weighted by factors which are reduced smoothly we go back in time as described in Eq. 2.8.

$$w_{i+1} = \frac{\sum_{k=N}^{1} K w_{i-N+k}}{\sum_{k=1}^{N}(k)} \tag{2.8}$$

The use of WMA filter ensures that prediction is aligned with the variations in the workload rather than being merely shifted in time. More complete comparative study on averaging filters which includes in addition the Least Mean Square filter (LMS) is presented in [81].

In [82], the authors apply the averaging filters proposed in the above works to MPEG video decoding. Then, they compare the PAST policy with the exponential weighted moving average filter (EWMA) described in the Eq. 2.9.

$$w_{i+1} = \alpha.w_{i-i} + (1-\alpha).w_i. \tag{2.9}$$

The coefficient α represents the degree of weighting decrease, a constant smoothing factor between 0 and 1. A higher α discounts older observations faster.

The results obtained in this study show that using EWMA allows to save more energy in case of a video workload than the simple PAST policy proposed in [79]. However, the impact on the video quality of service was not evaluated.

In [83], the authors used auto regressive models [84], which is a formal mathematical modeling tool allowing to calculate the weight of the averaging filters. In this study, they consider an MPEG video workload and claim better energy saving as compared to moving average technique.

In [85], the authors achieved more extended experiments on MPEG video decoding while using moving average filters discussed above. Moreover, they consider the quality of service (QoS) expressed in terms of the number of deadline misses. They highlighted that weighted averaging technique needs to tune the weights to have a good result. Moreover, they show that the tuned weights may not work for other applications or even the same application with different input. Based on these observations and the QoS of the video decoding, they claim that the simple PAST policy is better in case of video decoding.

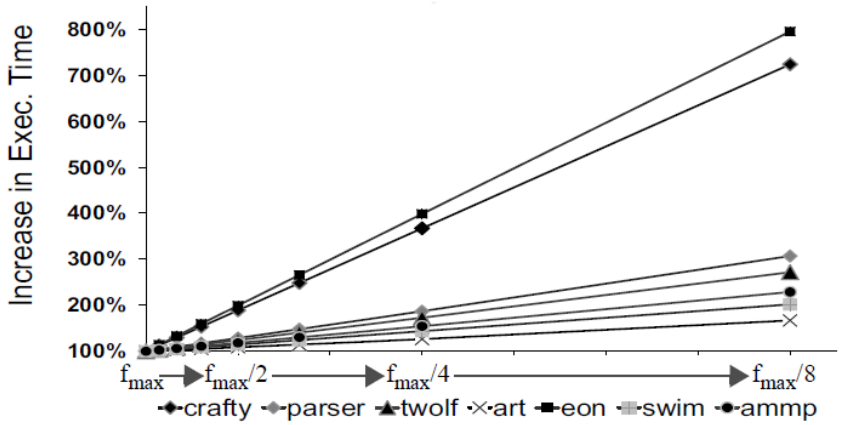

Figure 2.11: Impact of memory latency on performance scaling [4]

2.5.1.3 Memory-aware performance models

Many DVFS studies assume that if the predicted workload takes C_i processor cycles, then the execution time of this workload will be $\frac{C_i}{f}$. This assumption is true only in case of CPU bound programs. However, video decoding is both CPU and memory bound and thus, due to the off-chip memory access latency and the processor stall, the execution time does not scale linearly with the frequency [86].

This issue was observed first in [87, 71, 85, 88, 4] where the authors highlighted the effect of the memory latency on performance scaling using DVFS. They highlight the importance of considering this constraint in the frequency setting policy. For example, in [4], the authors show (see Fig. 2.11) that the programs included in SPEC CPU2000 benchmark[8] [89], react differently to the processor frequency scaling depending on their memory access rate.

Several studies considered this behavior in their proposed DVFS policies. In [90], the authors observe that in program phases with high rate of memory access, the frequency of the processor can be lowered without severe losses in performance. Accordingly, they propose an online solution using event counters available in some processors. Their solution consists in monitoring the memory access and scaling down the frequency if some threshold is reached. This allows to save energy without impacting severely the performance.

In [41, 91], the authors proposed an on-line memory-aware frame-by-frame DVFS. First, they observe that, in MPEG video decoding application, the execution time of

[8]The SPEC CPU2000 benchmarks are intended to exercise the CPU itself, the memory hierarchy to evaluate how much memory do they actually use.

memory bound instructions tend to be constant from a frame to another. They propose to separate a frame decoding time into two parts: a frame-dependent part and a frame-independent part. The frame independent part remains constant regardless of the frame type and tends to correspond to the memory-bound instructions. On the other hand, the frame dependent part highly relies on the type of each frame and corresponds to the CPU bound instruction. The amount of memory bound executed instruction is calculated based on the number Layer 2 cache miss rate provided by a processor event counter. The execution time of the CPU-bound instruction is estimated using a moving average filter. The combination of the two information is used to estimate the frame decoding time and to adjust the processor frequency accordingly.

2.5.2 Video decoding energy consumption modeling

We discuss hereafter the existing approaches used to model and estimate the energy consumption of video decoding at different levels.

2.5.2.1 Application level

In [78], the authors proposed an energy consumption model for H.264/AVC video decoding based on empirical performance and power models. The performance model estimates the complexity of a frame video decoding (number of cycles) in terms of the aggregation of basic operations executed at each decoding module. On the other hand, the power model was built empirically on Intel and ARM platforms. It estimates the consumed power in terms of the processor clock frequency and a set of platform dependent constant parameters. By using the proposed energy model, the authors estimate the amount of energy to be saved using DVFS and showed that the predicted values are very close to the measured ones.

In [92], the authors investigate the power-rate constrained video adaptation for video streaming applications. Towards this goal, the authors developed a video decoding complexity model with the focus on quality scalability, which can be translated to the power consumption model for mobile processor. In this study, the proposed model allows to estimate the power scalability in term of quality scalability expressed in bit-rate, which makes the power-rate constrained scalable video adaptation analytically tractable. Accordingly, the authors propose to solve the power-rate optimized mobile video streaming problem, so as to maximize the video quality given the limited access

network bandwidth and battery lifetime for mobile devices.

All the above studies consider the clock frequency parameter to estimate the power scalability. However, they make the assumption that decoding time scales linearly with the frequency which is not true as discussed in section 2.5.1.3. As far as we know, there are no studies considering the impact of the memory access latency in the energy consumption model of video decoding.

There exits other video decoding energy consumption modeling studies considering other types of processor architectures. In [93], it is proposed a power consumption model for H.264/AVC video decoding using hardware accelerator on popular mobile platforms. Their proposed model is expressed as the product of the power functions of video spatial resolution (i.e., frame size) and temporal resolution (i.e., frame rate). The authors have demonstrated that the proposed model is applicable to different video hardware accelerator on other platforms.

On the other hand, in [94], the multi-objective energy-video-quality issue is addressed based on experimental measurement on a DSP decoder. This study aims to find the video bit-rate and resolution maximizing the video quality without reducing the mobile device autonomy. In this study, the performances are modeled empirically and the power consumption values are extracted from the used processor data-sheet.

2.5.2.2 System level

In [95], the authors proposed power and energy models of three basic services of the embedded OS : the scheduling, the context switch and inter-process communication (IPC). Their modeling methodology was then applied on a video decoding use case to estimate the energy consumption. In this work, based on energy characterization on a real embedded platform, the authors estimated the energy consumption overhead of the context switch, the IPC and the scheduling as 27%, 4% and 2% respectively.

In [96], the authors propose an energy estimation methodology for a full embedded system including an OS, various processor architecture, memories and bus. Their proposed energy estimation framework is interfaced with different energy modeling tools. For example, *Softexplorer* [97] is used to estimate the energy consumption of program execution on ARM processor and DSP. In this study, the energy consumption of a full video decoding process is estimated with an error rate of 10%.

2.5.2.3 Architecture level

Architecture level Energy modeling aims to evaluate at early design phase the impact of architecture choices on the energy efficiency of video decoding. Usually, simulators are used to estimate the energy consumption of a running application without using physical measurement tools. For example, Wattch [19], based on SimplerScalar processor simulator framework [63], uses a suite of parameterizable power models for different hardware structure. It is based on a per-cycle resource usage count generated through cycle accurate simulations. In the same way and more recently, the McPAT framework [98] is proposed to estimate the energy consumption of multi-core architectures.

For example, in [99], the authors proposed the estimation of the energy saving achieved using the parallelization of video coding over a number of cores ranging from 8 to 24. For this purpose, they use the *Sniper* architecture simulator to estimate performance of the video coding. The obtained timing information are then fed to McPAT simulator to estimate the consumed energy. This study shows also that the use of frequency scaling on the multi-core processor allows 50% energy saving.

2.5.3 Discussion

Table 2.3 summarizes the above cited related works on performance and energy modeling of video decoding.

In case of video decoding performance modeling, the different studies were classified according to the complexity prediction granularity: frame or interval. As discussed in section 2.3.1, the use of each model type depends mainly on whether or not a decoding latency is accepted.

We paid a particular attention to highlight the performance model applicability on realistic system where the off-chip memory access delay makes the performance prediction corresponding to a given frequency non-trivial. This is highlighted in Table 2.3 in the "Off-chip memory access awareness" column.

In this thesis, we propose an average performance model (interval-based) which takes into consideration the impact of off-chip memory access. The added value of the proposed model as compared to the previous studies [90, 41, 91], is that it integrates in addition the video quality parameter. In fact, the proposed model is able to describe how the performance scaling varies in terms of both the processor frequency and the video quality. This may be useful to develop DVFS policies in a context of adaptive

	Performance modeling			
	Granularity	Model type	off-chip memory access awareness	
[71, 72]	Frame	Empirical (Frame size)	no	
[74]	Frame	Empirical (Frame size, type of MB)	no	
[75]	Frame	Statisical (Kalman filter)	no	
[3]	Frame	Statisical (Particle filter)	no	
[77, 78]	Frame	Metadata	no	
[41, 91]	Frame	Empirical (Hardware envent counters)	Yes	
[80]	Interval	Statisical (MA filter)	no	
[81]	Interval	Statisical (WMA filter)	no	
[82]	Interval	Statisical (EWMA filter)	no	
[83]	Interval	Statistical (Autoregressive)	no	
[90]	Interval	Empirical (Hardware envent counters)	Yes	
[This study]	Interval	Empirical	Yes	
	Energy modeling levels parameters			
	Application	System	Architecture	methodology
[78]	Video quality		(ARM,x86)	empirical
[92]	Video quality scalability		ARM	empirical
[93]	Video quality		Hardware codec	empirical
[94]	Video quality scalability		DSP	empirical
[100]		IPC, Context switch, scheduling	ARM	Simulation
[96]		IPC, Context switch, scheduling	ARM/DSP	Simulation
[99]		Processor frequecy	Homogenous multi-core processor	Simulation
[This study]	Video quality	Processor frequency, IPC	ARM, SDP, multi-core ARM, hardware codec	empirical

Table 2.3: Summary of studies on performance and energy modeling of video decoding

video decoding.

We have also referenced different studies for energy consumption modeling of video decoding. One conclusion which can raise from surveying these studies is that application level energy models are usually built empirically based on energy measurement. On the other hand, when it comes to describe the energy consumption into low level system and architecture parameters, it is more suitable to use a simulation frameworks which provides flexibility to tune the parameters and analyze accordingly their impact on the energy consumption. They can thus provide energy estimation for a wide range of configurations. However, simulation frameworks are hard to build and the existing tools may not cover exhaustively a full embedded multimedia system including a hardware codec, multi-core processor and DSP.

In this thesis, we present an experimental study based energy modeling methodologies. The proposed energy model is mainly empirical based, however, thanks to multi-level characterization; it is able to describe the energy consumption in term of comprehensive parameters which can be mapped to properties at application, system and architecture levels. The proposed model achieves a balance between the prediction accuracy and the level of the described details.

2.6 Conclusions

In this chapter we have described the energy saving issue of video decoding from different abstraction levels. We have highlighted also the importance to consider both the performance and the energy consumption properties to save the energy of video decoding. For this purpose, we have surveyed and classified the most important studies in the literature focusing on the characterization and the modeling of the performance and the energy consumption of video applications. The positioning of the different thesis contributions were defined as compared to these surveyed studies.

These contributions will be described in detail in the next four chapters. In chapter 3, 4 and 5, we propose a unified methodology for performance and energy consumption characterization and modeling of video decoding on ARM processors and DSP. In this step, we consider the video qualities ranging from *qcif* to *4cif* resolution. In chapter 6, we describe some possible usages of the obtained results then we will present a preliminary study on the energy efficiency of High Definition video decoding using hardware codec and parallel multi-core processors.

CHAPTER 3

Methodology

Contents

3.1 Introduction 51

3.2 Characterization methodology 52

3.2.1 Video complexity characterization 53

3.2.2 Operating-system level characterization 54

3.2.3 Video-frame level characterization 56

3.2.4 Video sequence level characterization 56

3.3 Modeling methodology 57

3.3.1 Video rate Sub-model 58

3.3.2 Power sub-model 59

3.3.3 Decoding-time sub-model 59

3.3.4 Models validation 60

3.4 Experimental methodology 60

3.4.1 Hardware setup 61

3.4.2 Power consumption measurement 62

3.4.3 Software setup 67

3.5 Conclusion 73

3.1 Introduction

In this chapter, we describe the methodology that we used to model the performance and energy consumption of video decoding. As illustrated in Fig. 3.1, the proposed methodology is composed of three main parts :

- Characterization methodology.
- Modeling methodology.
- Experimental methodology.

The characterization methodology focuses on defining the different experimental test scenarios and the performance/energy consumption metrics to be measured. We ensured that this methodology is independent from any underlying experimental environment to make it generalizable to other hardware/software platforms.

The modeling methodology is based on the obtained measurement results of the characterization part. It consists in building comprehensive performance and energy mathematical models using regression analysis and model fitting.

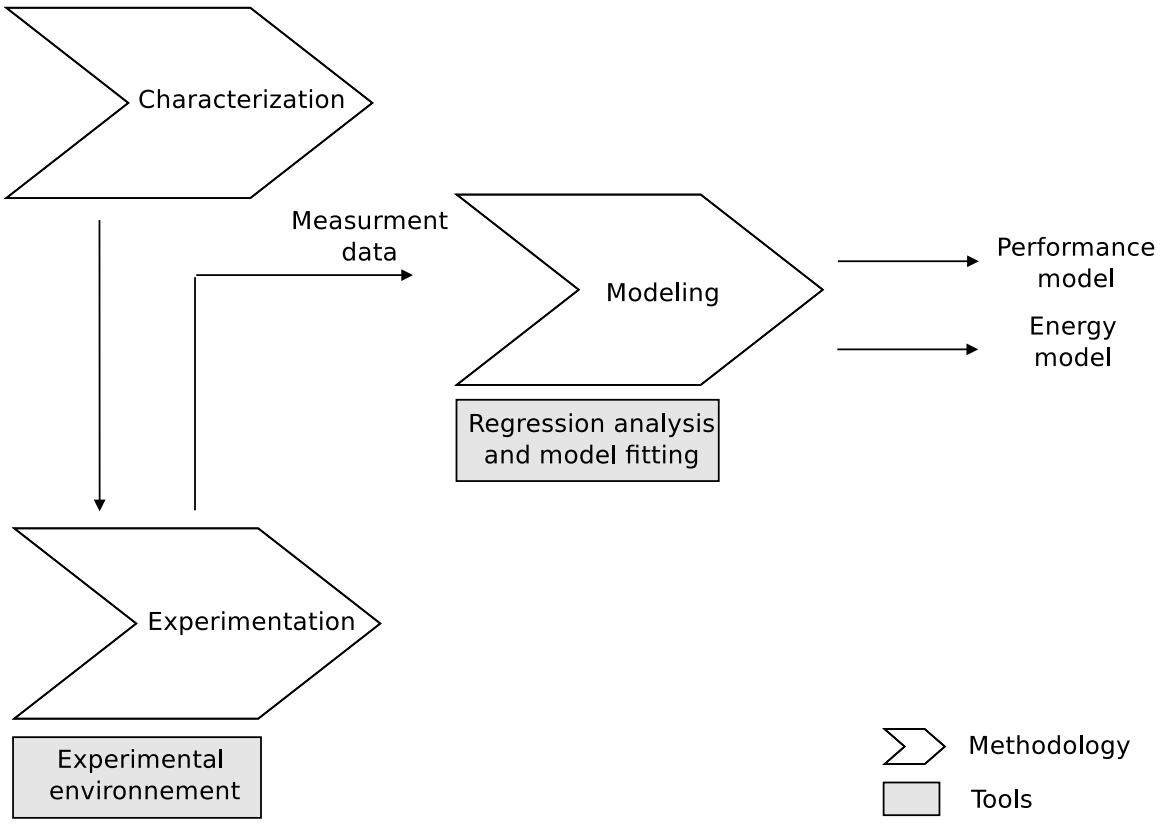

Figure 3.1: Overview of the modeling methodology

Finally, the experimental methodology describes the execution of the characterization methodology on a hardware/software platforms. In this part, we focus mainly on explaining the energy measurement tools, the hardware instrumentation and the software configuration we have used to achieve accurate measurement of the performance and the energy consumption.

In this chapter, we focus on standard definition video decoding on GPP (ARM) and DSP mono-core. High definition video decoding on multi-cores processor and hardware accelerator will be addressed in section 6.

In sections 3.2 and 3.3, we will start by describing receptively the characterization and the modeling parts regardless of any underlying software/hardware platform. Then, we describe in section 3.4 the experimental methodology on a specific target platforms including representative low power SoCs, embedded OS, video decoder and measurement tools.

3.2 Characterization methodology

We consider a video decoding process on a given hardware architecture containing a GPP and a DSP within a *SoC*. Each processor supports DVFS and different discrete clock frequencies. The value of the clock frequency is denoted f.

A video sequence v consists of a set of video frames. It is characterized by a displaying-rate $R_{display}$ (expressed in Frames/s), a bit-rate r (expressed in Kb/s), a spatial resolution s (size of a frame in terms of pixels number) and a complexity c, which represents complexity characteristics of the video. The video complexity may be temporal (motion intensity from a frame to another) or spacial (texture of frames).

The videos are encoded using H.264/AVC, a widely adopted coding standard [101]. H.264/AVC allows encoding a video according to different qualities and profiles. A quality is defined by a bit-rate and a resolution. On the other hand, a profile is defined by the set of tools used in the coding algorithm. In our experimentations, we used the restricted base profile; a basic coding scheme which is suitable for use in processing-resources constrained embedded devices [101].

We define a video decoding configuration as a set of constant and variable parameters. The constant parameters are: the type of the processor architecture, the video resolution and the complexity. For a given video sequence and a mobile device, these parameters are not supposed to change. On the other hand, the variable parameters

are the video bit-rate r and the processor clock frequency f. The bit-rate may vary depending on the network bandwidth capabilities and the processor frequency is driven by system frequency scaling policy.

The characterization of the performance and energy consumption of video decoding aims to evaluate and analyze the impact of the above-cited parameter on the performance and energy consumption of video decoding. For this purpose, we follow the different characterization steps listed below and illustrated in Fig. 3.2.

1. Video complexity characterization.
2. Operating system level characterization.
3. Video frame level characterization.
4. Video sequence level characterization.

The first step is achieved at the video encoding phase while the remaining ones are executed at the video decoding phase (on an embedded hardware platform).

3.2.1 Video complexity characterization

The main objective of this step is to extract information related to the complexity of the used video. Unlike the video quality properties such as the bit-rate or the resolution, the video complexity information is not available to the decoder.

However, the video encoder is able to evaluate the video scene and motion complexity at the encoding phase, more precisely, at the transform phase (refer to section 2.2.1). In fact, the more a video is complex, the less compact are the transformed coefficients.

If the encoder is configured to encode the video in a constant bit-rate (CBR), it tunes dynamically the quantization parameters to fit the targeted bit-rate (refer to section 2.2.1). The more a video has a complex scene and motion, the less compact are the transformed coefficients and the higher are the values of the used quantization parameters[1].

Thus, to extract the video complexity information, we kept trace of the mapping between each bit-rate and the average quantization parameter used by the encoder to

[1]Increasing the value of the quantization parameter allows to increase the number of transform coefficients set to zero

encode the videos. We define the average quantization parameter qp_{avg} used to encode a video sequence as :

$$qp_{avg} = \frac{\sum_{i=1}^{N} qp_i}{N} \tag{3.1}$$

where qp_i is the quantization parameter associated to the frame i and N is the number of frames. Therefore, the information about the video complexity can be provided by the couple $c = (r, qp_{avg})$ where r is the bit-rate associated to the video.

3.2.2 Operating-system level characterization

This step is a kind of a calibration operation which aims to measure the power consumption related to the different states of the used processors regardless of any video decoding process.

As introduced in section 2.2.2.1, modern processors are characterized by two kinds of power states: P-states and C-states. The P-states are voltage/frequency pairs that set the speed and power consumption of the processor. The transition between the different P-states is implemented using the DVFS technique. On the other hand, C-states are idle power saving states, in contrast to P-states, which are execution power saving states. During a P-state, the processor is still executing instructions, whereas during a C-state, the processor is idle, meaning that nothing is executing.

Within a video decoding process, the operating system may intervene to make the used processors transiting to *idle* or lower power states in order to save energy. This depends on the dynamic power management configured in the operating system and the impact that it can have on the performance of the decoding process.

To investigate performance and energy considerations related to those states, the power consumption of both the GPP processor and the DSP at the different supported C-states and P-states are measured regardless of any video decoding process. The objective is to have a set of reference power consumption values that helps to understand and quantify the energy consumption of different video decoding process phases at the frame-level fine granularity (see Fig. 3.2).

Moreover, at this phase, the overhead due to the transition of the processor to lower power mode is measured in order to evaluate its impact on the performance and decide whether to use or not this kind of power saving during the video decoding process.

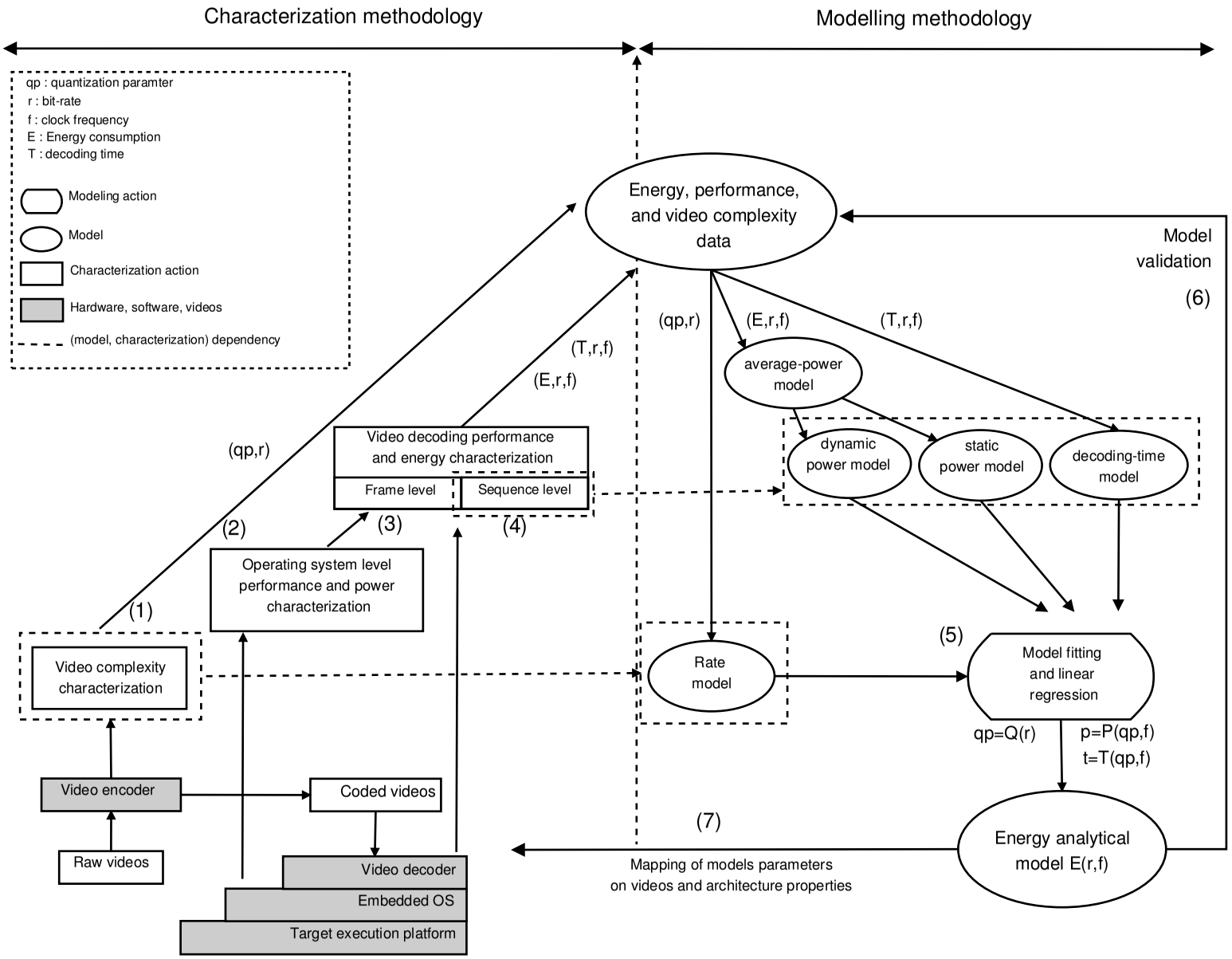

Figure 3.2: Characterization and modeling methodology

3.2.3 Video-frame level characterization

This level relies on the preceding operating system level characterization, which helps in identifying transitions of the processor to *active/idle* phases during the decoding process. The objective of this step is to understand how the processing elements are used and where goes the energy when decoding a single frame in order to explain the global performance and energy consumption of video decoding. This is particularly useful to understand complex video decoding processes involving both GPP and DSP.

For this purpose, the elementary video frame decoding is characterized at a fine granularity in terms of system metrics such as the amount of buffer transfers, the GPP/DSP communication latency.

We also evaluated during this step the overhead of video decoding when using both GPP and DSP. As discussed in section 2.4.1.2, decoding a video comes down to a sequence of frame processing periods. Each period is composed of a set of actions consisting in retrieving the coded frame from the input buffer, decoding it and transferring it to the output buffer. The actual frame decoding step is thus a sub-part of a frame processing period. In the frame processing period, we define the overhead as the actions which are not a part of the frame decoding step. For example, in case of DSP decoding, this may be related to cache coherency maintenance and inter-processor communication.

3.2.4 Video sequence level characterization

In this step, the number of decoded frames per second (FPS) of H.264/AVC decoding of the overall video sequence is evaluated on both GPP and DSP. This evaluation covers different bit-rate, resolution, and processor clock frequency configurations. The considered performance criteria is the video displaying rate of the decoded video. In fact we considered that a decoding rate which is lower than the displaying rate of the coded video is not sufficient for playing-back the video with respect to real-time constraints.

The overall energy consumption is then calculated by multiplying the sum of the elementary measured power values by the decoding time. The average energy per frame (mJ/frame) is then obtained by dividing the overall energy by the total frame number.

Each used video was decoded using GPP and DSP processors. This decoding was repeated for all the available clock frequencies. For each bit-rate, resolution and clock

frequency, the decoding time and the energy consumption were measured.

As shown in Fig. 3.2 (block 4), the results of this phase are triplet data sets (T, r, f) and (E, r, f) describing the decoding time and energy variation in terms of r and f. These data are used in the modeling phase, described hereafter, to build a power and performance model for video decoding.

3.3 Modeling methodology

In our modeling methodology, we used a top-down model decomposition approach in which the energy model is decomposed as a function of two sub-models: a decoding-time model T and an average power model P_{avg}.

$$E(r, f) = P_{avg}(r, f).T(r, f) \tag{3.2}$$

P_{avg} is then decomposed into a dynamic $P_{avg_{dyn}}$ and a static $P_{avg_{stat}}$ power models :

$$P_{avg}(qp_{avg}, f) = P_{avg_{dyn}}(r, f) + P_{avg_{stat}}(r, f) \tag{3.3}$$

As discussed previously in section 3.2.1, we propose to use the mapping between the bit-rate r and the average quantization parameter (qp_{avg}) to add the video complexity information to our model. For this purpose, we first use the video (qp_{avg}) instead of the bit-rate (r) as a model parameter for the power and performance models. Therefore, the model described in Eq. (3.2) giving the energy in terms of the clock frequency f and the bit-rate r becomes:

$$P_{avg}(qp_{avg}, f) = P_{avg_{dyn}}(qp_{avg}, f) + P_{avg_{stat}}(qp_{avg}, f) \tag{3.4}$$

To obtain an energy model as a function of the clock frequency and the bit-rate, we used a rate model Q which describes qp_{avg} in terms of the bit-rate r and some video complexity related coefficients.

$$E(r, f) = (P_{avg_{stat}}(Q(r), f) + P_{avg_{dyn}}(Q(r), f)).T(Q(r), f). \tag{3.5}$$

The energy model described in Eq. (3.2) is thus decomposed into: 1) a rate model, 2) a static/dynamic power model and 3) a time model. Then, based on these sub models, the energy model is built. A model fitting and regression analysis is performed

on the characterization results to develop a mathematical form for the related models (see block (5) in Fig. 3.2).

To validate the accuracy of each model, we compare the data predicted by the models with the measured ones (see block (6) in Fig. 3.2). Finally, based on the characterization results, we try to map the abstract parameters of the developed models onto the properties of the used platform (see block (7) in Fig. 3.2). We will show in section 5 how this can help to generalize the developed models on other platforms.

Details on development/validation of the sub-models and the experimental measurement are described below.

3.3.1 Video rate Sub-model

In order to express the video bit-rate as a function of the quantization parameters, we used the model proposed in [102]. The authors of this paper assume that the video bit-rate can be described as follows:

$$r = (\frac{q}{q_{min}})^{-a}.r_{max} \tag{3.6}$$

where q is the step-size which is defined as follows [101]:

$$q = 2^{(qp_{avg}-4)/6} \tag{3.7}$$

a is an exponent which represents how fast the video rate changes in term of the step-size parameter. q_{min} is the lowest step-size parameter used to encode the video with the highest bit-rate r_{max}.
By substituting Eq. (3.7) in (3.6) we obtain the following rate model :

$$qp_{avg} = 4 + 6.ln_2(q_{min}.(\frac{r}{r_{max}})^{-1/a}) = Q(r) \tag{3.8}$$

Using the values (r, qp_{avg}) returned by the encoder, the parameters q_{min}, r_{max} and a can be calculated using model fitting for each coded video.

3.3.2 Power sub-model

The measured power consumption is the sum of the static and the dynamic power values. Actually, the static power is defined as :

$$\begin{aligned} P_{static} &= L_g(V_{dd}I_{sub} + |V_{bs}|I_j + V_{dd}I_g), \\ I_{sub} &= K_3 e^{K_4} V_{dd} e^{K_5} V_{bs}, \\ I_g &= K_6 e^{K_7} V_{dd} \end{aligned} \tag{2.2}$$

where L_g , V_{bs} , I_j , K_3 , K_4 , K_5 , K_6 and K_7 are constants which depend on the circuit fabrication technology. To model a static power consumption of a microprocessor, one solution is to define all these constant parameters by fitting the measured values with the above described models.

In this work, we consider the static power as a platform dependent parameter. We do not aims to model it as a function of the above cited low level details. Instead, we estimate it based on the processor data-sheet [103] providing the value of the static power corresponding to each voltage level.

The dynamic power consumption values are obtained by subtracting the static power consumption from the total power consumption. Knowing the V and f values, the obtained data are then fitted with the model $C_{eff}.V^2.f$ to extract the C_{eff} parameter (see Eq. 2.3).

In some cases, the static and dynamic power values cannot be known separately. For example, some processor data-sheet do not provide information about the static power consumption. To overcome this issue, the total power consumption (static + dynamic) can be fitted with the following model as suggested in [78].

$$P_{tot} = af^b + c \tag{3.9}$$

Both of the above described methods will be used in our study.

3.3.3 Decoding-time sub-model

In our methodology, the objective of the decoding-time modeling is to estimate the execution time of video decoding in terms of video quality and processor frequency. As proposed in [78], one approach consists in estimating the decoding complexity in term

of processor cycles C, then the decoding time t is calculated by dividing the estimated cycles by the processor frequency as described in Eq. 3.10 :

$$t = T(C) = \frac{C}{f} \tag{3.10}$$

However, as highlighted in section 2.3.1.3, the performance scaling when varying the processor frequency highly depends on the off-chip memory latency. Thus, the processor cycles cannot be converted directly into decoding time.

For this reason, to develop the video decoding time model T, we have measured the decoding time corresponding to each video quality and processor frequency. Then, we used an observation on the experimental results (see section 4.3.3.1) which reveals a linear relation between $1/t$ (where t is the decoding time) and both the clock frequency f and the quantization parameters qp_{avg}. This linear relation was validated using a multi-linear regression of $1/t$ in terms of f and qp_{avg}.

3.3.4 Models validation

To validate the built models, we use the R^2 statistical metric to verify the accuracy of the predicted data as compared to the measured ones. If we suppose m_i are a set of measured data and p_i are the corresponding predicted values by a given model M, then :

$$R^2 = 1 - \frac{\sum_i (m_i - \overline{m_i})^2}{\sum_i (m_i - p_i)^2} \tag{3.11}$$

R^2 will give information about the goodness of fit of a model. In other words, it tells how well a given model approximates the real measured data. The more R^2 is close to 1, the higher accurate is the related model.

3.4 Experimental methodology

As highlighted in the beginning of this chapter, we propose in this section a methodology to execute the characterization steps on a given software/hardware platform. In addition to the description of the hardware and the software we have used in the experimentation, we explain the hardware instrumentation and software configurations we have used to make the power consumption measurement as accurate as possible.

3.4.1 Hardware setup

In our experimentations, we have used the Open Multimedia Applications Platform (OMAP) SoC manufactured by Texas Instrument (TI). This is motivated by our desire to use hardware platforms which are representative of those used in mobile devices. In fact, OMAP SoCs were used in many popular mobile devices as illustrated in Table 3.1.

In what follows, we describe the MistralEVM3530 and the Panda development boards which contain the OMAP 3530 and OMAP 4460 respectively.

3.4.1.1 MistralEVM3530

The power measurements were conducted on the OMAP3530EVM board (shown on Fig. 3.3) containing the low-power OMAP3530 *SoC*. This *SoC* is based on a 65-nm technology and consists of a Cortex A8 ARM processor supporting ARMv7 instruction set and a TMS320C64x DSP.

The OMAP3530 supports 6 P-states and 7 C-states. The P-states correspond to different frequencies ranging from 125 MHz to 720 MHz for the ARM and from 90 MHz to 520 MHz for the DSP. On the other hand, the C-states ranges from C_0 to C_6. C_0 is the active state, C_1 is the *idle* state (clock gating) and C_2 is the state corresponding to the deactivation of almost all the SoC blocks retaining the data in registers and cache memory. Starting from C_3 state and above, the SoC blocks are switched off and their data are saved in the external memory.

SoC Family	Mobile device
OMAP3X	Nokia N900
	Motorola Droid
OMAP4X	Samsung Galaxy Nexus
	Amazon Kindle Fire
	Archos Tablet
	Motorola Droid 3

Table 3.1: Mobile devices using OMAP SoCs

3.4.1.2 PandaBoard

The PandaBoard is a low-power and low-cost single-board computer development platform based on the Texas Instruments OMAP4430 SoC. In our experimentations, we used the newer version (PandaBoard-ES) based on the OMAP4460 SoC. The OMAP4460 SoC is based on a 45nm technology and contains a dual-core Cortex A9 ARM processors. Each core supports four clock frequencies: 350 MHz, 700 MHz, 920 MHz and 1.2 GHz.

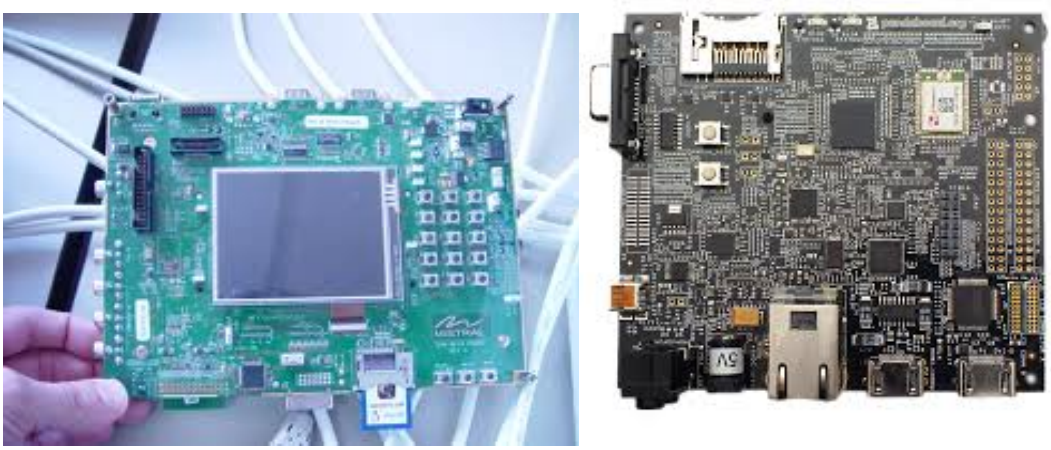

Figure 3.3: Mistral EVM3530 Board (left) and PandaBoard ES (right)

3.4.2 Power consumption measurement

In this section, we describe the used material and the achieved instrumentation to measure accurately the energy consumption on the above described boards.

3.4.2.1 Open-PEOPLE platform

The power consumptions is measured using the Open-PEOPLE platform [23], a multi-user and multi-target power and energy estimation and optimization platform. As illustrated in Fig. 3.4, the platform includes a set of accurate power measurement instruments (see Table 3.2) hosted in a rack in which we can integrate the embedded boards that we want to measure.

The advantage of using Open-PEOPLE platform is that the power measurement can be achieved remotely thanks to a set of software hosted in a server which automates the management of the measurement equipment and the embedded boards.

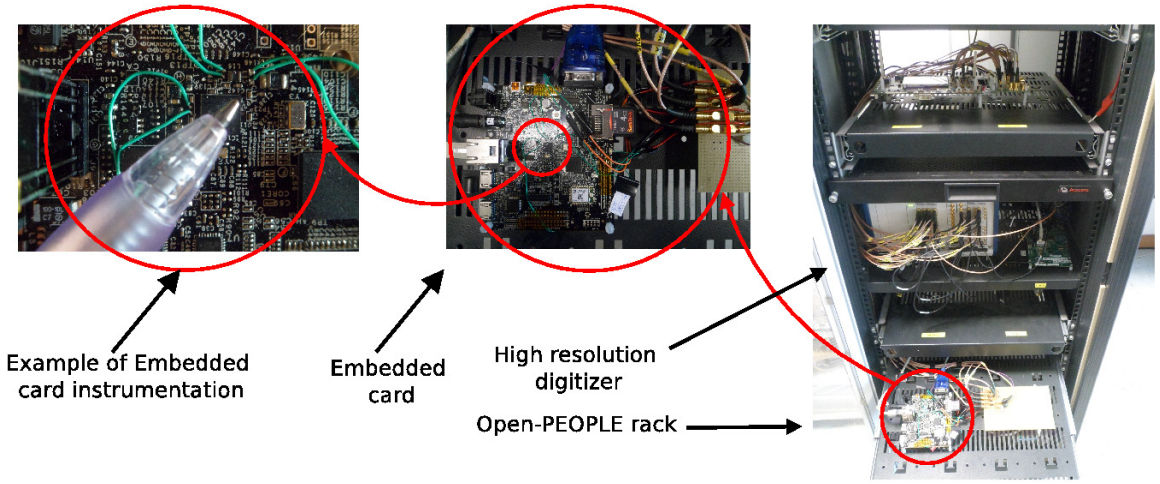

Figure 3.4: A view on Open-PEOPLE rack

The user who wants to utilise the platform should use a dedicated software. Its role is to manage the authentication process with the server, upload the power measurement test case (an archive file), monitor its execution status then download the results (see Fig. 3.5).

The test case archive is a self-contained archive consisting of the actual power measurement test case. It includes an XML configuration file and eventual additional resources. In the configuration file, the user can specify:

- the targeted embedded board and the measurement points.
- the sampling rate
- the cross-compiled binaries for remote execution.
- the cross-compiled Linux kernel to be executed remotely.

Any additional resources (libraries, data files, etc) can be added to the archive to be installed on the remote target. The platform is able to integrate any board using Linux OS and *uboot* boot loader.

Equipment	Sampling rate	Precision	Description
N6705A DC	100 KS/s	1.5 mV/15 μA	High precision Power Analyzer
NI PCI-4472	100 KS/s	1.19 μV	High precision digitizer
NI PCI-5105	60 MS/s	7.3 mV	High density digitizer
Agilent M9149A	N/A	N/A	Switch multiplexer

Table 3.2: Power measurement materiel

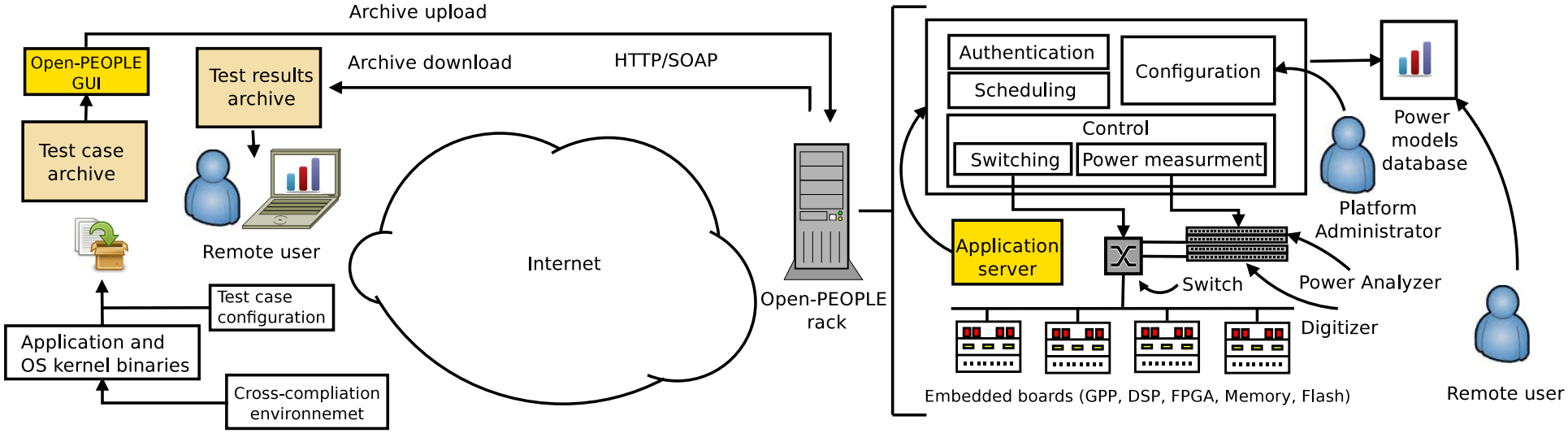

Figure 3.5: Open-PEOPLE platform architecture

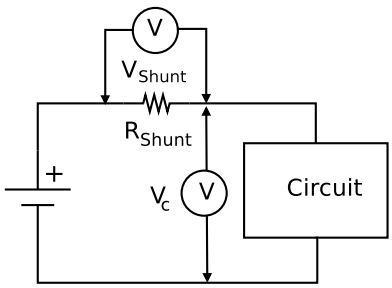

Figure 3.6: Power consumption measurement using a shunt resistor

In this thesis, the use of the Open-PEOPLE platform was a real added value. In fact, our experimentations required huge amount of tests corresponding to the combination of different processor architecture, processor frequencies, video bit-rate and resolution. In addition, an extensive software configuration was necessary to install DSP driver, customize and profile the Linux kernel and the *GStreamer* video decoder. Thanks to the flexibility of the Open-PEOPLE platform, all these configurations were achieved remotely from offices located in Université de Bretagne Occidentale in Brest (France).

3.4.2.2 Power consumption measurement methodology

The power consumption depends on the voltage V and the current intensity I.

$$P = V.I \tag{3.12}$$

To measure the power consumption of a given component, we used a shunt resistor in series with the component we want to measure. Then we measured V_{shunt} and V_c, the voltage across a shunt resistor R_{shunt} and the component respectively (see Fig. 3.6). The current intensity is $I = \frac{V_{shunt}}{R}$, therefore, $P = \frac{V_c.V_{shunt}}{R}$. To avoid a high voltage drop across the target circuit (which may affect a correct operating of the circuit), the shunt resistor should have a low value (usually, some tens of milliOhms). Consequently, the voltage around R_{shunt} is very low. It thus requires a very sensitive equipment in order to be measured accurately.

The energy consumption is the amount of consumed power within an time interval.

$$E(t) = \int_0^t P(t)\mathrm{d}t \tag{3.13}$$

In practice, to measure the energy consumption, the integral formula is approximated

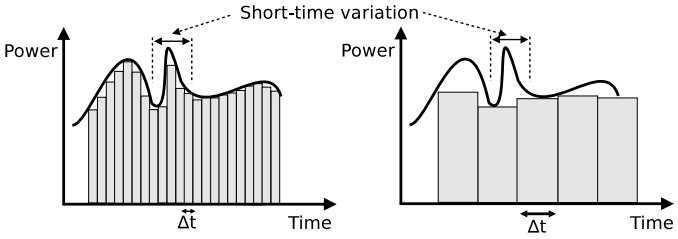

Figure 3.7: Sampling rate of the energy consumption measurement

by a discrete sampling measurement.

$$E(t) = \sum_{i=0}^{n} P_i.\Delta t \tag{3.14}$$

Δt is the sampling interval and P_i is the measured power at the i^{th} interval. The higher is the sampling rate $(1/\Delta t)$, the more accurate is the energy consumption measurement. As illustrated in Fig. 3.7, measuring with a high sampling rate allows evaluating accurately the energy consumption with a short-time variation. In the field of embedded systems, this may correspond to events like a frequency switching, a context switch in process scheduling or inter-processor communication in SoC. As we will discuss in section 4.3.2.2, the use of high sampling rate will help in measuring the energy overhead in inter-processor communication.

3.4.2.3 Boards instrumentation

We describe hereafter the different steps for integrating physically the target boards in the Open-PEOPLE platform.

Mistral OMAP3530EVM

The OMAP3530EVM board is already instrumented for power consumption measurement. It has an on-board shunt resistors and provides jumpers for measuring the voltage. These shunt resistors allow to measure the power consumption of the (ARM + DSP) processors, the memory and the core of the SoC.

Because there is one shunt resistor for the ARM and the DSP, the measured power consumption is the sum of the power consumptions of both the ARM processor and the DSP. In case of ARM video decoding, the measured power represents the ARM dynamic consumption plus the ARM and DSP static power. In case of DSP video decoding, both the ARM and the DSP are involved. In fact, the ARM controls the

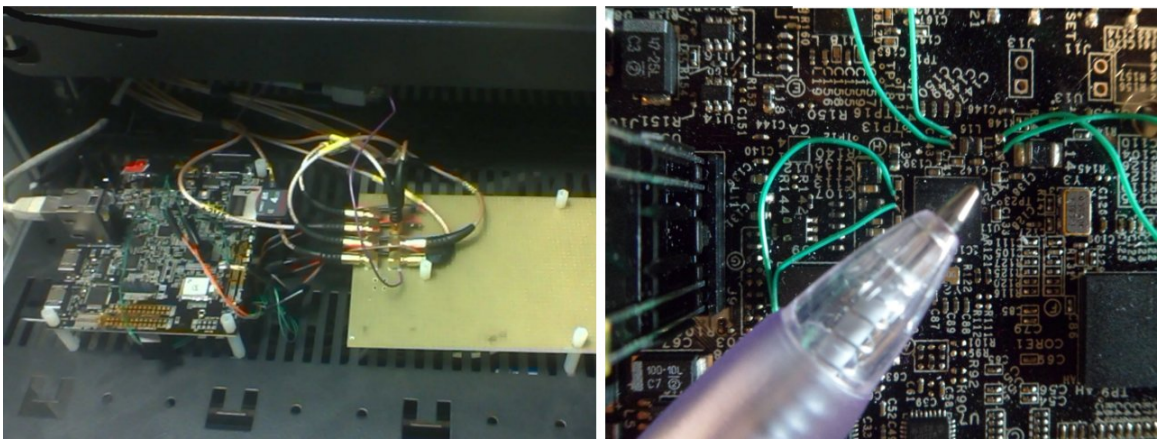

Figure 3.8: PandaBoard-ES Instrumentation

DSP which executes the actual video decoding process. The measured power is thus the sum of the static and the dynamic power values of both the ARM and the DSP. In what follows, P_{static}, is the sum of the ARM and the DSP static power. The ARM and the DSP dynamic powers are denoted $P_{dyn_{arm}}$ and $P_{dyn_{dsp}}$, respectively. The total power consumption of the ARM and the DSP is denoted P_{tot}.

PandaBoard

The Pandaboard does not provide initially power consumption measurement points. Thus, some instrumentation was done on this board to allow power consumption measurement. Firstly, in the board schematics, we looked for some appropriate resistors to plug the digitizer sensors. Only some inductors between the power supply and the ARM, the memory and core subsystems were found. The card was breakdown by removing the inductors and replacing them by the equivalent inductors plus a shunt resistor with well-known value (0.05 Ohm). These new inserted inductors and shunt resistor were placed on external annex board where the digitizer sensors were plugged around the inserted shunt resistors. See Fig. 3.8.

3.4.3 Software setup

We describe hereafter the different software tools we have used in our experimentations.

3.4.3.1 Video encoder

To encode the used test video sequence, we used *x264* encoder [104], a popular and open source H.264/AVC video encoder.

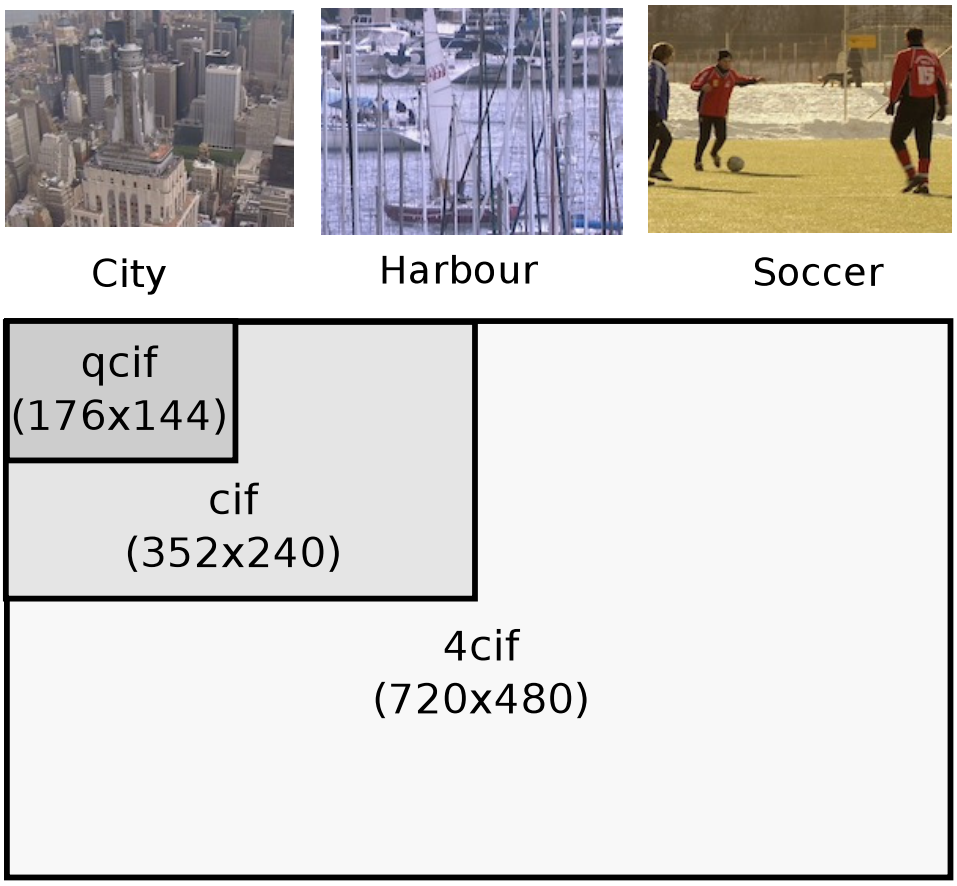

Figure 3.9: Used video test sequences

A set of representative raw video sequences (with different scene complexities, bit-rate and resolutions) were selected and encoded with different constant bit-rates using the H.264/AVC encoder. Thus, the following well-known YUV raw video sequences were used: City, Soccer and Harbor were encoded in H.264/AVC base-line profile (see Fig. 3.9). These sequences represent respectively a low, a medium and a high video complexity. As illustrated in Fig. 3.9, each video is available in three resolutions (*qcif* (176x144), *cif* (352x288) and *4cif* (704x576)).

After each coding operation, the values of qp used in the encoding process were extracted from the encoder log and the average quantization parameter qp_{avg} was then calculated accordingly.

3.4.3.2 Operating system

On this hardware platform, the Linux operating system version 2.6.32 provided in the Mistral EVM BSP (Board support package). It is a standard Linux OS integrating additional drivers for some specific devices such as the DSP and the LCD screen. The

DPM and DVFS drivers installed in the kernel are also specific to the OMAP3530 SoC.

Dynamic power management

To investigate the power consumption at diffrent C-state levels of both the ARM processor and DSP, we configured the *cpuidle* driver in the Linux kernel. Then, in order to trigger transition to low power mode of the ARM processor, we execute the following script [105] :

```
$ echo 1 > /sys/devices/platform/omap/omap_uart.0/sleep_timeout
$ echo 1 > /sys/devices/platform/omap/omap_uart.1/sleep_timeout
$ echo 1 > /sys/devices/platform/omap/omap_uart.2/sleep_timeout
$ echo 1 > /sys/kernel/debug/pm_debug/sleep_while_idle
$ sleep 1
```

The first three lines of the code allow to disable the UART device which prevents the processor to *idle*. The last line tells to the driver to transit the processor to deep low power mode when *idling*.

The DSP was running the *DSPBios* operating system and was driven from the Linux/GPP side using a driver named *DSPlink*. The DSP power management feature (activation/deactivation of the DSP) was handled using Local Power Manager (LPM) driver [106].

```
$ /usr/share/ti/ti-lpm-utils/lpmON.xv5T
$ /usr/share/ti/ti-lpm-utils/lpmOFF.xv5T
```

The *lpmON.xv5T* and *lpmOFF.xv5T* binaries allow to activate and deactivate the DSP processor respectively.

Frequency scaling

To control the ARM and the DSP clock frequencies, we used *cpufreq* driver for the OMPA3530 SoC [107]. The user-space governor, a frequency scaling policy allowing to control the clock frequency at the application level, was activated. A file system interface is provided by the kernel to allow the selection of the frequency scaling policy and the frequency value. The below script contains the commands to select the user-space governor and to set up the processor frequency.

```
$ echo userspace > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
$ echo 125000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed
```

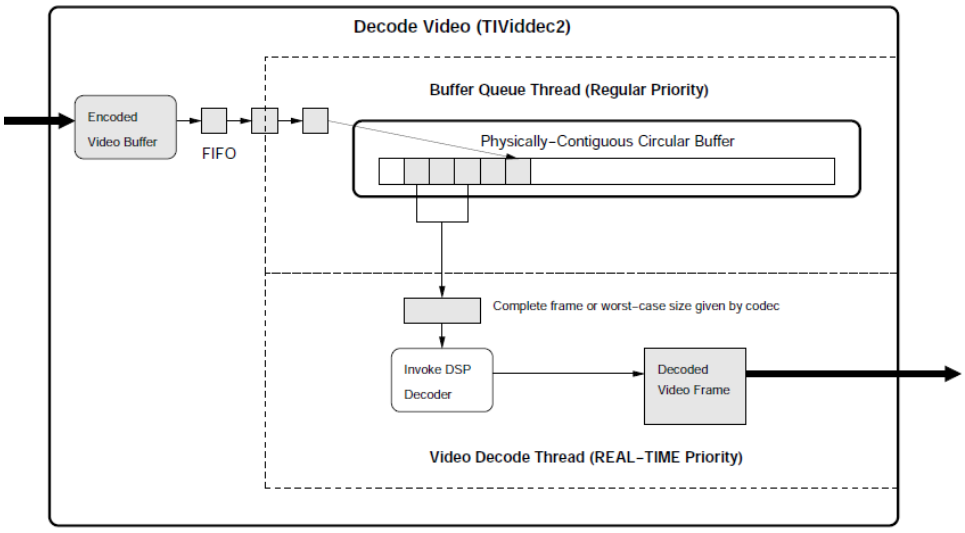

Figure 3.10: GStreamer DSP video decoding plug-in

```
$ echo 250000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed
$ echo 500000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed
$ echo 550000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed
$ echo 720000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed
```

The corresponding DSP frequencies (90 MHz, 180 MHz, 360 MHz, 400 MHz, 430 MHz, 520 MHz) are set automatically by the driver when setting the ARM frequency.

3.4.3.3 Video decoder framework

On the previously described embedded boards, the H.264/AVC video decoding task was achieved using *GStreamer* [108], a multimedia development framework. The use of GStreamer permits an accurate GPP/DSP decoding comparison thanks to its modular design allowing to execute both the GPP and the DSP video decoders into the same software environment. The ARM decoding, was performed using *ffdec_h264*, an open-source plug-in based on the widely used *ffmpeg/libavcodec* library compiled with the support of the NEON SIMD instruction set. According to [109], NEON boosts performance by 60-150% for video codecs. For DSP decoding, we used *TIViddec2*, a proprietary GStreamer H.264/AVC baseline profile plug-in provided by *Texas Instrument*. Its internal design is illustrated in the right side of Fig. 3.10. The video frames are moved from the encoded video buffer (input buffer which contain the coded frames) to a circular buffer by a queuing thread. The video decoder-thread invokes the DSP decoder via the *dsplink* DSP driver. The DSP codec executes a cache invalidation

operation so that it can see the right data in the shared memory, decodes the frame and transfers it to the decoded frame buffer using DMA. Table 3.3 gives a complete summary of the hardware and the software setups.

Table 3.3: Hardware and software setup summary

Applications	GStreamer	ARM plug-in	ffdec_h264
		DSP plug-in	TIViddec
	Operating System	GPP	Linux 2.6.32
		ARM	DSPBios
		ARM/DSP DVFS driver	cpufreq
		DSP driver	DSPLink
		DSP power management driver	LPM
MistralEVM3530	DSP	Model	TMS320C64x
		Frequencies (MHz)	90, 180, 360, 400, 430, 520
	ARM	Model	Cortex A8 + NEON
		Frequencies (MHz)	125, 250, 500, 550, 600, 720
	SoC	Model	OMAP3530
		Voltages levels	0.975, 1.05, 1.2, 1.27 , 1.35, 1.35
PandaBoard	ARM	Model	Cortex A9 + NEON
		Frequencies (MHz)	350, 700, 920, 1200
	SoC	Model	OMAP4460
		Voltages levels (V)	Unknown

Elimination of the I/O interference

GStreamer is multi-threaded application. The buffering operations (transfers of the video frames from the file system to the input buffer in the memory) may interleave with the video decoding operations. This makes the performance and the energy consumption related only to the decoding phase difficult to measure. In fact, the file system I/O operations may impact the accuracy of the measured video decoding time and power consumption. To avoid this situation, we developed a customized video decoder using the *GStreamer* API. As shown in Fig. 3.11, the decoding thread was kept initially in a pause state while the video stream was copied in an input buffer (*GStreamer queue* element) by the buffering thread. The transmission of the QoS messages containing the buffer level information were activated in a *GStreamer queue* element. These messages were then monitored by a handler that wakes-up the decoding thread when the entire video stream is held in the input buffer. On the other hand, the

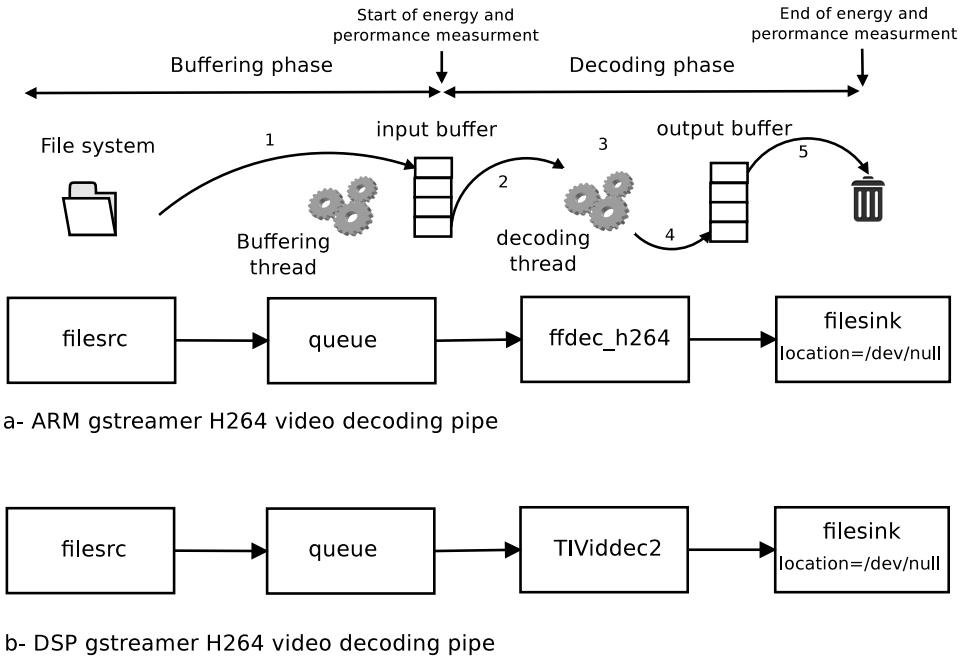

Figure 3.11: GStreamer GPP and DSP video decoding pipes

decoded frames were redirected to */dev/null* in order to disable the processing related to the frames copy from the output buffer to the display driver memory or to a file.

In Fig. 3.11, the GStreamer pipes reproducing the above experimentations are shown. *filesrc*, *queue* and *filesink* are the GStreamer modules allowing to read a video file, buffer it into a memory and send it to a destination location. These modules are shared between the GPP and DSP pipes and we suppose that they generate the same load in the DSP and the GPP decoding process. The measured phases are steps 2, 3 and 4. Step 5 is negligible since no video frame copy is performed there due to the use of redirection to */dev/null*.

Overhead calculation

To calculate the time overhead of video decoding, the ARM frame decoding time was measured by displaying timestamps information at the beginning and at the end of the function named *avcodec_decode_video2()* in *libavcodec* library. In case of DSP decoding, the tracing was enabled in the DSP API as described in [110]. Enabling the tracing does not impact the frame decoding time since no debug information is displayed within the decoding process. The timing information were provided as a function of the number of clock cycles which were converted into time duration by dividing it through the DSP clock frequency [110].

Similarly, the energy overhead is calculated by subtracting the sum of the frame decoding energy values from the overall video decoding energy. The frame decoding energy is obtained by multiplying the average frame decoding power consumption by the frame decoding time already calculated. In fact, we have noticed that the average power consumption corresponding to the frame decoding phase is constant for a given video resolution.

3.5 Conclusion

In this chapter, we described an end-to-end characterization and modeling methodology for performance and energy consumption of video decoding on GPP ARM processor and DSP. The methodology starts from the encoding phase until the decoding phase while considering different levels of abstraction.

The proposed methodology is generic and can be applied on any video decoding platform. As a study case, we have shown how to execute it on the low power OMAP3530 and OMAP4460 SoC including both an ARM GPP and a DSP. The obtained results on executing the characterization methodology on the OMAP3530 will be discussed in chapter 4. In chapter 5, we will discuss how to build a comprehensive performance and energy models from the characterization results. We will also show how to make use the conclusions from the experimentations conducted on the OMAP3530 platform to build fast performance and energy models on other architectures. We will consider the OMAP4460 as a study case.

CHAPTER 4

Performance and Energy Consumption Characterization

Contents

4.1 Introduction 75

4.2 Video complexity characterization 75

4.3 Video decoding performance and energy characterization 76

4.3.1 Operating-system level 77

4.3.2 Video-frame level 83

4.3.3 Video-sequence level 88

4.4 Conclusion 94

4.1 Introduction

The objective of this chapter is to describe the results of the performance and the energy consumption characterization of video decoding executed according to the methodology discussed in chapter 3.

As illustrated in Fig. 4.1, the characterization methodology is executed at both the encoding and the decoding phases while considering different levels of abstraction.

At the encoding phase, we focus on characterizing the video scene complexity of the used sequences regardless of any performance or energy consumption metrics. The objective is to extract relevant video complexity information for the used video sequences.

On the other hand, at the decoding phase, we focus on collecting data related to the performance and the energy consumption of video decoding while considering different videos and processing configurations (video quality, processor frequency) executed on both ARM processor and DSP.

As illustrated in Fig. 4.1, the output of this characterization phase will be used in the modeling phase which will be discussed in the next chapter.

4.2 Video complexity characterization

To characterize the video complexity, the quantization parameters used to encode a video using various bit-rates are extracted from the encoder. The average quantization parameter is then calculated based on the per-frame *qp*. The obtained values represent

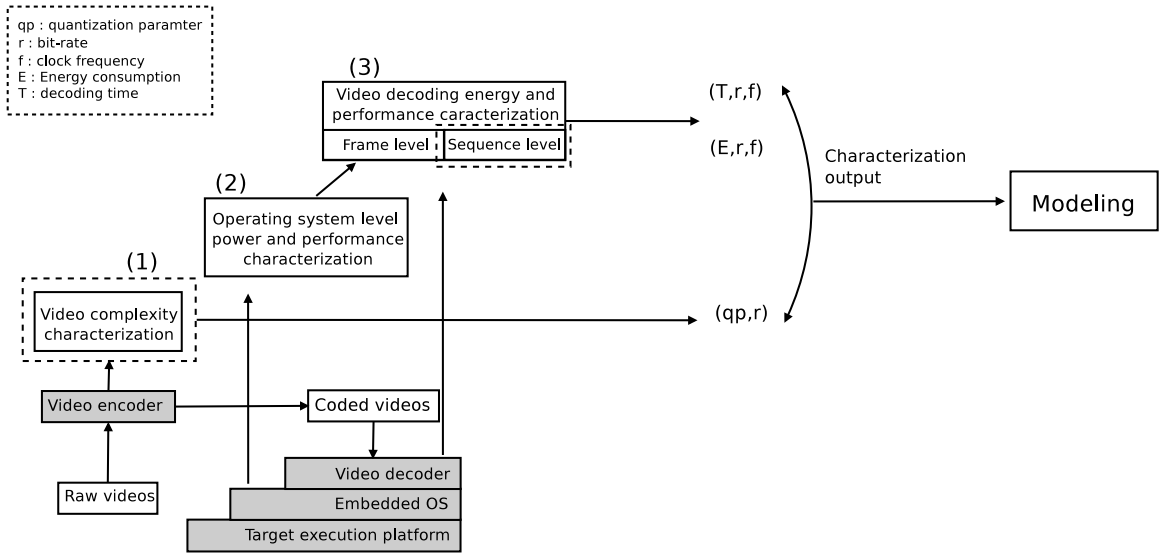

Figure 4.1: Characterization methodology

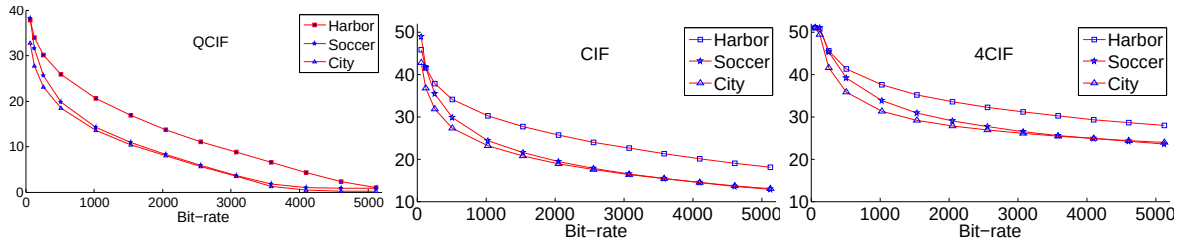

Figure 4.2: Mapping between the video qp_{avg} and the bit-rates

the overall complexity of the complete used video sequences.

Table 4.1 shows the correspondence between the resolution, the bit-rate, and the quantization parameters extracted for the considered sequences. The plots of the data in this Table (see Fig. 4.2) show that for the same resolution and bit-rate, the average quantization parameter qp_{avg} of the City, Soccer and Harbor videos are in an increasing order. This is explained by the fact that the Harbor video sequence is more complex than Soccer which is more complex than City[1]. In fact, the encoder considers dynamically the variation in the scene complexity and accordingly sets the used quantization parameters. The rule is that, given a constant target bit-rate, the more complex the video is, the lower is the used quantization parameters.

Thus, the couples (r, qp_{avg}) extracted from the encoder and associated to each sequence can provide information about the scene complexities of the used videos. At this step, we keep trace of this information and will use it later when modeling both the performance and the energy consumption of video decoding.

4.3 Video decoding performance and energy characterization

The aim of this part of the experiment is to characterize the impact of the video quality, the processor type and the processor frequency on both the performance and the energy consumption of video decoding. As explained previously in the methodology, three levels are considered: operating system, frame and video sequence levels. The first two steps aim to characterize the performance and the energy consumption at fine granularity to help the understanding of where goes the processor cycles and power budget during the decoding process. This helps to explain the performance and energy consumption behavior of both DSP and ARM video decoding. For example, we

[1]In the characterization methodology, we have selected these videos as representative of low, medium and high complexities

Table 4.1: Mapping between qp and the bit-rate

	Average quantization parameters qp_{avg}								
	City			Soccer			Harbor		
Bit-rate(Kb/s)	qcif	cif	4cif	qcif	cif	4cif	qcif	cif	4cif
64	32.81	42.74	51.00	38,22	48,81	51	37,78	45,84	51
128	27.69	36.79	49.38	31,65	41,55	50,99	33,94	41,56	50,7
256	23.06	31.83	41.52	25,57	35,43	45,27	30,07	37,78	45,5
512	18.49	27.28	35.76	19,84	29,76	39,09	25,83	34,12	41,26
1024	13.51	23.07	31.27	14,22	24,38	33,75	20,66	30,21	37,48
1536	10.43	20.71	29.14	10,89	21,47	30,87	16,84	27,64	35,18
2048	7.97	18.92	27.83	8,25	19,35	29,01	13,72	25,65	33,53
2560	5.69	17.50	26.86	5,87	17,77	27,61	11,08	23,98	32,21
3072	3.51	16.33	26.09	3,59	16,49	26,5	8,75	22,55	31,11
3584	1.31	15.34	25.45	1,8	15,41	25,6	6,51	21,27	30,15
4096	0.44	14.48	24.89	1,05	14,45	24,85	4,37	20,11	29,31
4608	0.22	13.70	24.40	0,91	13,6	24,17	2,31	19,03	28,56
5120	0.14	12.99	23.94	0,82	12,83	23,57	0,95	18,01	27,88

will show how the information extracted at these levels allows explaining the different energy consumption behavior for ARM and DSP decoding process depending on the video quality. On the other hand, at the video sequence level, the performance and the energy consumption properties are aggregated in average metrics which are the average number of decoded frames per second (FPS) and the average milli-Joules per frame (mJ/frame). As we will describe in chapter 5, these two metrics will be used in the modeling phase to build both the performance and the energy models.

4.3.1 Operating-system level

In this section, we will focus on characterizing the power consumption of the different power states (P-states and C-states) supported by the ARM processor and the DSP in the OMAP3530 target platform. This characterization is achieved regardless of any video decoding process. As we will discuss in the next section, this helps to define a set of reference values to understand the power consumption variation during the video decoding process.

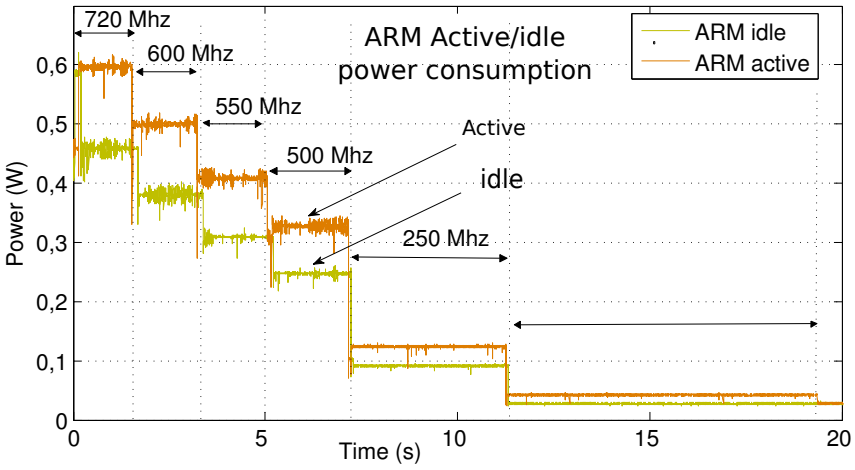

Figure 4.3: Cortex A8 power consumption

We will also investigate the impact of the transition between the different power modes on the performance. Accordingly, we discuss the opportunity of using these power modes during the video decoding process.

4.3.1.1 ARM processor

As already discussed in section 3.4.3.2, the Cortex A8 support 6 P-states and 7 C-states. In what follows, we will measure the power consumption corresponding to the P-states : 125 MHz, 250 MHz, 500 MHz, 550 MHz, 600 MHz and 720 MHz. On the other hand, we will focus on C1 (*idle*) and C2 states (switch-off with data retention) power consumption. The other C-states (switch-off without data retention) are usually used during long inactivity period (see section 3.4.1.1).

Figure 4.3 shows both the *active* and *idle* power consumption of the ARM processor corresponding to the six available clock frequencies. The power consumption in the *active* state is almost 35% larger than the one of the *idle* state. This is due to the Wait For Interrupt (WFI) ARM instruction called when entering the *idle* state. WFI puts the processor in a low power state by disabling most of the clocks in the processor while keeping it powered up. Note that the WFI power consumption is not equal to the processor static power, which corresponds to the state where all the clocks are gated. Actually, it corresponds to the C1 state where the processor is not executing instructions but can return to an executing state almost instantaneously.

More energy saving can be achieved using lower C-states. Figure 4.4 shows the power consumption level of the OMAP3530 SoC (ARM and DSP) during standby

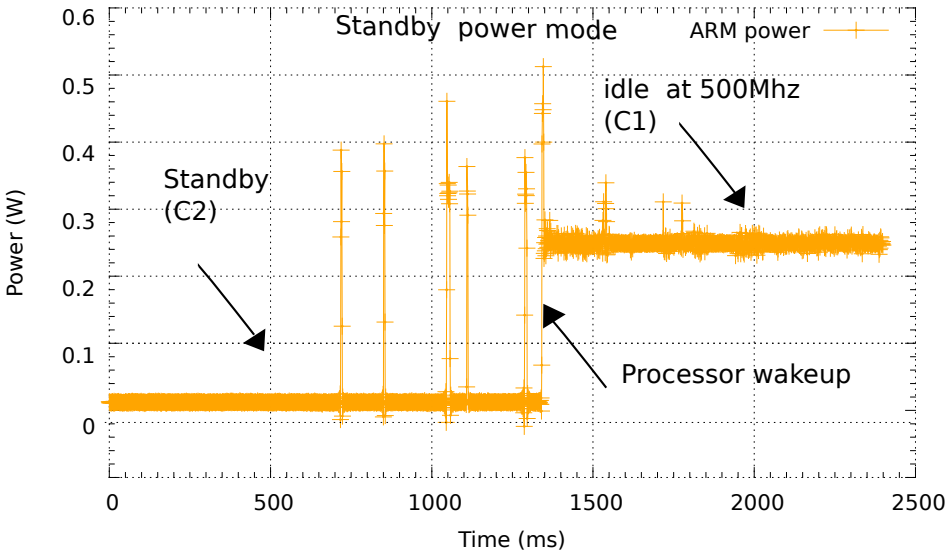

Figure 4.4: Power consumption of standby mode

mode corresponding to C2 state (refer to section 3.4.1.1 for more detail on C2 state). One can observe that deeper C2 reduces drastically the power consumption (around 10 mW) as compared to idle *state*. The level of power consumption in this state is below the static power consumption the OMPA3530 SoC (see the data in the last column of Table 4.2[2]). This is due to the fact that during this state, some SoC blocks are powered off which eliminates both static and dynamic power.

[2]The static power consumption values corresponding to the different voltage levels are extracted from the OMPAP3530 data-sheet [103].

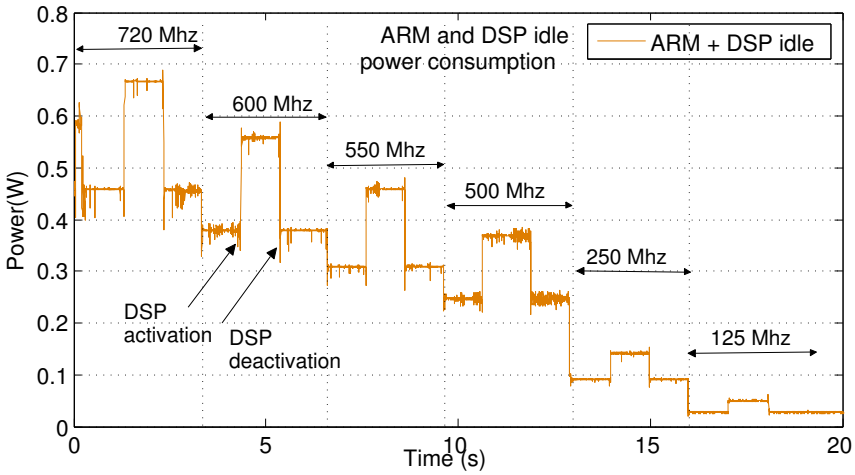

Figure 4.5: TMS320C64x DSP power consumption

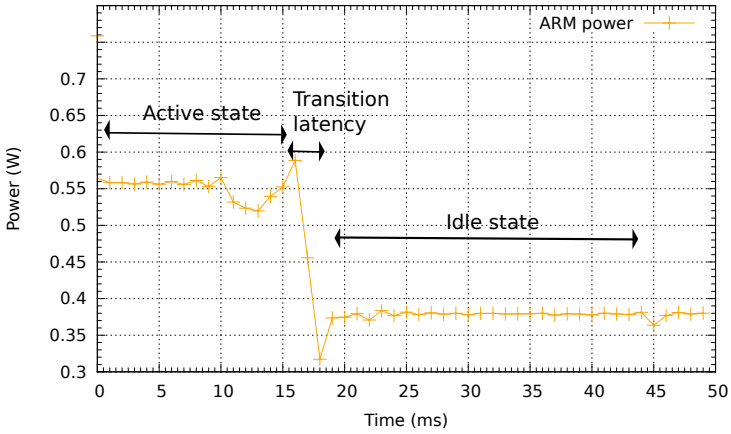

Figure 4.6: ARM *idle* mode transition latency

4.3.1.2 DSP processor

Like the ARM processor, the used DSP supports *idle* state and standby low power state to save power when the DSP is no longer used. To measure the power consumption of the DSP at the idle state, the total power consumption is measured, before and after deactivating the DSP. The DSP *idle* power is the difference between the two previous measured values. Figure 4.5 shows the variation of the power consumption during the above described calibration process. Depending on the clock frequency, the DSP *idle* power ranges from 20 mW (at 125 MHz) to 200 mW (at 720 MHz). On the other hand, the power consumption at standby mode is around 10 mW including the ARM processor.

4.3.1.3 C-states transition overhead

In the previous section, we have measured the power consumption corresponding to the different power state levels of both ARM and DSP processor. These power states, especially standby modes, allow saving considerably the power consumption during inactivity time, however they may induce additional latency. The transition to *idle* state is triggered by the Linux operating system scheduler during inactivity time. Usually, it consists calling the WFI instruction. On the other hand, lower C-states require more elaborated operations implemented in the *cpuidle* driver which depends on the underlying platform. In fact, the *idle* driver needs to know exactly the platform specific instruction for shutting down the different SoC blocks and saving/restoring their data.

In what follows, we will focus on measuring the latencies of the power states tran-

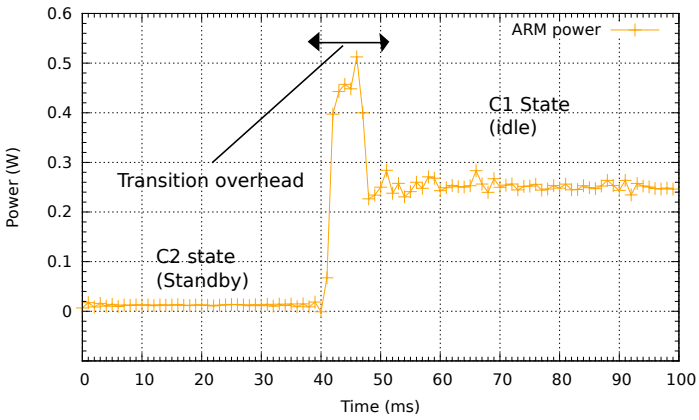

Figure 4.7: ARM standby mode transition latency

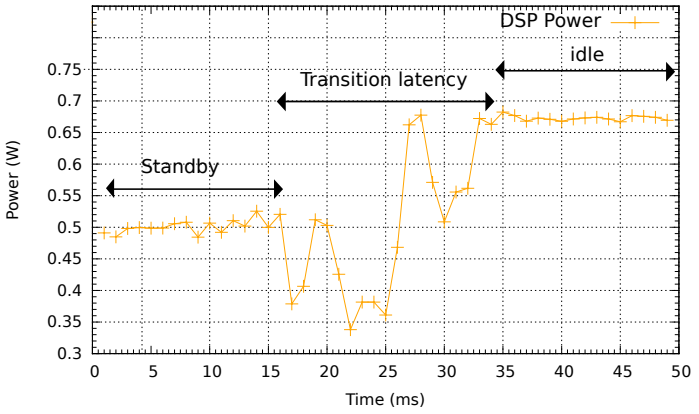

Figure 4.8: DSP standby mode transition latency

sitions and discuss the use of these states during video decoding process.

Figure 4.6 illustrates a zoom on the power consumption variation during a transition from the active state to the idle state of the ARM processor. We can observe that the transition is very fast. It consumes approximately few milliseconds. On the other hand, the transition to standby mode illustrated in Fig. 4.7 consumes more time (around 10 ms). In general, the deeper is the power state; the higher is the transition latency time. In fact, at *idle* mode, only the clock frequency is gated for some processor blocks while in standby mode, some circuits are switched off and their execution context is saved to avoid data loss.

The transition latency from the standby mode of the DSP is illustrated in Fig. 4.8. On can observe that the transition latency time is much higher than of the ARM processor (around $40ms$). This is due to the fact that the transition to standby mode is initiated by the ARM processor via a dedicated driver [106], then a standby command is sent to the DSP where it is executed by DSPBios DSP operating system.

V_{dd}	f_{arm}	f_{dsp}	$P_{act_{arm}}$	$P_{idle_{arm}}$	$P_{idle_{dsp}}$	P_{static}
V	MHz		W			
1.35	720	520	0.5965	0.4342	0.2312	0.0308
1.35	600	430	0.4997	0.3801	0.1778	0.0308
1.27	550	400	0.4087	0.3089	0.1490	0.0251
1.2	500	360	0.3276	0.2476	0.1217	0.0201
1.05	250	180	0.1238	0.0913	0.0498	0.0138
0.975	125	90	0.0421	0.0275	0.0224	0.0109

Table 4.2: Summary of OMAP3530 measured power consumption

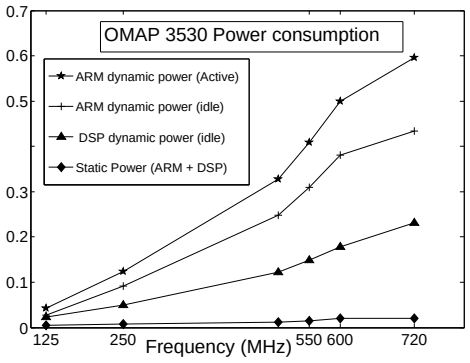

Figure 4.9: OMAP3530 power consumption

Discussion

Table 4.2 summarize the power consumption levels corresponding to the different power states of the ARM and DSP processors,

First, we can notice that the dynamic power represents the major part of the total power as compared to the static power consumption. This can be explained by the fact that the OMAP3530 SoC is based on 65 nm technology. One might expect that the ratio of the static power will be more important in case of lower feature size (ex. 45 nm, 32nm or 28nm). On such a platform, power saving based on clock gating and frequency scaling may save considerable amount of power consumption. For example, at 720 MHz, the power consumption can be reduced by 30% when transiting to *idle* mode and by 85% when scaling down the frequency to 125 MHz.

Second, notice that much more power can be saved using standby modes for both ARM and DSP processors. In fact, the power consumption can be reduced to only 10 mW. However, these modes require more latency to come-back to active state. This latency is around 10 *ms* for the ARM processor and 20 *ms* for the DSP. Knowing that a video frame decoding time is generally executed in few tens of *ms*, it is clear that such a level of latency is too long to be used in a context of video decoding. Thus, in all the following video decoding experiment, we have deactivated the *cpuidle* driver which disables the transition to standby modes.

Finally, the power consumption levels measured in these steps provide precious reference power information on the amount of consumed power corresponding to each processor state. These reference values will be used to understand and profile the power consumption of video decoding at a frame level granularity as described in the next

section.

4.3.2 Video-frame level

The objective of this step is to understand where goes the consumed energy at a frame granularity and how much energy is consumed in the processing overhead as defined in section 3.2.3. For this purpose, we have analyzed at a frame granularity the video decoding process.

Analyzing the power consumption at a fine granularity is not easy. In fact, the measured power samples using the digitizer are not synchronized with the timing information we can get from the video decoder. Thus, to map between a given decoding step and its corresponding measured power consumption phase, we refer to the power reference values extracted in the previous section. For example, based on the decoding time of one frame and the active power consumption of the processor, we can identify precisely the corresponding power consumption phase in the measured data.

This mapping is achieved manually. Thus, we focused on a sample of 16 frames extracted from the Harbor sequence. These frames are coded in 4cif (4 Mb/s), cif (1 Mb/s) and qcif (128 Kb/s) resolutions. We used 720 MHz clock frequency for the ARM and DSP processors. More extended experiments using other configurations are described at the video sequence level in the next section. As already highlighted in the methodology, we have used a high sampling rate (100K) for power consumption measurement so that to allow a fine grained power consumption measurement during the video decoding process.

Figure 4.10-a and Fig. 4.10-b show the power consumption levels of *4cif* and *qcif* DSP video decoding. The DSP frame decoding phase is represented by the strip varying between 0.7 W and 1.1 W corresponding to [32 ms, 62ms] (see Fig. 4.10-a) and [6.2 ms, 7.5ms] (see Fig. 4.10-b) intervals. This phase is terminated by a burst of DMA transfers of the decoded frame macro-blocks from the DSP cache to the shared memory. This phase corresponds to the intervals [56 ms, 62ms] (see Fig. 4.10-a) and [7.2 ms, 7.5ms] (see Fig. 4.10-b) and is illustrated by an increase in memory power consumption. When the DSP terminates the frame decoding, it returns to the GPP the execution status and enters into *idle* state. This event occurs, for example, at 25 ms in Fig. 4.10-a. The ARM wake-up latency is represented by the power level of 0.66 W which is the sum of the power consumption of both ARM and DSP in *idle* state

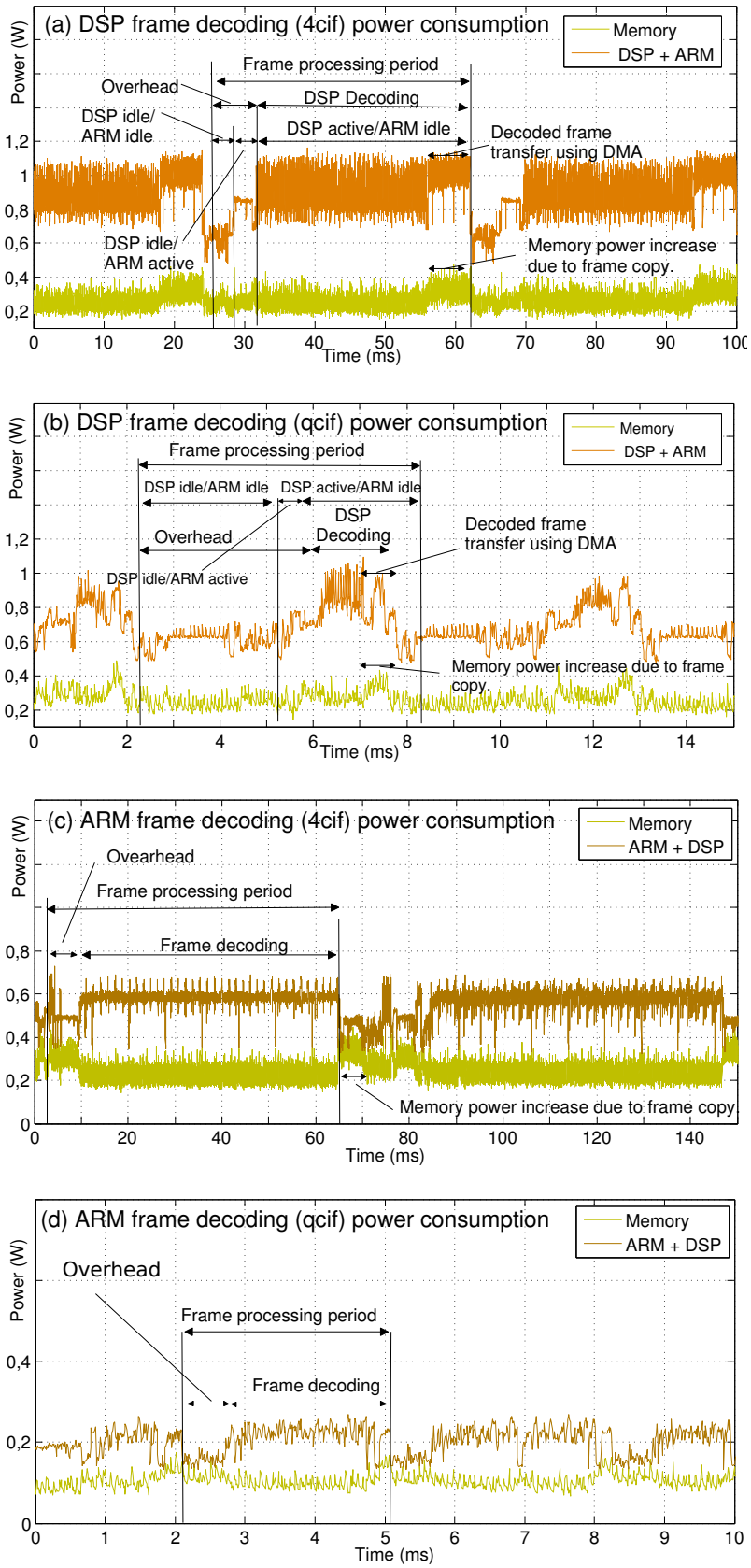

Figure 4.10: ARM and DSP video frames decoding

(0.43 W +0.23 W) as described in Table 4.2. The ARM wake-up is represented by the power transition to 0.83 W level which is the sum of the ARM *active* state (0.59 W) and the DSP *idle* state (0.23 W). Then, the ARM sends the parameters (the next frame to decode) to the DSP codecs and triggers a DSP decoding function.

Figure 4.10-c and 4.10-d show the power consumption variation in case of *4cif* and *qcif* ARM decoding. Like the DSP decoding, the frame decoding phase is characterized by an increase in the power consumption. The decoded frame copy does not appear clearly as in the case of DSP decoding since the frames are decoded in the ARM cache and evicted when no space is left in the cache. We can also notice that the frame decoding time is lower than the frame decoding period, which is due to a *GStreamer* overhead.

4.3.2.1 Inter-processor communication time overhead

We can observe that the amount of time spent in frame decoding as compared to the total video decoding time varies according to the video resolution. For example, in case of *qcif* DSP decoding (Fig. 4.10-b), the frame decoding time represents almost 50%. The complete measured time and energy overhead (as described in section 3.2.3) are given in Table 4.3.

Table 4.3: ARM and DSP decoding time overhead

Resolution	ARM decoding time(ms/frame)			DSP decoding time (ms/frame)		
	Processing	Total	Overhead (%)	Processing	Total	Overhead (%)
qcif (128kb)	2.19	2.87	10.04	1.97	4.16	52.64
cif (1024kb)	10.85	12.04	9.88	6.016	8.36	28.11
4cif (5120 kb)	47.23	52.39	9.86	23.73	25.93	8.48

The time overhead percentages are 52%, 28% and 8% of the total frames decoding time in case of *qcif*, *cif* and *4cif* DSP decoding. On the other hand, it is almost constant (10%) in case of ARM decoding. We note that the overhead is not negligible as compared to the total decoding time. For example, the total *qcif* DSP decoding time is higher than the one of ARM although the frames are processed faster by the DSP.

The above obtained results are verified through an application and system profiling performed on the ARM and DSP video decoding using *Oprofile* tool [111], a system-

	Functions	qcif	cif	4cif
ARM	libavcodec	42%	66%	75%
	omap3_pm_idle	26.5%	13%	5%
	libc	4.5%	4%	4.6%
	libgstreamer	2.5%	2%	2%
	other	31.5%	15%	18%
DSP	omap3_pm_idle	61%	69%	84%
	libc	8%	6%	4%
	libgstreamer	2.6%	2%	1.15%
	dsplinkk	0.8%	1.14%	0.6%
	other	27,6%	23%	11%

Table 4.4: Results of video decoding profiling

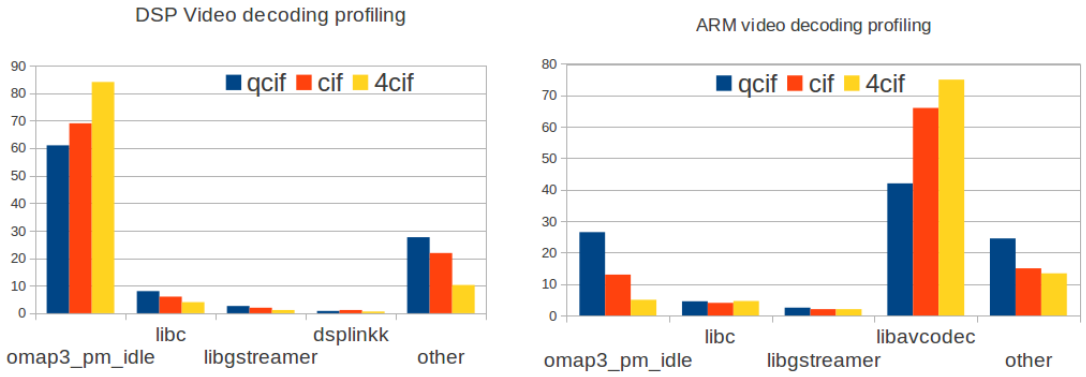

Figure 4.11: Profiling result of ARM and DSP video decoding

wide profiler for Linux systems.

Table 4.4 shows the obtained results. In the case of ARM video decoding, most of the decoding time is spent in the execution of the *libavcodec* library, which implements the H.264/AVC *GStreamer* plug-in. Although the decoder is configured to run in best effort mode (the synchronization with the displaying process is deactivated), there is an amount of time spent in *idle* state corresponding to the execution of the *omap3_pm_idle* kernel function. This can be explained by the thread blocking states while synchronizing the different *GStreamer* pipe elements. One can observe that the *idle* time ratio decreases when the video resolution increases. Indeed, frames with higher resolution are decoded in a longer time (refer to section 4.3.3.1) leading to shorter *idle* times.

In the case of DSP video decoding, most of the time, the ARM processor is in *idle* state waiting for the DSP to decode the video frames. This explains the increase of *idle* time ratio when increasing the video resolution. In fact, as in the case for ARM decoding, the frames with higher resolution are decoded by the DSP in a longer time than low resolution leading to a longer ARM *idle* time. The control of the DSP from the ARM side corresponds to the call of the *DSPlink* driver implemented in *dsplinkk* kernel module listed in Table 4.4. The actual video decoding process does not appear in the profiling results since it is executed on the DSP side.

4.3.2.2 Inter-processor communication energy overhead

In this section, we focus on analyzing the energy consumption at the different phases of video decoding. Firstly, one can observe that during the frame decoding process, the power consumption varies considerably depending on the state of the processors.

In case of DSP video decoding, while the DSP is waiting for the next frame to be sent by the ARM processor (inactivity period), it enters into *idle* state. During this time, the DSP consumes about 0.23 W at 520 MHz (refer to Table 4.2) without executing any task. During this phase, the DSP power consumption can be considerably reduced to 0.01 W if it is put into standby mode, however, this may impact considerably the performance due to the high latency of entering/exiting this mode. The standby latency is almost equal to few tens of milli-seconds which is approximately the time required to decode one 4cif frame. (see section 4.3.1.3).

In case of ARM video decoding, during overhead phase, the power consumption corresponds to the values of the *idle* state. As discussed above for the DSP, more energy saving may be achieved at this phase if the ARM processor is configured to enter in standby state. However, this may impact considerably the performance.

Table 4.5: ARM and DSP decoding energy overhead

Resolution	ARM decoding energy (mJ/Frame)			DSP decoding energy (mJ/frame)		
	Processing	Total	Overhead (%)	Processing	Total	Overhead (%)
qcif (128kb)	1.20	1.54	10.01	1.71	2.33	30.48
cif (1024kb)	6.18	6.87	9.97	5.35	6.72	20.38
4cif (5120 kb)	27.39	28.4	3.55	21.59	22.16	2.5

The energy overheads corresponding to this phase for both ARM and DSP video decoding are listed in Table 4.5. We can observe that it is higher in case of DSP decoding because the ARM/DSP communication contributes to a significant part of the energy consumption especially in case of low video resolutions. For example, in case of qcif decoding, the DSP is not used for more than 50% of the time, but still consumes *idle* power.

4.3.2.3 Discussion

We can conclude from this section that the overhead due to the video decoder framework and the operating system may contribute considerably in using the processing resources and consuming the energy. This is especially true is case of the video decoding is executed on an external processor (DSP) requiring an extra processing to manage the inter-processor communication. We have noticed also that the overhead rate highly depends on the video quality. For example, in case of low video quality, we have observed that most of time and energy are spent in inter-processor communication which leads to a drop in the energy efficiency.

In the next section, we will use these observations to explain the overall performance and energy consumption balance of video decoding on both ARM processor and DSP.

4.3.3 Video-sequence level

In this section, we analyze the average performance and the energy consumption of the video decoding at a video sequence level (a set of frames). At a second phase, we come back to the low level details discussed above to explain the performance and the energy consumption properties of ARM and DSP decoding and discuss particularly the impact of the video quality.

4.3.3.1 Decoding time

We analyze the video decoding performance in term of the number of frame decoded per second (frames/s) which is equal to N/t. N is the total number of frames (300 in our tests) and t is the decoding time. Figure 4.12 shows a comparison between ARM and DSP video decoding performance in case of *4cif*, *cif* and *qcif* resolutions for the considered video sequences. The dark (red) flat surface represents the acceptable reference video displaying rate (30 Frames/s).

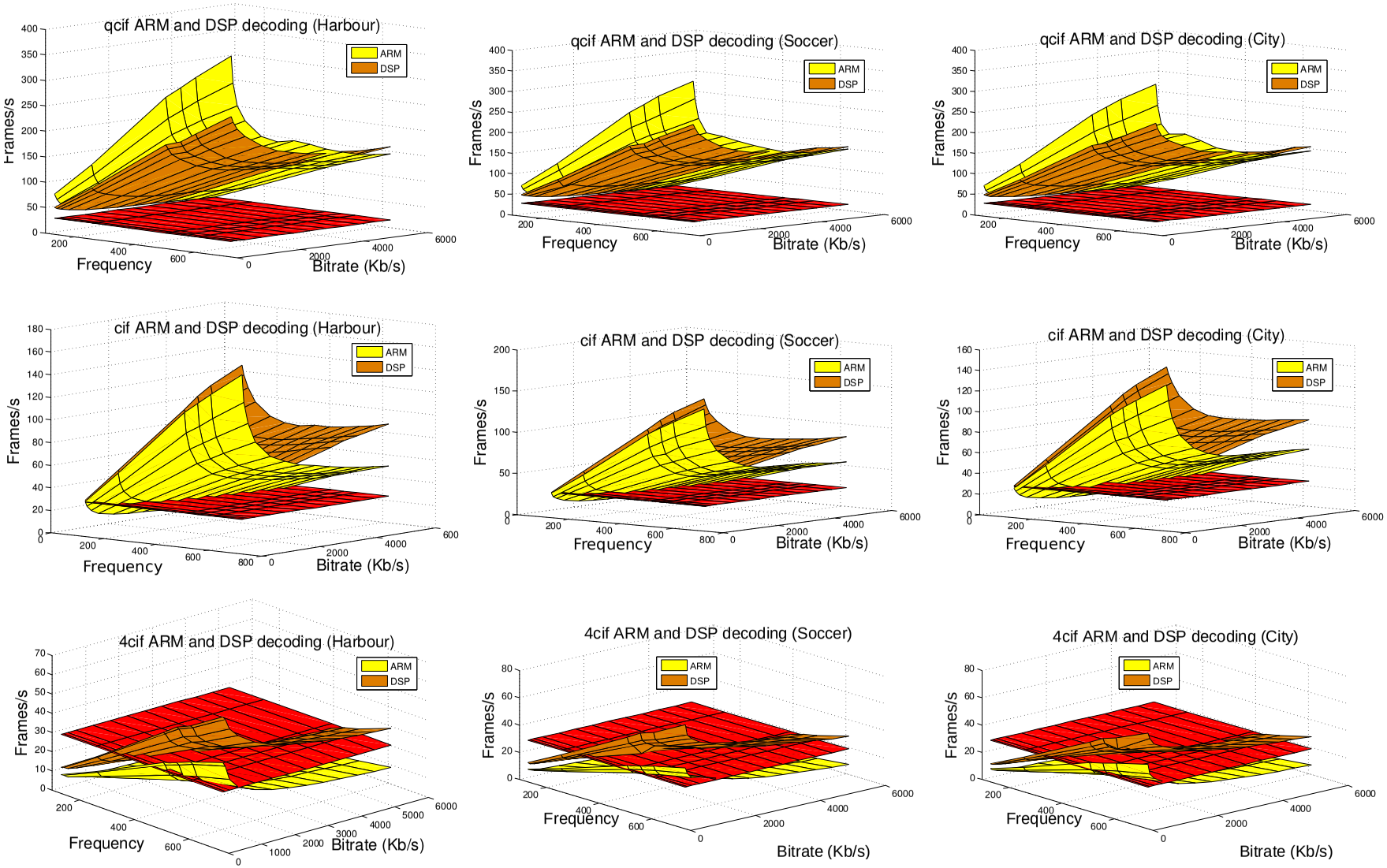

Figure 4.12: ARM and DSP video decoding performance

The first observation which can be made, in the case of *qcif* and *cif* resolutions, is that the video is decoded at a higher rate than the displaying rate (30 frames/s) even for low clock frequencies and regardless of the video bit-rate and the processor type. The ratio between the actual decoding speed and the displaying rate increases for high clock frequencies and low bit-rates.

In the case of *4cif* resolution, a decoding rate higher than 30 Frames/s can be performed by the DSP starting from 180 MHz frequency (i.e. 250 MHz ARM frequency) for low bit-rates and starting from 430 MHz frequency (i.e. 600 MHz ARM frequency) for high bit-rates (see Table 4.2 for all the mapping between ARM and DSP frequencies).

The performances of the ARM processor and the DSP are almost equivalent in case of *qcif* resolution. However, the ARM decoding speed is 43% higher than the one of DSP in case of 64 Kb/s bit-rate while the DSP decoding speed is 14% higher than one of the ARM in case of 5120 Kb/s bit-rate. For *cif* and *4cif* resolutions, The DSP decoding is almost 50 % faster than the one of the ARM in case of *cif* resolution and 100% in case of *4cif*. This ratio decreases drastically for low bit-rates.

4.3.3.2 Power consumption

Figure 4.13 illustrates the variation of the average power consumption of the ARM and the DSP video decoding according to the video resolution and bit-rate in case of the Harbor video (the Soccer and City video sequence gave similar results). We notice that the power consumption depends mainly on the clock frequency, which is explained by the dominance of the dynamic power model as compared to the static one. For example, at 720 MHz, the static power is 30,8 mW (see Table 4.2) which represents 3.4% and 2.8% of *qcif* video ARM and DSP decoding total power consumption (540 mW and 700 mW respectively).

We can also observe that, unlike the ARM decoding average power consumption, the DSP power consumption increases when the video resolution increases. The DSP power consumption is thus 30%, 40%, and 50% higher than the ARM's in case of *qcif*, *cif* and *4cif* resolutions, respectively. This can be explained by the results obtained in section 4.3.2 regarding the overhead evaluation. In fact, we found that the percentage of time overhead (overhead ratio) is almost constant in case of ARM decoding. On the other hand, in case of DSP video decoding, the overhead ratio decreases when the

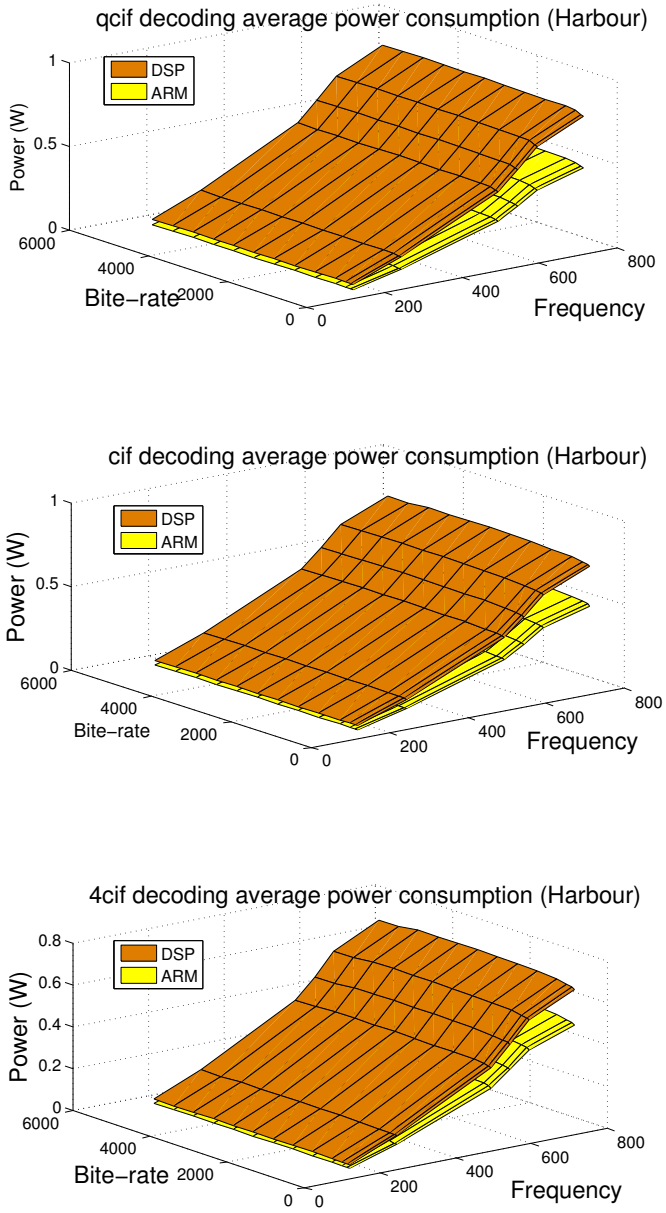

Figure 4.13: ARM and DSP video decoding power consumption

video resolution increases. A frame level power characterization (see previous sections) showed that the overhead phases correspond to a decreased power consumption due to entering the *idle* state (see to Fig. 4.10). Consequently, the higher is the overhead, the lower is the average power consumption. The same observation was highlighted in [69] in case of parallel mutli-core video decoding (see section 2.4.2.2).

4.3.3.3 Energy consumption

Previous results showed a very large variation of the DSP/ARM performance and power consumption, which depends on the clock frequency, the video bit-rate and the resolution. The energy consumption is the combination of the power consumption and the decoding time properties. Figure 4.14 shows the energy consumption of the ARM and the DSP video decoding (mJ/Frame) in case of *4cif*, *cif* and *qcif* resolutions for Soccer, Harbor and City video sequences.

The DSP *qcif* video decoding consumes 100% more energy than the ARM in case of low bit-rate and 20% for high bit-rates. This is explained by a lower performance and a higher power consumption of the DSP decoding as compared to the ARM because of the system overhead (see Table 4.5). On the other hand, the DSP *4cif* video decoding consumes less energy than the ARM although it consumes 60% more power. This is due to a better DSP decoding performance, which can be 100% higher than the one of the ARM. In case of *cif* resolution, we noticed a crossing between the ARM and the DSP energy consumption levels. In fact, for low bit-rate starting from 1Mb/s, the ARM consumes less energy than the DSP. Similarly, the inverse is true for high bit-rates videos.

4.3.3.4 Discussion

The analysis of video decoding results shows that the overall performance and the energy efficiency of the DSP as compared to the ARM processor depend mainly on the decoded video quality (bit-rate and resolution). In fact, the DSP video decoding is the best performance and energy efficient choice in case of *4cif* resolution and the use of ARM decoding is better in case of *qcif* resolution and *cif* resolution with a bit-rate less than 1Mb/s.

Table 4.6 summarizes the previous results and provides some guidelines for selecting the processor type (ARM Cortex A8 or DSP TMS320C64x) which offers the best performance and energy properties for decoding a video according to the bit-rate and the mobile device type.

On the other hand, the analysis of the experimental results according to the processor clock frequency reveals that in many cases, even if the clock frequency is scaled down, the video can still be decoded while meeting the displaying deadlines. For example, in case of 64 Kb/s *qcif* resolution, when using the maximum frequency (720 MHz),

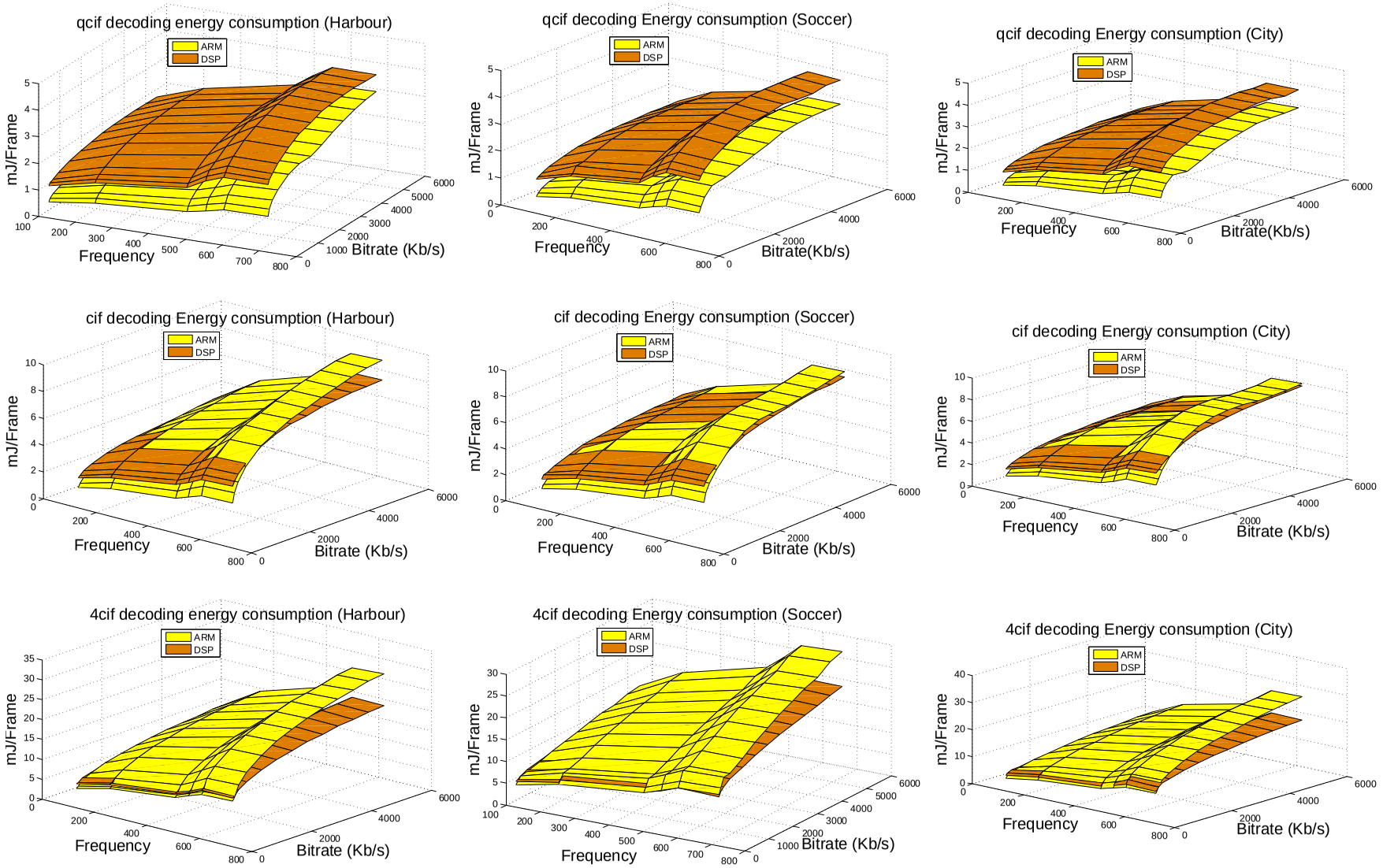

Figure 4.14: ARM and DSP video decoding energy consumption

Table 4.6: Energy efficiency *vs* Processor type *vs* video quality

	qcif		cif		4cif	
	Performance	Energy	Performance	Energy	Performance	Energy
< Mb/s	ARM	ARM	DSP	ARM	DSP	DSP
> Mb/s	ARM	ARM	DSP	DSP	DSP	DSP

the video can be decoded 10x faster than the displaying rate on the ARM processor. This is a typical configuration where an energy saving can be achieved by scaling down the processor clock frequency.

4.4 Conclusion

In this chapter, we have analyzed the performance and the energy consumption of video decoding on a GPP (ARM) and a DSP at operating system, frame and sequence levels.

At the operating system level, it was shown the potential of power saving provided by the different power states of the used ARM processor and the DSP. On the other hand, we have shown that, although standby modes allow considerable power saving, they are not suitable to be used in a context of video decoding due to high latency which may impact considerably the performance.

At a frame level, it was highlighted the importance of considering all the decoding steps while analyzing the performance and the energy efficiency of video decoding. In fact, some processing tasks, related to the decoding framework and/or the operating system, which does not belong to the actual video decoding, may contribute considerably in using the processing resources and in consuming the energy.

At a video sequence level, this is interpreted by a high variation in the energy consumption efficiency depending on the video quality. For example, it was shown that DSP processing is not always more energy efficient than the GPP as one may expect [2].

The proposed methodology gave a particular attention to achieve accurate and precise performance and the energy consumption measurement. In fact, the use of the same multimedia framework allowed to compare objectively different video codec.

On the other hand, the high-precision power consumption measurement tools used in the experimentation allows to measure fine-grained details necessary to understand the overall performance and energy consumption balance.

The obtained results are specific to the used platform (OMAP3530). However, the proposed characterization methodology provides a comprehensive experimental steps allowing the understanding of the performance and the energy consumptions behavior on modern SoC regardless of any low level details. Thus, it can be generalized to other platforms based on different processor architectures.

In the next section, based on the measured data obtained from this characterization step, we will build empirically a mathematical performance and energy models. Our objective is use the conclusions from the characterization step to make the developed models comprehensive and generalizable to other platforms.

CHAPTER 5

Performance and Energy Consumption Modeling of Video Decoding

Contents

5.1 Introduction 97

5.2 Video rate sub-model 98

5.2.1 Parameters discussion 99

5.3 Power sub-model 99

5.3.1 Static power sub-model 100

5.3.2 Dynamic power sub-model 100

5.3.3 Parameters discussion 102

5.4 Decoding time sub-model 103

5.4.1 Parameters discussion 105

5.5 Energy model 106

5.6 Models validation 106

5.6.1 Models accuracy on OMAP3530 107

5.6.2 Models generalization: OMAP4460 SoC case study 110

5.7 Conclusion 113

5.1 Introduction

The results obtained in chapter 4 provide experimental-based information on the performance and energy consumption of video decoding using ARM Cortex A8 processor and DSP TMS320C64 at different abstraction level details. In this chapter, we aim to synthesize these information within a unified mathematical performance and energy models for both the studied target processors and investigate the generalization of these to other processor types.

As described in Fig. 5.1 and discussed previously in section 3.3, the proposed modeling methodology is based on a sub-model decomposition approach. The key idea behind this approach is to model separately a set of characterization data then to combine them to construct a comprehensive energy model.

First, we first describe how to build a rate, performance and power sub-models (for ARM Cortex A8 processor and DSP TMS320C64). Then, we show how the combination of these sub-models allows to build an accurate energy analytical model for both GPP and DSP video decoding. Finally, we investigate the generalization of the developed models to other platforms. In this latter part, we show how to use the

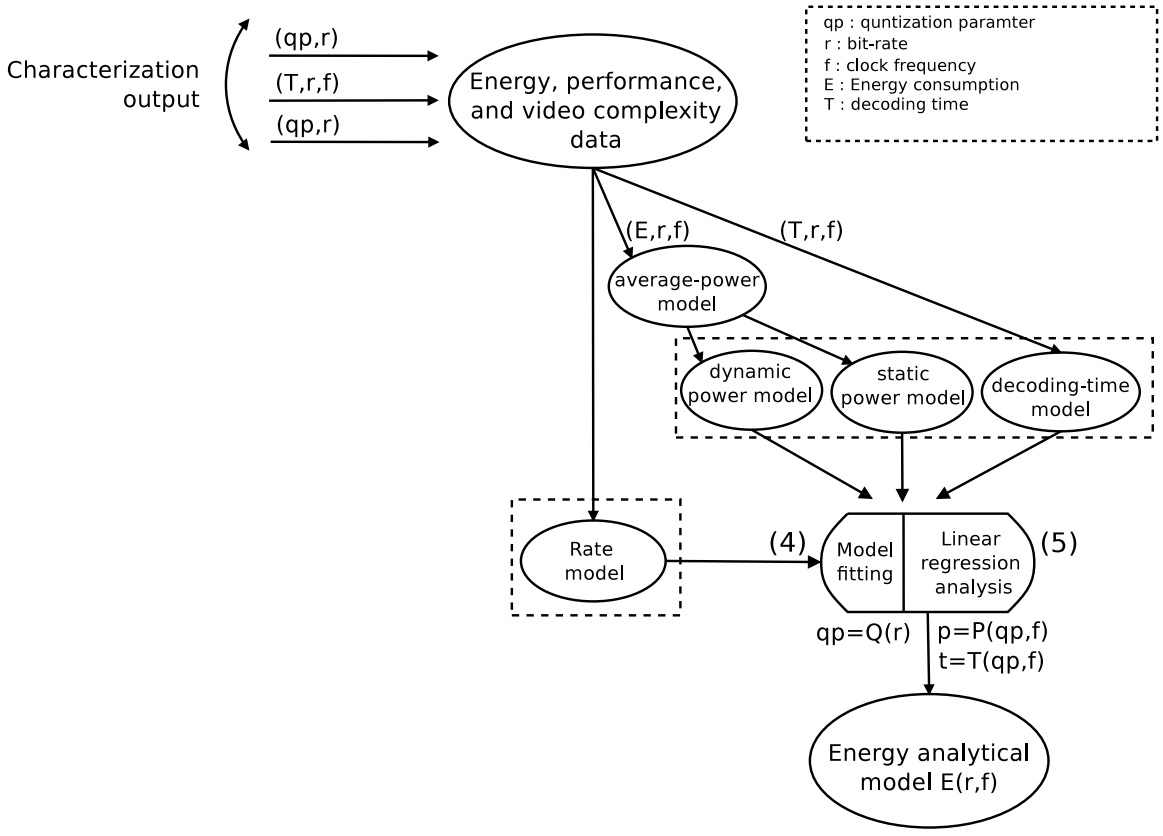

Figure 5.1: Modeling methodology

resolution	Video	a	r_{max} (Kb/s)	q_{min}	R^2
qcif	Harbor	1.031	475.5	10.15	0.9615
	Soccer	0.993	177.5	18.14	0.9978
	City	1.026	75.72	54.74	0.9726
cif	Harbor	1.393	1422	14.54	0.9937
	Soccer	1.084	327.9	32.84	0.996
	City	1.353	106.3	47.53	0.9972
4cif	Harbor	1.538	644.3	63.24	0.9913
	Soccer	1.25	552.7	54.06	0.9855
	City	1.34	115	147	0.9805

Table 5.1: Model fitting results of the bit-rate model

conclusions from the extensive experimentation described in chapter 4 allows reducing the time to build performance and energy models for video decoding on other processor types. We use the OMAP4460 SoC as a study case to validate the proposed generalization methodology. Although the OMAP4460 belongs to the same SoC family (OMAP) as the OMAP3530, it uses a different Cortex A9 processor (*vs* Cortex A8 for the OMAP3530) which is based on 45nm technology (*vs* 65 nm for OMAP3530).

5.2 Video rate sub-model

The first sub-model we built aims to represent the video properties (quality and complexity) within the expected final energy model. In addition to the video quality metric, we will show that this model is also able to represent the video scene complexity which may differ from a video sequence to another. First, a model fitting is performed on the characterization results of the video encoding (see section 4.2). The (qp, r) values obtained from the encoder (refer to Table 4.1) are fitted with the model described in Eq. 3.8. The a, r_{max} and q_{min} parameters are approximated accordingly using Matlab model fitting toolbox.

$$qp_{avg} = 4 + 6.ln_2(q_{min}.(\frac{r}{r_{max}})^{-1/a}) \tag{3.8}$$

Table 5.1 shows the model fitting results. This model has a very good precision especially for high resolution videos (R^2 values around 97%). Figure 5.2 illustrates

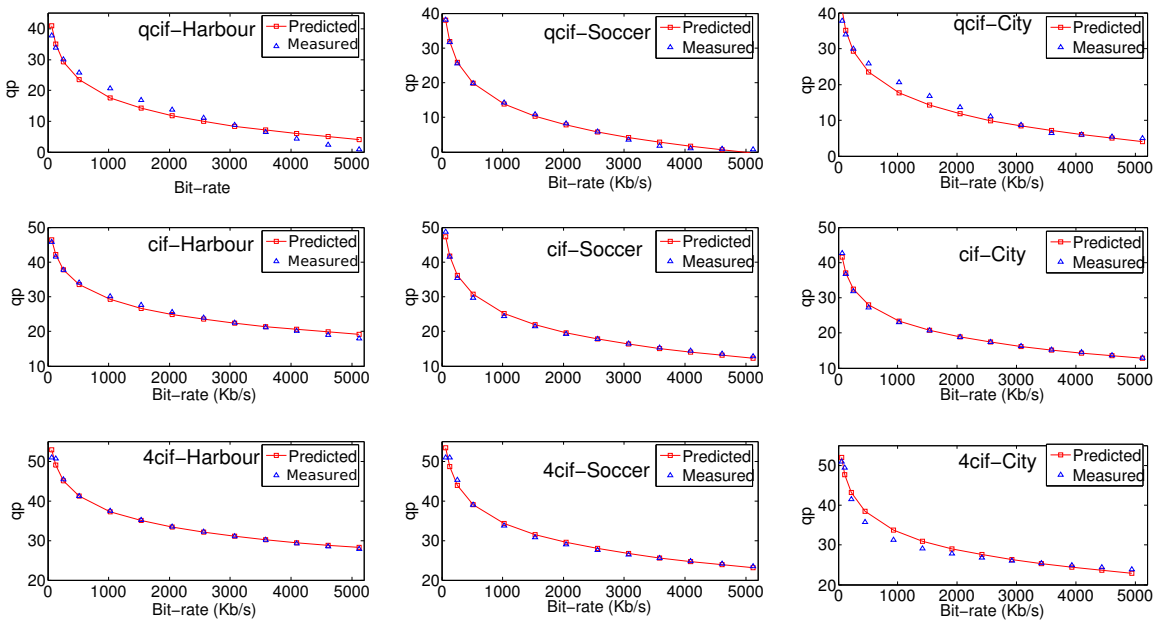

Figure 5.2: Rate model fitting

graphically the precision of the model in case of Harbor, soccer and City videos.

5.2.1 Parameters discussion

All the parameters extracted in this model are only related to the video. The parameter a is an exponent which controls how fast the video rate changes in terms of the step-size parameter. On the other hand, r_{max} and q_{min} depend on the video complexity. The more complex the video is, the higher is its corresponding r_{max} parameter and the lower is q_{min} [102]. Indeed, one can observe from Table 5.1 that the more the decoded video is complex (Harbor has the highest complexity and City has the least complexity), the higher is the value of r_{max} and the lower is the value of q_{min}. The value of the parameter a seems to be independent from the video complexity.

5.3 Power sub-model

In this section, we model the average power consumption of the ARM and the DSP during video decoding (see Fig. 4.13). Actually, two sources of power consumption were measured in the experimentation. Thus, we estimate separately each one of these sources in order to develop a model for the total power consumption.

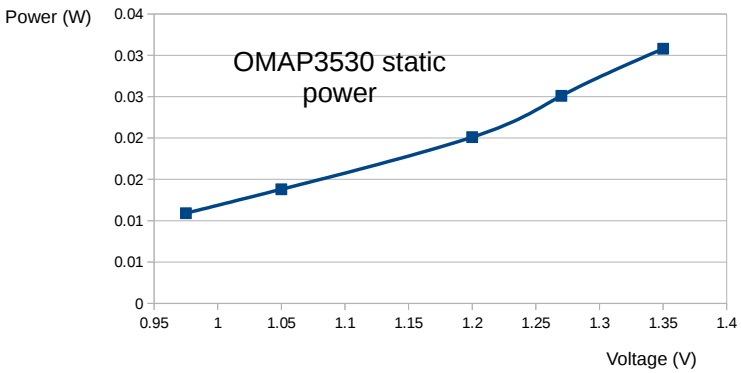

Figure 5.3: OMAP3530 static power

5.3.1 Static power sub-model

The static power consumption of a microprocessor depends exclusively on the voltage and the circuit fabrication technology. The running application does not influence it. Thus, in order to estimate the static power, we used the model provided by the OMAP3530 data-sheet [103] plotted in Fig. 5.3. We highlight that the values extracted from the data-sheet correspond to the sum of static power of the ARM processor and the DSP.

5.3.2 Dynamic power sub-model

The dynamic power is calculated by subtracting the static power estimated above from the total power consumption measured in the experimentations.

5.3.2.1 ARM video decoding

In case of ARM video decoding, the dynamic power can be represented mathematically as follows:

$$P_{dyn} = C_{eff_{arm}}.V^2.f_{arm} = P_{tot_{arm}} - P_{static} \tag{5.1}$$

where $C_{eff_{arm}}$ is the average effective capacitance of the ARM processor during the video decoding.

5.3.2.2 DSP video decoding

In case of DSP video decoding, both ARM and DSP processors are involved in the decoding process. Thus, the dynamic power can can be represented analytically as

Resolution	Video	$C_{eff_{arm}}$	$C_{eff_{arm-dsp}}$
qcif	Harbor	4.20E-007	5.48E-007
	Soccer	4.19E-007	5.53E-007
	City	4.17E-007	5.51E-007
cif	Harbor	4.28E-007	6.03E-007
	Soccer	4.23E-007	6.05E-007
	City	4.21E-007	6.04E-007
4cif	Harbor	4.18E-007	6.47E-007
	Soccer	4.17E-007	6.46E-007
	City	4.15E-007	6.44E-007

Table 5.2: Model fitting results of the dynamic power model

follows :

$$P_{dyn} = C_{eff_{dsp}}.V^2.f_{dsp} + C_{eff_{arm}}.V^2.f_{arm} = P_{tot_{arm}} + P_{tot_{dsp}} - P_{static} \tag{5.2}$$

where $C_{eff_{dsp}}$ is the average effective capacitance of the DSP.

Actually, Eq. (5.2) can be simplified using the linear relation between the ARM clock frequencies and those of the DSP[1]. In fact, the linear regression analysis on the frequency values listed in Table 4.2 leads to :

$$f_{dsp} = 0.72.f_{arm} \tag{5.3}$$

Therefore, Eq. 5.2 becomes :

$$P_{dyn} = (C_{eff_{arm}} + 0.72.C_{eff_{dsp}}).V^2.f_{arm} = P_{tot_{arm}} + P_{tot_{dsp}} - P_{static} \tag{5.4}$$

[1]In case this condition is not verified, one should model separately the power consumption of the DSP and the ARM processors to obtain the model for the total power consumtion. This needs two separate power measurement points on the board for the two processors.

5.3.2.3 Dynamic power modeling

From the above section, in both ARM and DSP video decoding, the dynamic power consumption should follow the model described in Eq. :

$$P_{dyn} = C_{eff}.V^2.f \tag{2.3}$$

Based on the values of P_{tol} (from power measurement) and P_{static} from the OMAP3530 data-sheet, we calculate the value of $P_{dyn} = (P_{tot} - P_{static})$. Knowing the values of V and f, we execute a model fitting on Eq. 2.3 to calculate the corresponding C_{eff} parameters. Table 5.2 shows the obtained results for Harbor, Soccer and City video sequences.

Column $C_{eff_{arm}}$ represents the effective capacitance of the ARM processor in case of ARM video decoding. On the other hand, the column $C_{eff_{arm-dsp}}$ represents the effective capacitance of ARM + DSP processors when using the DSP to decode a video.

5.3.3 Parameters discussion

The effective capacitance parameter C_{eff} of the processing resources is the parameter which determines the power consumption level. Actually, this parameter fluctuates depending on the executed instructions. The values calculated in the above power models represent the average power consumption over the overall video decoding time.

At system level, one technique for reducing the C_{eff} of the processor is to enter the *idle* state while the processor is not used. In fact, in this state, almost all clocks are deactivated which reduces the overall circuit effective capacitance. Thus, the higher is the time spent in *idle* mode, the lower is the average C_{eff}.

This can be veried by the correlation between the calculated C_{eff} and the profiling results discussed in section 4.3.2. In fact, we have noticed that the time spent by the DSP/ARM processors in *active* or *idle* state highly depends on the decoded video resolution. In the developed energy model, this is reflected by a various values of the C_{eff} parameter (see Table 5.2). The more the processor (GPP or DSP) spends time in *idle* state, the smaller is its effective capacitance.

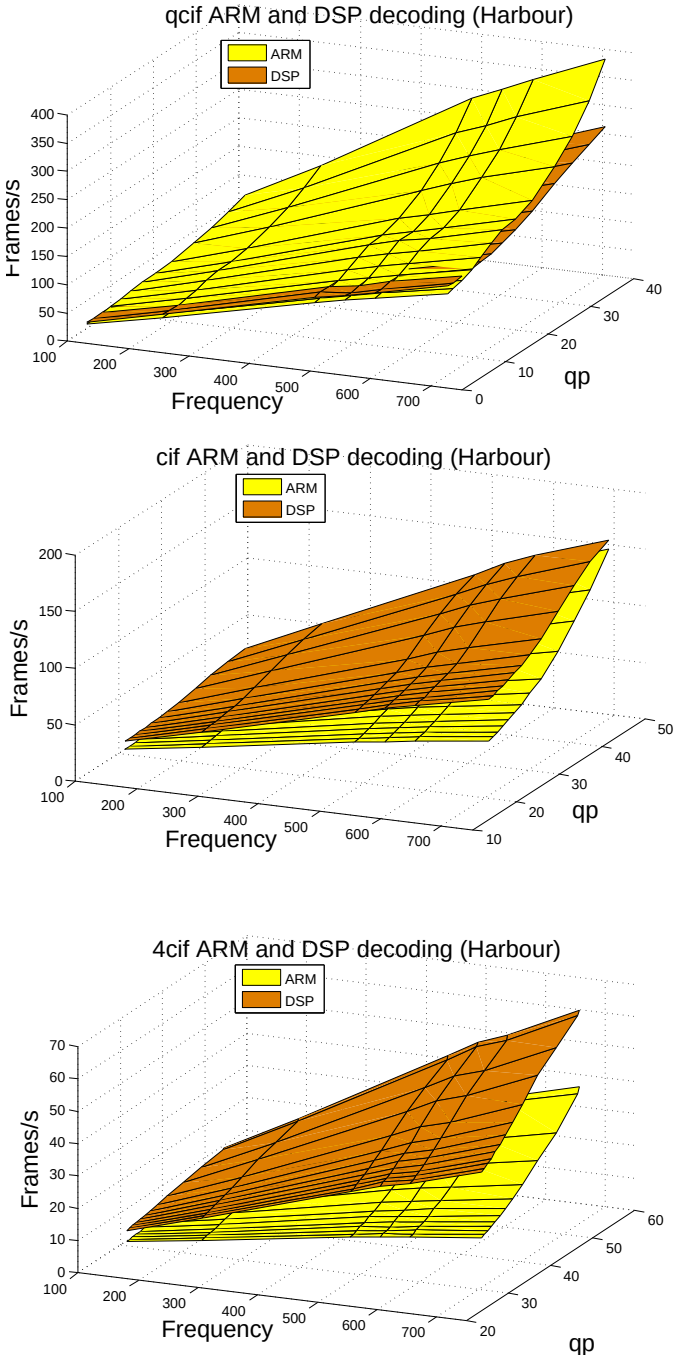

Figure 5.4: ARM and DSP video decoding time in terms of f and qp

5.4 Decoding time sub-model

To model the video decoding time, we use the linear relation observed between $(1/t)$ and both f and qp_{avg} as illustrated in Fig. 5.4. A multi-linear regression analysis using Matlab verfyed the observation. The results of this regression showed that the decoding time can be described by Eq. 5.5. The values of the coefficients α_o, α_1, α_2 and α_3, obtained by the multi-linear regression analysis, are shown in Table 5.3.

$$1/t = \alpha_o + \alpha_1.f + \alpha_2.qp_{avg} + \alpha_3.f.qp_{avg} \tag{5.5}$$

The obtained results, in Table 5.3, clearly show the accuracy of the proposed model as it can be seen from the high R^2, which are calculated for each test defined by a video sequence, a resolution and a processor type (ARM or DSP).

To have a decoding-time model as a function of the bit-rate, the qp parameter in the above model can be expressed in terms of bit-rate using the rate-model (3.8). Equation (5.5) becomes :

$$1/t = \alpha_o + \alpha_1.f + (\alpha_2 + \alpha_3.f).(4 + 6.ln_2(q_{min}.(\frac{r}{r_{max}}^{-1/a})) \tag{5.6}$$

Table 5.3: Multi-linear regression of 1/t in terms of f and qp

Resolution	Video	Processor	α_0	α_1	α_2	α_3	R^2
qcif	Harbor	ARM	2.040e+00	1.895e-04	2.166e-01	7.994e-06	0.973
		DSP	2.756e+00	2.178e-04	6.841e-02	4.021e-06	0.991
	Soccer	ARM	3.482e+00	2.165e-04	1.733e-01	7.352e-06	0.992
		DSP	2.795e+00	2.300e-04	4.703e-02	3.795e-06	0.996
	City	ARM	3.332e+00	1.195e-04	1.365e-01	6.325e-06	0.989
		DSP	2.385e+00	1.908e-04	3.573e-02	2.911e-06	0.991
cif	Harbor	ARM	-1.680e+00	-1.943e-05	1.395e-01	4.825e-06	0.978
		DSP	-1.945e-01	5.243e-05	9.438e-02	3.877e-06	0.994
	Soccer	ARM	3.356e-02	3.622e-05	9.569e-02	3.342e-06	0.991
		DSP	1.181e+00	8.893e-05	3.783e-02	2.950e-06	0.996
	City	ARM	3.128e-02	3.021e-05	8.775e-02	3.229e-06	0.990
		DSP	1.021e+00	6.320-05	2.803e-02	3.120e-06	0.991
4cif	Harbor	ARM	-1.078e+00	-1.995e-05	5.405e-02	1.559e-06	0.991
		DSP	-4.915e-01	1.458e-06	2.906e-02	1.850e-06	0.998
	Soccer	ARM	-1.749e-01	6.355e-06	3.101e-02	9.765e-07	0.902
		DSP	6.244e-03	3.984e-05	1.502e-02	9.909e-07	0.865
	City	ARM	-1.591e-01	5.505e-06	2.913e-02	6.564e-07	0.952
		DSP	5.564e-03	7.654e-05	2.277e-02	6.097e-07	0.926

5.4.1 Parameters discussion

The performance model described in Eq. 5.5 describes the linear relationship between the reverse of the decoding time ($\frac{1}{t}$) and qp. This linear relationship is defined by α_0, α_1, α_2 and α_3 coefficients.

α_0 coefficient is not coupled to f neither qp. This means that α_0 represents the time spend in executing task which does not depend on f and/or qp. For example, the *idle* transition latency or ARM-DSP communication overhead which may occur within the decoding process.

On the other hand, we can observe from the analytical model given by Eq. (5.5) that $1/t$ depends on the frequency f, qp and the correlation existing between f and qp weighted with the coefficient α_3. We show hereafter that how we can interpret these constant parameters in terms of the performance of the memory hierarchy.

We have shown in section 2.5.1.3 that a program react differently to a frequency scaling depending on the rate of memory access. In the proposed performance model, this is represented by a two-way interactions multi-linear model (non-null α_3) [112]. In fact, a non-null α_3 value means that the decoding speed-up when scaling the clock frequency f is not the same for all the video qualities but depend on the related qp parameters.

The explanation of this behavior is related to the impact of the off-chip memory access latency on the performance scaling in the context of the use of DVFS. Indeed, it is well established that the memory access rate increases for high quality video (low qp value) [57, 65, 64]. However, unlike cpu-bound instructions, memory-bound instructions execution time do not scale when varying the clock frequency [58] due to the memory wall problem [113, 4]. Consequently, when increasing the processor frequency, the performance of decoding low quality video tends to scale better than when decoding higher quality. This is illustrated graphically by a twisted surface around qp and f axis in Fig. 5.4. This figure can be compared with Fig. 2.11 given in [4] where the authors observed this behavior in SPEC CPU2000 benchmarks.

At a constant qp, $1/t$ varies by a factor of $(\alpha_1 + \alpha_3.qp)$ in terms of f. This factor reaches its minimum value when $qp = 0$, which corresponds to a lossless H.264/AVC coding. On the other hand, at a constant f, $1/t$ varies by a factor of $(\alpha_2 + \alpha_3.f)$ in terms of qp. This factor reaches its minimum value when f is minimal. This can be explained by the fact that when f decreases, the difference between the processor

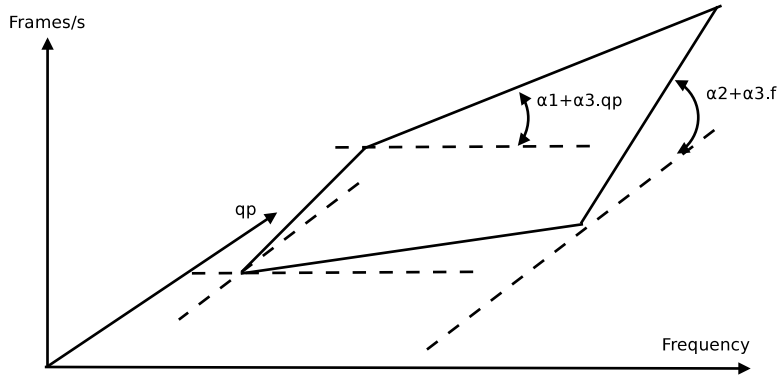

Figure 5.5: Performance scaling behavior

frequency and the memory frequency decreases. Consequently, the decoding time is not impacted considerably when the memory-bound instruction rate varies. Theoretically, this factor can be null (which means that the decoding time is independent from qp) in one of these cases: 1) the size of the cache memory is large enough to hold the entire video sequence or 2) the processor is clocked at the same frequency as the memory. However, these configurations are not realistic. Thus, the decoding time depends on the combination of α_1, α_2 and α_3. Figure 5.5 illustrates the graphical interpretation of the scaling factor $\alpha_1 + \alpha_3.qp$ and $\alpha_2 + \alpha_3.f$ discussed above.

5.5 Energy model

Based on the dynamic power model, the static power model and the decoding time model described respectively in Eq. (2.3) and Eq. (5.5), the video decoding energy consumption can be calculated as follows :

$$E = \frac{C_{eff}.V^2.f + P_{static}}{\alpha_o + \alpha_1.f + (\alpha_2 + \alpha_3.f).(4 + 6.ln_2(q_{min}.(\frac{r}{r_{max}}^{-1/a}))} \tag{5.7}$$

This model describes the energy consumption in terms of clock frequency f and bit-rate r in addition to the constant parameters C_{eff}, α_0, α_1, α_2, α_3, q_{min} , r_{max} and a. Table 5.4 summarizes the mapping of the extracted parameter to architecture, system and video properties.

5.6 Models validation

We have analyzed, in the previous, sections the accuracy of each developed sub-model (rate, time and power). The objective of this section is :

Table 5.4: Summary of the model constant parameters

Parameter	Architecture related	System related	Video related	What does the parameter reflect ?
α_1	x			- Memory hierarchy - Memory-bound/CPU-bound instructions rate
α_2	x			
α_3	x			
a			x	- How fast the video bit-rate varies when changing *qp*
q_{min}, r_{max}			x	- Video complexity
C_{eff}		x		- Scheduling. - Dynamic Power Management

- To analyze the accuracy of the performance and energy analytical models (Eq.s 5.6 and 5.7) resulted from the combination of the sub-models (Eq.s 2.3, 3.8 and 5.5).
- To investigate the generalization and validity of these models on another execution platform. We used the OMAP4460 SoC as a case study. We show how the sub-models decomposition approach proposed in this study reduces the effort of building the performance and the energy consumption models of video decoding for other platform.
- To provide some guidelines for online performance and energy estimation of video decoding on a given target execution platform.

5.6.1 Models accuracy on OMAP3530

5.6.1.1 Decoding time model

Table 5.5 shows the accuracy of the performance model described in Eq. 5.6 (Frames/s in terms of f and the bit-rate) obtained from the combination of the rate sub-model described in Eq. 3.8 (qp_{avg} in terms of the bit-rate) and the multi-linear time sub-model described in Eq. 5.5 (Frames/s in terms of f and qp_{avg}). The calculated R^2 coefficients are about 98%. Figure 5.6 shows the comparison between the predicted performance values and the measured ones.

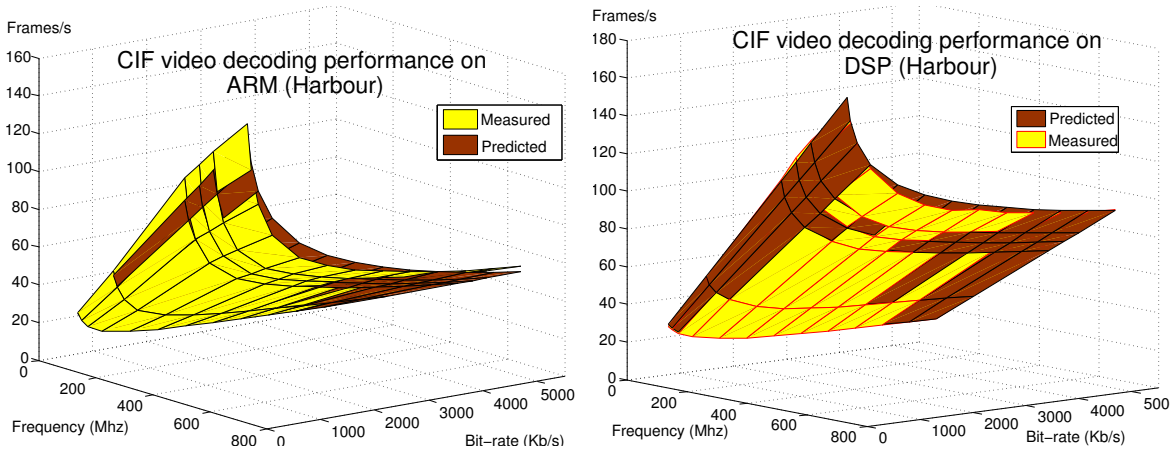

Figure 5.6: Measured vs predicted video decoding time (OMAP3530)

Table 5.5: Performance model R^2

Video	Processor	Energy model R^2		
		qcif	cif	4cif
Harbour	ARM	0.9851	0.9840	0.9886
	DSP	0.9750	0.9789	0.9787
Soccer	ARM	0.9753	0.9813	0.9811
	DSP	0.9699	0.9701	0.9687
City	ARM	0.9749	0.9803	0.9801
	DSP	0.9699	0.9689	0.9797

Table 5.6: Energy model R^2

Video	Processor	Energy model R^2		
		qcif	cif	4cif
Harbour	ARM	0.9851	0.9840	0.9886
	DSP	0.9750	0.9789	0.9787
Soccer	ARM	0.9753	0.9813	0.9811
	DSP	0.9699	0.9701	0.9687
City	ARM	0.9749	0.9803	0.9801
	DSP	0.9699	0.9689	0.9797

One can highlight an important observation, the combination of the same rate model (Eq. 3.8, which depends exclusively on the video properties), with the multi-linear time sub-model (Eq. 5.5, which depends on the execution platform), allows to build an accurate performance model for both ARM and DSP processors. We will verify this observation for the OMAP4460 platform in section 5.6.2.

5.6.1.2 Energy model

Table 5.6 shows the calculated R^2 values of the energy model (Eq. 5.7) as compared to the measured values. They vary around 97% for almost all the video sequences for both ARM and DSP (see Fig. 5.7).

To show the accuracy of the energy model for both ARM and DSP, we used it to predict analytically the bit-rates for which the ARM processor is more energy efficient than the DSP in case of *cif* video resolutions. Figure 5.8 shows the surface corresponding to the $E_{dsp} - E_{arm}$ function. One can observe that for the frequency $f = 720MHz$, the $E_{dsp} - E_{arm}$ is null for the bit-rate 1024 kb/s. This corresponds exactly to the

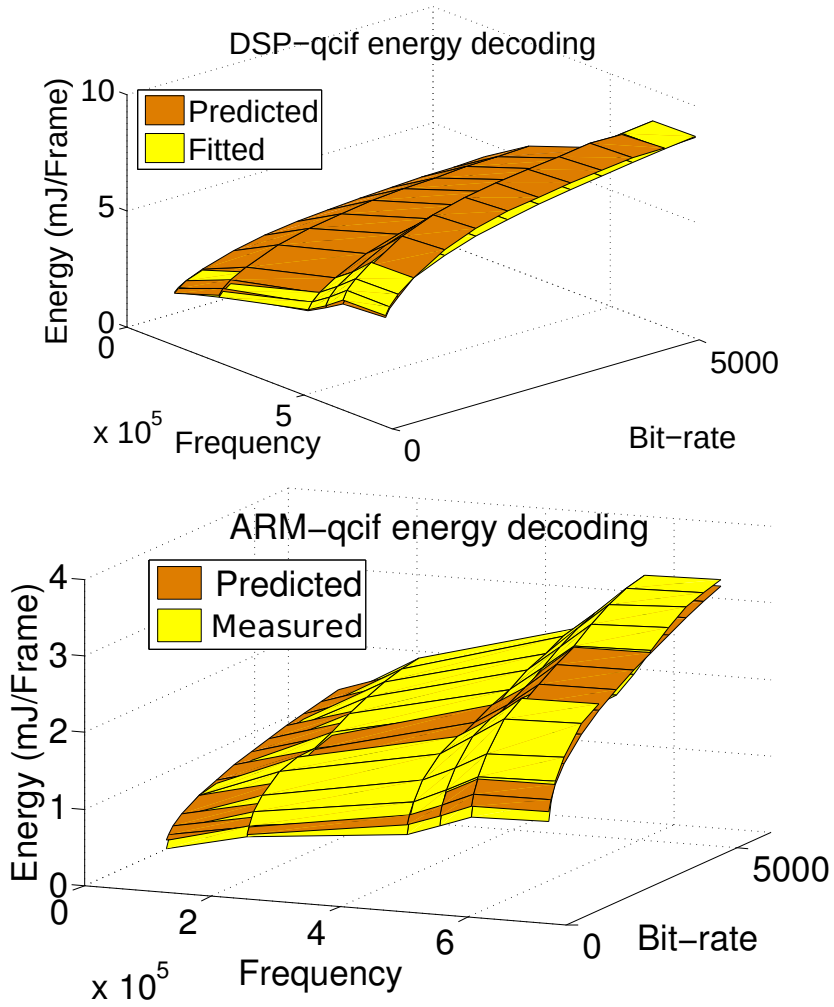

Figure 5.7: Measured vs predicted video decoding energy consumption (OMAP3530)

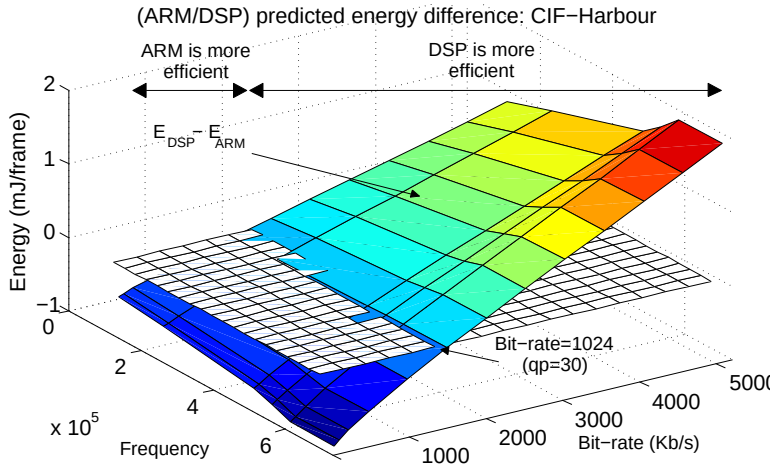

Figure 5.8: Difference between ARM and DSP energy consumption (OMAP3530)

results of the experimental measurement shown in Fig. 4.14 for *cif* decoding where we can notice the crossing between the energy surface at the bit-rate 1024 kb/s.

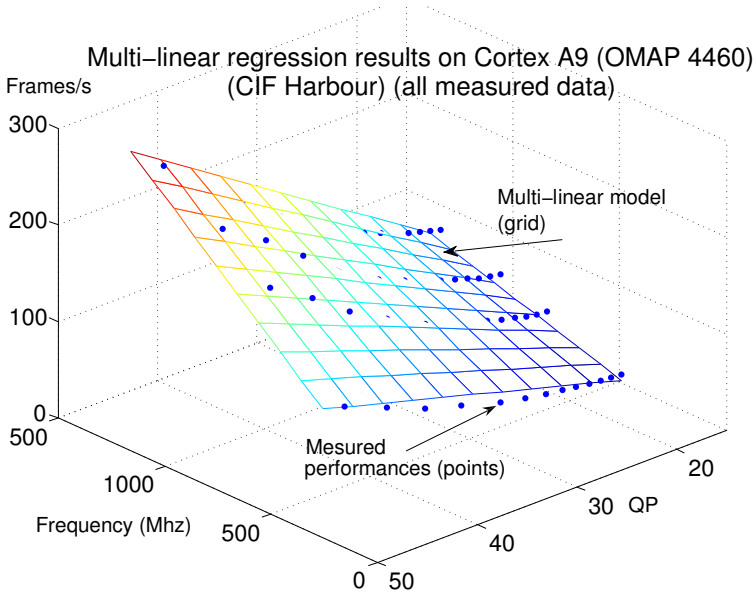

Figure 5.9: Multi-linear regression of the video decoding time (OMAP4460)

5.6.2 Models generalization: OMAP4460 SoC case study

The methodology could not be complete if one do not give how it can be applied to other platforms. To validate the methodology on another architecture, we used the OMAP4460 SoC [114] on a Pandaboard [115]. This SoC is based on 45nm technology (vs 65 nm for the OMAP3530) and contains a double core Cortex A9 processor. Each processor supports four frequencies : 350 MHz, 700 MHz, 920 MHz and 1.2 GHz. During the experiments executed on this platform, only one core was activated[2]. On this board was used the same software environment as for the OMAP3530: Linux Operating system and GStreamer video decoder. The board needed some instrumentation to allow a separate power measurement of the Cortex A9 processor. More details on the board instrumentation can be found in [23].

5.6.2.1 Decoding time model

Multi-linear regression using the model of Eq. 5.5 was performed on the measured FPS in terms of the frequency and the qp_{avg} for the OMAP4460 platform. The calculated R^2 coefficient was around 97% for all tested videos. Figure 5.9 illustrates the accuracy of the multi-linear model. When applying the rate model constants obtained previously in section 5.2, we obtained the performance model illustrated in Fig. 5.10 (Frames/s

[2]Energy efficiency of parallel multi-core video decoding will be introduced in chapter 6.

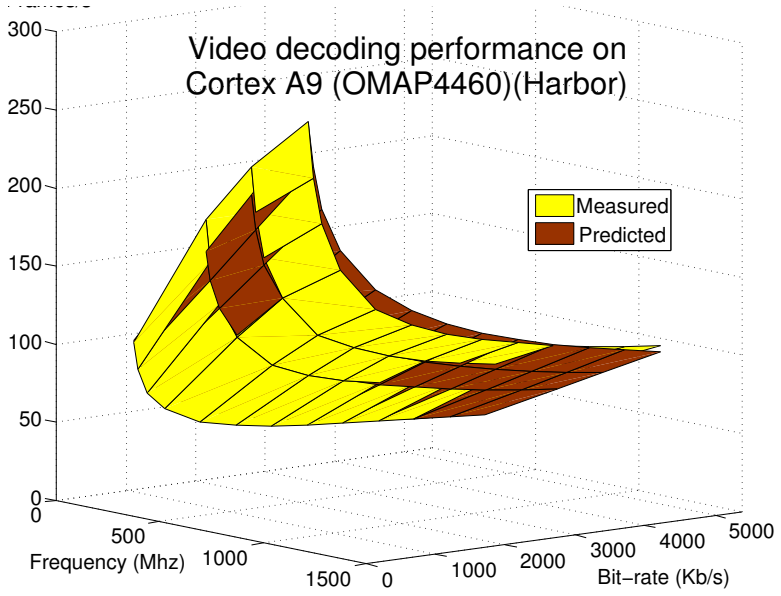

Figure 5.10: Measured vs predicted video decoding time (OMAP4460)

in terms of the frequency and the bit-rate). The model accuracy is around 96% for all tested videos.

5.6.2.2 Energy model

To obtain the energy model of the video decoding, the average power consumption values of video decoding corresponding to each frequency should be known. In the case of OMAP4460 SoC, we were not able to decompose the power consumption model into a static and a dynamic power models. This is due to the non availability of the static power in the OMAP4460 data sheet. To overcome this issue, the average power consumption (static + dynamic) of video decoding corresponding to the different frequencies was fitted with the model $af^b + c$ as suggested in [78]. The 512 Kb/s CIF video quality was used to measure the average power consumption values corresponding to each frequency. As highlighted in section 5.3, in case of ARM video decoding, the average power consumption is not impacted by the video quality. Therefore, the measured values can approximate the average power consumption for the other video qualities. Figure 5.11 shows the results of the power model fitting.

Based on the average power model and the performance model of the previous section, the energy was calculated analytically and compared to the measured energy values according to our methodology described in section 3.2.4. Figure 5.12 shows the surfaces representing the measured and the modeled energy values (mJ/frame). The

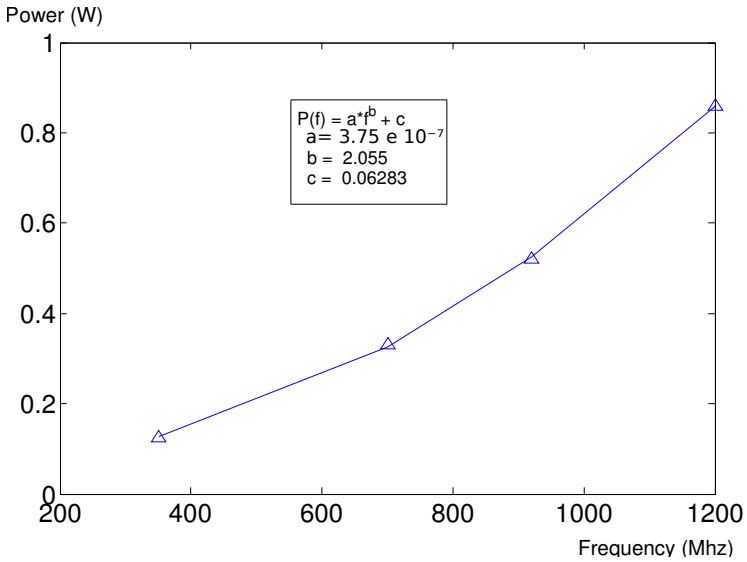

Figure 5.11: Cortex A9 power consumption model

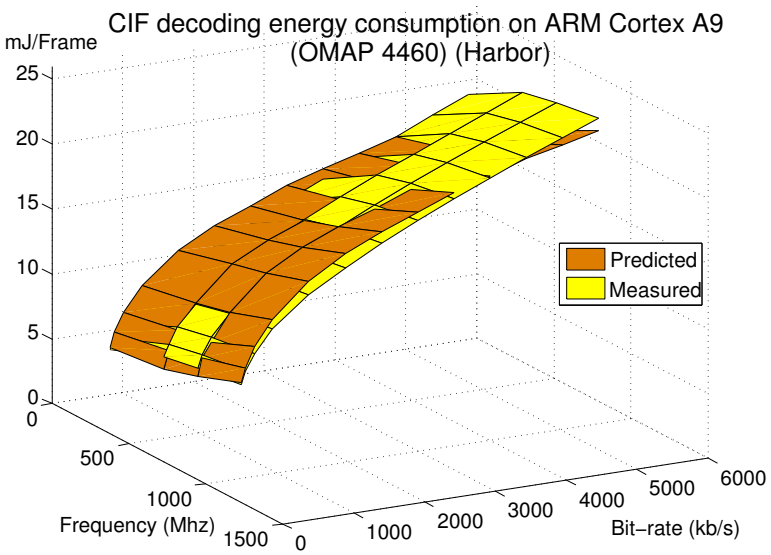

Figure 5.12: Measured vs predicted video decoding energy consumption (OMAP4460)

accuracy of the model (R^2) was about 95%.

The results of this study helped in building an accurate energy model of video decoding on the Cortex A9 processor in a much faster delay. In fact, the video complexity characterization data (a, q_{min} and R_{max} parameters) calculated in the first experimentation set (for the OMAP3550) were reused. The power consumption values of video decoding corresponding to the different processor frequencies can be measured using a simple oscilloscope. It was shown that this average value combined with the performance model allowed to estimate accurately the energy regardless of the processor

fabrication technologies: 45nm for the OMAP4460 and 65nm for the OMAP3530.

Although the proposed energy model is built empirically, based on the used sub-model decomposition approach, it can easily and efficiently be reused to build an accurate performance and energy models for other platforms. This presents an interesting advantage as compared to classical empirical energy models discussed in section 2.5.2.1 which may need a lot of effort to be rebuilt for other platforms.

5.7 Conclusion

In this chapter, comprehensive performance and energy analytical models were build based on the experimental results of the characterization phase. One of the strengths of the proposed analytical model is that it describes the performance and the energy consumption of the video decoding while distinguishing between the architecture, system and video related parameters. This was used to ease and speed up the generalization of the proposed models to other architectures by clearly identifying the parameters which depend on the underlying execution platform and those depending on the video content.

In the remaining of this thesis report, we investigate some applications of the results obtained from the characterization and the modeling of video decoding. Then, we explore the performance and energy consumption considerations in the new mobile architectures including multi-core processors and hardware accelerated video codecs.

CHAPTER 6

Applications and open issues

Contents

6.1 Introduction 115

6.2 Energy-aware video decoding in adaptive streaming . . . 115

6.2.1 Motivational example 115

6.2.2 Energy aware scheduling of video decoding on heterogeneous multi-core *SoCs* 116

6.2.3 Video-quality aware DVFS 120

6.3 Energy efficiency of high definition video decoding 123

6.3.1 Motivation 123

6.3.2 Experimental evaluation 124

6.3.3 Experimental results 126

6.4 Discussion 132

6.4.1 Per-core frequency scaling 132

6.4.2 Processing migration on asymmetric multi-cores 133

6.5 Conclusion 134

6.1 Introduction

In this chapter we present various ways to use the obtained results from the characterization and the performance/energy modeling of the video decoding . Then, we comment some issues that we plan to investigate as future works.

We, first, give some propositions for energy saving of video decoding in the context of adaptive quality video decoding. In fact, one of the strength of the proposed models is that they describe the variation of the performance and the energy consumption of video decoding in terms of video quality and processor frequency for both GPP (ARM) processor and DSP. We show that this is useful to select the best processing resources configuration (processor type and/or processor frequency) according to the desired video quality.

Secondly, we discuss the energy consumption issue in case of high definition quality which was not covered in the above discussed characterization and modeling study. We focus particularly on evaluating the energy efficiency of parallel multi-core and hardware accelerated video decoding. This is a preliminary work addressing the energy efficiency versus flexibility balance in video decoding.

6.2 Energy-aware video decoding in adaptive streaming

In this section, we first introduce through a motivational example, the adaptive video streaming and highlight the opportunity of saving energy in the context of dynamic video quality adaptation [12]. Then we present two techniques for saving energy in adaptive video decoding. The first one intervenes at the scheduling phase for selecting the best energy efficient processor according to the video quality. The second one, proposes a video-quality aware DVFS where the processor frequency is selected according to the video quality to minimize the consumed energy while guaranteeing the video decoding QoS.

6.2.1 Motivational example

To cope with the network bandwidth fluctuation and the heterogeneous mobile device capabilities, more and more video content providers such as Youtube and Netfilix, support adaptive video streaming [12]. As illustrated in Fig. 6.1, in adaptive video streaming, a video stream S is divided into sequential and independent elementary

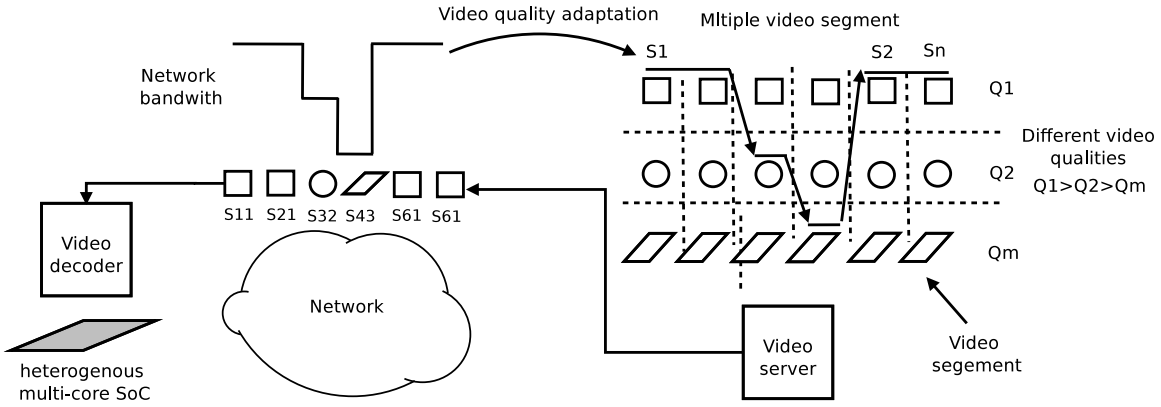

Figure 6.1: Adaptive streaming

segments S_1, S_2, $\ldots S_n$. Each segment S_i represents a few seconds video sequence and is coded into different video qualities Q_1, Q_2 $\ldots Q_m$. A video segment S_i having the quality Q_j is represented by a chunk S_{ij}. For example, a two minutes video sequence may be divided into 60 segments of 2 seconds duration each. Each segment may be coded into 512 Kb/s, 1 Mb/s, 2 Mb and 4 Mb/s bit-rate video qualities. The video decoder may then switch dynamically between the different video qualities according to the network bandwidth variation or its battery level.

In this context, adaptive video streaming poses a new challenge to energy aware video decoding. In fact, the video decoder should take into consideration at run time the dynamic variation of the video quality in dimensioning its processing resources to save energy.

6.2.2 Energy aware scheduling of video decoding on heterogeneous multi-core *SoCs*

In general, process scheduling over heterogeneous SoC provides interesting opportunities for saving energy [116]. In fact, the scheduler may consider both the performance and energy consumption properties of the underlying processing resources while assigning a processor to a given task. For example, in process scheduling over the big.LITTLE ARM SoC used recently in mobile device, the I/O-bound tasks are executed on little and energy efficient Cortex A7 cores, while processor-bound tasks are executed on big Cortex A15 core [117].

Based on this principle, we propose in what follows, a scheduling technique over ARM/DSP cores. This scheduling technique takes into consideration the video workload characteristics to select the best energy efficient core for decoding a video.

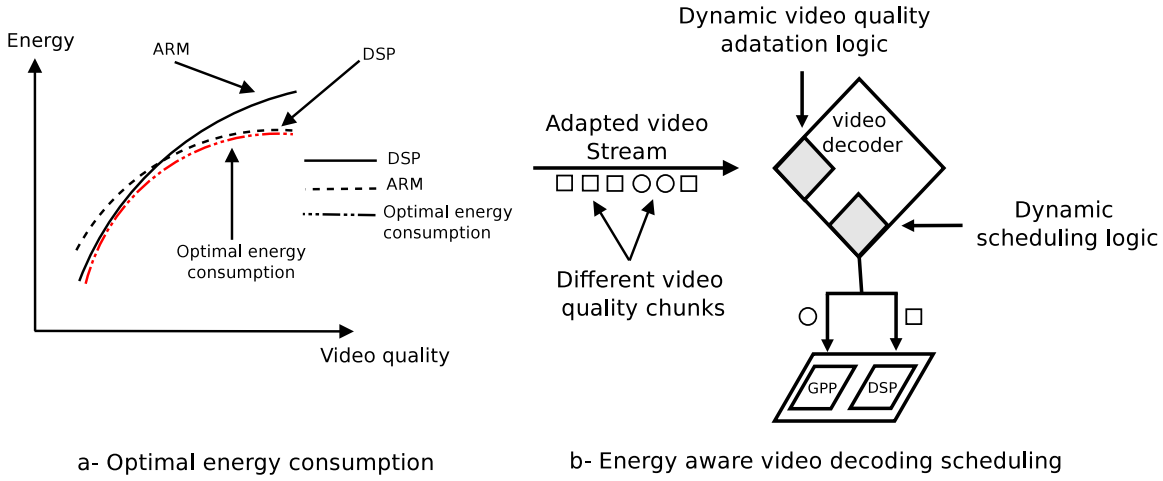

Figure 6.2: Video-quality and energy aware video decoding on heterogeneous *SoCs*

6.2.2.1 Principles

In section 4.3.3.3, we highlighted that when a video is decoded in an external specialized processor such as a DSP, a non-negligible part of the decoding time and the consumed energy is spent in the overhead due the I/O. We mentioned that if the actual video decoding workload is not very high (For example in case of low video quality), the overhead becomes significant and lead to a drastic drop in the performance and energy efficiency as compared to ARM video decoding. Through an example illustrated in Fig. 6.2-a which shows the combination of energy models developed for the ARM and the DSP, an optimal energy model may be obtained if the low video qualities are decoded on the ARM and the high video qualities on the DSP.

This is particularly interesting in the above introduced adaptive video decoding context. As illustrated in Fig. 6.2-b, when a video decoder adapts dynamically the quality of the video, it may decide accordingly on which processor it is decoded depending on the video quality.

6.2.2.2 Implementation

Based on the above discussed proposition, a basic video decoder was implemented on the OMAP3530 SoC to take into consideration the video quality to decide on which processor a video should be decoded. The scheduling decision criteria which was implemented is based on the observation that *cif*, *4cif* video decoding is more energy efficient on the DSP processor while *qcif* video decoding is more energy efficient on the ARM processor (see chapter 4).

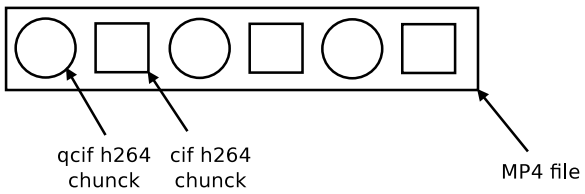

Figure 6.3: Embedding video chunks in MP4 file

To implement this scheduling policy, we have used *GStreamer* framework. In the proposed solution, we reproduced a typical adaptive streaming scenario where a video content was coded in different qualities and divided into small chunks. To simplify the implementation, we encapsulated these different chunks in one MP4 file[1] as illustrated in Fig. 6.3.

Then, a GStreamer pipe (see section 3.4.3.3) is built to decode the video chunks contained in the mp4 file. The corresponding *GStreamer* pipe (see section 3.4.3.3) consists of the following elements:

- *filesrc* : for reading the video file.
- *qtdemux* : for extracting the video content (H.264/AVC coded data) from the MP4 files.
- *ffdec_h264* or *TIViddec2*: for decoding the coded H.264/AVC data using the ARM processor or the DSP.
- *xvimagesink* : for displaying the video content.

As illustrated in Fig. 6.4: a video file is read using *filesrc* element (1) then its video content is extracted (2) using *qtdemux* demuxer. This element raises a "*new pad*"[2] event (3) when it detects a new video chunk. According to the "pad" properties (in our case, the resolution of the transported video), a dedicated event handler plugs dynamically (4) the demuxer to the *ffdec_h264* or *TIViddec2* decoder element. The next detected pads are queued in a list (5). When a video chunk is totally played, an "End Of Stream" (EOS) message is sent via the communication bus (6). Each time an EOS is sent, a message listener treats it by retrieving a pad from the list (7) and plugs it to a decoder element (ARM or DSP decoder) according to the video quality

[1]Actually, in adaptive streaming, each video chunk is coded in a separate container file. A manifest XML file which provides to the decoder the description (location and quality) of each chunk.

[2]In GStreamer framework, a *pad* is software component allowing to connect pipe elements

(8). The processor switching is achieved at this step. The selected decoder is then connected to the *xvimagesink* display element. All these functionalities are controlled from the application using an API provided by the *GStreamer* framework.

6.2.2.3 Evaluation

The player with the proposed scheduling policy achieved a transparent processor switching according to the decoded video resolution. Figure 6.5 shows the power consumption plots resulting from decoding consecutive (*cif*, 4Mb/s) and (*qcif*, 512 Kb/s) chunks when disabling (Fig. 6.5-a) and enabling (Fig. 6.5-b) the processor switching. We can observe that switching to ARM decoding in case of *qcif* resolution allows reducing considerably the power consumption. A 30% energy saving was achieved in this example as compared to the static scheduling on DSP, without impact on the performance.

The proposed scheduling policy selects dynamically the appropriate processor for decoding a video according to its resolution. However, it can easily be extended to

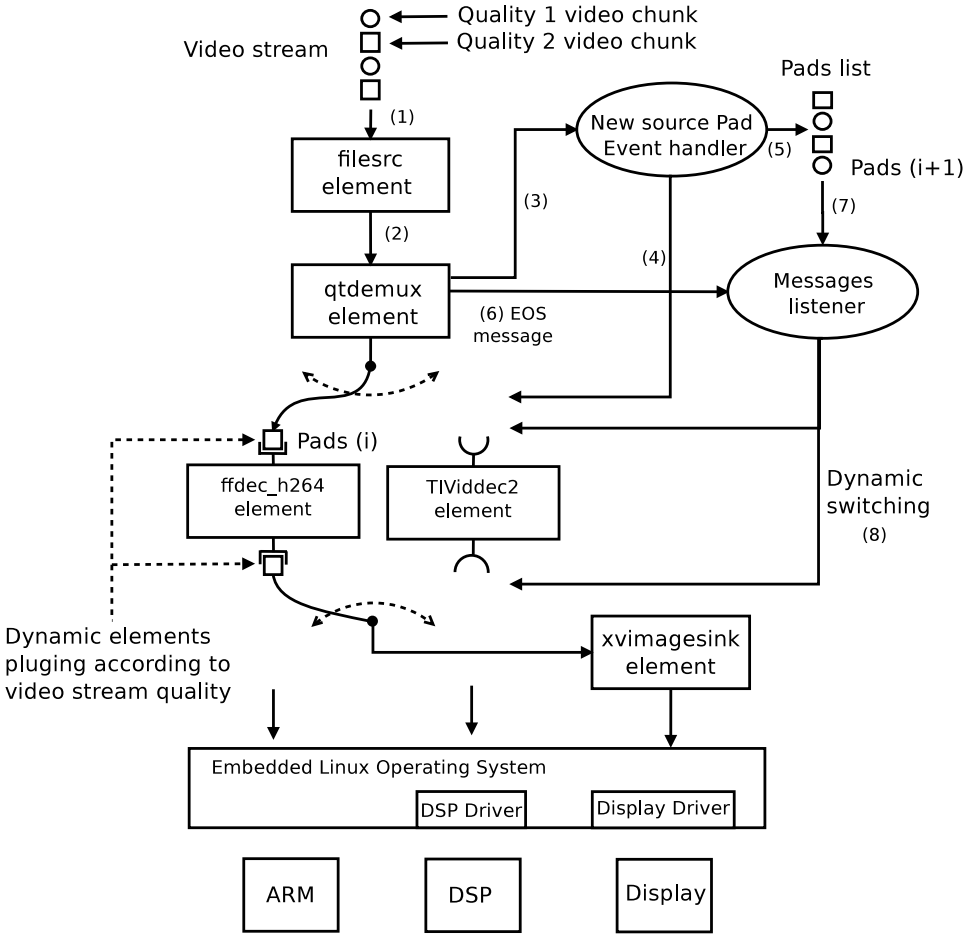

Figure 6.4: Dynamic processor switching solution design using GStreamer

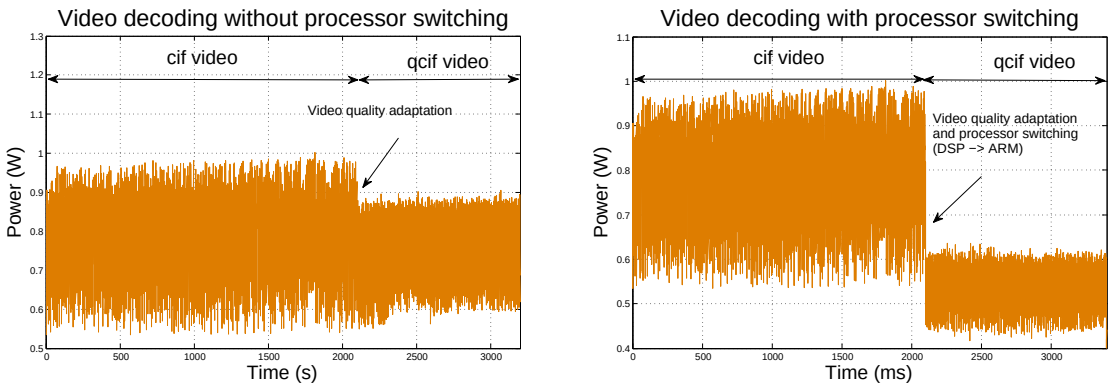

Figure 6.5: Impact of dynamic processor switching on video decoding power consumption

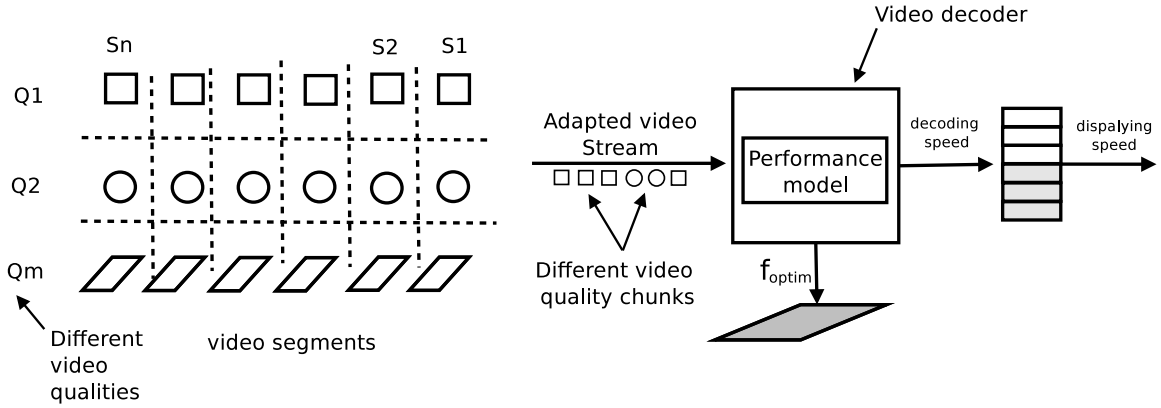

Figure 6.6: Video-quality aware DVFS

take into consideration the bit-rate (see Table 4.6) or any other video property.

6.2.3 Video-quality aware DVFS

In this section, we provide some guidelines on how to use the developed performance models to select the processor frequency for a given video quality in case of adaptive video decoding.

6.2.3.1 Problem description

As shown in section 4.3.3.1, the video decoding performance highly depends on the video quality. Thus, the required processing resource configuration may vary when the video quality changes. For example, in case of the use of DVFS, an adaptive video decoder should react to a video quality changes to adjust the processor frequency accordingly. Figure 6.6 illustrates such a scenario, where the video quality can change dynamically. Based on a video-quality aware performance model, the video decoder

should adjust dynamically the processor frequency.

We can highlight that the frequency adjustment can be achieved at a frame or average decoding rate basis. In the latter case, in order to decouple the constant frames displaying speed from the frame-to-frame workload variation, a buffer may be inserted between the decoder and the displaying device as suggested in [40] and presented in section 2.3.1.2.

6.2.3.2 Proposed solution

In the context of the above situation, we provide a possible use of the proposed performance model to drive the processor frequency in case of adaptive video decoding.

One of the advantages of the performance model described in Eq. (5.6) is that it distinguishes between the architecture/system related parameters and the video parameters.

$$1/t = \alpha_o + \alpha_1.f + (\alpha_2 + \alpha_3.f).(4 + 6.ln_2(q_{min}.(\frac{r}{r_{max}}^{-1/a})) \tag{5.6}$$

Accordingly, we propose hereafter a methodology (illustrated in Fig. 6.7) were both the video encoder and the video decoder collaborate to ease the online performance model building. The objective is to have a video decoding performance model which allows to select the adequate processor frequency for a given video quality.

First, since a, q_{min} and R_{max} parameters are video dependent, one can suggest to calculate their values at the encoding phase and send them to the decoder as a metadata, such as proposed in the studies presented in section 2.5.1.1[3] (see step 1 in Fig. 6.7).

Then, if the video parameters (a, q_{min} and R_{max}) are known, the video decoder can calculate qp_{avg} for the corresponding video sequence using the rate model described in Eq. (3.8) (step (2) in Fig. 6.7).

Assuming the multi-linear model described in Eq. 5.5, online performance model building becomes easier. For example, an adaptive linear filtering technique [119] may be used to calibrate α_0, α_1, α_2 and α_3 parameters online. The parameters adjustment (step (3) in Fig. 6.7) is driven by the error of prediction fed from online performance

[3]The use of metadata for energy aware video decoding is subject to an active discussions in the Green Metadata standardization initiative conducted by MPEG [118]

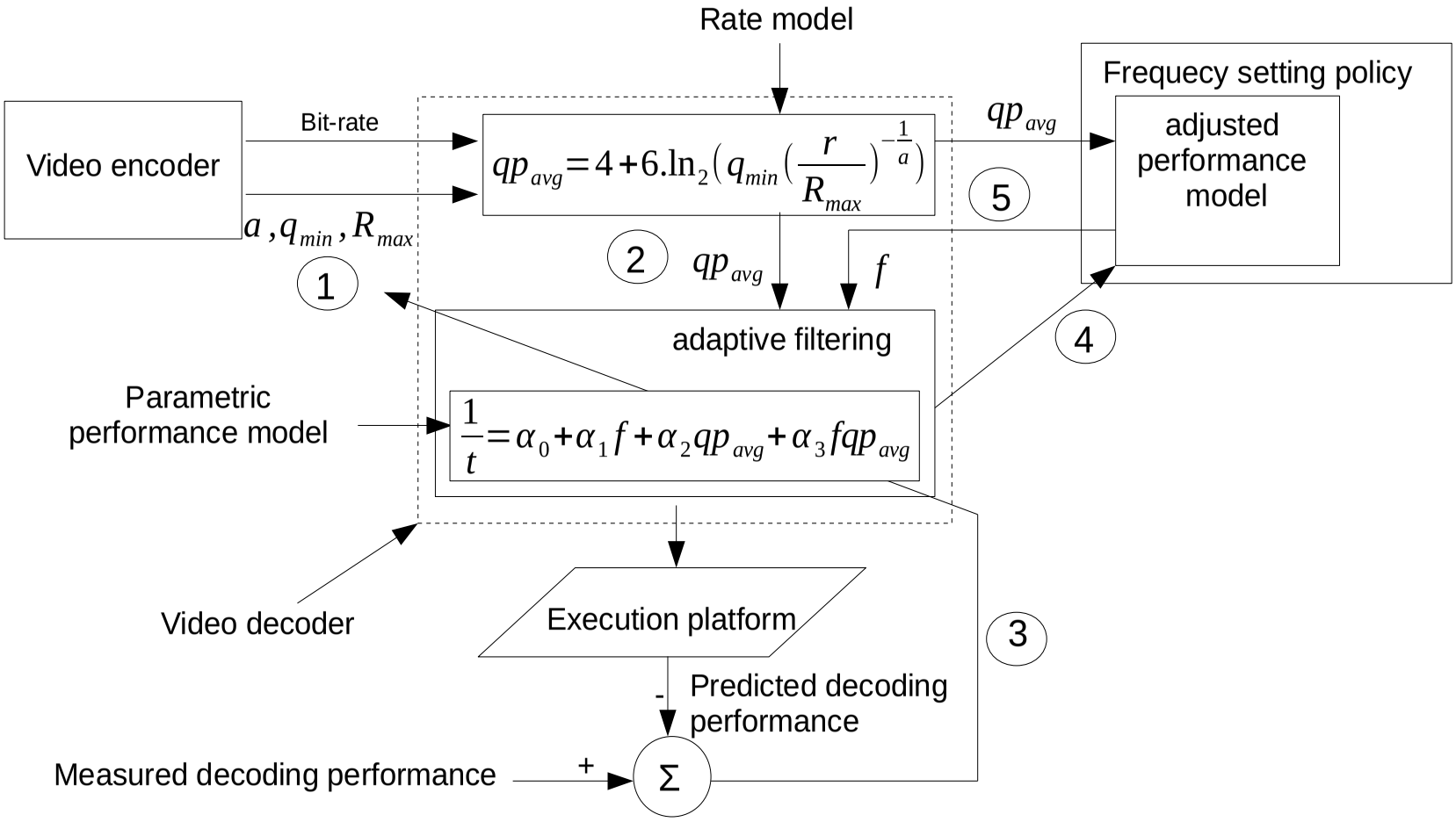

Figure 6.7: Performance and energy consumption models building

calculation of video decoding. In this calibration phase, the decoder has to alternate between different video decoding configurations (video quality and processor frequency) to calibrate the model parameters. The largest are the decoded configuration, the more accurate is the calibrated model.

Once the performance model is calibrated (ex. the error rate is below a given threshold), it can be used within a DVFS policy (step (4) in Fig. 6.7) to set proactively the processor frequency for future video sequences depending on the video quality (step (5) in Fig. 6.7).

The model parameters calibration phase provides to this technique a kind of auto-learning capabilities. This may present a interesting advantage as compared to the previously proposed reactive feedback systems presented in 2.5.1.2. In fact, these latter techniques need to continuously monitor the performance of video decoding to adjust accordingly the processor frequency. However, as highlighted in [85], the tuning of such system is not always easy and may depend on the video workload.

The above described methodology provides a guideline to build video-quality aware DVFS. Its main advantage is that it is based on well validated performance model allowing a proactive frequency scaling according to the video quality. As a future work, we plan to implement it within a video decoder and evaluate the energy saving in case of real adaptive video decoding scenario.

6.3 Energy efficiency of high definition video decoding

6.3.1 Motivation

The video qualities targeted previously ranges from *qcif* (176 x 144) to *4cif* (704 x 576) resolutions. However, currently, due to the increase of bandwidth capacity and mobile terminal displaying capabilities, there is more and more demand on high definition (HD) video (1280 x 720).

HD video quality requires a huge processing which cannot be provided by single ARM or DSP processors. Currently, most of modern smartphones and tablets use hardware accelerator to decode HD videos. However, the drawback of hardware video codecs is that they are not flexible. In fact, video standards evolve quickly and hardware codec do not adapt to those changes [49]. For example, hardware accelerators for the new MPEG HEVC (High Efficiency Video Coding) standards are still not available on mobile devices at the date of writing this document.

With the advent of multi-core architecture, parallel decoding on multiple ARM core may be an attractive solution to fit with the HD decoding processing requirement. The question which may raise in a context of the use of modern SoC including more and more cores is how much energy efficient is the parallel HD video decoding on multiple cores as compared to Hardware codec. In other words, may parallel video decoding on multi-core ARM conciliate between the flexibility of software codec and the energy efficiency of the hardware codecs ?

In the following section we will describe a preliminary experimental study where we compare the performance and the energy efficiency of parallel video decoding as compared to the use of hardware codec.

6.3.2 Experimental evaluation

We have used the same environment described in 3.4. To evaluate parallel and hardware video decoding, we have integrated a new embedded board and achieved some additional configuration on the video decoder.

6.3.2.1 Hardware and software setup

We used in our experiment the *SABRE* development board containing the low-power i.MX6 Quad-core SoC. This SoC consists of the Quad Cortex A9 ARM processors and a set of specialized processing units such as a Graphical Processing Unit (GPU) and a Video Processing Unit (VPU) (see Fig. 6.8). Each Cortex A9 processor supports 3 frequencies: 400 MHz, 800 MHz and 1000 MHz. The VPU is a hardware accelerator implementing H.264/AVC encoding/decoding standard. It is clocked at 264 MHz and supports full HD video decoding up to 60 Hz rate. In what follows, the VPU term serves to designate the video hardware accelerator.

On this hardware platform, the Linux kernel version 3.0.17 was used with *cpufreq* enabled to drive the ARM cores frequency scaling. The *userspace* governor was activated to allow the control of the clock frequency at the application level. The H.264/AVC video decoding was achieved using GStreamer. The ARM decoding, was performed using *ffdec_h264* plug-in based on the *libavcodec*. For the hardware accelerated decoding, we used *vpudec*, a proprietary GStreamer H.264/AVC plug-in provided by *Freescale*.

The *libavcodec* library supports both slice and frame multi-threaded decoding. To fully make use of the available processors, we use slice-based multi-threaded decoding

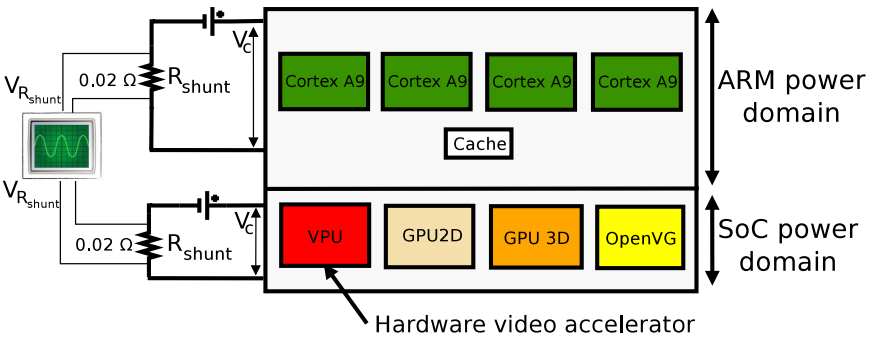

Figure 6.8: i.MX6 SoC power domains

in our experiments. In fact, in slice-based parallelism, each slice can be decoded independently from each other one. On the other hand, in frame-based parallelism, there may exist some dependencies between the frame which limits the number of parallel threads (see section 2.3.2).

The *ffdec_h264* plug-in does not allow to select explicitly which method to use and the automatic selection mechanism tends to select systematically the frame-level multi-threading. To fix this issue, the plug-in was forced to use the slice-level method by setting *active_thread_type* = FF_THREAD_SLICE in the *pthread.c* source file.

As a test video, we used the well-known *Big Buck Bunny* sequence. We encoded it in 720p resolution (1280x720), 2Mb/s bit-rate and 24Hz rate using *x264* encoder. We configured the encoder to set the number of slices per frame to 4 by means of the *–slices* option. The objective is to fully make use of the 4 available ARM processors on the i.MX6 SoC while decoding the video.

6.3.2.2 Performance measurement

We started by measuring the performance of video decoding using a single core, dual-core, quad-core decoding at all the available clock frequencies (400, 800 and 1000 MHz) and the VPU decoding. The number of cores used for decoding the video was selected by setting the value of *max_threads* parameter of the *ffdec_h264* plug-in. The VPU and multi-core video decoding is selected by choosing the corresponding *GStreamer* plug-in: (*ffdec_h264* or *vpudec*). For each configuration, we measured the number of decoded frames per second (FPS).

	400 MHz	800 MHz	1000 MHz
1 core	7,16	13,71	17,03
2 cores	12,30 (x 1.71)	24,55 (x 1.79)	28,02 (x 1.64)
4 cores	18,35 (x 2.56)	33,36 (x 2.43)	39,80 (x 2.33)
VPU	90,57 (x 12.64)	90,61 (x 6.60)	91,05 (x 5.34)

Table 6.1: HD video decoding performances (fps) (i.MX6)

6.3.2.3 Energy consumption measurement

The used *SABRE* board has two power domains which can be measured separately. The ARM power domain includes the four ARM cores plus the cache memory and the SoC power domain includes the VPU and other specialized graphical and image processing units [120]. At each power domain was inserted R_{shunt}, a 0.02 Ω shunt resistor (see Fig. 6.8).

The power consumptions is then measured using the Open-PEOPLE framework [23], used previously is our experimentation (see section 3.4.2.1). In case of multi-core ARM video decoding, only the ARM power domain consumption is measured. On the other hand, the sum of the ARM power domain and the SoC power are measured in case of VPU decoding since both the ARM cores and the VPU are involved in the decoding process.

6.3.3 Experimental results

We discuss hereafter the measured performance and energy consumption of video decoding on ARM processors and hardware accelerator.

6.3.3.1 Video decoding performances

Table 6.1 shows the performance results of the video decoding. We can observe that in case of multi-core decoding, the decoding speed is higher than the displaying rate using 2 cores or 4 cores starting from 800 MHz clock frequency. In case of VPU video decoding, the decoding speed is (x 3.75) higher than the displaying rate regardless of the ARM cores frequency[4]. This is illustrated in Fig. 6.9 where the flat red (dark) surface represents the displaying rate (24 Frames/s).

[4]The frequency of the VPU frequency (264 MHz) remains constant when varying the frequency of the ARM cores

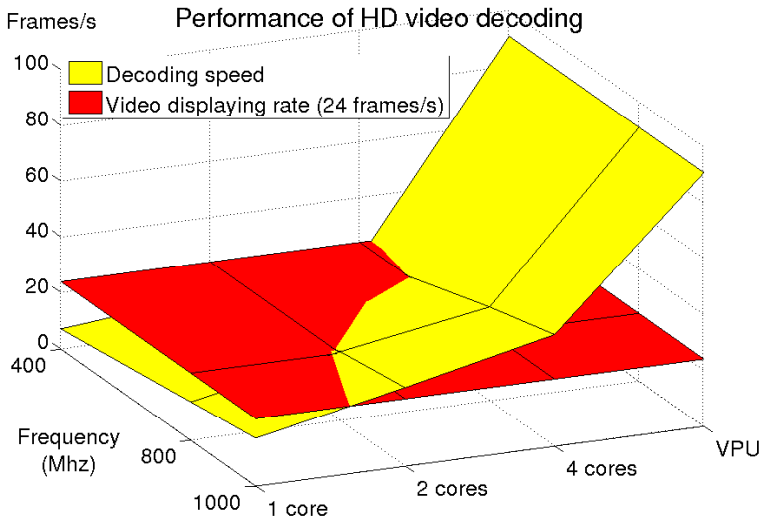

Figure 6.9: Performance of HD video decoding (i.MX6)

The values between parentheses in Table 6.1 represent the performance scaling factor as compared to mono-core video decoding. We can observe that using four ARM cores allows only x2.4 performance increase. This is mainly due to the unbalanced workload. In fact, the video encoder divides each frame into equal-sized slices. However, the decoding workload depends on the slice scene complexity. Thus, a decoding thread assigned to a given slice may terminate before the other ones. During this time, it goes into a blocked state waiting for the other threads to terminate.

Notice that, the scaling factor is much higher (from x5 to x12) in case of VPU decoding. This is due to MB level parallelism implemented in the VPU.

The measured processor usage values[5] illustrated in Fig. 6.10 confirm these obser-

[5] $processor\ usage = (\sum_i T_i)/T_{exe}$ where T_i is the time that the i^{th} thread got a processor core

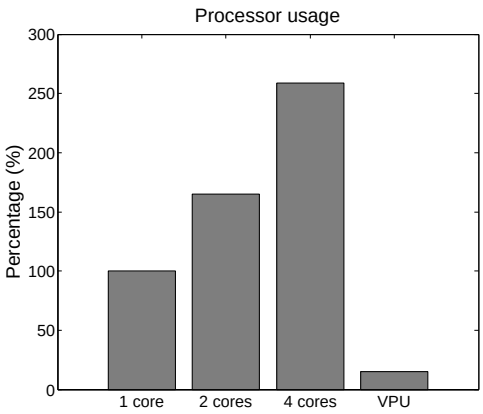

Figure 6.10: Processor usage of HD video decoding

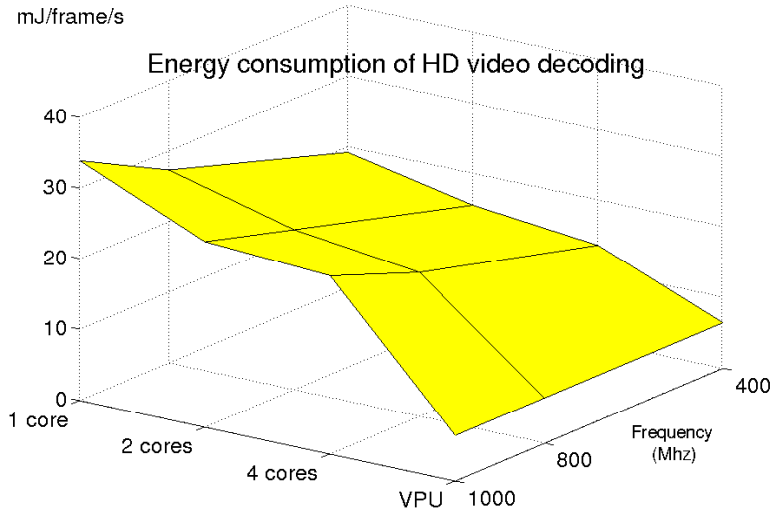

Figure 6.11: Energy consumption of HD video decoding (i.MX6 SoC)

	400 MHz	800 MHz	1000 MHz
1 core	19.16	27.24	33.85
2 cores	15.46 (x 0.80)	22.57 (x 0.82)	26.16 (x 0.77)
4 cores	13.55 (x 0.70)	20.30 (x 0.74)	25.12 (x 0.74)
VPU	6.41 (x 0.33)	6.53 (x 0.23)	6.61 (x 0.18)

Table 6.2: HD video decoding energy consumption (mJ/Frame) (i.MX6)

vations. In fact, in case of single-core video decoding (one thread), the processor usage is 100% which means that the decoding thread is all time in active state. However, it is around 160% and 260%[6] in case of dual-core and quad-core decoding respectively. However, when using the VPU, the processor usage is about 15% because the ARM cores are in idle mode almost all the time waiting for the frame to be decoded by the VPU.

6.3.3.2 Video decoding energy consumption

Table 6.2 shows the energy consumption of video decoding using the ARM cores and the VPU. The values between parentheses in Table 6.1 represent the energy saving factor as compared to single core decoding using the same frequency.

As expected, for a given clock frequency, increasing the number of cores allows to

(active time), T_{exe} is the decoding time.

[6]The processor usage is higher than 100% because the sum of the the times spent by the threads in different cores is higher than the total decoding time

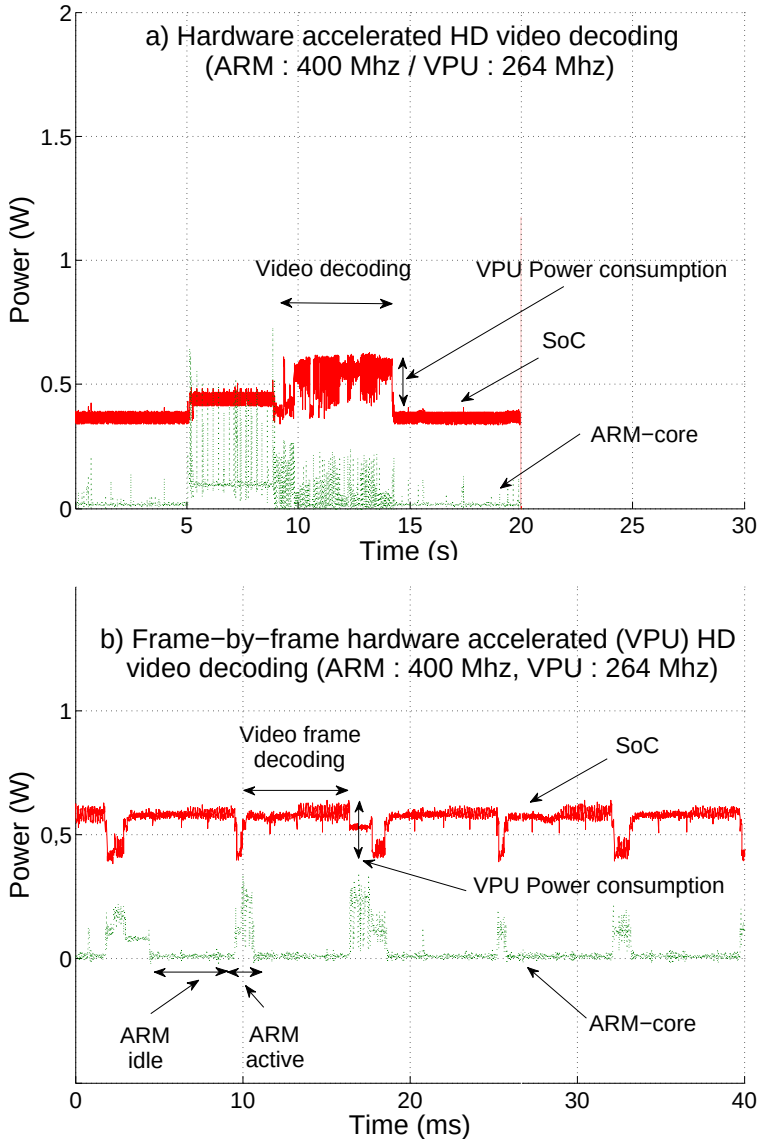

Figure 6.12: VPU HD Video decoding power consumption

reduce the energy consumption (see Fig. 6.11). For example, as compared to mono-core decoding, the optimal multi-core configuration (4 cores, 800 MHz)[7] deceases the energy by a factor of x0.74 while increasing the performance by a factor of x2.43.

On the other hand, the energy saving is much higher in case of VPU video decoding (0.23 scaling factor) as compared to mono-core decoding at 800 MHz and x0.36 as compared to the optimal multi-core video decoding (4 cores, 800 MHz). This can be explained by both a high decoding performance and a very low power consumption of the VPU. As illustrated in Fig. 6.12-a, we can observe that the decoding of the 480 video frames terminates in almost 5 seconds. During this decoding phase, the power

[7]The configuration which consumes the lowest amount of energy with a decoding speed higher than 24 fps

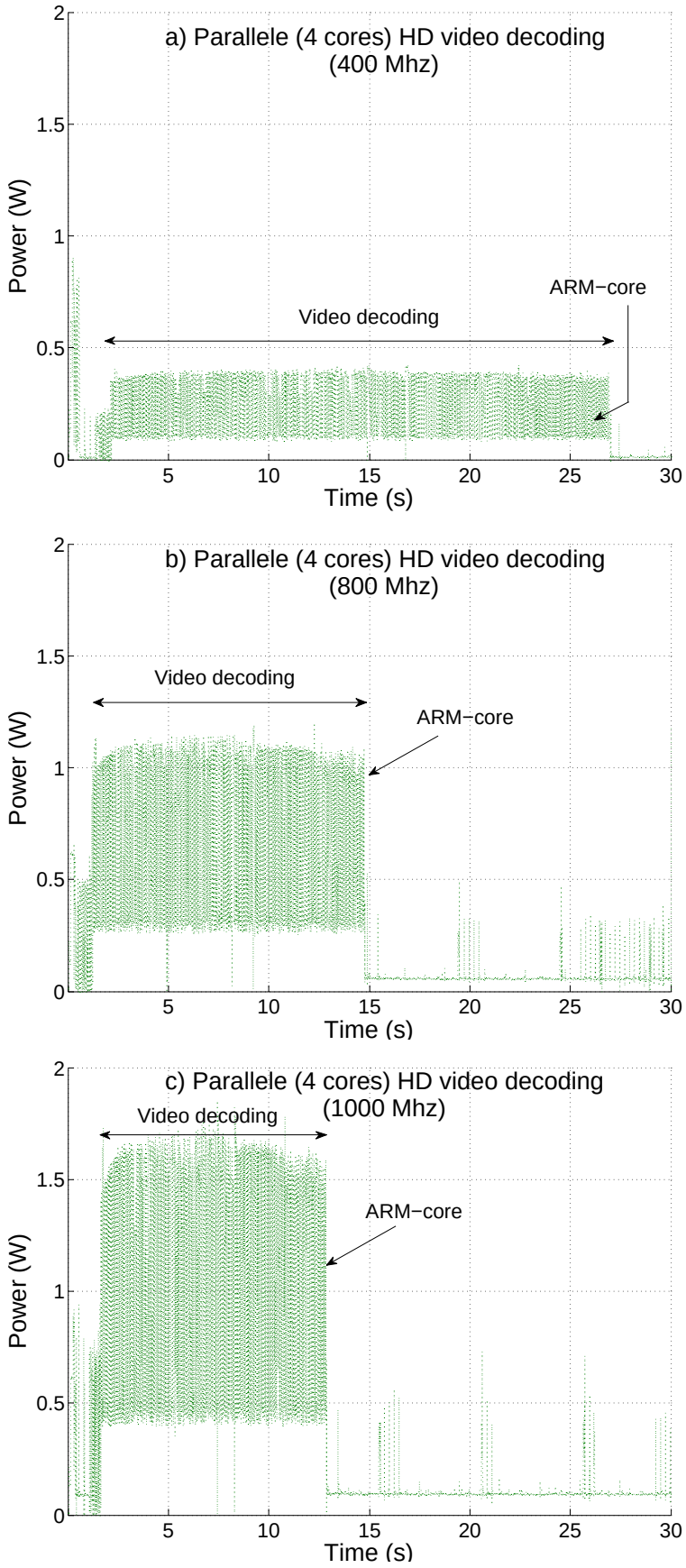

Figure 6.13: Parallel multi-core HD video decoding energy consumption

consumption of the SoC power domain increases with only 0.2 W which corresponds to the VPU power consumption. This low value can be explained by the low frequency (264 MHz) of the VPU. During this time, the ARM cores power consumption is negligible. In fact, as illustrated in Fig. 6.12-b showing the frame-by-frame power consumption variation, the ARM cores are almost all the time in idle state waiting for the VPU to decode a video frame. In the idle state, the ARM cores execute the WFI (Wait For Interrupt) instruction where most of the processor clocks are gated to reduce the power consumption.

Unlike the VPU decoding, multi-core video decoding cannot conciliate the performance and the energy efficiency. As illustrated in Fig. 6.13, at 400 MHz frequency (see Fig. 6.13-a), the power consumption is low (= 0.3 mW), but the decoding time is very long. On the other hand, at higher frequencies, the decoding time decreases but the power consumption increases considerably (see Fig. 6.13-b and c).

We can highlight that the unbalanced workload over the processor cores may be source of energy inefficiency. In fact, during a thread waiting time, the processor core continues to consume energy while doing nothing (see section 6.4 for more detailed comments).

The main conclusions we can draw from the above experimentations are twofold. First, when considering the overall system energy balance, the gap between the energy consumption of software and hardware accelerated video decoding is not so important as one may expect. In fact, while the difference between raw energy consumption of GPP and specialized processor is expected to be about two orders of magnitude (see section 2.3.3), we have noticed that the ratio between a hardware accelerated and 1 core ARM processor video decoding is 1/5 (see Table 6.2).

Second, the gap between the energy consumption of software hardware accelerated video decoding can be bridged using parallel decoding on GPP processors. In fact, a trivial parallel video decoding setup (4 cores, 800 MHz) without using any optimization leads to a ratio of only 1/3 between software and hardware decoding energy consumption. One may expect that the use of optimizations allows further reduction of this gap. To achieve this objective, it is necessary to develop performances and energy models considering this kind of multi-core architectures. In the following section, we discuss the main challenges and issues for developing such performance and energy models.

6.4 Discussion

The developed models in this thesis considered sequential video decoding on heterogeneous processing with different instruction set architectures (GPP and DSP). The application of the proposed models consists in either scaling the frequency of the used processors or the decoding migration between the DSP or the GPP. Both approaches are based on performance models related to a video sequence (a set of frames).

The use of parallel processing on muti-cores architecture allows to deal with high processing requirement of HD video decoding. However, to enhance the energy efficiency of the used cores, it is necessary to have fine-grained performance models to adjust the processing resources at small video unit basis. In fact, the decoder should be able to predict the decoding workload at a slice or frame basis and take an optimization decision accordingly. The new trends in multi-cores architecture offer two possible optimization strategies :

6.4.1 Per-core frequency scaling

We have shown that the parallelization of video decoding over multiple cores may lead to unbalanced workload (see section 6.3). The waiting time due to early decoding termination on a given core is a source of energy inefficiency. One approach to fix this issue is to set the clock frequency of each core depending on the assigned decoding workload (a slice or a frame).

In [121], the author shows that per-core DVFS enhances the energy efficiency of various benchmark workload. This needs to be investigated more deeply in case of video decoding. The main challenge faced to implement such strategy in case of video decoding is to predict the future workload accurately at a frame or slice granularity. In this context, the new possibilities offered by the Green Metadata standard may help in driving the decoder processing capabilities very accurately using assisting metadata sent by the encoder [118]. Currently, the standard support sending metadata at a video sequence and a frame basis. It would be interesting to study the impact of per-core DVFS in case of slice-based parallel video decoding to evaluate the relevance of sending metadata at a slice basis.

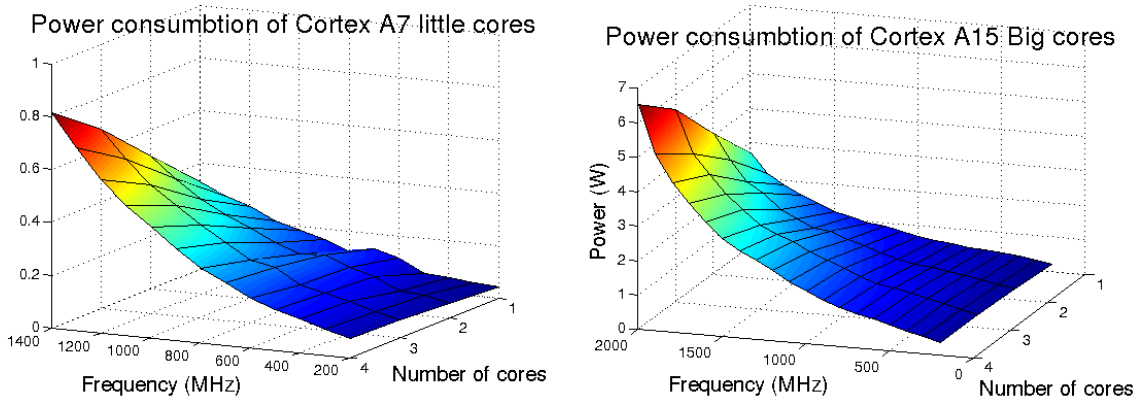

Figure 6.14: Power consumption of little and big core in Exynos 5422 SoC

6.4.2 Processing migration on asymmetric multi-cores

New trends in mobile multi-cores architecture are more and more moving toward the use of single instruction set architecture asymmetric cores. For example, the last big.LITTLE ARM architecture proposes a SoC based on four cortex A7 energy efficient cores and four Cortex A15 high throughput cores. In addition to the parallelism, the principle of energy saving on these architectures is to move low workload on little energy efficient core and high workload on the big cores. As compared to per-core frequency scaling, this approach has two advantages [122] : 1) The workload migration from one core to another has a lower overhead than scaling the core frequency. 2) Moving a workload to an energy efficient core is more energy efficient than simply scaling down the processor frequency. In fact, thanks to an optimized design (simpler pipe architecture, smaller cache memory, ...), the little cores offer better energy saving opportunities. For example, Fig. 6.14 illustrates the large gap between the power consumption of both little as compared to big cores in the Samsung Exynos 5422 Octa core SoC.

To leverage the energy efficiency of parallel video decoding on such kinds of architectures, the main faced challenges are to develop performance models which help the decoder to select which type of core to use to process a given workload. Although the decoder may be assisted by metadata information, it should have the capability to interpret them differently and accurately on the available heterogeneous architectures. This assumes the development of mechanism to map matadata workload information onto different target architectures.

6.5 Conclusion

In this chapter, we showed how to use the obtained results for enhancing the energy efficiency of adaptive video decoding. We have particularly proposed a proof of concept of a scheduling approach for saving energy of video decoding on heterogeneous SoC. Then, guidelines were provided for building a DVFS governor for driving the processor frequency in case of adaptive video decoding. Both proposed solutions need more extensive experimentation to be validated. We plan to address those issues in future works.

As a complementary work, we addressed in this chapter the issue of the energy efficiency of High Definition video decoding which was not covered in the proposed characterization and modeling methodology. We particularly focused on two architectures which are hardware accelerator and parallel multi-cores. The experimentations results suggest that parallel multi-cores video decoding is a promising technique which may have energy consumption close to hardware decoder if further optimization are used. Accordingly, we have pointed out the main challenges for enhancing the energy efficiency on parallel video decoding on such kind of multi-cores architecture.

CHAPTER 7

Conclusions and future works

This thesis focused on the energy consumption issue of video decoding which is one of the most energy intensive applications running on mobile devices. Due to ever growing of both mobile video traffic and the power consumption of the hardware, this issue is addressed actively by the community. The objective of our work is to contribute to this effort to bridge the gap between the explosion of power consumption and the mobile device autonomy.

In this this work, we aimed to enhance the understanding of the power consumption behavior of video decoding on modern heterogeneous SoCs. Accordingly, it was proposed an end-to-end methodology for characterizing and modeling the performance and the energy consumption of video decoding applications on heterogeneous SoC.

The proposed methodology is based on extensive performance and energy consumption measurement. This was motivated by our desire to build performance and energy models which represent as close as possible real life scenario. In addition, we gave a particular attention in our work to derive from the achieved experimentation the maximum information for making the developed models reusable and generalizable to other platforms different from those used in the experimentations.

For this purpose, our works was divided into two phases : characterization and modeling. The characterization part was achieved on different processing configurations including mono core GPP processor, DSP, multi-cores ARM processors and hardware video codec. On these architecture configurations, a large set of system and video parameters was covered. For example, the system overhead was evaluated and its impact on the overall performance and energy consumption was analyzed. Moreover, a wide range of video configurations was tested including different representative video

sequences coded into low, standard and high definition qualities.

The contribution of the characterization part of this work is twofold. First, the use of a unified methodology executed within a common multimedia framework allowed to evaluate in the same condition the performance and the energy consumption of decoding different video qualities on various processing architectures. Second, the multi-level characterization of the video decoding process highlighted the importance to consider different parameters which may pertain to different abstraction levels in evaluating the overall energy efficiency of a given system.

For instance, it was shown that considering jointly the system energy overhead and the video quality may lead to configurations where the GPP is more energy efficient than a DSP for video decoding. On the other hand, the analysis of the overall energy balance of parallel muti-core HD video decoding as compared to hardware codec, shows that the gap between the two approaches is not much important as it was expected.

The modeling part of this thesis covered two processor architectures : ARM processors and DSP. Based on the obtained characterization empirical results, mathematical performance and energy models were developed. Using a sub-model decomposition approach, the proposed models describe the performance and the energy consumption in terms of the video bit-rate and the clock frequency in addition to a set of comprehensive constant parameters related to the video complexity and the underlying architecture. The developed models are very accurate ($R^2 = 97\%$) for both GPP and DSP video decoding.

Moreover, it was shown that the combination of these different sub-models allows to build an accurate high level performance and energy model for video decoding. This result was used to provide a reduced complexity and fast energy model building and generalization methodology for a given target architecture. The key idea is to identify clearly the model parameters depending on the video and those depending on the underlying system and architecture. Only the latter should be calculated again when the target architecture changes.

Finally, some propositions for using the results of this study were proposed. Particularly, We explain how to use the proposed models (which consider the video quality and the processor frequency parameters) to dimension the processing resource in the context of adaptive video. These propositions are guidelines which need to be investigated more deeply as we will discuss hereafter in the future works that we plan.

Future works

The results of this thesis allowed to highlight interesting applications and open issues which need a deeper investigation. Hereafter, are listed some of them that we are planning to address in the future.

Video quality aware frequency scaling

We have provided guidelines for online video decoding performance model construction (see section 6.2) to adapt the processor frequency according to the video quality. One of the most challenging issue to implement such technique is the calibration of the model parameters to fit the target architecture.

Under the assumption that the multi-linear relationship of the model (which was verified for 3 processor architectures including a DSP), the regression coefficients can be calculated online using appropriate techniques such as adaptive filtering or neural network based regression. The objective is to provide to the video decoder some auto-learning capabilities for predicting the upcoming workload based on the workload history.

Performance scalability vs memory performance

One of the strength of the proposed performance model is that it describes analytically how the performance scaling varies when changing the video quality.

Based on previous studies in the literature reporting a relationship between the memory access rate and cache miss ratio, from one side, and the video quality from the other side, we have expressed the abstract model parameters in terms of memory hierarchy properties (see section 5.4.1). This mapping has been done a posteriori and need to be verified rigorously using experimentation. This was not possible on real platform used in our experimentation since it is not possible to modify their memory configuration.

As a future work, we plan to explore this issue using architecture simulators offering much more flexibility for configuring such low level details. According to our preliminary investigations, the GEM5 simulator [123] suits very well our needs. In fact, it supports cycle accurate DVFS simulation of various processor and memory architectures.

Generalization to other video codecs

In this study we focused on H.264/AVC, however, the methodology can be extended to H.265 (HEVC), the successor of H.264/AVC. In fact, although H.265 has introduced a lot of changes in the internal coding algorithms, it is still based on the same principles used in almost all MPEG codecs family. The high level video related parameters used in this study (qp, bit-rate, step-size) are still valid in H.265 standard. Thus, the characterization and the modeling methodology can be executed without any changes since they are independent from the codec internal details. However, the regression analysis may lead to different analytical forms of the rate and performance models. This will be the focus of some future investigations.

Energy efficiency of parallel video decoding

We have presented in this thesis a preliminary experimental study to evaluate the energy efficiency of parallel multi-core HD video decoding as compared to hardware codec. Indeed, the hardware codec is more energy efficient than the parallel GPP cores, however, the gap separating the two approaches is not much important. One may expect to reduce it if some optimizations are achieved in the decoding process. The objective is to conciliate the flexibility of the software codecs and energy efficiency of the hardware accelerated.

From our point of view, a load balancing strategy combined with a per-core DVFS scaling and/or thread migration over heterogeneous asymmetric cores is a promising technique to save energy of parallel video decoding. This is one of the open issues we plan to investigate in our future works.

Participation to the MPEG Green meta-data standardization

Finally, we would like to conclude this work by highlighting the ever growing interest of the mobile device industry for the energy saving considerations of video applications. This has been concertized recently by the MPEG Green meta-data standardization initiative [118] grouping the most important actors in the mobile devices industry. This may be an interesting opportunity to federate the research works in the field of energy consumption of video applications and to encourage the use of their results in real industry products.

In this context, a collaboration work is in progress with the GreenVideo FUI project

[124] to use the results obtained in this thesis in an experimental video decoder. Particularly, we are working on integrating the proposed performance model in a DASH video player supporting Green meta-data. This player is developed by *Thomson Video Network*, one of the industrial partners involved in the GreenVideo project.

Bibliography

[1] International Technology Roadmap for Semiconductors. Design. `http://www.itrs.net/Links/2012ITRS/2012Chapters/2012Overview.pdf`, 2012.

[2] M. Horowitz. Computing's energy problem (and what we can do about it). In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International*, pages 10–14, Feb 2014.

[3] Jae-Beom Lee, Myoung-Jin Kim, Sungroh Yoon, and Eui-Young Chung. Application-support particle filter for dynamic voltage scaling of multimedia applications. *Computers, IEEE Transactions on*, 61(9):1256–1269, Sept 2012.

[4] Georgios Keramidas, Vasileios Spiliopoulos, and Stefanos Kaxiras. Interval-based models for run-time DVFS orchestration in superscalar processors. In *Proceedings of the 7th ACM International Conference on Computing Frontiers*, CF '10, pages 287–296, New York, NY, USA, 2010. ACM.

[5] Y. Benmoussa, J. Boukhobza, E. Senn, and D. Benazzouz. Evaluation of the performance/energy overhead in dsp video decoding and its implications. In *Digital System Design (DSD), 2013 Euromicro Conference on*, 2013.

[6] Yahia Benmoussa, Jalil Boukhobza, Yassine Hadjadj Aoul, Loïc Lagadec, and Djamel Benazzouz. Behavioral system level power consumption modeling of mobile video streaming applications. In *GDR SoC SIP Workshop*, 2012.

[7] Michel Broussely and Graham Archdale. Li-ion batteries and portable power source prospects for the next 5-10 years. *Journal of Power Sources*, 136(2):386–394, October 2004.

[8] Cisco. Cisco visual networking index: Global mobile data traffic forecast update. `http://bit.ly/bwGY7L`, 2013.

[9] OOYALA. Ooyala Global Video Index Q2 2013. `http://go.ooyala.com/wf-video-index-q2-2013.html`, 2013.

[10] A. Carroll and G. Heiser. An analysis of power consumption in a smartphone. *Proceedings of the USENIX Annual Technical Conference*, pages 21–28, 2010.

[11] Aaron Carroll and Gernot Heiser. The systems hacker's guide to the galaxy energy usage in a modern smartphone. In *Proceedings of the 4th Asia-Pacific Workshop on Systems*, APSys '13, pages 5:1–5:7, New York, NY, USA, 2013. ACM.

[12] Thomas Stockhammer. Dynamic adaptive streaming over HTTP : standards and design principles. *Proceedings of the second annual ACM conference on Multimedia systems*, pages 133–144, 2011.

[13] R. Trestian, O. Ormond, and G.-M. Muntean. Enhanced power-friendly access network selection strategy for multimedia delivery over heterogeneous wireless networks. *Broadcasting, IEEE Transactions on*, 60(1):85–101, March 2014.

[14] Shekhar Borkar. Thousand core chips: A technology perspective. In *Proceedings of the 44th Annual Design Automation Conference*, DAC '07, pages 746–749. ACM, 2007.

[15] D. Markovic, V. Stojanovic, B. Nikolic, M.A. Horowitz, and R.W. Brodersen. Methods for true energy-performance optimization. *Solid-State Circuits, IEEE Journal of*, 39(8):1282–1293, 2004.

[16] A. Wang and A. Chandrakasan. Energy-efficient DSPs for wireless sensor networks. *Signal Processing Magazine, IEEE*, 19(4):68–78, 2002.

[17] C. H. (Kees) Van Berkel. Multi-core for mobile phones. *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 1260–1265, 2009.

[18] B. Ouni, C. Belleudy, S. Bilavarn, and E. Senn. Embedded operating systems energy overhead. In *Design and Architectures for Signal and Image Processing (DASIP), 2011 Conference on*, pages 1–6, Nov 2011.

[19] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. In *Computer Architecture, 2000. Proceedings of the 27th International Symposium on*, pages 83–94, 2000.

[20] Gilberto Contreras and Margaret Martonosi. Power prediction for intel xscale® processors using performance monitoring unit events. In *Low Power Electronics and Design, 2005. ISLPED'05. Proceedings of the 2005 International Symposium on*, pages 221–226. IEEE, 2005.

[21] V. Spiliopoulos, A. Bagdia, A. Hansson, P. Aldworth, and S. Kaxiras. Introducing dvfs-management in a full-system simulator. In *Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), 2013 IEEE 21st International Symposium on*, pages 535–545, Aug 2013.

[22] Yahia Benmoussa, Jalil Boukhobza, Eric Senn, and Djamel Benazzouz. Gpp vs dsp: A performance/energy characterization and evaluation of video decoding. In *Proceedings of the 2013 IEEE 21st International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems*, MASCOTS '13, pages 273–282. IEEE Computer Society, 2013.

[23] Yahia Benmoussa, Eric Senn, Jalil Boukhobza, Mickael Lanoe, and Djamel Benazzouz. Open-PEOPLE, a collaborative platform for remote & accurate measurement and evaluation of embedded systems power consumption. *in Proceedings of the IEEE 22nd International Symposium On Modeling, Analysis And Simulation Of Computer And Telecommunication Systems*, 2014.

[24] Yahia Benmoussa, Jalil Boukhobza, Eric Senn, and Djamel Benazzouz. On the energy efficiency of parallel multi-core vs hardware accelerated hd video decoding. *SIGBED Rev.*, 11(4), February 2014.

[25] Y. Benmoussa, J. Boukhobza, E. Senn, and D. Benazzouz. Energy consumption modeling of h.264/avc video decoding for gpp and dsp. In *Digital System Design (DSD), 2013 Euromicro Conference on*, pages 890–897, Sept 2013.

[26] Yahia Benmoussa, Jalil Boukhobza, Eric Senn, Yassine Hadjadj-Aoul, and Djamel Benazzouz. A methodology for performance/energy consumption char-

acterization and modeling of video decoding on heterogeneous soc and its applications. *Journal of Systems Architecture*, 61(1):49 – 70, 2015.

[27] Yahia Benmoussa, Jalil Boukhobza, Eric Senn, Yassine Hadjadj-Aoul, and Djamel Benazzouz. Dyps: Dynamic processor switching for energy-aware video decoding on multi-core socs. *SIGBED Rev.*, 11(1):56–61, February 2014.

[28] Quan Huynh-Thu and Mohammed Ghanbari. Scope of validity of psnr in image/video quality assessment. *Electronics letters*, 44(13):800–801, 2008.

[29] ClemensC. Wst, Liesbeth Steffens, WimF.J. Verhaegh, ReinderJ. Bril, and Christian Hentschel. Qos control strategies for high-quality video processing. *Real-Time Systems*, 30(1-2):7–29, 2005.

[30] Zhijian Lu, J. Lach, M. Stan, and K. Skadron. Reducing multimedia decode power using feedback control. *Computer Design, 2003. Proceedings of the 21st International Conference on*, pages 489–496, 2003.

[31] Ravindra Jejurikar, Cristiano Pereira, and Rajesh Gupta. Leakage aware dynamic voltage scaling for real-time embedded systems. In *Proceedings of the 41st Annual Design Automation Conference*, DAC '04, pages 275–280, New York, NY, USA, 2004. ACM.

[32] T.D. Burd and R.W. Brodersen. Energy efficient CMOS microprocessor design. *System Sciences, Proceedings of the Twenty-Eighth Hawaii International Conference on*, 1:288–297, 1995.

[33] N.S. Kim, T. Austin, D. Baauw, T. Mudge, K. Flautner, J.S. Hu, M.J. Irwin, M. Kandemir, and V. Narayanan. Leakage current: Moore's law meets static power. *Computer*, 36(12):68–75, 2003.

[34] Konstantin Moiseev, Avinoam Kolodny, and Shmuel Wimer. Timing-aware power-optimal ordering of signals. *ACM Trans. Des. Autom. Electron. Syst.*, 13(4):65:1–65:17, October 2008.

[35] Jacob R. Lorch and Alan Jay Smith. Improving dynamic voltage scaling algorithms with pace. *SIGMETRICS Perform. Eval. Rev.*, 29(1):50–61, June 2001.

[36] D. Meisner, C.M. Sadler, L.A. Barroso, W. Weber, and T.F. Wenisch. Power management of online data-intensive services. In *Computer Architecture (ISCA), 2011 38th Annual International Symposium on*, pages 319–330, June 2011.

[37] Tajana Simunic, Luca Benini, Peter Glynn, and Giovanni De Micheli. Dynamic power management for portable systems. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, MobiCom '00, pages 11–19, New York, NY, USA, 2000. ACM.

[38] Frank Hansen and Gert Kjærgård Pedersen. Jensen's inequality for operators and löwner's theorem. *Mathematische Annalen*, 258(3):229–241, 1982.

[39] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced cpu energy. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, FOCS '95, pages 374–, Washington, DC, USA, 1995. IEEE Computer Society.

[40] V. Gutnik and A.P. Chandrakasan. Embedded power supply for low-power dsp. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 5(4):425–435, 1997.

[41] Kihwan Choi, R. Soma, and M. Pedram. Fine-grained dynamic voltage and frequency scaling for precise energy and performance tradeoff based on the ratio of off-chip access to on-chip computation times. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 24(1):18–28, Jan 2005.

[42] Cor Meenderinck, Arnaldo Azevedo, Mauricio Alvarez, Ben Juurlink, and Alex Ramirez. Parallel scalability of h. 264. In *Proceedings of the first Workshop on Programmability Issues for Multi-Core Computers*, 2008.

[43] M. Horowitz, E. Alon, D. Patil, S. Naffziger, Rajesh Kumar, and K. Bernstein. Scaling, power, and the future of cmos. In *Electron Devices Meeting, 2005. IEDM Technical Digest. IEEE International*, pages 7 pp.–15, Dec 2005.

[44] Rehan Hameed, Wajahat Qadeer, Megan Wachs, Omid Azizi, Alex Solomatnikov, Benjamin C. Lee, Stephen Richardson, Christos Kozyrakis, and Mark Horowitz. Understanding sources of inefficiency in general-purpose chips. *SIGARCH Comput. Archit. News*, 38(3):37–47, 2010.

[45] Wajahat Qadeer, Rehan Hameed, Ofer Shacham, Preethi Venkatesan, Christos Kozyrakis, and Mark A Horowitz. Convolution engine: balancing efficiency & flexibility in specialized computing. In *Proceedings of the 40th Annual International Symposium on Computer Architecture*, pages 24–35. ACM, 2013.

[46] Ke Xu, Tsu-Ming Liu, Jiun-In Guo, and Chiu-Sing Choy. Methods for power/throughput/area optimization of H.264/AVC decoding. *Journal of Signal Processing Systems*, 60(1):131–145, 2010.

[47] K. Iwata, T. Irita, S. Mochizuki, H. Ueda, M. Ehama, M. Kimura, J. Takemura, K. Matsumoto, E. Yamamoto, T. Teranuma, K. Takakubo, H. Watanabe, S. Yoshioka, and T. Hattori. A 342 mw mobile application processor with full-hd multi-standard video codec and tile-based address-translation circuits. *Solid-State Circuits, IEEE Journal of*, 45(1):59–68, Jan 2010.

[48] Olli Silvén and Tero Rintaluoma. Energy efficiency of video decoder implementations. In *Mobile Phone Programming*, pages 421–439. Springer, 2007.

[49] Gerard J.M. Smit, André B.J. Kokkeler, Pascal T. Wolkotte, and Marcel D. van de Burgwal. Multi-core architectures and streaming applications. In *Proceedings of the 2008 international workshop on System level interconnect prediction*, SLIP '08, pages 35–42. ACM, 2008.

[50] Michael Macedonia. The gpu enters computing's mainstream. *Computer*, 36(10):106–108, 2003.

[51] Guobin Shen, Guang-Ping Gao, Shipeng Li, Heung-Yeung Shum, and Ya-Qin Zhang. Accelerate video decoding with generic gpu. *Circuits and Systems for Video Technology, IEEE Transactions on*, 15(5):685–693, 2005.

[52] Bart Pieters, Dieter Van Rijsselbergen, Wesley De Neve, and Rik Van de Walle. Performance evaluation of h. 264/avc decoding and visualization using the gpu. In *Optical Engineering+ Applications*, pages 669606–669606. International Society for Optics and Photonics, 2007.

[53] Huifang Deng, Chunhui Deng, and Jingjing Li. Gpu-based real-time decoding technique for high-definition videos. In *Intelligent Information Hiding and Mul-*

timedia Signal Processing (IIH-MSP), 2012 Eighth International Conference on, pages 186–190. IEEE, 2012.

[54] Texas Instruments. Tms320 dsp/bios users guide. *reference spru423b*, 2002.

[55] A.P. Chandrakasan, S. Sheng, and R.W. Brodersen. Low-power CMOS digital design. *IEEE Journal of Solid-State Circuits*, 27(4):473 –484, 1992.

[56] Suhwan Kim and Marios C Papaefthymiou. Reconfigurable low energy multiplier for multimedia system design. In *VLSI, 2000. Proceedings. IEEE Computer Society Workshop on*, pages 129–134. IEEE, 2000.

[57] M. Alvarez, E. Salami, A. Ramirez, and M. Valero. A performance characterization of high definition digital video decoding using H.264/AVC. pages 24–33, October 2005.

[58] Matthew J. Holliman, Eric Q. Li, and Yen kuang Chen. Mpeg decoding workload characterization. *Proceedings of Workshop on Computer Architecture Evaluation Using Commercial Workloads*, 2003.

[59] Shiao-Li Tsao and Sung-Yuan Lee. Performance evaluation of inter-processor communication for an embedded heterogeneous multi-core processor. *Journal of Information Science and Engineering*, 28(3):537–554, 2012.

[60] P. Ramachandra and M. R. Satish. H.264 main profile video decoding implementation techniques on OMAP3430IVA. *Signal Processing (ICSP), IEEE 10th International Conference on*, pages 271–274, 2010.

[61] S. Kant, U. Mithun, and P. S S B K Gupta. Real time H.264 video encoder implementation on a programmable dsp processor for videophone applications. *Consumer Electronics, 2006. ICCE '06. 2006 Digest of Technical Papers. International Conference on*, pages 93–94, 2006.

[62] Tse-Tsung Shih, Chia-Lin Yang, and Yi-Shin Tung. Workload characterization of the h. 264/avc decoder. In *Advances in Multimedia Information Processing-PCM 2004*, pages 957–966. Springer, 2005.

[63] T. Austin, E. Larson, and D. Ernst. Simplescalar: an infrastructure for computer system modeling. *Computer*, 35(2):59–67, 2002.

[64] Nathan T. Slingerland and Alan Jay Smith. Cache performance for multimedia applications. In *Proceedings of the 15th International Conference on Supercomputing*, ICS '01, pages 204–217, New York, NY, USA, 2001. ACM.

[65] Z. Xu, S. Sohoni, Rui Min, and Y. Hu. An analysis of cache performance of multimedia applications. *Computers, IEEE Transactions on*, 53(1):20–38, Jan 2004.

[66] Cor Meenderinck, Arnaldo Azevedo, Ben Juurlink, Mauricio Alvarez Mesa, and Alex Ramirez. Parallel scalability of video decoders. *J. Signal Process. Syst.*, 57(2):173–194, November 2009.

[67] Chu-Hsing Lin, Jung-Chun Liu, and Chun-Wei Liao. Energy analysis of multimedia video decoding on mobile handheld devices. In *Multimedia and Ubiquitous Engineering, 2007. MUE'07. International Conference on*, pages 120–125. IEEE, 2007.

[68] Hendrik Eeckhaut, Harald Devos, and Dirk Stroobandt. The energy scalability of wavelet-based, scalable video decoding. In *PATMOS*, pages 363–372, 2007.

[69] Yi-Chu Wang and Kwang-Ting Cheng. Energy and performance characterization of mobile heterogeneous computing. In *Signal Processing Systems (SiPS), 2012 IEEE Workshop on*, pages 312–317. IEEE, 2012.

[70] Elias Baaklini, Santhosh Rethinagiri, Hassan Sbeity, and Smail Niar. Scalable row-based parallel h.264 decoder on embedded multicore processors. *Signal, Image and Video Processing*, pages 1–15, 2014.

[71] Johan Pouwelse, Koen Langendoen, and Henk Sips. Dynamic voltage scaling on a low-power microprocessor. *Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 251–259, 2001.

[72] Johan Pouwelse, Koen Langendoen, Inald Lagendijk, and Henk Sips. Power-aware video decoding. In *in 22nd Picture Coding Symposium, Seoul, Korea*, pages 303–306, 2001.

[73] Andy C. Bavier, A. Brady Montz, and Larry L. Peterson. Predicting mpeg execution times. *SIGMETRICS Perform. Eval. Rev.*, 26(1):131–140, June 1998.

[74] Ying Tan, P. Malani, Qinru Qiu, and QingWu. Workload prediction and dynamic voltage scaling for mpeg decoding. In *Design Automation, 2006. Asia and South Pacific Conference on*, pages 6 pp.–, Jan 2006.

[75] Sung-Yong Bang, Kwanhu Bang, Sungroh Yoon, and Eui-Young Chung. Run-time adaptive workload estimation for dynamic voltage scaling. *Trans. Comp.-Aided Des. Integ. Cir. Sys.*, 28(9):1334–1347, September 2009.

[76] M. Mattavelli, S. Brunetton, and D. Mlynek. Implementing real-time video decoding on multimedia processors by complexity prediction techniques. In *Consumer Electronics, 1998. ICCE. 1998 Digest of Technical Papers. International Conference on*, pages 264–265, June 1998.

[77] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro. H.264/AVC baseline profile decoder complexity analysis. *Circuits and Systems for Video Technology, IEEE Trans on*, 13(7):704–716, 2003.

[78] Zhan Ma, Hao Hu, and Yao Wang. On complexity modeling of H.264/AVC video decoding and its application for energy efficient decoding. *IEEE Trans on Multimedia*, 13(6):1240 –1255, December 2011.

[79] Mark Weiser, Brent Welch, Alan Demers, and Scott Shenker. Scheduling for reduced cpu energy. In *Proceedings of the 1st USENIX Conference on Operating Systems Design and Implementation*, OSDI '94, Berkeley, CA, USA, 1994. USENIX Association.

[80] Kinshuk Govil, Edwin Chan, and Hal Wasserman. Comparing algorithm for dynamic speed-setting of a low-power cpu. In *Proceedings of the 1st Annual International Conference on Mobile Computing and Networking*, MobiCom '95, pages 13–25, New York, NY, USA, 1995. ACM.

[81] Amit Sinha and Anantha P Chandrakasan. Dynamic voltage scheduling using adaptive filtering of workload traces. In *VLSI Design, 2001. Fourteenth International Conference on*, pages 221–226. IEEE, 2001.

[82] T. Pering, T. Burd, and R. Brodersen. The simulation and evaluation of dynamic voltage scaling algorithms. In *Low Power Electronics and Design, 1998. Proceedings. 1998 International Symposium on*, pages 76–81, Aug 1998.

[83] Xiaotao Liu, Prashant Shenoy, and Weibo Gong. A time series-based approach for power management in mobile processors and disks. In *Proceedings of the 14th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '04, pages 74–79, New York, NY, USA, 2004. ACM.

[84] George Edward Pelham Box and Gwilym Jenkins. *Time Series Analysis, Forecasting and Control*. Holden-Day, Incorporated, 1990.

[85] Dirk Grunwald, Charles B. Morrey, III, Philip Levis, Michael Neufeld, and Keith I. Farkas. Policies for dynamic clock scheduling. In *Proceedings of the 4th Conference on Symposium on Operating System Design & Implementation - Volume 4*, OSDI'00, pages 6–6, Berkeley, CA, USA, 2000. USENIX Association.

[86] Rustam Miftakhutdinov, Eiman Ebrahimi, and Yale N. Patt. Predicting performance impact of dvfs for realistic memory systems. In *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-45, pages 155–165, Washington, DC, USA, 2012. IEEE Computer Society.

[87] Thomas L. Martin and Daniel P. Siewiorek. The impact of battery capacity and memory bandwidth on cpu speed-setting: A case study. In *Proceedings of the 1999 International Symposium on Low Power Electronics and Design*, ISLPED '99, pages 200–205, New York, NY, USA, 1999. ACM.

[88] T.L. Martin and D.P. Siewiorek. Nonideal battery and main memory effects on cpu speed-setting for low power. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 9(1):29–34, Feb 2001.

[89] Spec2000. www.spec.org/cpu2000/analysis/memory/, 2014.

[90] Andreas Weissel and Frank Bellosa. Process cruise control: Event-driven clock scaling for dynamic power management. In *Proceedings of the 2002 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*, CASES '02, pages 238–246, New York, NY, USA, 2002. ACM.

[91] J. Choi and H. Cha. Memory-aware dynamic voltage scaling for multimedia applications. *Computers and Digital Techniques, IEE Proceedings* -, 153(2):130–136, March 2006.

[92] Zhan Ma, F.C.A Fernandes, and Yao Wang. Power-rate-quality optimized scalable video adaptation. In *Information Sciences and Systems (CISS), 2012 46th Annual Conference on*, pages 1–6, March 2012.

[93] Xin Li, Zhan 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 (VCIP), 2012 IEEE*, pages 1 –6, 2012.

[94] Eduardo Jurez, Fernando Pescador, Pedro J. Lobo, A. Groba, and C. Sanz. Distortion-energy analysis of an OMAP-Based H.264/SVC decoder. *Mobile Multimedia Communications*, (77):544–559, January 2012.

[95] Bassem Ouni, Ccile Belleudy, and Eric Senn. Accurate energy characterization of os services in embedded systems. *EURASIP Journal on Embedded Systems*, 2012(1), 2012.

[96] Saadia Dhouib, Eric Senn, Jean-Philippe Diguet, Dominique Blouin, and Johann Laurent. Energy and power consumption estimation for embedded applications and operating systems. *Journal of Low Power Electronics*, 5(4):416–428, 2009.

[97] Eric Senn, Johann Laurent, Nathalie Julien, and Eric Martin. Softexplorer: Estimation, characterization, and optimization of the power and energy consumption at the algorithmic level. In Enrico Macii, Vassilis Paliouras, and Odysseas Koufopavlou, editors, *Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation*, volume 3254 of *Lecture Notes in Computer Science*, pages 342–351. Springer Berlin Heidelberg, 2004.

[98] Sheng Li, Jung-Ho Ahn, R.D. Strong, J.B. Brockman, D.M. Tullsen, and N.P. Jouppi. Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*, pages 469–480, 2009.

[99] Muhammad Shafique and Jörg Henkel. Low power design of the next-generation high efficiency video coding. In *ASP-DAC*, pages 274–281, 2014.

[100] Young-Hwan Park, Sudeep Pasricha, Fadi J Kurdahi, and Nikil Dutt. System level power estimation methodology with h. 264 decoder prediction ip case study.

In *Computer Design, 2007. ICCD 2007. 25th International Conference on*, pages 601–608. IEEE, 2007.

[101] T. Wiegand, G.J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the H.264/AVC video coding standard. *IEEE Trans on Circuits and Systems for Video Technology*, 13(7):560 –576, July 2003.

[102] Zhan Ma, Meng Xu, Yen-Fu Ou, and Yao Wang. Modeling of rate and perceptual quality of compressed video as functions of frame rate and quantization step-size and its applications. *IEEE Trans on Circuits and Systems for Video Technology*, 22(5):671 –682, May 2012.

[103] Texas Instruments. OMAP3530 Power Estimation Spreadsheet. `http://processors.wiki.ti.com/index.php/OMAP3530_Power_Estimation_Spreadsheet`, 2012.

[104] Loren Merritt and Rahul Vanam. x264: A high performance H.264/AVC encoder. 2006.

[105] OMAPPEDIA. Power management debug and profiling. http://www.omappedia.org/wiki/Power_Management_Debug_and_Profiling, 2012.

[106] Texas Instruments. Local Power Manager Driver. `http://software-dl.ti.com/dsps/dsp_public_sw/sdo_sb/targetcontent/lpm/index.html`, 2012.

[107] Venkatesh Pallipadi and Alexey Starikovskiy. The ondemand governor: past, present and future. *Proceedings of Linux Symposium*, pages 223–238, 2006.

[108] Chase Maupin Don Darling and Brijesh Singh. Gstreamer on texas instruments OMAP35x processors. *Proceedings of the Ottawa Linux Symposium*, pages 69–78, 2009.

[109] ARM. The ARM NEON general-purpose SIMD. www.arm.com/products/processors/technologies/neon.php, 2012.

[110] Texas Instruments. Codec Engine Profiling - Embedded Processors Wiki. `http://processors.wiki.ti.com/index.php/Codec_Engine_Profiling`, 2013.

[111] John Levon and Philippe Elie. Oprofile: A system profiler for linux. url-http://oprofile.sf.net, 2004.

[112] Mathworks. Linear regression with interaction effects. fr.mathworks.com/help/stats/linear-regression-with-interaction-effects.html, 2014.

[113] J. Choi and H. Cha. Memory-aware dynamic voltage scaling for multimedia applications. *Computers and Digital Techniques, IEEE Proceedings*, 153(2):130 – 136, march 2006.

[114] Texas Instruments. OMAP4 mobile applications platform. `http://www.ti.com/lit/ml/swpt034b/swpt034b.pdf`, 2012.

[115] PandaBoard Project. Pandaboard. `http://www.pandaboard.org`, 2013.

[116] Kenzo Van Craeynest, Aamer Jaleel, Lieven Eeckhout, Paolo Narvaez, and Joel Emer. Scheduling heterogeneous multi-cores through performance impact estimation (pie). *SIGARCH Comput. Archit. News*, 40(3):213–224, June 2012.

[117] Kisoo Yu, Donghee Han, Changhwan Youn, Seungkon Hwang, and Jaechul Lee. Power-aware task scheduling for big.little mobile processor. In *SoC Design Conference (ISOCC), 2013 International*, pages 208–212, Nov 2013.

[118] The Moving Picture Experts Group. MPEG systems technologies part 11: Energy-efficient media consumption (green metadata). `http://mpeg.chiariglione.org/sites/default/files/files/\standards/parts/docs/w14344-v2-w14344.zip`, 2014.

[119] Xiaorui Wang, Kai Ma, and Yefu Wang. Adaptive power control with online model estimation for chip multiprocessors. *Parallel and Distributed Systems, IEEE Transactions on*, 22(10):1681–1696, 2011.

[120] *i.MX 6Dual/6Quad Power Consumption Measurement*, Freescale Semiconductor, 2012.

[121] Wonyoung Kim, M.S. Gupta, Gu-Yeon Wei, and D. Brooks. System level analysis of fast, per-core dvfs using on-chip switching regulators. In *High Performance*

Computer Architecture, 2008. HPCA 2008. IEEE 14th International Symposium on, pages 123–134, Feb 2008.

[122] Krishna K Rangan, Gu-Yeon Wei, and David Brooks. Thread motion: fine-grained power management for multi-core systems. In *ACM SIGARCH Computer Architecture News*, volume 37, pages 302–313. ACM, 2009.

[123] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R Hower, Tushar Krishna, Somayeh Sardashti, et al. The gem5 simulator. *ACM SIGARCH Computer Architecture News*, 39(2):1–7, 2011.

[124] Green video project. `http://greenvideo.insa-rennes.fr`, 2014.