Power optimization strategies within a H.264 encoding system-on-chip

Ngoc-Mai Nguyen

VNU
ĐẠI HỌC QUỐC GIA HÀ NỘI
Vietnam National University, Hanoi

UNIVERSITÉ GRENOBLE ALPES

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

préparée dans le cadre d'une cotutelle entre l'*Université Grenoble Alpes* et l'*Université Nationale du Vietnam, Hanoi*

Spécialité : **Automatique - Productique**

Arrêté ministériel : le 6 janvier 2005 - 7 août 2006

Présentée par

Ngoc-Mai NGUYEN

Thèse dirigée par **Xuan-Tu TRAN** et **Suzanne LESECQ**
codirigée par **Edith BEIGNÉ**

préparée au sein des **Laboratoire d'Electronique et des Technologies de l'Information, CEA Grenoble et Laboratoire SIS, VNU-UET, Hanoi**

dans **l'École Doctorale d'Electronique, Electrotechnique, Automatique et Traitement du Signal (EEATS) et l'UET du VNU**

Stratégies d'optimisation de la consommation pour un système sur puce encodeur H.264

Thèse soutenue publiquement le **29/06/2015**,
devant le jury composé de :

M. Nicolas MARCHAND
Laboratoire GIPSA-Lab, Président
M. Alain MERIGOT
Université Paris XI, Rapporteur
Mme. Nadine AZEMARD
LIRMM, Rapporteur
M. Xuan-Tu TRAN
Université Nationale du Vietnam, Directeur de thèse
M. Amara AMARA
ISEP, Membre
M. Huu-Duc NGUYEN
Université Nationale du Vietnam, Membre
Mme. Edith BEIGNÉ
CEA, Invité
Mme. Suzanne LESECQ
CEA, Co-directeur de thèse, Invité

Contents

List of Abbreviations

Abbreviation	Description
AHB	Advanced High-performance Bus
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
ASIC	Application-Specific Integrated Circuit
AVS	Audio Video coding Standard
ATSC	Advanced Television Systems Committee
AVC	Advanced Video Coding
BBC R&D	British Broadcasting Corporation Research and Development
CABAC	Context Adaptive Binary Arithmetic Coding
CAVLC	Context Adaptive Variable Length Coding
CCITT	Comité Consultatif International Téléphonique et Télégraphique (International Telegraph and Telephone Consultative Committee), renamed ITU-T in 1993
CD	Compact Disc
CIF	Common Intermediate Format
DCSS	Dynamic Clock Supply Stop
DF	De-blocking Filter
DFS	Dynamic Frequency Scaling
DPM	Dynamic Power Management
DVD	Digital Video Disc
DVFS	Dynamic Frequency and Voltage Scaling
DVS	Dynamic Voltage Scaling
Exp-Golomb	Expotential Golomb (coding)
FIFO	First In, First Out
FME	Fractional Motion Estimation
FPGA	Field-Programmable Gate Array

Abbreviation	Description
fps	frame per second
HD	High Definition
HDL	Hardware Description Language
HEVC	High Efficiency Video Coding
HTML	HyperText Markup Language
IC	Integrated Circuit
IEC	International Electrotechnical Commission
IME	Integer Motion Estimation
IP	Intellectual Property
ISDN	Integrated Services for Digital Network
ISO	International Organization for Standardization
ITU	International Telecommunication Union
ITU-T	ITU Telecommunication Standardization Sector
JVT	Joint Video Team
kbps	kilobits per second
LUT	Look-Up Table
MAU	Memory Access Unit
Mbps	Mega-bits per second
MC	Motion Compensation
ME	Motion Estimation
MPEG	Moving Picture Experts Group
NCC	Normalized Cross Correlation
NTSC	National Television System Committee
P2P	Point-to-Point
PAL	Phase Alternating Line
PMRME	Parallel Multi-Resolution Motion Estimation
PSNR	Peak Signal-to-Noise Ratio
QCIF	Quarter Common Intermediate Format
QoS	Quality of Service
RAM	Random-Access Memory
ROM	Read-Only Memory
RTL	Register Transfer Level
SAD	Sum of Absolute Differences
SATD	Absolute Transformed Differences
SD	Standard Definition
SHD	Sum of Hamming Distances
SMPTE	Society of Motion Picture and Television Engineers
SoC	System-on-Chip

Abbreviation	**Description**
SSD	Sum of Squared Differences
SW	Search Window
TV	Television
UHDT	Ultra High Definition Television
VCD	Video Compact Disc
VCEG	Video Coding Experts Group
VENGME	Video Encoder for the Next Generation Multimedia Equipment
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
VLC	Variable Length Coding
VLSI	Very-Large-Scale Integration

List of Figures

List of Tables

Acknowledgements

I would like to express my deepest gratitude to my PhD supervisors, Dr. Suzanne Lesecq, senior scientist at CEA-LETI and Dr. Xuan-Tu Tran, professor, Head of the Department of Science, Technology and International Relations, Deputy Head of the UET-Key Laboratory for Smart Integrated Systems, at Vietnam National University, Hanoi for their excellent guidance, caring, patiently correcting my writing and helping me to improve my French.

I am also grateful to my adviser, Mrs. Edith Beigne, senior scientist at CEA-LETI, for sharing expertise, and sincere and valuable guidance and encouragement to me.

I would like to thank Prof. Nicolas Marchand, Head of Control Systems Department, Deputy director of GIPSA-lab, Directeur de Recherche CNRS, for doing me great honour by presiding the jury.

I would like to thank Prof. Alain Merigot, professor at University Paris XI and Dr. Nadine Azemard, professor at LIRMM, for accepting to be reporters.

I would also like to thank Prof. Huu-Duc Nguyen, Deputy Director of the Vietnam National University, Hanoi and Prof. Amara Amara, Deputy Managing Director Research and International Cooperation at ISEP, for their participation in the jury.

This PhD work was conducted in the collaboration between the Vietnam National University, Hanoi and CEA-LETI. I would like to thank both organizations, many thank to Mr. Hervé Fanet, for the supports that permit me to do this thesis.

I take this opportunity to express gratitude to Mr. Fabien Clermidy and Mr. Vincent Olive for their welcome and for providing me excellent atmosphere for doing research.

Special thanks goes to my colleagues, Van-Huan Tran, Duy-Hieu Bui, Nam-

Khanh Dang, Pascal Vivet, David Coriat, Sebatien Thuries, Warody Lombardy and Mikhailo Zarudniev, who shared to me their advices and supports during my PhD work. I also place on record, my sense of gratitude to all the dear colleagues, students in the three laboratories, the UET-Key Laboratory for Smart Integrated Systems (SIS), at Vietnam National University, Hanoi, the Laboratory of Silicon Integration and Digital Architecture (LISAN) and the Laboratory of Infrastructure and Software Tools for Chip (LIALP) at CEA-LETI, France, for warm and friendly working environment.

I would never have been able to finish my PhD thesis without the help from friends, and support from my family. Many thanks go to my friends who always support me and encourage me with their best wishes. I also thank my parents and my elder sister for the unceasing encouragement, support and attention. I would like to thank my husband. He is always there cheering me up and stood by me through the good times and bad. Finally, a special thank to my little son, Trung-Kien, Tony Vu, for coming into my life in September 2014.

Grenoble, June 2015.

Bonne lecture !

List of publications

This is the list of my publications during my thesis:

N.-M. Nguyen, E. Beigne, D.-H. Bui, N.-K. Dang, S. Lesecq, P. Vivet and X.-T. Tran. An Overview of H.264 Hardware Encoder Architectures including Low-Power Features. In *REV Journal on Electronics and Communications (JEC)*, pp. 8-17, Vol. 4, No. 1-2, January - June, 2014, ISSN: 1859 – 387X.

N.-M. Nguyen, E. Beigne, S. Lesecq, D.-H. Bui, N.-K. Dang and X.-T. Tran. H.264/AVC Hardware Encoders and Low-Power Features. In *Proceeding of the IEEE Asia Pacific Conference on Circuits And Systems*, APCCAS 2014, pp. 77-80, Okinawa, Japan, November 2014.

N.-M. Nguyen, W. Lombardi, E. Beigne, S. Lesecq, and X.-T. Tran. FIFO-level-based Power Management and its Application to a H.264 Encoder. In *Proceeding of the IEEE Industrial Electronics Conference* , IECON14, pp. 158-163, Dallas, TX, USA, October 2014.(**Best Presentation Award**)

N.-M. Nguyen, E. Beigne, S. Lesecq, P. Vivet, D.-H. Bui, and X.-T. Tran. Hardware implementation for entropy coding and byte stream packing engine in H.264/AVC. In *Proceeding of the International Conference on Advanced Technologies for Communications*, ATC 2013, pp. 360-365, Ho Chi Minh city, Vietnam, October 2013.

N.-M. Nguyen, X.-T. Tran, P. Vivet, and S. Lesecq. An efficient Context Adaptive Variable Length coding architecture for H.264/AVC video encoders. In *Proceeding of the International Conference on Advanced Technologies for Communications*, ATC 2012, pp. 158-164, Hanoi, Vietnam, October 2012. (**Best Student**

Paper Award)

Introduction

Context of video encoding

Video codec (compression-decompression) chips have been recently used in various applications, for instance video conferencing, security and monitoring systems, entertaining applications. To meet the need of mobile applications, video codec is preferred to be implemented in hardware than in software, due to better power efficiency and real-time capacities. One of the latest and most efficient standards for video applications is the H.264 Advanced Video Coding (H.264/AVC) [H.264] which provides better video quality at a lower bit-rate than the previous ones [Joch02P]. It has been specified in the co-operation between the ITU-T Video Coding Experts Group and the ISO/IEC Moving Picture Experts Group. The H.264/AVC specification adopts many advanced methods and algorithms for video compressing and also improves them. This makes the standard more efficient but requires more complex hardware implementation and consumes a huge amount of power.

Recently, the H.264 successor, namely the H.265/High Effciency Video Coding (HEVC) has been released. The H.265/HEVC recommendation [H.265] has been formally published in 2013, two years after this PhD thesis was started. It was jointly developed by the ITU-T and ISO/IEC organizations. It is promisingly bandwidth saving with 40-52% of bit rate reduction in comparison to the H.264/AVC [Rodr13P]. This enables Ultra High Definition Television (UHTV). However, this new standard also exhibits high computing complexity, much more intensive than the H.264 standard, which leads to shortened battery lifespan and higher power consumption. With these higher costs, the switch to the new standard has to be carefully thought and the H.264/AVC video codecs are still under use.

The "Video Encoder for the Next Generation Multimedia Equipments" (VENGME) project, a research project granted by Vietnam National University, Hanoi (VNU), aims at designing and implementing a hardware H.264/AVC encoder targeting mobile platforms. The current design is optimized for CIF video. However, the VENGME architecture can be extended for larger resolutions by enlarging the reference memory and the search window.

To bring the H.264/AVC standard into commercial products, especially for hand-held devices, designers need to apply low-power techniques when designing the video codec. Targeting mobile applications, the VENGME platform was designed using low-power design features. Some blocks in the platform present nice power figures and performances, similar to state-of-the-art designs. However, the whole platform power results are not outstanding in comparison to state-of-the-art hardware video encoders. To improve the platform power results, its imbalance workload can be exploited via different power control techniques. This includes applying Dynamic Power Management (DPM) methods, for instance, Dynamic Voltage and Frequency Scaling (DVFS) approaches [Yada12A].

Motivations and objectives

Many research efforts have been made to produce electronic circuits and systems that can perform more tasks with limited resources while consuming less energy. Technology evolution has increased significantly the operating speed and integration density. Meantime, energy and power consumption have become a tremendous problem because of their effects on the system reliability, cooling cost, and battery lifetime, especially for mobile devices, along with environmental considerations. Therefore, power consumption reduction is strongly required when designing complex system-on-chips.

Static and dynamic techniques have been proposed in order to decrease the power consumption. Static techniques are applied at design time while dynamic ones adapt the circuit functioning during the operating time [Lu01C].

For System-on-Chips, power consumption is influenced by four main factors, namely the supply voltage, the switching activity in the circuit, the switching capacity load, and the clock frequency. Techniques to reduce the power consumption are based on the fact that the chip never works at its maximum power capacity and that there are always some idle or

lower speed operating parts of the chip during the operating time. There are numerous existing low-power optimization mechanisms ranging from circuit level (power gating [Chio06T], clock gating [Khan99A], etc.) up to system (DVFS [Sylv07A, Ma10E, Vive10O]) and software level (idle mode programming, automatic wake up, etc.) [Deva95A]. Recent study, [Choi05F, Qu07P, Sula08U, Herb12E, Yada12D, Stan13E, Zhur13S] provide new solutions and improvements for these techniques.

The objective of this PhD thesis is firstly to design the EC-NAL module of the VENGME video encoder. Secondly, the VENGME architecture is analyzed to extract the power properties. Thirdly, we propose new mechanisms to control power consumption for a fully hardware H.264 encoder system-on-chip, the VENGME platform. Then, a low-level hardware control solution is proposed to reduce the power consumption of the system. Finally, hardware implementation and validation are realized to prove nice features and performances of the hardware control solution and evaluate the proposed mechanism.

Contributions of the work

To answer the objectives, this thesis proposes the following three contributions for both module and platform levels:

- **Design of the entropy coder and data packer (EC-NAL) module**

  This contribution requires a full understanding of the H.264 encoder techniques. The H.264 implementations are reviewed and the available implementations on hardware are classified and analyzed.

  The entropy coder and data packer (EC-NAL) is the final module in the encoding path of a video encoder. Its role is to eliminate the static redundancy in the video data. It receives data from most of the modules in the platform (all modules in the encoding path) and compresses this data using some entropy coding techniques. There are three main entropy coding techniques possible for H.264, namely, Exp-Golomb, CAVLC and CABAC [H.264]. In our entropy coder, the techniques applied are Exp-Golomb and CAVLC, equivalent to the Baseline or the Main profile of the H.264 standard.

To understand the H.264 encoding process and the operation of the VENGME H.264 video encoder, we have proposed, designed and implemented the EC-NAL module in the VENGME H.264 video coding platform. The module has been integrated and validated into the complete platform.

This contribution has been published in:

- "An efficient Context Adaptive Variable Length coding architecture for H.264/AVC video encoders" [Nguy12A] presents the CAVLC encoder, one of the main engines of the Entropy Coder (EC);
- "Hardware implementation for entropy coding and byte stream packing engine in H.264/AVC" [Nguy13A] presents the design and implementation of the whole VENGME EC-NAL module.

- **Power estimation at RTL level for the VENGME H.264 encoder platform**

  An early power estimation is performed on the full VENGME platform to obtain a first evaluation of the power composition and the needs for applying power management methods (e.g. is there any chance to decrease the power consumption).

  This power estimation result are presented in the publication: "H.264/AVC Hardware Encoders and Low-Power Features" [Nguy14H].

- **Propose a method to the reduce the power consumption based on the FIFO occupancy level**

  Methods to manage and optimize power consumption aim at changing some of the four factor influencing the power consumption of digital CMOS circuits, given a means to monitor the workload status of the circuit. Some techniques based on controlling the status of a memory buffer were studied and analyzed.

  The method proposed in our work is based on automatics control theory. The control decision is made according to the FIFO occupancy level in the communication link between two calculating submodules. The method was modeled and verified using Matlab. The controlled system is also proved to be stable.

The proposed method is modeled, implemented, validated and applied onto the FIFO embeded in the EC-NAL module of the VENGME platform.

The controller was then modeled in VHDL, implemented in hardware. Its operation is verified via the integration into the EC-NAL module of the VENGME platform. The method was also validated with power estimation before and after applying it.

This work is presented in the publication "FIFO-levelbased Power Management and its Application to a H.264 Encoder" [Nguy14F].

Explanation for the different CMOS technologies and operating frequencies in the results presented in the thesis

This thesis work is conducted in three laboratories, namely the Smart Integrated Systems (SIS) laboratory in Vietnam National University, Hanoi (VNU), Vietnam, the Laboratory of Silicon Integration and Digital Architecture (LISAN) and the Laboratory of Infrastructure and Software Tools for Chip (LIALP) in CEA-LETI, France. The VENGME platform architecture is designed in the SIS laboratory. Low-power techniques, e.g. DFS techniques, are acquired from the LISAN laboratory. The automatic control is studied with the LIALP laboratory.

As a consequence, the author had the opportunity to work with large-node technologies and with more advanced ones. Table 0.2 shows the different CMOS technologies and operating frequencies used during the thesis. As can be seen, many technology changes have been done. These changes are imposed by the tools available in the laboratories at the time the developments have been done.

In the beginning phase of the thesis, the Entropy Coder and Network Abstraction Layer data packer (EC-NAL) module was designed. The CAVLC encoder, an important engine in the Entropy Coder (EC) submodule, was designed firstly, in 2012, in Vietnam. The engine was then synthesized in CEA-LETI, France, on the $65nm$ CMOS technology from STM. At the moment, to explore the operating capability of the design, the CAVLC encoder was synthesized with a wide range of frequencies, from $400MHz$ to $715MHz$.

Table 0.2: Different CMOS technologies used during the thesis

Year	**Design**	**Task**	**Tool**	**Place**	**Technology**	**Frequency (*MHz*)**
2012	CAVLC encoder	synthesis	Design Compiler from Synopsys	CEA-LETI France	CMOS 65nm from STM	400-715
2013	EC-NAL module	synthesis	Design Compiler from Synopsys	VNU Vietnam	CMOS 180nm from AMS	100
2013	VENGME platform	Power simulation @ RTL level	SpyGlass from Atrenta	CEA-LETI France	CMOS 32nm from STM	50
2013	EC-NAL module	Power simulation @ RTL level	SpyGlass from Atrenta	CEA-LETI France	CMOS 32nm from STM	50
2014	VENGME platform	synthesis	Design Compiler from Synopsys	VNU Vietnam	CMOS 130nm from Global Foundary	100
2014	EC-NAL module	Synthesis, Power estimation @ gate level	RC from Cadence PrimeTime from Synopsys	CEA-LETI France	FDSOI 28nm from STM	50 & 100

Then, the EC-NAL module was designed in France and was sent to VNU, Vietnam, at RTL level, to be integrated in the platform. Thus, the module synthesis was performed, in Vietnam, in 2013, using the technology at hand, which is the 180nm CMOS technology from AMS. The operating frequency was decided to be 100MHz according to the VENGME project specification.

Simultaneously, the VENGME platform, at RTL level, was sent to France, for the low-power strategy sake. In this platform, the defined operating frequency was 50MHz. During this phase of the thesis, the power simulation at RTL level using SpyGlass from Atrenta was performed in CEA-LETI, with the 32nm CMOS technology from STM. The power simulation results at platform level allow to analyze the power distribution over the various modules of the platform and to seek for extra power optimization.

The FIFO/buffer-level-based DFS method was, then, proposed and the EC-NAL module was selected for the method to be firstly implemented in. To enable DFS, the FIFO within the EC-NAL module was modified from a synchronous FIFO to an asynchronous one. Power simulations, for the EC-NAL module, at RTL level using SpyGlass were performed with the

same technology and operating frequency. The power results shows that the modification of the FIFO in the EC-NAL module increases the module power consumption. However, this increase seems to be small enough in comparison to the expected power gain from applying the DFS method.

In 2014, the whole VENGME platform was synthesized on the $130nm$ CMOS technology from Global Foundary, with the operating frequency of $100MHz$. All the synthesis and power simulation at RTL level results will be presented in Chapter 2.

The proposed DFS method proposed was modeled in VHDL and implemented in hardware. To evaluate the power gain, synthesis and gate level power estimation were performed in CEA-LETI, on the $28nm$ FDSOI technology from STM. The main operating frequency is $50MHz$ and for the DFS part, the maximum frequency is $100MHz$. The power estimation at gate level and DFS method evaluation will be presented in Chapter 4.

This explains the different CMOS technologies and frequencies that have been used during this PhD thesis.

Organization of the manuscript

Apart from this introduction, the document contains four chapters and a conclusion. The first chapter introduces the H.264 video coding standard in relation with other video coding standards in use. Some basic concepts of video coding in general and of H.264 video coding in particular are also presented. Moreover, at the end of the chapter, a state-of-art of H.264 encoding hardware implementations is provided and analyzed. This contents firstly shows some issues in implementing the H.264 video coding standards and lastly review the available hardware implementations of H.264 video encoder.

The second chapter introduces the VENGME platform. This is a hardware of H.264/AVC video encoder targeting mobile applications. The VENGME platform has been designed by the research group in Vietnam in the framework of the VENGME project. The architecture, modeling and validation of the EC-NAL module, one of the main contributions, is also presented in this chapter. The results of the power simulations at RTL level for the whole VENGME platform and for the EC-NAL module help in seeking a power optimization solution for the VENGME platform.

In the third chapter, FIFO-based power management and optimization methods are reviewed and analyzed. Then, a FIFO-level-based method to control the power consumption is proposed and applied to the FIFO embedded in the EC-NAL module. All the steps from system modeling, control design, stability analysis and MATLAB simulation results are in turn presented.

The fourth chapter presents the hardware implementation of the FIFO-level-based method proposed in the previous chapter. Simulation and power estimation results enable verifying the control operation and validating the method.

Finally, the conclusion summarizes the works performed during this thesis and it also draws some future works perspectives.

Chapter 1

The H.264/AVC video coding standard and its implementations

Thanks to technology evolution, video application, even for basic applications such as television, has become one of the most used technology applications nowadays. It is more and more popular in daylife applications, for instance, in television broadcast over cable or satellite; in many storage medias as DVD, Blu-ray disc, in video on internet, video chat, video telephone, video conferencing, etc. High video quality, large video size are increasingly required for these applications. Moreover, storage space and bandwidth are limited and expensive. Thus, video compression technology that reduces the video data size is obviously essential. To allow international video exchange and make video data formats world-wide-accepted leading to video codecs (compressing and decompressing) cheaper, international standardization is mandatory.

The objectives of this chapter are to firstly review video coding formats and standards from the past to present, and their application range. Then, the basic concepts of video coding techniques and the H.264 Advanced Video Coding (H.264/AVC) standard will be presented. Lastly, the main difficulties and trends in implementing the H.264/AVC will be discussed with an overview and analysis of different implementations.

1.1 H.264 video coding standard in the history of video coding standards

This section provides a brief review of video coding standards and their applications. The details on coding techniques are not in the scope of the present document. The reader interested in such details can consult [Scha95D, Joch02P, Jaco07A]. Table 1.1 presents a chronology of some popular video coding formats and standards. As can be seen, the first standard considered, H.120, has been standardized in the 1980s while the latest evolution of the H.264, named H.265, has been published in 2013, after the beginning of the present work.

1.1.1 The video coding standards before the H.264/AVC

The first digital video coding standard is the H.120 recommendation [H.120], standardized by the ITU-T[1] in 1984. Its target was to transmit video over dedicated point-to-point (P2P) data communication lines, NTSC (National Television System Committee) at $1544kbps$, or PAL (Phase Alternating Line) at $2048kbps$ [Jaco07A]. The H.120 was superseded [itu] and it is essentially not in use today.

In 1988, the H.261 recommendation [H.261] was approved by the ITU-T Video Coding Experts Group (VCEG). It was the first successful practical digital video coding standard, operating at $64 - 2048kbps$ [Sull05O]. The recommendation H.261 designed the coding algorithm to support the transmission of colour video over Integrated Services for Digital Network (ISDN) at $p \times 64kbps$ with low delay, where p is between 1 and 30. The targeting resolutions were Common Intermediate Format (CIF) 352×288 pixels and Quarter CIF (QCIF) 176×144 pixels [Scha95D].

The International Electrotechnical Commission of the International Organization for Standardization [iso] (ISO/IEC) [iec] also developed a video coding standard for the coding of moving pictures and associated audio at up to about $1.5Mbps$ (the MPEG-1) [mpeg-1]. The MPEG-1 was in use more frequently than the H.261, especially in Video CD (VCD) in Asia [Sull05O]. However, the lack of interlaced features [Jaco07A] has limited its applications.

[1] The Telecommunication Standardization Sector of the International Telecommunication Union

Table 1.1: Chronology of video coding formats and standards

Year	Video coding formats/standards	Developed/ Standardized by	Application range
1984	**H.120**	CCITT (ITU-T now)	Video transmission (NTSC or PAL) over P2P data communication lines
1988	**H.261**	CCITT (ITU-T now)	Video transmission over ISDN lines
1993	**MPEG-1**	ISO/IEC	Digital satellite/cable TV; Video CD; ...
1994	**H.262/MPEG-2**	(joint) ITU-T & ISO/IEC	DVD players; camcorders; video recorders; distribution networks
1995	**H.263**	ITU-T	Video conferencing; cellphone codec; World Wide Web-distributed video
1998	**MPEG-4 Visual**	ISO/IEC	DVD; mobile multimedia; internet multimedia; broadcast TV; videophone; ... [MP4Apps]
2000	**VP3**	On2 Tech.	Released into the open source community in 2001 by On2 Tech.
2001	**VP4**	On2 Tech.	
2002	**VP5**	On2 Tech.	
2003	**H.264/MPEG-4 AVC**	(joint) ITU-T & ISO/IEC	HD video like Blu-ray Disc; HDTV broadcast over terrestrial, cable and satellite; video streaming (Vimeo, YouTube, iTunes Store, ...); ...
2003	**VP6**	On2 Tech.	Adobe's Flash video; YouTube video; video conference over IP; satellite broadcasts
2005	**VP7**	On2 Tech.	Adobe's Flash video; video streaming; ...
2005	**AVS**	The government of the People's Republic of China	within mainland China
2006	**SMPTE 421M/VC-1**	Microsoft - SMPTE	Blu-ray Disc; Window media; HD DVD; ...
2008	**VP8**	On2 Tech. - Google	Adobe's Flash video; YouTube video; ... (open source by Google)
2010	**Dirac/VC-2**	BBC Research - SMPTE	Ultra HDTV; used by BBC to transmit HDTV
2013	**VP9**	Google	HTML5 video; some smart TVs; YouTube for 4K resolution; ...
2013	**H.265/HEVC**	(joint) ITU-T & ISO/IEC	Ultra HDTV; used by ATSC, DVB & Blu-ray disc

The H.262/MPEG2 video coding standard [H.262] was developed jointly by the ITU-T VCEG and the ISO/IEC Moving Picture Experts Group (MPEG) in 1994 as the successor of the MPEG-1 (in 1993). It is compatible with the MPEG-1 standard and widely used. It was designed for a bit rate from 3 to 10$Mbps$ and supports both standard-definition (SD, $720\times480/576$ pixels) and high-definition (HD, 1920×1080 pixels) [Jaco07A].

In 1995, the ITU-T provided the H.263 video coding standard [H.263]. The H.263 successfully dominated video conferencing and cell phone video applications. Then, different versions (H.263+, H.263++, ...) and several annexes were published.

The ISO/IEC MPEG has developed the MPEG-4 Visual video compression format, the second part of the MPEG-4 standard [mpeg-4]. The standard was finalized in 1998. It is applied in several means of video storage and video transmission, such as mobile multimedia, Digital Video Disc (DVD), video streaming on internet, broadcasting, digital television (TV) box,... [MP4Apps]. During its enhancement, many functionalities have been added. For instance, interactive graphics, object and shape coding, face modeling, scalable coding, 3D graphics,... are the novel coding concepts added to the MPEG-4 standard. However, few of these techniques have been successfully used in commercial applications due to their complexities [Jaco07A].

1.1.2 The H.264 Advanced Video Coding standard

The H.264/AVC video coding standard [H.264], the successor of the H.263, is one of the most widely used video compression standard. It was recommended by the Joint Video Team (JVT), that was formed by the ITU-T VCEG and the ISO/IEC MPEG groups in 2001. Using block-based coding, it was a step "back to basics" in comparison to MPEG-4 Visual coding techniques [Jaco07A]. Its primary technical objectives are to improve significantly the coding efficiency; to have high loss/error robustness; to exhibit low latency capability and to enable "network friendliness" i.e. flexibility for user over various systems and networks [Wieg03O]. The first recommendation was finalized in 2003.

The H.264/AVC contains a wide set of video coding tools to support a variety of applications ranging from mobile services, video conferencing, digital broadcast for IPTV, HDTV and digital storage media. Compared

to previous standards such as MPEG-4, H.263, MPEG-2, the H.264/AVC standard achieves respectively 39%, 49%, and 64% of bit-rate reduction [Joch02P]. The H.264/AVC standard will be discussed in more details in Section 1.2.

1.1.3 Video coding standards from other organizations, H.265/HEVC standard and the consideration of standard selection

After the H.264/AVC standard was introduced, video coding standards have not been given by a unique standardization group, e.g. ITU-T or ISO/IEC. Several companies and/or organizations have developed their own video compression formats [Jaco07A]. For instance:

- the Audio Video coding Standard (AVS) was developed in 2005 by the government of the People's Republic of China to avoid paying royalties for the patents on international standards. Some semiconductor companies support the standard, but it is mainly used within mainland China [Jaco07A];
- the TrueMotion video codec series, known as VP3 to VP7, were developed by the On2 Technologies company [Jaco07A]. In 2001, On2 has released the VP3 codec implementation as open-source. The VP6 was applied in the Adobe Flash video codec. With the acquisition of On2 Technologies in 2010, the Google Inc. [Goog] has obtained the rights on the VP8 video codec. It has been announced as a royalty-free alternative to the H.264/AVC video codec. Several tests have been performed to compare both codecs [Fell11T]. They have shown that, up to a compression ratio of six, the H.264/AVC (the reference software implementations *x264* and *JM*) shows better quality than the VP8 codec by means of Peak Signal-To-Noise ratio (PSNR). Then, at a higher compression ratio, the quality of both codecs is comparable. Moreover, the *x264* codec encoding speed is 350% higher than the VP8 one [Fell11T];
- the Windows Media Video video codec version 9 was developed by Microsoft [Micr]. This video format was then standardized by the Society of Motion Picture and Television Engineers (SMPTE) [smpte]

as the SMPTE 421M (or VC-1) video coding standard. The standard does not provide cost-saving by avoiding the use of patented technologies from third parties. Moreover, it compresses less well than the H.264 [Jaco07A]. The Dirac video compression format was developed by the BBC Research and Developement (BBC R&D) [bbcrd]. Its I-frame-only subset, Dirac Pro, was standardized by the SMPTE and known as VC-2.

Recently, the H.264 successor, H.265/High Efficiency Video Coding (HEVC), and its royalty-free alternative VP9 by Google, have been released. The H.265/HEVC recommendation [H.265] has been formally published in 2013. It was jointly developed by the ITU-T and ISO/IEC organizations. It is promisingly bandwidth saving with 40-52% of bit rate reduction in comparison to the H.264/AVC [Rodr13P]. This enables Ultra High Definition Television (UHTV). However, this new standard also requires computing complexity much more intensive than the H.264 standard, which leads to shortened battery lifespan and higher power consumption. **With these higher costs, the switch to the new standard has to be carefully thought.**

1.2 Overview of H.264 video coding standard

This section provides readers with some basic concepts of video coding techniques in general. It also summarizes advanced techniques used in the H.264 video coding standard.

1.2.1 Video coding basic concepts

Video coding techniques are applied during the *encoding process* to compress an original video data to a lower bit rate one. The reduced data is then stored or transmitted via a communication channel. To display the video, the *decoding process* is performed to decompress the video data into the original format, see Figure 1.1. As in other data compression techniques, lossy video coding techniques can be used: the decoded and the original videos are different. For lossless video coding techniques, the decoding process produces the same video as the original one (i.e. without loss).

A digital video signal is the result of sampling the natural visual scene [Rich03H]. Thus the video signal contains a sequence of *video frames*, i.e.

image samples at points in time. Temporal sampling is usually performed at the intervals of 1/25 or 1/30 second. A frame might be composed of two fields containing odd- and even- numbered lines of spatial sampling. *Progressive sampling* samples the complete frame at once. *Interlaced sampling* gives the interlaced fields which are sampled at two different temporal points. Each spatial sample, i.e. a *pixel*, is a square picture element where the image information is supposed identical. The size of the video frame in pixels (*width dimension* $\times$ *height dimension*) is called the *display resolution*. During the *encoding process*, a frame may be divided into blocks of $m \times n$ pixels, or into slices (i.e. specific groups of pixels).

If the video picture is monochrome, it requires only the brightness to represent image information. A colour video contains both luminance (luma), i.e. brightness, and colour information. In RGB colour space, the image is represented with three additive primary colours of light (Red, Green, and Blue). A given colour is created by a particular combination of Red, Green and Blue. Because the human visual system is less sensitive to colour than to luminance, if we separate the colour information into luma and colour, we can store the colour information with a lower resolution than the one of the luma information. The YCbCr (or YUV) colour space is one of the ways to represent a colour image in luma and colour separately. Y is the luma component which is calculated as:

$$Y = k_r R + k_g G + k_b B \tag{1.1}$$

where k_r, k_g, k_b are the weighting factors. The colour information is represented with colour differences, named chrominance (chroma):

$$\begin{aligned} Cr &= R - Y \\ Cg &= G - Y \\ Cb &= B - Y \end{aligned} \tag{1.2}$$

Because $Cr + Cg + Cb$ is a constant, it requires only two chroma components to represent the colour information. Actually, the Red and Blue differences are more significant than the Green one. Therefore, in the YCbCr colour space, the luma Y and red and blue chroma Cr, Cb are used. The sampling format 4:4:4 means that three components, Y, Cb and Cr, have the same resolution. In the sampling format 4:2:2 (YUV2), Cb and Cr have the same vertical resolution as Y, but half its horizontal resolution. The sampling format 4:2:0 (YV12) halves both the vertical and horizontal resolutions of the chroma components in comparison to the luma resolution.

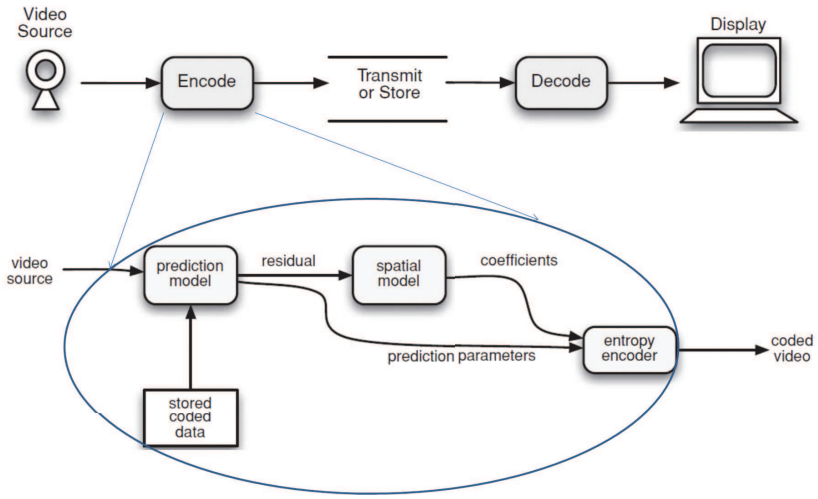

Figure 1.1: Fundamentals of video coding, modified from [Rich10T].

The video encoding process is a dataflow process. Figure 1.1 describes the main phases in the video encoding process, namely the prediction model, the spatial model and the entropy encoder. The prediction model can be intra or/and inter prediction from the stored coded data that generates a residual frame. The spatial model can be a transformation, scaling and/or quantization that produces (quantized) transformed coefficients. These video coding techniques are now summarized.

The video coding techniques are based on the four dimensions of the digital video signals [Watk04T]:

- the attributes of the pixel;
- the two spatial dimensions (horizontal and vertical);
- the time.

Rounding the binary representation of luma or chroma samples can reduce data according to the first dimension, i.e. the attributes of the pixel.

In the spatial dimensions, since there is always high correlation between neighbouring pixels, we can remove the spatial redundancy within a frame. This is achieved by spatial prediction techniques, creating a predicted version of the current block from the previously coded samples in the same frame (*intra prediction*). Similarly, to remove temporal redundancy, the *inter prediction*, based on high correlation between adjacent frames, generates the predicted version of the current block by using the previously

coded frames (called *reference frames*). The inter predicted version of one block can be the highest correlated block (with the current one) found in the reference frames (P inter prediction). It can also be interpolated from the highest correlated block(s) (with the current one) found in the reference frames (B inter prediction). The process of searching for the "best matching" block(s) in the reference frames with the current frame is called *motion estimation*. *Motion compensation* then substracts the predicted block and the current one to create *residual data*. Only the residual, i.e. the difference between the predicted and the real versions of the current block, is then used in the next steps of the encoding process. At the decoding side, the samples are reproduced by adding the received residual to the previously decoded samples.

Based on the fact that the human visual system is less sensitive to high frequency image signals than to low frequency ones, we can also remove the subjective redundancy parts. Transformation and quantization processes are used for this purpose. The transformation converts the image or residual data into another domain. Several transforms can be used, e.g. a Discrete Cosine Transform (DCT), a Discrete Wavelet Transform (DWT or Wavelet), etc. The selected transform should be decorrelated and compact, i.e. most of the energy of the transformed data should be concentrated into a small number of values, reversible, and easily applied. The quantization process reduces the range of possible values for the data. It may be used to remove the less important values in the transformed data. The quantization reduces the precision of the image (or video) data. Hence, this coding technique is lossy. Therefore, the reference samples used in the prediction must be the reconstructed samples from a decoding path inside the encoder instead of using directly the samples in the original video.

The final phase in the video encoding process is the entropy coding, used to losslessly remove the stastical redundancy in data. The entropy coding represents a series of data elements into a bitstream. It can be Variable Length Coding (VLC) or Arithmetic Coding.

1.2.2 H.264/AVC basic concepts

The general architecture of the H.264/AVC encoders is depicted in Figure 1.2. In order to achieve high compression ratio while keeping the encoded video quality, the H.264/AVC standard has adopted several advances in

coding technology to remove spatial and temporal redundancies. These prominent techniques are summarized hereafter:

- a new way to handle the quantized transform coefficients has been proposed for trading-off between compression performance and video quality to meet the applications requirements. For example, an efficient method, i.e. Context-Adaptive Variable Length Coding (CAVLC) is used to encode residual data. In this coding technique, Variable Length Coding (VLC) tables are switched according to already transmitted syntax elements. Since these VLC tables are specifically designed to match the corresponding image statistics, the entropy coding performance is impressively improved in comparison to schemes that use only a single VLC table [Rich10T];

- the H.264/AVC adopts motion prediction with variable block size to provide more flexibility. The intra prediction can be applied either on 4×4 blocks individually or on entire 16×16 macroblocks (MBs). Nine different prediction modes exist for a 4×4 block while four modes are defined for a 16×16 block. After comparing among the cost functions of all possible modes, the mode having the lowest cost is selected. The interprediction is based on a tree-structure where the motion vector and prediction can adopt various block sizes and partitions ranging from 16×16 MBs to 4×4 blocks. To identify these prediction modes, motion vectors and partitions, the H.264/AVC specifies a complex algorithm to derive them from their neighbors [H.264]. Motion compensation is improved from the previous half-sample-accurate to quater-sample-accurate motion compensation. Weight prediction weights the predicted signal and its offset by amounts specified by the encoder to improve the coding efficiency in specific situations, for instance fading scenes [Rich10T];

- the forward transform and inverse transform also operate on blocks of 4×4 pixels to match the smallest block size. The transform is a Discrete Cosine Transform (DCT) with some fundamental differences compared to those in previous standards. It requires only 16-bit arithmetic for the transform computation instead of 32-bit as in previous standards. It is also the first standard that achieves the same quality of decoded video content from different decoders [Wieg03O];

- the in-loop deblocking filter in the H.264/AVC depends on the so-called Boundary Strength parameters (BS) to determine whether the current block edge should be filtered. The derivation of the BS is highly adaptive because it relies on the modes and coding conditions of the adjacent blocks. An adaptive deblocking filter can improve both the objective and subjective video qualities.

For robustness to data errors and losses, several new aspects are also defined for the H.264/AVC standard:

- the H.264/AVC parameter-set structure contains all significant header information. This information is separated flexibly to reduce the risk of losing a few bits within this information;

- the Network Abstraction Layer Unit (NALU) syntax structure allows the customization of packing video data to match different specific networks ("network friendliness" features);

- many other flexible features were added to the H.264/AVC to the enhance robustness to data errors [Wieg03O], for instance, the flexible slice size, the flexible macroblock ordering, data partitioning, etc.

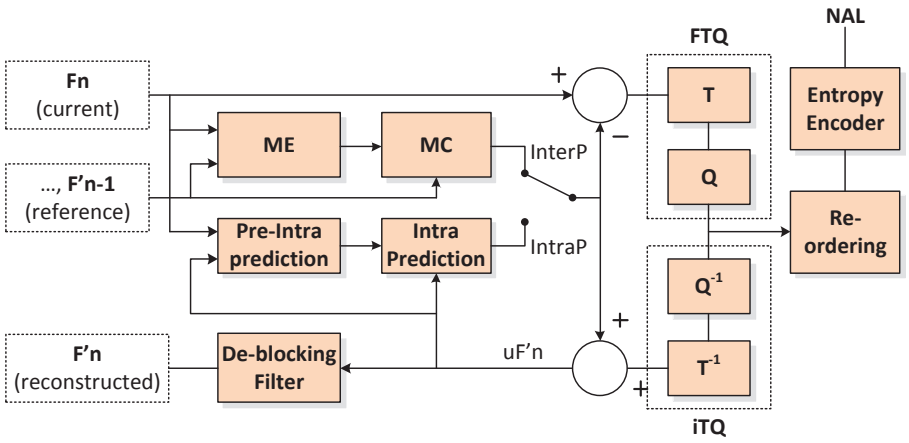

Figure 1.2: Functional diagram of the H.264/AVC encoder [Nguy14A].

The general architecture of the H.264/AVC encoder is built of different functional blocks.

The encoding path consists of the Intra prediction block, the Inter prediction containing the Motion Estimation (ME) and the Motion Compensation (MC) blocks, the Forward Transform and Quantization (FTQ), the Re-ordering, and the Entropy encoder. The Intra prediction block predicts the current macroblock (MB, a block of 16×16 pixels) based on the previously encoded pixels in the current frame, to remove spatial redundancies of the video data. In order to remove temporal redundancies of video data, the inter prediction block estimates the motions of the current MB based on the previously encoded pixels in different frame(s). The residual data, i.e. the differences between the original current MB and the predicted one, is then transformed and quantized. The post-quantization coefficients are then re-ordered and entropically encoded to lastly remove statistical redundancies. The encoded video might be encapsulated into Network Abstraction Layer (NAL) units. A decoding path that contains the Inverse Transform and de-Quantization (ITQ) and the De-blocking filter is also built in the video encoder to generate reference data for the prediction processes. The intra prediction directly uses the data from the ITQ, while the inter prediction refers to reconstructed frames from the de-blocking filter.

Each standard defines different *profile*s as sub-sets of coding tools or algorithms that can be used. *Level*s that are the constraints on key parameters of the bitstream are also defined. At its early stage, the H.264/AVC defined three profiles, namely, the Baseline, Main and Extended ones [Wieg03O].

1.3 H.264/AVC implementation solutions

This section presents typical solutions to implement the H.264/AVC specification. The first sub-section will review some full software implementations of the H.264/AVC standard. The two last sub-sections will present challenges in the implementation of the H.264/AVC standard and some trends in its hardware implementation.

1.3.1 Software implementations

The Joint Model (JM) is a C software program developed by the JVT as a reference implementation of the H.264/AVC standard. It is available on the Internet and the current version is JM 18.6 [JM]. The JM program contains an encoder and a decoder that can encode video files (raw YCbCr

format) into the coded H.264 files and vice versa. It applies all the coding techniques described in the specification. The coding parameters can be fixed by the user. It also enables the user to trace the syntax elements during the encoding and decoding processes. The JM codec runs very slowly. It is usually used for testing and research purposes instead of practical real-time applications.

The x264 is another free software library and application to encode videos into H.264/AVC format. It has been developed by the x264 team [x264]. It is released under the term of the GNU General Public License (GNU GPL). Different from JM, x264 is used in many practical coding applications such as television broadcast, Blu-ray low-latency video applications and web video. It performs well in terms of bitrate, video quality and processing time. For example, it encodes at least four 1080p streams in real-time on a single computer [x264]. It contains a large number of coding features defined in the H.264/AVC standard. Psychovisual optimizations are also applied to enhance the subjective video quality.

Many other software codecs containing the H.264/AVC have been developed by different companies. For instance, QuickTime Pro is developed by Apple Inc. [QuTime], the Nero Multimedia Suit is from Nero AG [Nero], etc. Due to proprietary issues, these codecs are not freely downloadable.

1.3.2 Challenges in implementing the H.264/AVC and solutions

The main challenges in implementing the H.264/AVC are the long coding path and the data dependency. As presented in Section 1.2.2, the encoding process consists of a variety of tasks from prediction (intra and/or inter), transformation, quantization to entropy coding. One has also to implement the fetching of the source video data at the entrance of the encoder, and the data packing into syntax structure at the beginning and at the end of the coding process. Moreover, the predictive features lead to coding efficiency but they cause strong data dependencies.

Data dependency exists within a task and between tasks. The data dependency between tasks is obvious: the output of the previous task is the input of the next one because of the natural dataflow structure. The entropy coding task even requires the information from all the prediction, transformation, quantization tasks for processing one macroblock. Regarding data

dependency within a task, many blocks of the encoding process require previously processed data as inputs for the current data. For instance, the intra prediction block needs the egde pixels of the left, left-top, top and right-top neighbouring blocks when processing the current block in 4×4 prediction mode, see Figure 1.3. The inter prediction block even requires the data in the previously coded frames. In the entropy coding block, e.g. the CAVLC block, the previously coded data elements are required to select the "best" VLC table to encode the current data element. The deblocking filter uses also the left and the top or the top, the bottom, the left and the right neighbouring blocks when processing the current block. Note that data dependency restricts parallel computing capacity and requires high memory access.

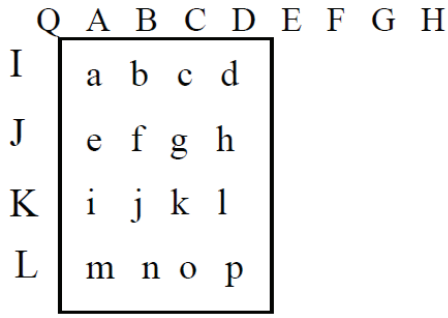

Figure 1.3: Intra prediciton, mode 4 pixels, for pixels $a - p$ of a block using the samples $A - Q$ from the neighbour blocks [Khan08T].

The coding efficiency of the H.264/AVC standard leads to its use in challenging applications, e.g. HD video and/or real-time ones. Note that a HD video source increases the amount of input data for the encoder while real-time applications require high processing speed. These challenging features are driven by emerging demands in video applications nowadays.

Actually, the advanced coding techniques such as many prediction modes, various block sizes, quater-pixel-accurate motion compensation, weight prediction, CAVLC, etc. improve the coding efficiency. However, they exhibit high computational complexity. Along with long coding path, data dependency and real-time HD applications make the implementation of the H.264/AVC very difficult, especially in the context of low power platform.

Even if the data dependency restricts the parallel computing capability, the implementation in parallel of tasks is possible to increase the encoding speed. Long coding path and data dependency lead to pipelining process as

the most appropriate solution. The pipelining solution naturally uses the dataflow character of the video coding process. It maps each video data unit, e.g. macroblock (MB), slide, or frame, to one stage of the video coding process. For instance, Figure 1.4 illustrates an example of a pipelining solution for video encoding. In this pipeline, after one MB, e.g. *MB_0*, is processed in the Integer Motion Estimation (IME) stage, it moves on to be processed in the next stage, i.e. Fractional Motion Estimation (FME), leaving the IME engine to the next MB, *MB_1*. When the FME for the *MB_0* is done, the third stage processes *MB_0*, the FME stage processes *MB_1* and the IME starts processing *MB_2*, and so on. The pipeline solution does not reduce the time to process each MB. However, it increases the system throughput, meaning that more MBs can be processed in a certain time interval. Several H.264/AVC encoders are implemented in parallel DSPs, multi threads processors [Zrid09H, Chan11S], stream processors [Wen11D], multicore systems [Le07A, Xiao11A] and in dedicated Very-Large-Scale Integration (VLSI) hardware architectures [Huan05A1, Chen06AA, Liu07A, Moch07A, Chen08A, Lin08A1, Chan09A, Chen09A, Iwat09A, Kim11P, Zuo12A].

Zrida et al. [Zrid09H] did not implement a H.264/AVC encoder in a real multiprocessor system but they proposed a parallelization methodology for a multiprocessor implementation. They used parallel programming models of computation that work on a general-purpose processor platform. They use the JM 10.2 with its main profile at level 4 as reference software.

Both Le [Le07A] and Xiao [Xiao11A] implemented their H.264/AVC encoders on an Asynchronous Array of Simple Processors (AsAP) platform. The platform provides a new method of programming over a large number of simple processors, see Figure 1.5. The encoder in [Le07A], in baseline profile, was mapped on 167 cores which are highly flexible and parallelizable, see Figure 1.6. It supports both QCIF and CIF video formats. The encoder in [Xiao11A] was mapped on an array of 25 small processors.

To overcome data dependency, Chang *et al.* [Chan11S] implemented concurrent encoding of multiple independent slices, parallel luminance/chrominance residual coding and reconstruction, and motion vector search for multiple reference frames. Wen *et al.* [Wen11D] and Chen *et al.* [Chen06AA] proposed a modified motion vector prediction in the inter prediction block to be less dependent on the previously coded data. The required motion vectors from the left macroblock, which is late available, are replaced by the top-left ones. Then, in the intra prediction block, Wen *et*

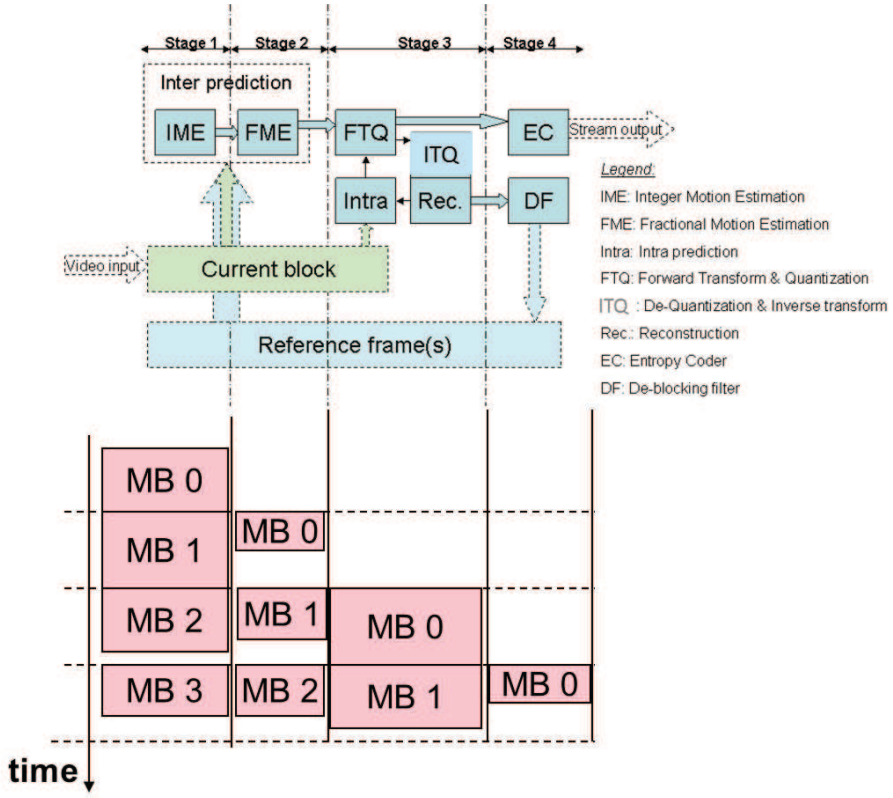

Figure 1.4: An example of pipelining solution for video encoding.

al. [Wen11D] processed all independent blocks in parallel. In CAVLC, they implemented a parallel bitstream generating method to process a row of macroblocks of a frame parallely. In [Iwat09A], a parallel pipelining architecture where two pipelines operate at the same time to encode the odd and even lines of macroblocks in a frame was proposed. Mochizuki *et al.* [Moch07A] and Chang *et al.* [Chan09A] proposed new rate control schemes while Chen *et al.* [Chen09A] and Lin *et al.* [Lin08A1] placed the reconstruction task across the last two piplining stages to solve inter-task data dependency.

1.3.3 Trends in hardware implementation of the H.264/AVC coding

With coding techniques of high computational complexity and long coding path, hardware implementation is better than other solutions in terms of speed, area and power for its flexibility, parallelization and optimization

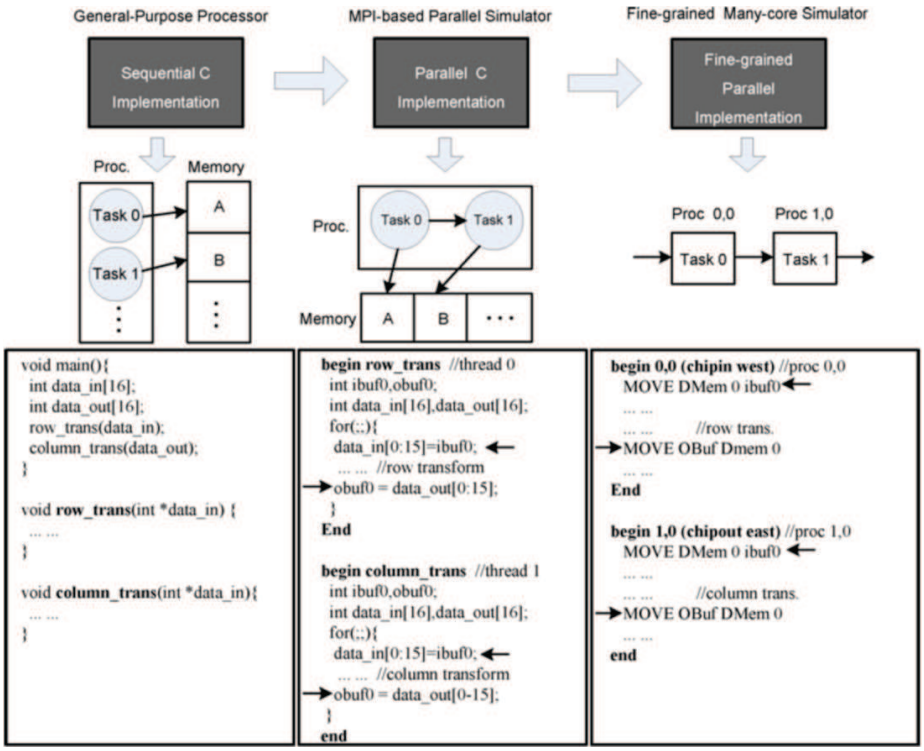

Figure 1.5: Program method for fine-grained Asynchronous Array of simple Processors (AsAP) system [Xiao11A].

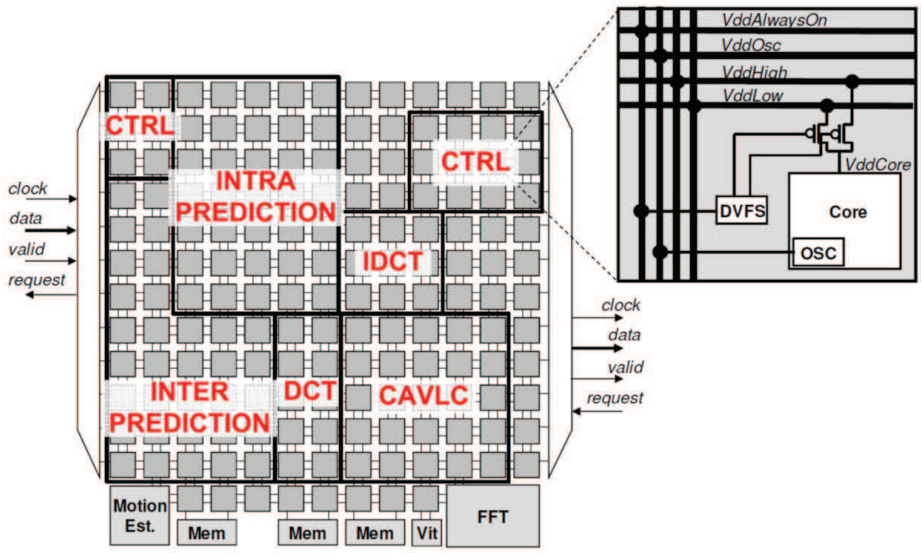

Figure 1.6: H.264 encoder mapped onto a AsAP chip [Le07A].

capabilities. Moreover, some coding tools are even more efficiently implemented in hardware than in software. For instance, in the DCT process, most of the calculating operations are add and shift ones, which are highly efficient when being implemented in hardware.

Currently, reducing the power consumption is mandatory in electronic systems design, not only because of limited battery lifespan for mobile devices but also for cooling cost and environmental issues. Several H.264/AVC hardware implementations have focused on power consumption reduction [Moch07A, Lin08A1, Baker08P, Chan09A, Chen09A, Kim11P]. This has proved again the efficiency of hardware solutions. For example, the design in [Wang03A] used a $130MHz$ ARM966 processor. However, it is only capable of QCIF decoding at 7.5fps. Even if some software designs can achieve QCIF at 30fps the power consumption is relatively large and may be not suitable for mobile applications. Using hardware implementation solutions, we can also overcome high memory access rate during the coding process. Usually, an off-chip memory is used for reference frames to reduce the total area and energy cost. Then some on-chip buffers are implemented to reduce the external bandwidth requirements, thus reducing timing cost and power consumption. It is proved that an embedded local search window buffer can reduce the external bandwidth from 5570TB/s in the software implementation to 700MB/s in a hardware one [Wen11D].

Because our work is in the encoding process, the next section will review some state-of-the-art hardware implementations of the H.264/AVC encoders.

1.4 Overview of H.264/AVC hardware encoder implementations

The specification recommended by ITU-T and ISO/IEC only provides the H.264/AVC encoded data syntax and how to decode the video. Here some H.264/AVC hardware encoders found in the literature are analyzed. These encoders range from basic pipelining architectures to several innovative improvements.

1.4.1 Basic pipelining design

Due to the long encoding path of the H.264 standard, pipelining architectures at MB level are usually implemented. Figure 1.7 shows the major modules of a four-stage H.264 encoder [Huan05A1]. The Motion Estimation (ME) block, operating with the Motion Compensation (MC) one to perform inter prediction, is a potential coding tool but with a huge computational complexity. The ME module with full search may spend more than 90% of the overall computation [Chen09A]. Hence, in pipelining architectures, the ME task is separated in two sub-tasks (integer (IME) and fractional (FME)). To achieve a balanced schedule, the intra prediction (IntraP) is placed in the third stage. The Intra mode decision requires the Forward Transform and Quantization/de-Quantization and Inverse Transform (FTQ/ITQ) and the reconstruction results of the current MB in the same stage as IntraP. The last stage contains two independent modules, Entropy Coder (EC) and De-blocking Filter (DF). In order to reduce the size of the buffer between stages, the pipeline is usually scheduled to operate at the MB level rather than at the frame level. The four-stage pipelining architecture cuts the coding path in a balanced manner which eases the task scheduling but it also increases the overall encoder latency.

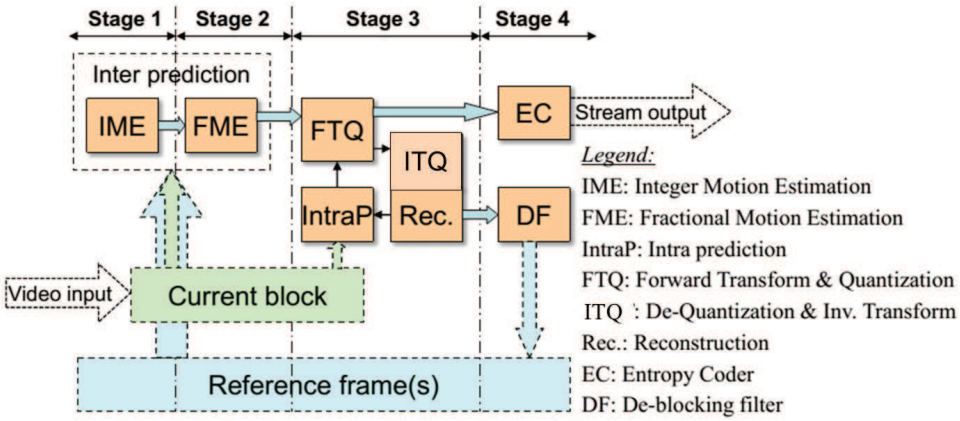

Figure 1.7: A conventional four-stage pipelining architecture for H.264 hardware encoder.

Many works have implemented H.264 encoders based on this basic four-stage pipelining architecture, e.g. [Chen06AA, Moch07A, Chen08A, Chan09A]. The pipeline implemented by S. Mochizuki *et al.* [Moch07A] is a 6-stage

one.

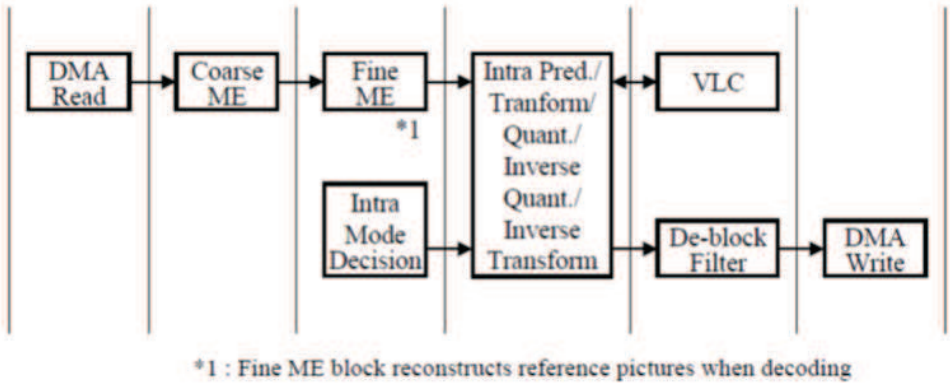

Figure 1.8: An pipelining architecture of H.264/AVC hardware encoder closed to the clasical pipeline [Moch07A].

However, the main encoding tasks are performed in the four middle stages in a way close to the classical pipeline. As can be seen in Figure 1.8, the ME operates in the two earlier stages. The Intra and the transform-quantization occupy the next stage. Then, the entropy coder (VLC: Variable-Length Coder) and the DF are placed in the fifth stage. The first and last stages are for Direct Memory Access (DMA) reading and writing, respectively. Moreover, the intra-prediction is modified to enable high picture quality. As specified in the H.264 standard, the "best" mode can only be selected after all predictive blocks in an MB are processed through Intra, FTQ, ITQ then reconstruction (Rec.). In this 6-stage pipeline encoder, the mode decision part is performed before the other intra-prediction tasks, in the previous stage. The best mode is decided from the original image but not from the locally decoded image as in the classical intra-prediction engines. With a reduction of more than 300K logic gates and of the processing time in comparison to a classical intra prediction block, this solution avoids limiting the number of mode candidates while keeping high picture quality. With a faster Intra stage, the design in [Moch07A] is slightly less balanced than the one in [Chen06AA]. Most of its improvements are provided by dedicated techniques, such as prediction from original image in the intra prediction block, active skip prediction algorithm in the inter prediction block, etc. but not by the architecture.

Sometimes, the implementation of only three stages [Chen08A, Lin08A1, Chen09A] makes sense. For instance, FME and IntraP are grouped into one

stage to share the current block and pipeline buffers [Lin08A1]. This allows decreasing the latency on the entire pipeline [Chen08A]. Moreover, reducing the number of stages also decreases the power consumption for the data pipelining [Chen09A]. However, this scheme obviously leads to an unbalanced schedule. When IntraP and FME operate in parallel, too many tasks are put into the second stage. To avoid this throughput bottleneck, [Chen09A] has retimed the IntraP and Rec. blocks to distribute them into the last two stages, see Figure 1.9. The luminance data is first processed in the second stage, and then the chrominance one is treated in the third one. The FME engine is also shared for the first two stages [Chen09A].

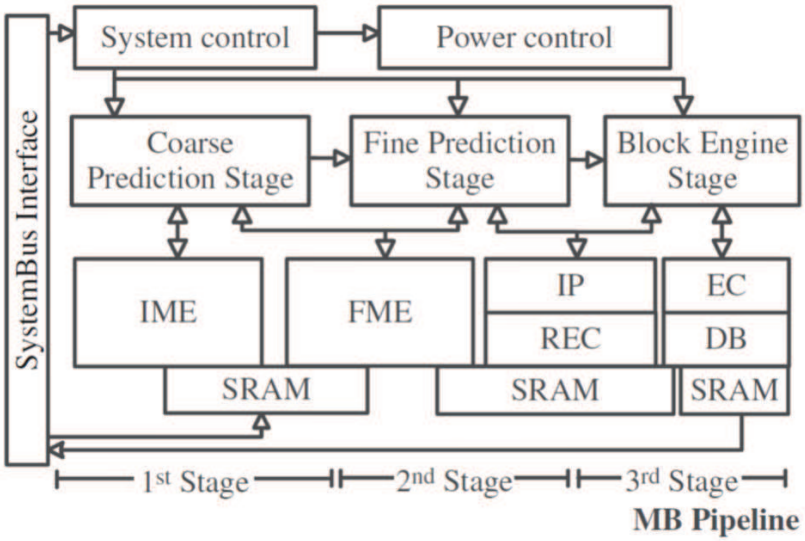

Figure 1.9: A 3-stage pipelining architecture of H.264/AVC hardware encoder proposed in [Chen09A].

To summarize, the basic pipelining architecture is naturally designed from the coding path of the H.264 standard. Some modifications can improve particular features for a given design. Different architectures that highly improve the coding speed or scalability are presented in the next sub-section.

1.4.2 Design improvements for specific purposes

H.264 hardware encoders can be split in 3 sets, namely scalability-oriented, speed-oriented and power-oriented ones. Hereafter, we will introduce the

two first groups, then power-oriented H.264 hardware video encoders will be presented in details in Section 1.5.

- **Scalability-oriented design for H.264 hardware encoder**

An H.264 encoder for high profile, which "firstly supported Scalable Video Coding" (SVC), was proposed in [Chen08A]. Figure 1.11 illustrates its 2-stage frame pipelining architecture and its B-frame parallel scheme.

The first stage contains a four-stage pipelining encoder as discussed in subsection 1.4.1. The Fine-Grain-Scalability (FGS) arithmetic coder was integrated in the second pipelining stage at frame level to enable the scalable quality feature. The proposed encoder also supports spatial scalability via inter-layer prediction and temporal scalability by using Hierarchical B-frame (HB).

In [Chen08A], many schemes were adopted to reduce external memory bandwidth and internal memory access. Two consecutive B-frames can be independently encoded and the encoder processes both frames in parallel to use the common reference data. This method reduces the external memory bandwidth of the loading searching window for Group Of Pictures (GOP) IBBP by 50%, and it reduces the external memory bandwidth for the HB by 25%, see Figure 1.11. In the next stages, the Intra, Rec. and EC blocks are duplicated to process two MBs concurrently. Then, the data reuse scheme in the ME engine enables both inter-layer prediction and HB while saving 70% of the external memory bandwidth and 50% of the internal memory access. Lastly, the FGS engine also integrates other techniques to reduce the memory bandwidth. From all these modifications and improvements, the high profile-SVC encoder, even with computation four times more complex than a baseline profile encoder, achieves comparable power consumption, i.e. only $306mW$ in high profile and $411mW$ with SVC for HDTV1080p video [Chen08A], see Table 1.2 on 39. The area cost of this design can be guessed quite large when compared to the basic pipelining architecture because of the complexity of the additional features.

- **Speed-oriented design for H.264 hardware encoder**

A high speed codec in high profile is proposed in [Iwat09A], see Figure 1.10. It makes use of a two-stage pipeline encoder at frame level. The second stage operating in the stream-rate domain contains only the VLC.

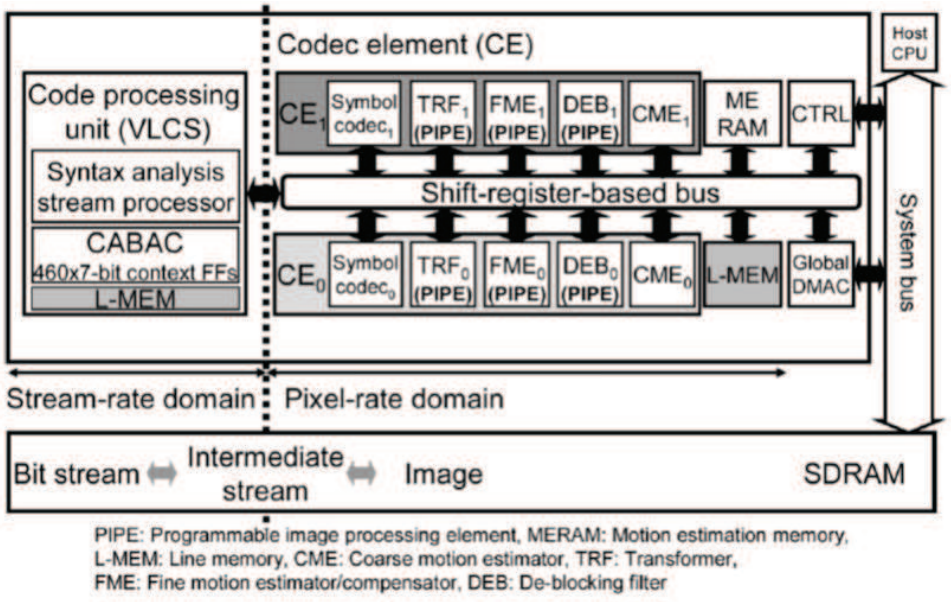

Figure 1.10: High speed pipelining architecture for H.264 hardware encoder, proposed in [Iwat09A].

All other tasks are performed in the first stage in the pixel-rate domain, which contains two parallel MB-pipeline processing modules named Codec Element (CE). This method increases the processing performance but it also increases the area cost and the power consumption, see Table 1.2. To support video-size scalability, on-chip connections among sub-modules are done via a shift-register-based bus network. This bus structure enables the scalability for the encoder in case more CEs are required for video-size scalability. The power consumption caused by the duplicated CE and the high computational complexity in high profile is decreased by the implementation of a Dynamic Clock-Supply-Stopping (DCSS) scheme. This low-power method will be discussed hereafter. The architecture with two parallel MB-pipelines doubles the coding throughput.

As can be seen from this short review, modified architectures can provide interesting improvements with respect to the scalability and the computational speed, at the price of higher power consumption and/or larger silicon area, leading for designers to a trade-off between various conflicting objectives. Power-oriented architectures that embed additional techniques to reduce the power consumption or to enable power-aware functioning are now discussed.

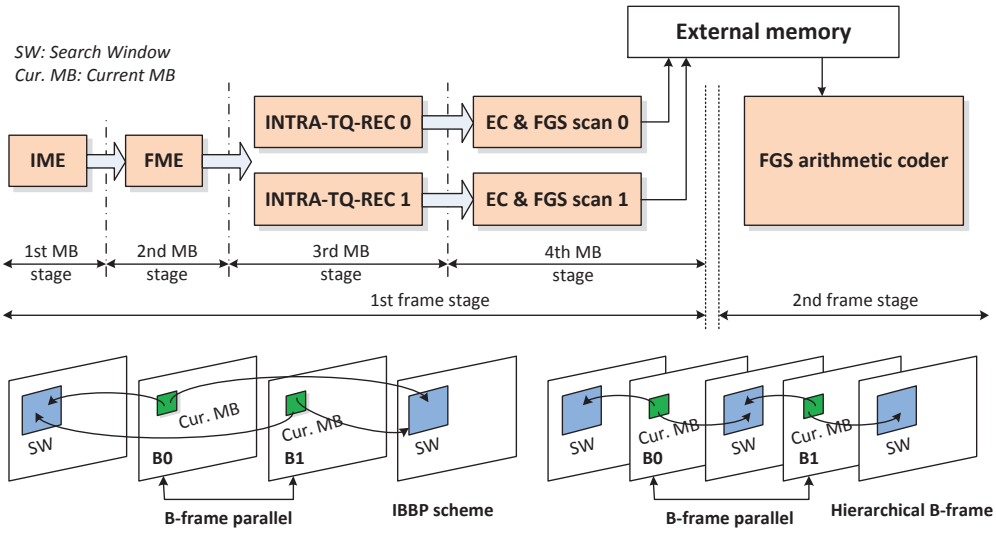

Figure 1.11: Two-stage frame pipelining H.264 encoder architecture, adapted from [Chen08A].

1.5 Power-oriented design for H.264/AVC hardware encoders

This section presents power-oriented designs for H.264/AVC hardware encoders. Subsection 1.5.1 introduces the fundamentals of low-power techniques in electronic circuits and systems design. Then power-oriented H.264 hardware video encoders will be surveyed.

1.5.1 Low-power design techniques

The power consumption in digital CMOS circuits can be modeled as follows [Chand92L, Weng03A]:

$$P_{total} = K.C_L.V_{dd}.V_{swing}.f + I_{sc}.V_{dd} + I_{leakage}.V_{dd} = P_{sw} + P_{sc} + P_l \quad (1.3)$$

The main part is the dynamic power P_{dyn} that contains the switching P_{sw} and short-circuit P_{sc} powers. The leakage power P_l depends on the technology at hand. In most cases, the voltage swing V_{swing} is equal to the supply voltage V_{dd} [Chand92L]. Hence, P_{sw} can be expressed as the product of the node transition activity factor K, the load capacitance C_L, the square of supply voltage V_{dd} and the clock frequency f. Thus, K, C_L, f and V_{dd} influence the power consumption.

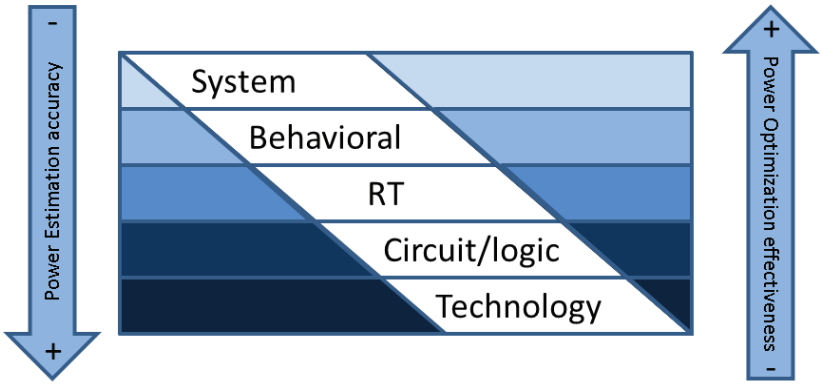

Figure 1.12: Power estimation and optimization based on abstraction levels, modified from [LecNENTS1].

Acting on V_{dd} is usually preferred because of the quadratic dependency on P_{sw} and its linear dependency on P_{sc} and P_l. Lowering the frequency f also scales down the power consumption. For idle components, it is also possible to stop the clock or enter a power-off mode. However, even in active states, components do not always need to work at their maximum performance level (i.e. clock speed). Therefore, f and V_{dd} can vary (i.e. scale down) to operate at lower power states [Beni00A].

Some techniques for power reduction can be summarized as follows:

- some power reduction approaches tried to reduce the switching activity K by re-encoding data, reconstructing the calculation circuits, modify the communication links, memory and hierarchy [Weng03A]. These architectural strategies are applied during the design phase;

- Dynamic Voltage Scaling (DVS) scales down the supply voltage V_{dd} to reduce the power consumption. This method is efficient because it scales approximately by V_{dd}^2 the power consumption. It also reduces the total energy of the consumption system;

- Dynamic Frequency Scaling (DFS) acts on the operating frequency f. DFS offers power reduction but no energy reduction;

- Dynamic Voltage and Frequency Scaling (DVFS) techniques implement the scaling of both factors;

- The Adaptive Body Bias (ABB) method is applied to the body voltage of both nMOS and pMOS transistors in CMOS technology. By this way, we can modify the effective threshold voltage and vary the leakage power P_l. This is applied at transistor circuit level.

As our H.264/AVC hardware encoder, namely the VENGME platform, was availabe at RTL level at the time the power reduction techniques were developed, DFS and/or DVS (DVFS) seem to be the most appropriate methods.

Indeed, power estimation and optimization can be realized at different levels, from transistor circuit level to hardware/software system level. Note that power management solutions have been investigated for the last few decades [Chand92L]. Figure 1.12 shows the dependency of power estimation accuracy and power optimization effectiveness on the various levels of abstraction, namely at system, behavioral, Register Transfer (RT), circuit and technology levels. At lower levels, with more physical information available, power estimation can be more accurate. Note that power optimization methods at lower levels are more generic, i.e. application independent. However, they are less efficient than the methods applied at higher levels [LecNENTS1].

1.5.2 Low-power techniques for H.264/AVC encoder hardware implementation

Low-power H.264 encoders also implement the pipelining architecture with additional low-power techniques. Among these techniques, Dynamic Clock Supply Stop (DCSS) [Moch07A] and fine-grained clock-gating [Chen09A], exploit the inactive state of sub-modules to reduce their idle power consumption. Figure 1.13 shows the schedule of the modules in a H.264 encoder example and the time slots when power can be saved [Nguy14A]. The different blocks correspond to the encoder architecture presented in Figure 1.7. Actually, an unbalanced schedule will offer more opportunities to reduce power thanks to the integration of power reduction techniques during the inactivity phases. While DCSS cuts off the supply clock signal per stage when all modules in this stage are not operating, clock-gating pauses the clock signal entering unused modules. DCSS was estimated to reduce up to 16% the power consumption [Moch07A] and fine-grain clock gating in [Chen09A] can save around 20% the power consumption. Thus, this later

seems to provide more power reduction but its control is more costly as it requires a control engine of the clock per block.

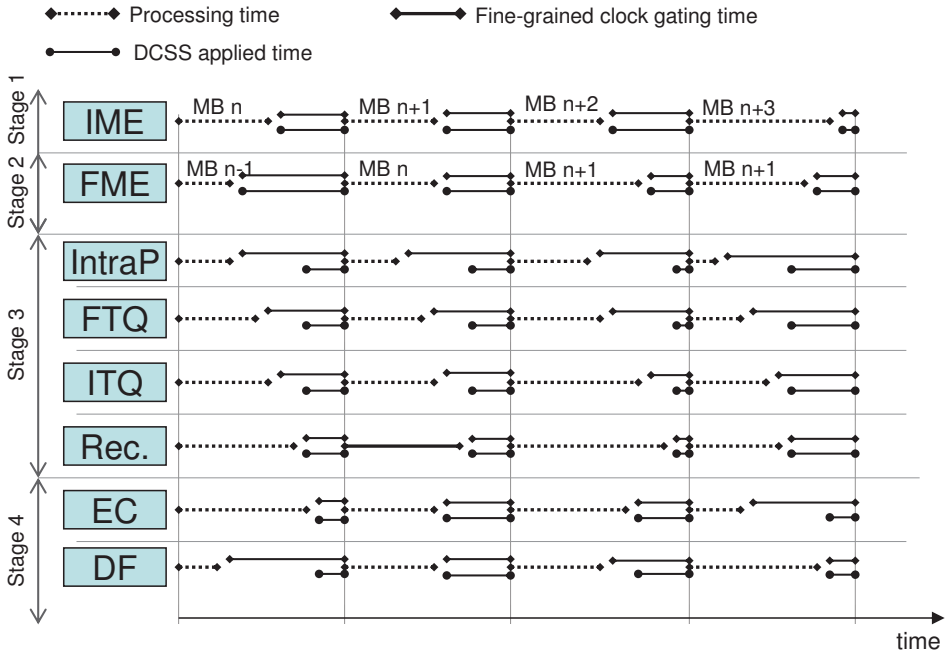

Figure 1.13: DCSS and fine-grained clock gating exploit schedule of H.264 encoder.

Memory access also consumes a lot of power. For instance, a cache memory can be implemented [Zuo12A] to pre-fetch reference images for the MC module in HD applications, leading to 75%-86% of MC external bandwidth reduction. The encoder in [Kim11P] applies three solutions: frame memory compression, early skip mode decision and reduced FME search range to save up to 49.9% bus and external memory power consumption.

The H.264 encoder proposed in [Lin08A1], see Figure 1.14, does not implement the above specific low-power techniques. However, many efforts have been done in order to reduce the memory access. Firstly, Intra and FME are both placed in the second stage to use common current block and pipeline buffers. Secondly, it implements eight-pixel parallelism intra-predictor to reduce the area cost and a particular motion estimation block that can deal with high throughput. Moreover, the high throughput IME with Parallel Multi-Resolution ME (PMRME) technique approach also leads to 46% of memory access reduction. Actually, PMRME only samples the necessary pixels to be stored in the local memory. This video

encoder achieves promising power figures, that is $6.74mW$ for CIF video and $176.1mW$ for 1080p video in baseline profile under technology $130nm$, see Table 1.2.

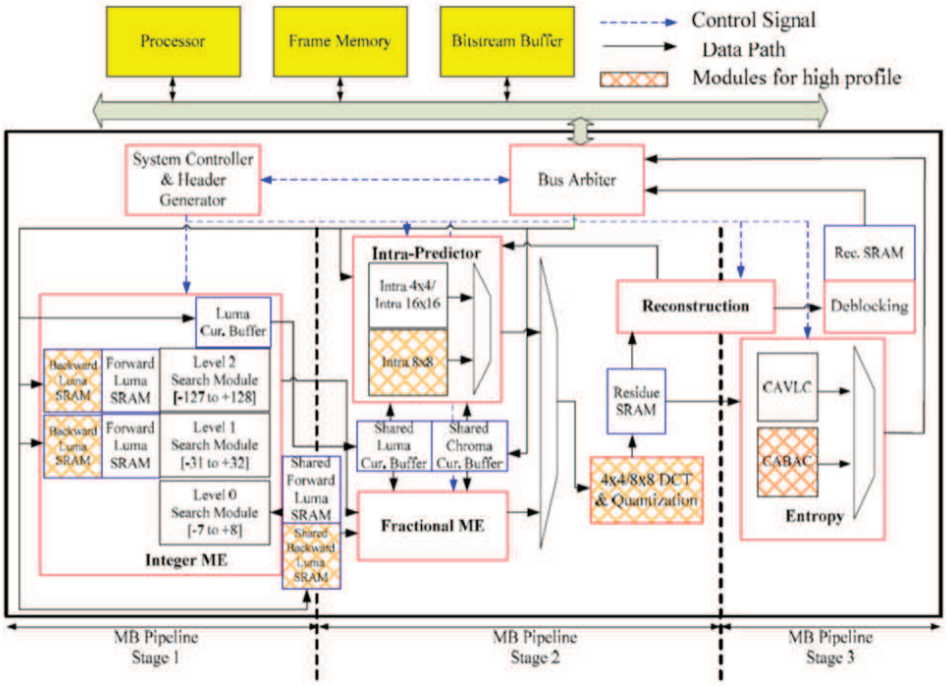

Figure 1.14: Low-power H.264/AVC hardware encoder architecture proposed in [Lin08A1].

A low-power ME module implementing low-power techniques is proposed in [Chen09A], see Figure 1.9, page 29. One of these low-power techniques is data reuse to save power during memory access phases. In the IME module, both intra-candidate data reuse and inter-candidate data reuse are applied. Intra-candidate calculates the matching cost of larger blocks by summing up the corresponding cost of smaller blocks (4×4). Inter-candidate shares overlapped reference pixels for two neighboring searching candidates. Differences among neighboring Motion Vectors (MVs) are also used to reduce the computation. In the FME block, an online interpolation architecture to reuse interpolated data and a mode pre-decision to reduce the number of mode candidates are adopted to save power consumption. The one-pass algorithm and its corresponding hardware implementation not only alleviate the memory access but they also increase the throughput of the FME sub-module. The IME data access solution that is proposed

consumes 78% less than a standard IME engine. The FME engine halves the memory access thus saving a large amount of power consumption for data access.

These designs prove that reducing the memory access is an efficient high-throughput low-power scheme. However, they require the design of many specific sub-modules, which leads to a complex design task.

Other designs propose not only low-power features but also quality scalability to improve the power consumption. As an example, [Kim11P] defines 10 power levels to adapt the power consumption depending on the remaining energy.

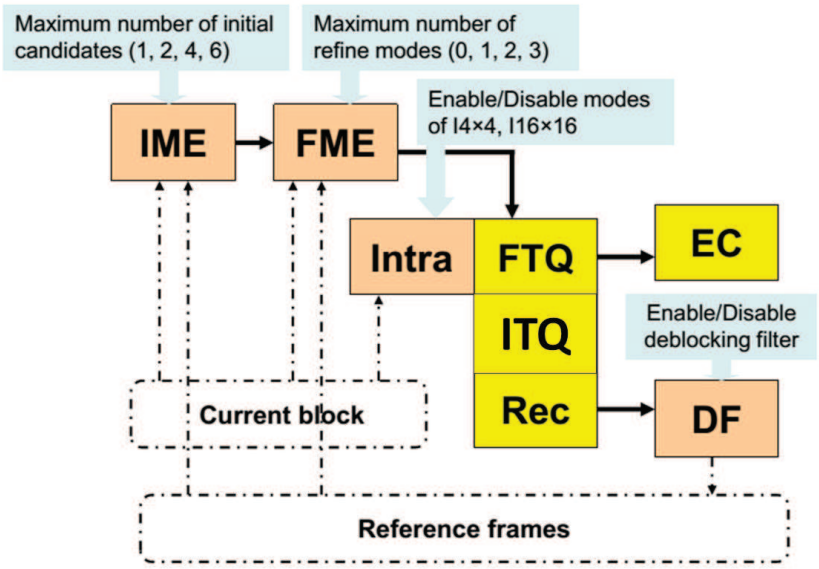

Figure 1.15: Parameterized H.264/AVC video encoder, modified from [Chen09A].

The H.264/AVC encoder proposed in [Chen09A] is dedicated to applications for mobile devices. Besides several low-power techniques as presented above, a pre-skip solution with a reconfigurable parameterized coding system together with a three-layer system architecture with flexible schedule enables power scalability. The pre-skip solution is indeed the very first step of the motion estimation module. For each MB, it compares the Sum of Absolute Differences (SAD) function of the candidate $(0, 0)$ to a threshold S in order to skip all the ME process, when possible. The parameterized coding system provides 128 different power modes based on the parameters of the IME, FME, Intra, and DF blocks. Figure 1.15 illustrates these param-

eters in the encoding system. The three-layer architecture is a hierarchical controlling system containing a system controller, a power controller and a processing engine controller. This architecture enables the clock gating technique at fine-grain level to be implemented so that the clock entering one processing engine can be stopped while the clock entering another processing engine in the same pipeline stage can be kept.

The work in [Chan09A] focuses not only on portable video applications but it supports a wider range of resolutions, up to HD720@30fps. For each resolution, four different quality levels with their corresponding power consumption level are provided. The quality-scalability feature is implemented with parameterized modules. For instance, they proposed a Half-word Down-sample Local Full Search (HDLFS) IME using three parameters: down-sample rate, candidate number and local full search range. Different operating frequencies are used in the different quality levels. Besides, some design techniques to reduce the complexity of the main modules and to decrease their power consumption are also applied.

1.5.3 Discussion

Table 1.2 summarizes the state-of-the-art solutions that have been discussed in this chapter. Various features are presented but only the power consumption one will be discussed.

When comparing power results, we also consider the difference in implementation technologies. In general, for applications requiring low speed performance, we can affort a large node technology. For instance, applications whose operating frequency is less than $200MHz$ are typically implemented with CMOS technologies higher than $130nm$. In applications with higher speed performances, we need more agressive technologies, typically smaller than $65nm$, at the cost of higher power consumption. Moreover, the profile and resolution obviously influence the operating frequency and thus the power consumption. Indeed, the encoders that support multiple profiles [Lin08A1] or multiple resolutions [Chen06AA, Moch07A, Lin08A1, Chan09A, Chen09A] operate at different frequencies and show different power consumption figures. Therefore, when the power results are compared, the resolution and profile that the encoders support have to be taken into account as well.

Both the specific low-power techniques [Moch07A, Chen09A] and the

Table 1.2: A survey of H.264 hardware video encoders

	[Liu07A]	[Chen08A]	[Iwat09A]	[Chen06AA]	[Chen09A]	[Moch07A]	[Lin08A1]	[Chan09A]	[Kim11P]
Target	Real-time	SVC, high profile	Performance, Low power Video size scalable	HW design for H.264 video codec	Low power Power aware Portable devices	Low power real-time high picture quality	High profile Low area High throughput	Dynamic Quality-Scalable Power aware	Low-power Power aware
Profile	Baseline level 4	High SVC	High level 4.1	Baseline level upto 3.1	Baseline	Baseline level 3.2	Baseline/High level 4	Baseline	N/A
Resol.	1080p30	HDTV 1080p	1080p30	720p SD/HD	QCIF 720SDTV	720p SD/HD	CIF to 1080p	CIF to HD720	CIF, HD1280×720
Techno. (*nm*)	UMC 180 1P6M CMOS	UMC 90 1P9M	65	UMC 180 1P6M CMOS	TSMC 180 1P6M CMOS	Renesas 90 1POLY-7Cu-ALP	UMC 130	130	N/A
Freq. (*MHz*)	200	120 for high profile; 166 for SVC	162	81 for SD 180 for HD	N/A	54 for SD 144 for HD	7.2 for CIF 145 for 1080p	10-12-18-28 for CIF; 72-108 for HD720	N/A
Gate count (*Kgates*)	1140	2079	3745	922.8	452.8	1300	593	470	N/A
Memory (*Kbytes*)	108.3	81.7	230	34.72	16.95	56	22	13.3	N/A
Power (*mW*)	1410	360 for high profile; 411 for SVC	256	581 for SD 785 for HD	40.3 for CIF 2 ref. 9.8-15.9 for CIF 1 ref. 64.2 for 720SDTV	64 for 720p HD	6.74 for CIF Baseline profile; 242 for 1080p high profile	7-25 for CIF; 122-183 for HD720	238.38-359.89 depends on power level

strategies implemented to reduce the memory access [Lin08A1, Kim11P] present appealing power consumption figures, e.g. 9.8-40.3mW for CIF video [Chen09A], 64mW for HD720p [Moch07A] and 242mW for 1080p high profile [Lin08A1]. In the baseline profile, with 6.74mW of power consumption for CIF video [Lin08A1], the technique applied for the memory access reduction seems to perform better than the one in [Chen09A] (9.8mW).

Recent encoders with power-aware ability [Chan09A, Chen09A] take even less silicon area and seem more suitable for mobile applications.

Note that the threshold of 100mW of consumption is widely admitted for portable media applications [Chen09A].

1.6 Conclusion

In this chapter, an overview of video coding standards and H.264/AVC standard was presented. State-of-the-art solutions on the design and hardware implementation of the H.264/AVC video encoders have been discussed.

The H.264/AVC standard adopted many advanced video coding techniques from previous standards. It is widely applied in many applications. Even if the new standard H.265 is now available, it is hard to say anything about its success in near future.

The trend is to implement H.264/AVC in dedicated hardware. As presented above, many techniques can be applied to improve the basic pipelining architecture in terms of scalability, speed and power consumption.

The next chapter is dedicated to the design of a H.264/AVC hardware encoder that targets mobile applications. This design has been conducted in the context of the "Video Encoder for the Next Generation Multimedia Equipment" (VENGME) research project (No. QGDA.10.02). The project is granted by Vietnam National University, Hanoi (VNU). The VENGME H.264/AVC video encoder platform was designed with improvement techniques applied in all modules. One of the contributions of the thesis is the design and implementation of the Entropy Coding and Network Abstraction Layer data packer (EC-NAL) module in the VENGME platform. The power simulation at RTL level on the whole VENGME platform and on the EC-NAL module are also performed to early analyze the power composition of the platform for power optimization.

With the power consumption of $58.25mW$, the VENGME H.264 video encoder meets the consumption requirement for mobile applications.

Chapter 2

The VENGME platform and its EC-NAL module

According to the state of the art for H.264/AVC video encoding hardware implementations provided in the previous chapter, H.264/AVC encoders are designed based on the basic pipelining architecture with different improvements added to serve specific design objectives, namely scalability, speed and power consumption. This chapter will introduce the design of the VENGME H.264/AVC encoder platform that targets mobile applications. The VENGME platform was designed in the framework of the "Video Encoder for the Next Generation Multimedia Equipment" (VENGME) project. This research project is granted by Vietnam National University, Hanoi (VNU).

Our VENGME platform adopts a pipelining architecture with different dataflow control which enables applying different low-power control methods. Note that $100mW$ seems to be the maximum power consumption acceptable for media coding components to be implemented in mobile portable devices [Etoh05A]. The verification and synthesis results show that with the power consumption of $58.25mW$, the VENGME video encoder can meet this requirement. Some VENGME modules, e.g. the Transformation-Quantization module, the Entropy Coder and Network Abstraction Layer (EC-NAL) module, present nice power and performance figures, similar to the ones obtained by state-of-the-art designs.

The main contribution of this work to the VENGME platform is the de-

sign of the Entropy Coder (EC) and the Network Abstraction Layer (NAL) bytestream data packer (EC-NAL) module. Another contribution is the power simulation at RTL level of the whole VENGME platform and of the EC-NAL module. This simulation was performed prior the VENGME synthesis. The power simulation results of the whole VENGME platform allow to analyze the power distribution over the various modules of the platform and seek for extra power optimization.

The thesis work is conducted in three laboratories, namely

1. the Smart Integrated Systems (SIS) laboratory in Vietnam National University, Hanoi (VNU), Vietnam;

2. the Laboratory of Silicon Integration and Digital Architecture (LISAN) in CEA-LETI, France;

3. the Laboratory of Infrastructure and Software Tools for computing platforms (LIALP) in CEA-LETI, France.

The architecture of the VENGME platform is designed in the SIS laboratory (Vietnam). Low-power techniques, e.g. DFS technique, are acquired from the LISAN laboratory (France). The control aspects are searched with the LIALP laboratory (France). As a consequence, the author had the opportunity to work with large-node technologies and with more advanced ones. This explains the various CMOS technologies that are used all along this chapters for the results., especially for results of power simulations that have been performed both in Vietnam and in France.

Section 2.1 presents the VENGME platform, including the main features, the overall architecture, the dataflow control, verification and implementation results. The power simulation results at RTL level at platform level is also presented in this section ($32nm$ CMOS technology, in France). Section 2.2 summarizes the specification of the data structure for the Entropy Coding and the bytestream Network Abstraction Layer (NAL) for H.264 video encoding. It also reviews the state-of-the-art for Entropy Coder and video data packer designs. Section 2.3 focuses on the design of the VENGME EC-NAL module. Its implementation results and the comparison with state-of-the-art designs are presented in Section 2.4. Finally, the power simulation results at RTL level for the VENGME EC-NAL module are analyzed in Section 2.5. ($32nm$ CMOS technology, in France)

2.1 The VENGME platform

The "Video Encoder for the Next Generation Multimedia Equipment" (VENGME) project is a research project (No. QGDA.10.02) granted by Vietnam National University, Hanoi (VNU). The objective is to design and implement a hardware H.264/AVC encoder using advanced coding techniques and targeting mobile platforms. As a consequence, low-power design techniques, design optimization and reconfigurable designs have to be focused on. This section provides an overview of the VENGME platform architecture and its task scheduling. The operations of the main modules and the VENGME video encoder verification and implementation will be also presented [Tran13B]. The design of the entropy coder (EC) and the Network Abstraction Layer (NAL) bytestream data packer (EC-NAL) module, which is the contribution of the thesis author, will be presented in details in Section 2.2.

2.1.1 Introduction to the VENGME platform

In the H.264/AVC recommendation that was firstly published in 2003, there were three profiles[1]: the *Baseline* profile is mainly designed for video conferencing applications; the *Main* profile is used for television broadcasting and video storage applications because of its high compression rate and high fidelity; and the *Extended* profile is used for HD applications. The standard has been extended, in 2009: it defined totally 11 profiles and 5 levels[2]. Most of the newly published "high profiles" were developed from the Main profile. The reader can refer to Figure 1.7 (page 27) for a generic H.264/AVC encoder architecture.

Besides mobile applications, the VENGME project also targets the extension to other applications. Therefore, the Main profile at level 2 is selected. The design performed during the project, also called the VENGME design/platform, is optimized for CIF video at a frame rate of 30*fps*. The architecture can be extended to larger resolutions by enlarging the reference memory and the search window.

With the Main profile that has been chosen, the VENGME platform supports the following features:

[1]See Chapter 1, Section 1.2.2
[2]See Chapter 1, Section 1.2.2

- Intra prediction to create a predicted version of the current block from the previously coded samples in the same frame;
- Inter prediction that contains motion estimation (ME) and motion compensation (MC), enabling high compression efficiency;
- B-slice. It means that the future frames (in display order) can be used as reference frames;
- Variable block-size enables precise segmentation of the moving regions;
- Quater-pixel-accurate motion compensation describes accurately the displacement of moving areas;
- CAVLC entropy coding. The Main profile enables the use of both Context Adaptive Variable Length Coding (CAVLC) and Context Adaptive Binary Arithmetic Coding (CABAC) techniques. However, to simplify the hardware architecture, the VENGME platform supports only the CAVLC technique. The CABAC is optional and may be developed later on;
- Weighted prediction is used to improve the motion compensation process;
- The de-blocking filter reduces blocking-artifacts for block-based video coding techniques and it increases the quality of video images;
- The sampling format is 4:2:0, which means that the vertical and horizontal resolutions of both chroma components (Cb and Cr) are equal to half of the resolutions of the luma, Y[3];
- NAL byte stream formatting allows a encoded video can be used in many network environments.

Table 2.1 summarizes the key features and targeted parameters for the VENGME design.

These features are fixed by regarding the Annex profiles and levels of the Recommendation [H.264].

[3]See Chapter 1, Section 1.2.1

Table 2.1: Targeted features and parameters of the VENGME design

Features	**Targeted parameters**
Features	I & P slices; B slice; multi reference frames; De-blocking filter; Entropy coding: CAVLC & Exp-Golomb; sampling format 4:2:0; 8-bit sample depth
Video format	CIF at 30*fps*
Maximal speed	11880 *MB/s*
Maximal frame size	396 *MBs*
Maximal bit rate	$\sim 2Mbps$
Compression rate	$\sim$ 30 times

2.1.2 Hardware architecture of the VENGME H.264/AVC video encoder

The overall hardware architecture of the VENGME H.264/AVC encoder is depicted in Figure 2.1. It contains all the blocks that appear in a general H.264/AVC encoder, namely, the Intra Prediction (IP), the inter prediction consisting of Motion Estimation (ME) and Motion Compensation (MC), the transformation and quantization grouped into the Forward Transformation and Quantization (FTQ), the de-Quantization and Inversed Transform (ITQ), the Entropy Coder (EC), the Network Abstraction Layer (NAL), the Reconstruction Macroblock (Recons. MB), the De-blocking Filter (DF), and the Rate-Distortion Optimization (RDO).

The video encoder communicates with the external system via an Advanced Microcontroller Bus Architecture (AMBA) bus. The configuration of the encoder parameters is done via the Advanced Peripheral Bus (APB), a slave bus interface, while video data are transferred via the Advanced High-performance Bus (AHB), a master bus interface. After the configuration phase, the encoder starts the H.264/AVC encoding process. The source input video, reference data and encoded data are stored in off-chip memories. Hence, the video encoder still needs to access these memory blocks via the AHB master bus during the encoding process. An internal bus is used to communicate between the Memory Access Units (MAUs) and the AHB master bus interface. In the VENGME architecture, there are four Memory

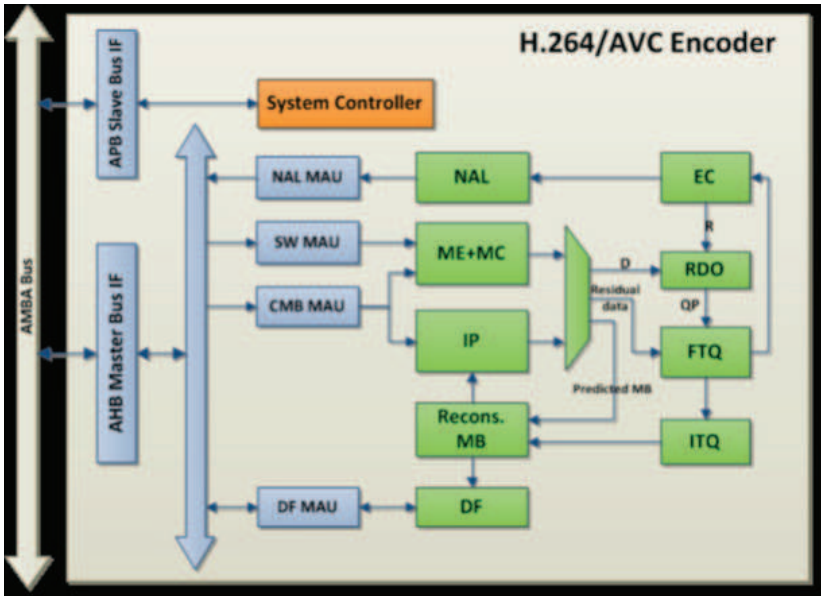

Figure 2.1: The VENGME H.264/AVC video encoder architecture.

Access Units (MAUs):

- the Current MB Memory Access Unit (CMB MAU) is used to fetch source data;
- the Search Window Memory Access Unit (SW MAU) is used to access reference data;
- the De-blocking Filter Memory Access Unit (DF MAU) is used by the DF module;
- the Network Abstraction Layer Memory Access Unit (NAL MAU) is used to write out the encoded data.

Among them, the NAL MAU has the highest priority to avoid a bottleneck in the dataflow.

We present now a brief description of the main modules of the VENGME platform. Some dedicated techniques to reduce the area cost and increase the throughput are applied to each module [Tran11C, Bui12A, Nguy12A, Dang13A, Nguy13A, Nguy13H].

- **The Intra prediction module**

With the input videos of sampling format 4:2:0[4], there are 17 intra prediction modes in one MB. To overcome the huge amount of data and calculation requirements in the intra prediction generation, parallel calculations and calculation reduction techniques must be implemented during the module design. Among several available cost functions, the Sum of Absolute Transformed Differences (SATD) has been chosen for most of the current intra prediction designs thanks to its simplicity, high efficiency and ability for calculation reduction [Huan05A].

Some researches tried to reduce the number of calculating modes using a threshold [Meng03E, Huan05A] or using the correlation between modes [Chen05F, Li07A]. Here, we use the most popular method, that is comparing the results of all the modes to optimize the prediction.

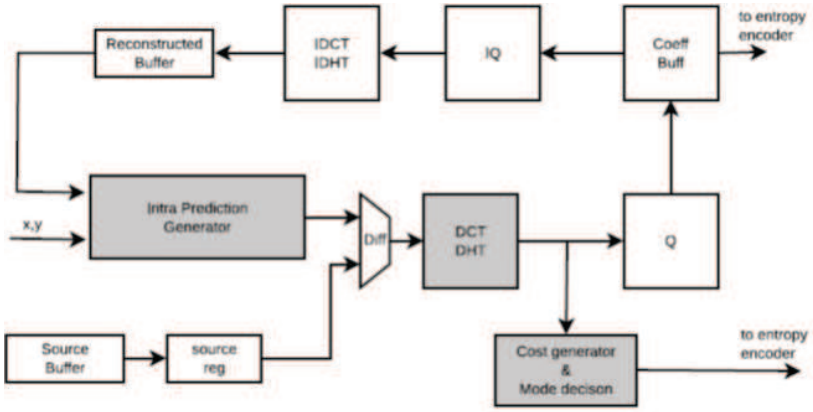

Figure 2.2: Dataflow of the Intra prediction module.

The details on the design of the Intra prediction module in the VENGME platform is presented in [Bui11H] and [Bui12A]. Three main submodules of the classical intra prediction architecture, namely, the intra prediction generator, the prediction efficiency estimation and the mode selection and the control part are equivalent to the main blocks of this module dataflow. They are Intra Prediction Generator, Discrete Cosine Transform and Discrete Hadamard Transform (DCT DHT) and Cost generator and Mode decision, as illustrated in Figure 2.2. The system can generate prediction of 4 modes in one clock cycle. The prediction data of the selected mode is discrete cosine transformed (in the DCT block), quantized (Q), dequantized (IQ) and inverse transformed (IDCT-IDHT) to reconstruct the image.

[4]See chapter 1, section 1.2.1

- **The Inter prediction module**

Inter prediction consists of two main functions, namely, the Motion Estimation (ME) and the Motion Compensation (MC). For supporting sub-pixel motion estimation, half-pixel and quater-pixel, the ME is divided into Integer Motion Estimation (IME) and Fractional Motion Estimation (FME). Figure 2.3 depicts the block diagram of the VENGME Inter prediction module. It contains three main submodules, namely, the IME, the FME and the MC. Besides, it uses also the CMB MAU and SW MAU for input, and the Residual/Predicted Memory (RES/PRED MEM) and Encapsulating Encoding Information (EEI) MEM for output.

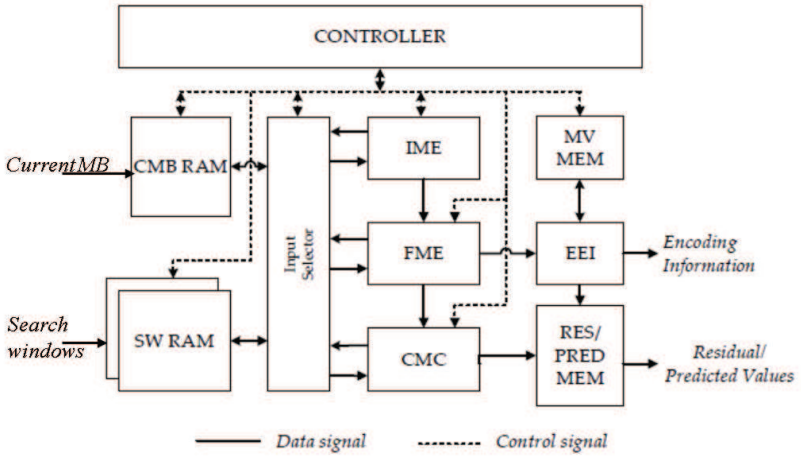

Figure 2.3: The architecture of Inter prediction module [Tran13B].

In the ME, a Full-Search algorithm is chosen because it provides motion vectors more precise than Fast ME algorithms, e.g. Three-Step Search [Chen07F], Diamond Search [Port11A, Sanc12D], or Multi-Resolution Search [Lee04V, Lin08A2, Yin10A]. For the block-size decision, we decided to use only the SAD function for its simplicity and its Silicon resource saving.

Bandwidth optimization methods are used for both on-chip and off-chip communications. Reusing search window data takes advantage of 2/3 data in common between two search windows to save 60% of the off-chip bandwidth. The ME switches between two motion vectors by reading only one column or row of 16 pixels additionally. This also allows calculating all the prediction modes (7 block-sizes) at the same time and giving out one motion vector in just one clock cycle.

Moreover, because the SAD function values of the IME and the FME are not too different, the mode decision placed just after the IME instead of the FME seems a good solution. Indeed, it decreases by 7 times the FME interpolating and calculating time and it enables integrating the MC into the FME submodule.

- **The Transformation and Quantization module**

With the sampling format 4:2:0, each MB contains one 16×16 block of luma component Y and two 8×8 of chroma components Cb and Cr. Each block is then divided into 2×2 or 4×4 to be transformed and quantized. There is one quantization parameter (QP) for one MB.

The overal architecture of the Transform-Quantization and Inverse Transform-Dequantization is illustrated in Figure 2.4. It is constructed using two one-dimension DCT blocks (1-D DCT), one transpose RAM and the quantizer. The input data is processed in the first 1-D DCT (DCT_0), stored in the transpose RAM and processed in the second 1-D DCT (DCT_1). The results after the second 1-D DCT are the post-transform coefficients. They will be scaled and quantized in the quantizer submodule. The memory block DC RAM is used to store the DC coefficients for the Hadamard Transform process. The Hadamard Transform has a similar calculating structure as the DCT. In the design, both transform techniques are combined in just one calculating circuit.

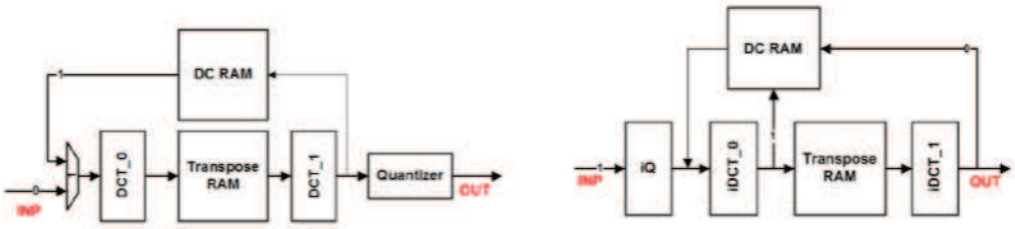

Figure 2.4: Architecture of the Transformation and Quantization module [Tran13B].

- **The Entropy coder and bytestream NAL data packer (EC-NAL):**

The entropy coder (EC) in the VENGME video encoder performs Exp-Golomb and CAVLC coding for all the H.264 video information, i.e., prediction mode, motion vector, residual data, etc. The EC processes data in

syntax elements, i.e., elements of data represented in the bitstream [H.264]. The entropy encoded elements are then packed into the correct syntax specified in the H.264/AVC recommendation so that the decoder can "understand" them. To enable "network friendliness", the data packer will give out the encoded video in the bytestream network abstraction layer (BSNAL) format.

As the EC and the BSNAL data packer (EC-NAL module) design is one of the contributions of this thesis, the design of the module is presented in Section 2.2 in details, with specification, state-of-the-art, architecture, technical improvements and implementation results.

- **The De-blocking Filter (DF)**

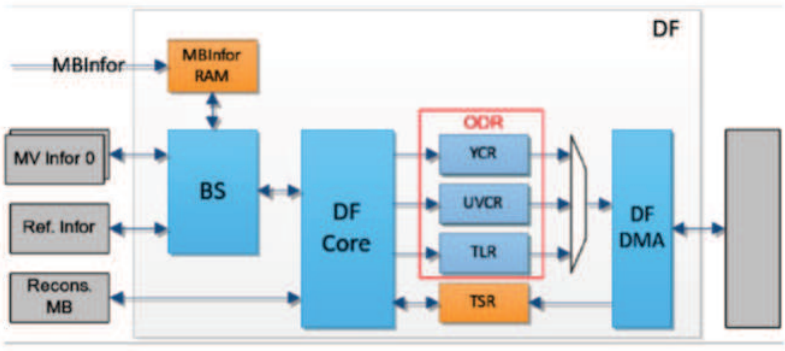

Figure 2.5: The architecture of De-blocking filter module [Tran13B].

The overal architecture of the DF module is presented in Figure 2.5. It consists of five main submodules, namely, the MBinforRAM containing the MB information, the BS calculating the Boundary Strength (BS) parameter, the DF Core performing the filter calculation, the ODR and TSR memory blocks that store the calculated results and the DF DMA which communicates with the outside memory.

2.1.3 Dataflow control

To overcome long coding path and data dependency, a 4-stage pipelining VENGME encoder schedule is proposed, see Figure 2.6. It is different from previous solutions on various aspects. Besides the first stage to load data, similar to [Kim11P], it cuts the coding path into three main stages,

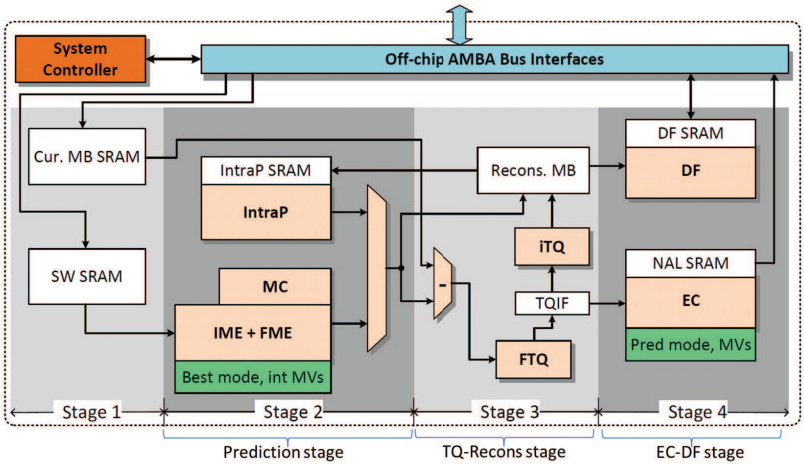

Figure 2.6: VENGME pipelining schedule.

namely Prediction, TQ-Recons. and EC-DF. Both Integer Motion Estimation (IME) and Fractional Motion Estimation (FME) are in the same stage for sharing the IME information and the data in the search window (SW) SRAM. The VENGME schedule takes advantages of a three-stage pipelining architecture that provides memory sharing [Lin08A1], pipeline latency reduction [Chen08A] and power consumption reduction [Chen09A]. This pipeline is even more unbalanced than the 3-stage ones found in the literature [Liu07A, Lin08A1, Chen09A]. However, an extra external memory access bandwidth and its associated power consumption can be saved, while the performance for the targeted applications remains unchanged. Because of the unbalancing schedule, this design is suitable for the implementation of power management techniques. Indeed, the speed of the two last stages can be adapted to the heavy prediction stage. In other words, while waiting for the heavy prediction stage, the two last stages can be placed into low-power mode.

Moreover, Inter Prediction (MC and IME + FME) and Intra Prediction (IP) in the same stage can be executed in parallel or separately, thanks to the system controller decision. In the separate mode, power can be saved via the switch off of the Intra prediction or Inter prediction module while the other one is active. In the parallel mode, the Intra Prediction will finish first, and its results are stored in the TQIF memory which is the memory block placed inside the Transformation and Quantization module. Then, it can be switched off to save power. The Inter Prediction module, including

Motion Compensation (MC), still searches for the "best" predicted pixels. After having inter prediction results, the TQIF memory can be invalidated to store new transformed results for Inter Prediction. The first solution, i.e. the parallel execution of Inter prediction and Intra prediction modules, currently in use, can be seen as a low-power mode for the system.

2.1.4 Verification and implementation results

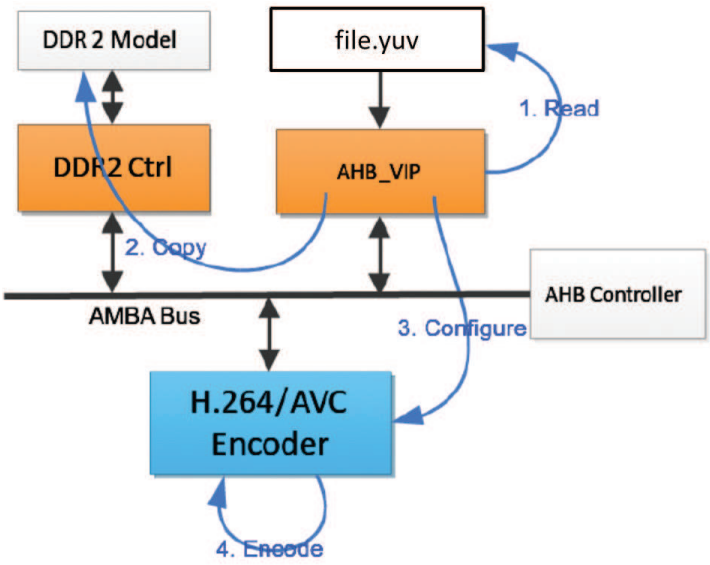

Figure 2.7: The VENGME verification method [Tran13B].

All the modules of the H.264/AVC encoder were modeled in VHDL with some RAM/ROM models in Verilog. It is important to verify and validate the whole integrated system. Figure 2.7 describes the verification method. A CPU or an IP core can control the VENGME encoder by accessing its control registers via an AMBA AHB/APB bus system communication. The memory block DDR2 model is used to store the input video and reference data. The source input videos are the standard reference videos [VidLib]. The block AHB_VIP reads the YUV video file and loads its content to the memory block. The encoded data is then decoded by the reference software decoder JM18 [JM].

It has been proved that the VENGME encoder can encode successfully video data into H.264 format. Figure 2.8 illustrates the frame extracted from the orignal video example (Suzie.yuv [VidLib]) on the left and the video encoded by using VENGME encoder on the right.

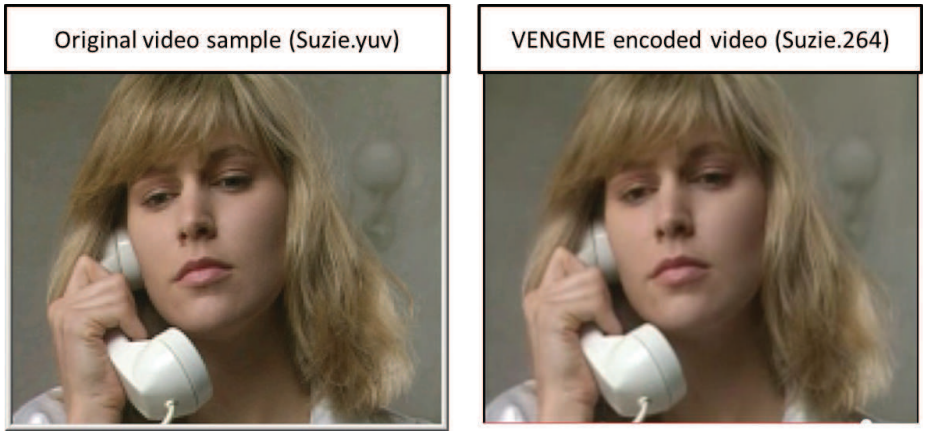

Figure 2.8: Video frames in the original and the VENGME encoded video streams.

Before the fabrication phase, the encoder must be validated to avoid unexpected faults. The System-on-Chip (SoC) for validation is illustrated in Figure 2.9. The Development and Education board (DE2) [DE2] from Altera is used to implement this SoC.

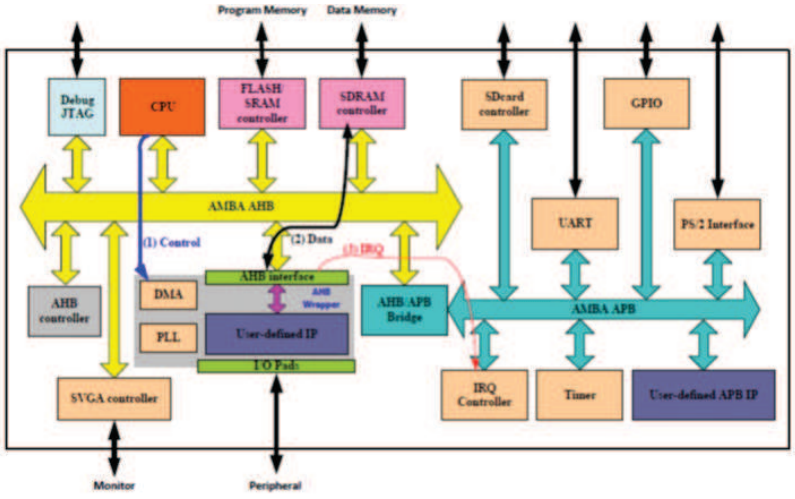

Figure 2.9: System-on-Chip for VENGME validation [Tran13B].

Some VENGME modules present nice performance and/or power figures, equivalent to the state-of-the-art designs [Tran11C, Bui12A, Nguy12A, Dang13A, Nguy13H, Tran13B]. The synthesis is performed in Vietnam by people involved in the VENGME project. Table 2.2 illustrates the VENGME results on the CMOS 130nm technology from Global Foundary

(GF). These synthesis results for the whole the VENGME platform have been obtained using Design Compiler from Synopsys [Syno].

Table 2.2: VENGME implementation results for the 130nm CMOS technology

	Intra prediction	Inter prediction	Transformation Quantization	De-blocking filter	EC-NAL	Memory and control	**VENGME**
Area cost (*Kgate*)	30.448	550.415	90.778	151.314	90.648	**27.953 *Kbytes***	**913.604** (excluding memory)
Power consumption (mW)	1.58	14.77	1.46	2.66	3.72	34.06	**58.25**

The VENGME encoder, targeting CIF resolution, requires a resource of 913.604 *Kgate* for functional blocks and 27.953 *Kbyte* for memory blocks. It consumes 58.25mW, which is suitable for mobile application.

In comparison to the other H.264/AVC hardware video encoders, see Table 1.2, page 39, the VENGME encoder does not have outstanding results. The power consumption and area cost of the VENGME design is lower than most of the designs in state-of-the-art survey, but both numbers are slightly higher than the ones of other designs with the same resolution target and technology. For instance, also targeting CIF resolution and being synthesized with the technology CMOS 130nm, the design in [Chan09A] consumes from 7mW to 25mW; the one in [Lin08A1] consumes 6.74mW. The design in [Chen09A], also targets to CIF resolution, using technology CMOS 180nm, consumes from 9.8mW to 40.3mW depending on the power level. However, the three design in [Lin08A1], [Chan09A] and [Chen09A] target Baseline profile while the VENGME design targets Main profile. This makes the comparison not fully accurate. The design in [Kim11P] also targets CIF resolution but its power consumption is rather high. It consumes from 238.38mW to 359.89mW depending on the power level. In conclusion, the VENGME video encoder results can be considered equivalent to the state-of-the-art designs. Applying some power management methods may help reduce its power consumption. The next section presents power simulation results at RTL level which are early power figures of the VENGME video encoder in looking for a suitable power management method

2.1.5 Power simulation for VENGME platform

We presented the architectural differences between the VENGME H.264/AVC encoder and state-of-the-art implementations. When we started to work on a power optimization solution for the VENGME platform, the platform at hand was modeled in VHDL at RTL level. In order to define a power optimization strategy to be applied onto the VENGME platform, knowing its power characteristic is essential. Power estimation using synthesis tools is usually based on the supposition of average activities of the circuit. To obtain more accurate power estimation results, simulation is required because it provides the real activities of the circuit. However, post-synthesis simulation takes long time. For this reason, we performed power simulations at RTL level, i.e. before synthesis phase, using the ModelSim from Mentor-Graphic [Ment] and the SpyGlass^{TM} Power tool from Atrenta [Atre]. The results provide an early view of the power consumption for the VENGME encoder. Although the comparison with previously H.264/AVC hardware encoders using power simulation at RTL level is provided in this subsection, our main objective is to analyze if a power management method can be usefully applied to the platform. The power simulation at RTL level for the VENGME platform is done at the CEA-LETI, France. In this simulation, we use the CMOS $32nm$ technology from STMicroelectronics [stm].

- **Power simulation results**

The power consumption is estimated at RTL level while encoding five frames (IBPBP) of a video with QCIF resolution, due to the limitation of the simulation space, in CMOS $32nm$ technology from STMicroelectronics. The results are provided in Table 2.3. The (estimated) total power consumption is $19.1mW$. Note that the leakage power at RTL level is not accurately estimated as it highly depends on gate choices. Actually it is over-estimated because the light synthesis performed by the SpyGlass tool cannot provide a netlist well-optimized as the final netlist provided by a synthesis tool. Thus, it can be assumed that this power consumption should be smaller.

- **Comparison with State-of-the-Art H.264 hardware encoder implementations**

Although the main objective of this power simulation is to seek some power management method for the VENGME platform, we try to compare

Table 2.3: Power composition of the VENGME H.264 encoder, obtained using SpyGlass, with 32nm CMOS technology

Power(mW)	**IntraP**	**InterP**	**TQ**	**EC**	**DF**	**SW DMA**	**H.264 encoder**
Total	0.32	7.86	0.82	0.9	0.71	6.39	19.1
Leakage	0.1	3.43	0.52	0.31	0.46	2.88	9.21
Internal	0.13	2.76	0.16	0.26	0.11	0.94	4.56
Switching	0.09	1.68	0.14	0.32	0.15	2.57	5.37

this early power result with the state-of-the-art designs. As mentioned above, this power result is not accurate. Moreover, in the power simulation, we use video with QCIF resolution, which is not targeted by the state-of-the-art designs. Hence, we can only make a very rough comparison.

The total power consumption of the VENGME platform estimated at RTL level using SpyGlass and in CMOS 32nm technology is 19.1mW which is equivalent to other HW encoders that target mobile applications, see some results provided in Table 1.2 (page 39). For instance, the H.264/AVC encoder in [Iwat09A], which aims at low power, video size scalable, consumes 256mW in CMOS 65nm technology. The solution in [Chen09A] consumes 40.3mW for CIF resolution with 2 reference frames and 9.8-15.9mW for CIF resolution with 1 reference frame in CMOS 180nm technology. The low-power H.264 encoder in [Kim11P] consumes from 238.38mW to 359.89mW, depending on the power level. The used technology is not mentioned in [Kim11P]. Note that our real power consumption is certainly smaller because the leakage has been over-estimated.

- **Discussion on power consumption composition and low-power solutions for the VENGME platform**

A power consumption of 19.1mW (in CMOS 32nm technology) makes the VENGME encoder suitable for mobile applications. However, there is still room for power consumption improvement via the implementation of adaptive power management techniques as the activity is highly unbalanced between modules. Of course, this will complicate the design but it is a worthwhile effort with respect to the power gain. The analysis of the relative power results between the different modules suggests a power management strategy could be implemented.

Figure 2.10 depicts the power composition of the VENGME modules. As expected, the Inter prediction module is the most consuming one. The

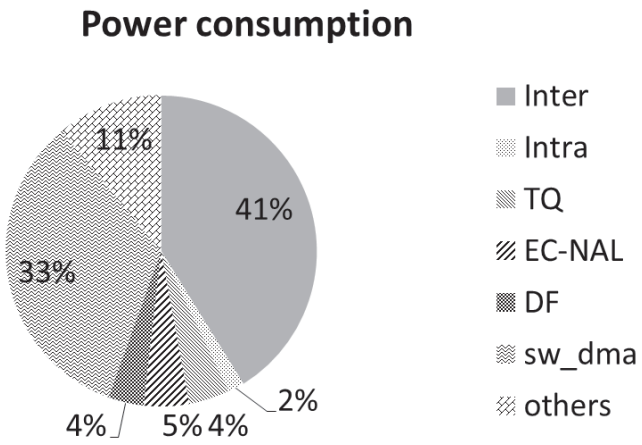

Figure 2.10: Power consumption composition of VENGME system.

second one is the SW_DMA (sw_dma), also related to inter prediction. Because of the complexity of the Inter prediction module, of its huge calculation and memory access requirements, it is reasonable that the two modules have highest power consumption compositions. We can eliminate the power consumption of these modules by using only intra prediction. Actually, the control decision to execute Inter prediction or Intra prediction or both ones in the second VENGME pipelining stage as mentioned in subsection 2.1.4 has been identified as one of the power management solutions.

When we started to work on a power reduction strategy, the VENGME platform was available at RTL level. Moreover, all modules were designed independently but in an optimized way from a power consumption point-of-view. The optimization solutions applied at the design phase, as reviewed in Subsection 1.5.2, namely, parameterization, memory access reduction and data reuse are only appropriate at the module level. Therefore, a low-power technique at platform level seems also suitable. Among the low-power techniques presented in subsection 1.5.1, Dynamic Frequency Scaling (DFS) is a good candidate because it is simple, independent from the technology and can provide power gain because of the highly unbalanced schedule. Moreover, it can be extended later to Dynamic Voltage Frequency Scaling (DVFS) to provide even more power gain.

In order to act on the operating frequency of the VENGME platform, we can firstly exploit the unbalanced schedule among the stages or modules, to gate the clock signal when the stage or the module is unused. Dy-

namic Clock Supply Stop (DCSS) [Moch07A] and fine-grained clock-gating [Chen09A] techniques can be implemented to gate the clock signal. However, the DFS technique can be applied even when the module is in active state. In Chapter 3, we will propose a method to implement a DFS technique. It is a general approach where the frequency is scaled according to the status of the buffer between modules. The method proposed is simple and suitable for the VENGME hardware platform as buffers are used to store temporal data between modules. Moreover, the method proposed in Chapter 3 is independent from the application, i.e. it can be widely applied for any hardware buffer-based system.

In the VENGME power composition, the EC-NAL module is number three in the power consumption ranking of the modules. However, unlike the inter prediction and its SW_DMA, EC-NAL works in every frame. Therefore, reducing the power consumption of the EC-NAL module is also important. Moreover, both the overall VENGME system and the EC-NAL module use memory blocks/FIFOs structure to store the temporal data between blocks. Because the EC-NAL module design is one of our contributions, it is well understandable by the author. Hence, we can focus on a buffer-based DFS power reduction method for the EC-NAL module as an example before applying it onto the whole VENGME platform. The control of this buffer-based DFS power reduction method will be discussed in details in Chapter 3.

The EC-NAL module design is now presented in details.

2.2 Design of the entropy coder and bytestream NAL data packer for H.264/AVC video encoder

The functionalities of the EC-NAL module are to receive information from the previous modules in the encoding path, prediction and transformation-quantization, to encode the data elements and to pack them in the correct BSNAL encoded video syntax. This module is one of our contributions and it is inserted and validated in the VENGME platform. This section presents in details the specification, designed architecture, implementation results for the VENGME EC-NAL module. A comparison with previous works is also provided. Techniques to improve the throughput, area cost

and power consumption in the EC-NAL module during the design phase are presented. The power reduction using buffer-based DFS method will be presented in Chapter 3.

2.2.1 Specification for the entropy coding and data packing in the H.264/AVC standard

The encoded video data is specified by the ITU-T Recommendation [H.264]. We now provide a brief introduction to the specification of the encoded video data and entropy coding.

2.2.1.1 Byte stream encoded video

Figure 2.11 shows the structure of an encoded video stream. Video data at the Video Coding Layer (VCL) consists of slices. The first slices of a video sequence are always Instantaneous Decoder Refresh (IDR) slices forming an IDR frame. Other IDR slices in the stream start new video sequences. Each slice is composed of a Slice Header (SH) and slice data. The SH contains information that supports the slice decoding process. The slice data is a sequence of interchanged macroblocks (MBs) and MBs skipped indication. Each MB contains a MB header and residual data. The MB header gathers the prediction information from the Intra prediction and Inter prediction modules and other information like the Quantization Parameter (QP) and Coded Block Pattern (CBP). Residual data are entropy encoded residual post-quantization coefficients.

Being encapsulated at the Network Abstraction Layer (NAL), the video output data consists of NAL Units (NALUs). Each VCL NALU contains one slice. Non-VCL NALUs are Sequence Parameter Set (SPS), Picture Parameter Set (PPS), End Of Sequence (EOSeq), End Of Stream (EOS), etc. The SPS is a set of parameters to decode one video sequence. The PPS gathers the parameters for pictures, i.e. either frames or fields. At least, there are one SPS and one PPS at the beginning of a video stream. EOSeq, EOS, and many other non-VCL NALUs are optional.

Each NALU starts with one special byte followed by Raw Byte Sequence Payloads (RBSP) bytes [H.264]:

- the first byte is composed of one **forbidden_zero_bit**, two bits representing **nal_ref_idc** and five bits representing **nal_unit_type**. If

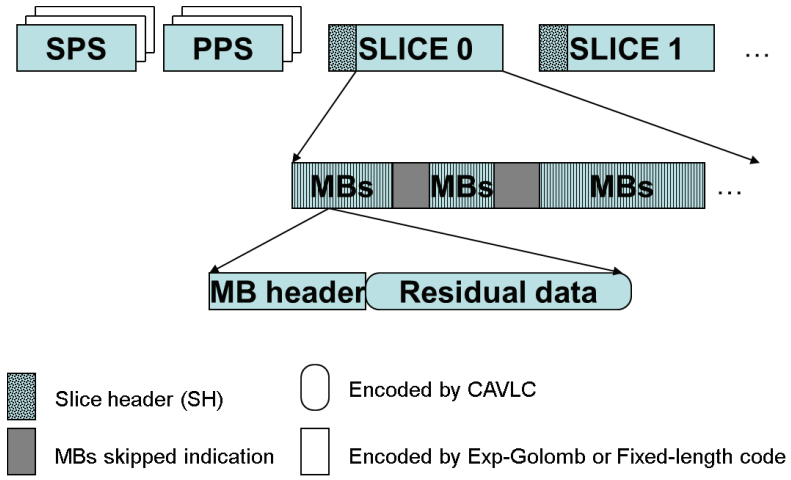

Figure 2.11: Encoded video data structure.

the NALU is referred by other NALUs, e.g. NALUs containing SPSs, PPSs, slices of reference frames, the **nal_ref_idc** value is not equal to zero. Otherwise, the **nal_ref_idc** value will be zero. The values of the **nal_unit_type** are specified in the ITU-T Recommendation;

- in the RBSP bytes, a byte named **emulation_prevention_three_byte**, valued 0x03, might be inserted to prevent the decoder from detecting a start code inside the content of the NALU. Hence, in the bytestream, if three bytes having values in the range from 0x000000 to 0x000003 appear, they will be then represented as four bytes from 0x00000300 to 0x00000303;

- if the NALU content is already byte-aligned, the final byte after the content will be 0x80 which contains one bit '1' called **rbsp_stop_one_bit** and seven bits '0' called **rbsp_alignment_zero_bit**. However, usually, the NALU content is not byte-aligned. The final byte, in this case, contains firstly the remaining bits of the content, then the **rbsp_stop_one_bit**, then the **rbsp_alignment_zero_bit**(s) if required.

The Bytestream NAL (BSNAL) syntax is specified in the Annex B of the ITU-T Recommendation [H.264]. A BSNAL unit may contain the elements as listed in Table 2.4. The first three bytes of a BSNALU, always valued 0x000001, are called **start_code_prefix**. The followed bytes are the contents of the NALU.

Table 2.4: Syntax of a byte stream NAL unit

Element	Leading zero 8bits	Zero byte	Start code prefix	NALU	Trailing zero 8bits
Value	0x00	0x00	0x000001	...	0x00
Quantity	0 or more	1	1	1	0 or more
Condition	Only the first NALU	SPS, PPS, the first slice of frame	All	All	Except the last NALU

In Main profile, Exponential Golomb (Exp-Golomb) coding can be used with Context-based Adaptive Variable Length Coding (CAVLC) or with Context-based Adaptive Binary Arithmetic Coding (CABAC). Because of the VENGME objectives, we implemented only Exp-Golomb coding and CAVLC in our Main profile entropy coding. As shown in Figure 2.11, residual data are encoded using CAVLC and the other data are encoded using Exp-Golomb or Fixed-Length Code (FLC). Both entropy coding techniques will be introduced in the sequel.

2.2.1.2 Zigzag scan and CAVLC

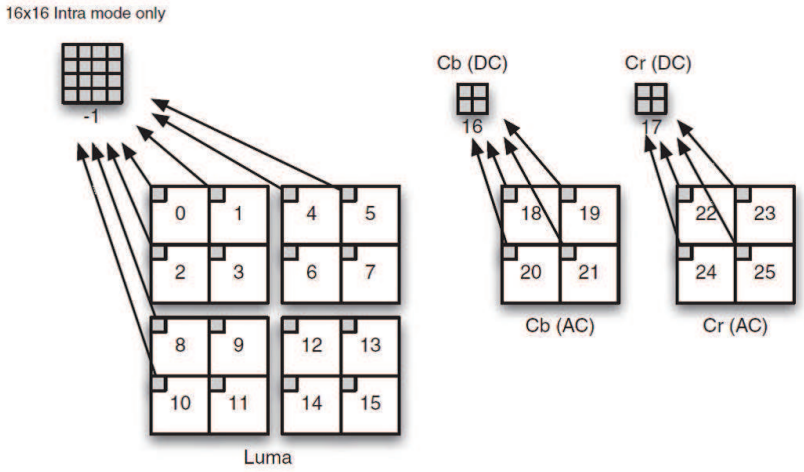

Figure 2.12: Block scanning order in one MB [Rich03H].

Because the Transformation-Quantization engine processes data in blocks of 4×4 or 2×2 coefficients, data entering the CAVLC is also in block format. After the transformation of a block, the DC coefficient is the one having

the lowest frequency, the other coefficients being the AC ones. The input data of each MB contains post-quantization DC luminance (lumaDC, only in $Intra\_16 \times 16$ mode), AC luminance (lumaAC), DC chrominance (chromaDC: Cb(DC) and Cr(DC)) and AC chrominance (chromaAC: Cb(AC) and Cr(AC)) coefficients. The order of the blocks entering CAVLC in MB is the order of the residual syntax [Rich03H], as illustrated in Figure 2.12.

The coefficients of a block are firstly rearranged in zigzag order, see Figure 2.13. For instance, the rearranged coefficients of the block in Figure 2.13 are (15, -9, 6, 1, 0, 0, 2, 0, -1, 0, 0, 0, 0, 0, 0, 0). The coding techniques are applied according to statistical characteristics of the block. The blocks of the quantized transform coefficients mostly contain zero coefficients. Thus, run-length coding is applied to encode the zero strings. In the zigzag order, non-zero coefficients are gradually smaller. The last non-zero coefficients of the blocks are usually $+1$ or -1. The last consecutive coefficients with magnitude 1 are called trailing ones. The maximum number of trailing ones is three. For the block in Figure 2.13, there is only one trailing one, which is the coefficient "-1". These coefficients are encoded in a special way. The other non-zero coefficients are called "level". For the block in Figure 2.13, levels are (15, -9, 6, 1, 2). Levels are encoded in inverse zigzag order because in this inverse order, the magnitude of the previous level can be used to predict the value of the next level. This prediction is then used to select the appropriate coding table. Because the numbers of non-zero coefficients in neighbouring blocks are correlated, the number of non-zero coefficients in upper and left blocks are used to predict the number of non-zero coefficients in the current block [Rich10T].

The ITU-T Recommendation [H.264] has introduced five syntax elements to be encoded in the CAVLC. They are the coefficient token (**coeff_token**), the signs of trailing ones (**T1s**), **level**(s), **total_zeros**, and run(s) of zero

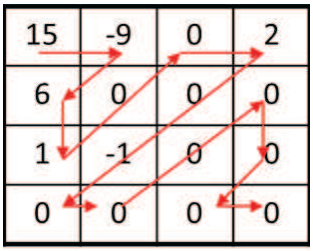

Figure 2.13: Zigzag scan order in one block [Nguy12A].

(**run_before**(s)), where:

- the coefficient token (**coeff_token**) represents a pair of values, namely, the total number of non-zero coefficients and the number of trailing ones in the block. For the block in Figure 2.13, the total number of non-zero coefficients is 6 and the number of trailing ones is 1. The selection of the VLC table depends on the number of non-zero coefficients in the upper and the left blocks. If the number of non-zero coefficients in the upper block is nU and the number in the left block is nL, then, the parameter nC is calculated as follows:

$$nC = ceiling(\frac{nU + nL}{2}) \tag{2.1}$$

  where *ceiling* is the function that maps a real number to the smallest following integer.

  The coeff_token of a 4×4 luma block is encoded using the coding table

$$\begin{cases} \text{VLC0 if } 0 \leq nC < 2 \\ \text{VLC1 if } 2 \leq nC < 4 \\ \text{VLC2 if } 4 \leq nC < 8 \\ \text{Fixed Length Coding (FLC) if } nC \geq 8 \end{cases} \tag{2.2}$$

  We also have a special table for encoding the **coeff_token** of 2×2 chroma DC blocks;

- signs of trailing ones (**T1s**) are from zero to three bits wide. They represent the signs of trailing one coefficients in the reverse zigzag order. If the coefficient is $+1$, the sign bit is zero ('0'). If it is -1, the sign bit is one ('1'). For the block in Figure 2.13, the sign bit is one ('1');

- **total_zeros** is the total number of zero coefficients before the last non-zero coefficient in the zigzag sequence. For the block in Figure 2.13, the **total_zeros** is equal to 3. **total_zeros** is encoded using 15 VLC tables selected by the number of non-zero coefficients in the luma blocks. Three other tables are used for the encoding of the chromaDC blocks;

- run(s) of zero (**run_before**(s)) are the numbers of zero coefficients standing before each non-zero coefficient in the zigzag order. For the

block in Figure 2.13, run(s) of zero (**run_before**(s)) are (0, 0, 0, 0, 2, 1). **run_before** elements are encoded in inverse zigzag order. The VLC table selection is done based on the $ZeroLeft$ information, that is, the number of zeros left after each **run_before** is encoded. The next $ZeroLeft$ is equal to the current $ZeroLeft$ minus the current **run_before**. There are 7 VLC tables used for **run_before** encoding.

- **level**s are values of each non-zero coefficient in the block, except the trailing one cases. They are also encoded in the reverse zigzag order. In the **level** encoding, seven VLC tables are used. The next VLC table is selected according to the current VLC table and the current magnitude of **level**. The first **level** is encoded using Level_VLC0. Then the VLC number is increased if the magnitude of the current **level** is larger than the correlated threshold of the current VLC. Table 2.5 shows the thresholds for the VLC tables.

  One exception is that if there are more than 10 nonzero coefficients and less than three trailing ones, the first **level** will be encoded using table Level_VLC1. Another exception is that if there are less than three trailing ones, the first **level** will be encoded with the magnitude decreased by 1 as mentioned in the Recommendation [H.264].

Table 2.5: Thresholds to increase the VLC number

Current VLC table	Threshold
Level_VLC0	0
Level_VLC1	2
Level_VLC2	3
Level_VLC3	12
Level_VLC4	24
Level_VLC5	48
Level_VLC6	-

The ITU-T recommendation [H.264] introduced the **level** encoding using three variables: *suffix length*, *level prefix* and *level suffix*. The coding tables can be found in the JVT document JVT-C028 [Bjon02C]. From this presentation of VLC tables, we can figure out the format of the codeword in three cases: one general case and two escape code cases. Signed **level**s are converted into unsigned *code-number*s:

- in general case, a codeword contains a *prefix* and a *suffix*. The *prefix* is a string of zero bits followed by one '1'. The *suffix length* is equal to the VLC number. The *suffix* value is equal to *code-number* minus the number of zero bits in the *prefix*. The maximum *prefix* is 13 in the table Level_VLC0. In the Baseline and Main profiles, the maximum *prefix* is 14 in the other VLC tables [Wieg03O];
- if the VLC number is VLC0 and the *code-number* is greater than 13 and smaller than 31, the codeword is in the first escape format: the *prefix* is equal to 14, the *suffix length* is 4;
- if the VLC number is VLC0 and the *code-number* is greater than 30, or in the other VLC tables, the *code-number* is greater than the maximum *prefix* plus maximum *suffix*, the second escape format is applied: the *prefix* is 15; the *suffix length* is 12 in baseline and main profiles.

Each syntax element above is encoded using several different coding tables. The table selection is performed based on the previous encoded information. This gives the "context-based adaptive" feature of the CAVLC coding technique. The syntax structure of the output bitstream for one block data is in the order: **coeff_token**, **T1s**, **level**(s), **total_zeros** and **run_before**(s). A conventional CAVLC encoder is composed of five encoders to generate the five syntax elements of the current block. Each encoder contains several look-up tables to store the VLC tables. The table selection is done by the previous coded syntax elements with some special cases. It requires in total 41 different coding tables in the CAVLC encoder.

2.2.1.3 Exp-Golomb coding

The Exp-Golomb coding encodes each unsigned code number ($codeNum$) to produce a bit string [H.264]. The $codeNum$ is given by the following expression:

$$codeNum = 2^M + N - 1, \text{ where } \begin{cases} M, N \geq 0, \text{ and} \\ 2M > N \end{cases} \tag{2.3}$$

The encoded bit string is then constructed from *prefix* bits and *suffix* bits. *prefix* is a series of M zero bits followed by one bit '1'. The *suffix* is

also M bits long whose value is equal to N. Table 2.6 gives some examples of the code construction principle. Hence, this coding technique is efficient when low-value $codeNum$s occur more frequently than high-value ones.

Table 2.6: Examples of Exp-Golomb coding

$codeNumK$	**Exponential Expression**	**Bit string**
0	$2^0 + 0 - 1$	**1**
1	$2^1 + 0 - 1$	**010**
2	$2^1 + 1 - 1$	**011**
3	$2^2 + 0 - 1$	**00100**
4	$2^2 + 1 - 1$	**00101**
...	**...**	**...**

In the H.264 standard, Exp-Golomb coding is used alternately with Fixed-Length coding (FLC) to represent all the syntax elements in the NALUs. It processes syntax elements individually. Based on the statistical characteristics, each kind of elements is represented by a $codeNum$ in a different way:

- if a syntax element is always larger than or equal to zero and the more frequently occurred values are the lower values, the process applied is called Unsigned Exp-Golomb coding (**ue**). The value of the corresponding $codeNum$ is equal to the value of the unsigned element;

- if a syntax element is signed and the expectation value is zero, the process applied is called Signed Exp-Golomb coding (**se**). The value of the corresponding $codeNum$ is mapped to the syntax element value k as follows:

$$codeNum = \begin{cases} 2|k|, \text{ when } k \leq 0 \\ 2|k| - 1, \text{ when } k > 0 \end{cases} \tag{2.4}$$

- if an unsigned element has a different statistical characteristics from the one of **ue**, its corresponding $codeNum$ is then mapped to its value in a special way as the mapping tables indicated in the standard [H.264]. The process applied is called the Mapped Exp-Golomb coding (**me**);

- if an unsigned element has for largest possible value 1, then the Truncated Exp-Golomb (**te**) process is applied: the bit representing the syntax element is the inversed value of the element.

2.2.2 Analysis of previous works

In the literature, several entropy coding (EC) and data packing architectures for H.264 video encoding have been proposed. Here, as there is no CABAC in the VENGME platform, only the Exp-Golomb and CAVLC coding techniques are discussed for comparison purpose.

Actually, the EC is used in the syntax structure of the SPS, PPS at NAL layer. However, many hardware EC engines only encode the data at MB level, i.e. the MB header and residual [Chen05D, Tsai06L, Huan08H]. In [Chen05D], a coding process for SPS, PPS and Slice header has been implemented in software. [Tsai06L] encoded the slice header in software while the data in SPS and PPS are not mentioned.

Meanwhile, [Lu08V] implemented all the EC to encode SPS, PPS, Slice header and MB in hardware. Hence, [Chen05D, Lu08V] proposed full-EC engines with data packer at NAL layer while other works just present a simple data packer to produce video data at VCL layer [Tsai06L] or even lower [Huan08H].

Many methods, mostly focusing on the CAVLC encoder, have been introduced to enhance some features of the EC. For instance, parallel coding or pipelining increases the processing speed, modified coding tables or calculating circuits can reduce area cost, etc. Now, we review in details these implementations.

2.2.2.1 Implementations related to processing speed

Pipelining architectures are usually used for the CAVLC [Chen06AD, Chen10T] or for the entire EC [Chen05D] to increase the throughput. A two-stage pipeline architecture can halve the time to process a block in average, but it requires to double the buffer size to store all the statistic information of one block before the data is encoded. Therefore, in their CAVLC encoders, Chien [Chien06A] and Tsai [Tsai06L] introduced a mechanism that scans the coefficients in inverse zigzag order. Moreover, they proposed a special buffer structure to maintain the buffer size at the size of one block.

In the CAVLC encoder, parallel symbols encoding is also an efficient method in terms of speed. However, it obviously doubles the area cost for symbol encoders. In addition, this method needs to handle data dependency, that is, in the two symbols encoded in parallel, the encoding of the later symbol depends on the previous one. Thus, the statistical information of both symbols is pre-calculated before the encoding process [Chien06A]. In [Yi08H], the later symbol is encoded in two parallel encoders. Then, the results are selected based on the output of the previous symbol encoder. This solution triples the hardware cost of the symbol encoder.

Ramos et al. [Ramo10A] presented an efficient method to overcome the bottleneck at the scan phase by scanning the coefficients in parallel. This method halves the required time of the scan phase. Parallel coding for level and run-before is also applied.

The coefficients entering the CAVLC encoder can be pre-flagged. Actually, the processing time is reduced when the encoder already knows the characteristics of a data block. In [Chen05D] and [Tsai06L], some flags indicate the zero coefficients. In [Huan08H], the flags are used not only for zero coefficients but also for the signs of non-zero ones.

W. J. Lu et al. [Lu08V] also tried to increase the throughput in data packer while avoiding the stall state when the **emulation_prevention_three_byte** is added to the data stream. However, the stall state occurs very rarely [Chen05D]. As a consequence, this improvement does not gain much.

2.2.2.2 Implementations focusing on area cost

To reduce the area cost, [Lai05A, Chien06A, Rahm07C, Huan08H, Lu08V, Kim10I] try to reduce the memory size. Actually, some register blocks can be implemented instead of RAM [Huan08H] to store the reference information of neighbouring blocks. The regularity calculation method [Huan08H] implemented in nC unit decreases the number of registers in use. Based on the fact that codewords start with zero-bit sequences that are much longer than the representation of the codeword length, an optimized table to encode **coeff_token** is proposed in [Lu08V]. In the CAVLC encoder proposed by Kim et al. [Kim10I], instead of storing a 16-bit word for each symbol in **coeff_token** encoding, the VLC tables are modified into tables of 9-bit words. In some designs [Lai05A, Rahm07C, Huan08H, Lu08V] the level coding process is realized with calculation circuits instead of a Look-Up

Table (LUT). This latest solution reduces as well the area cost.

2.2.2.3 Implementations focusing on power consumption

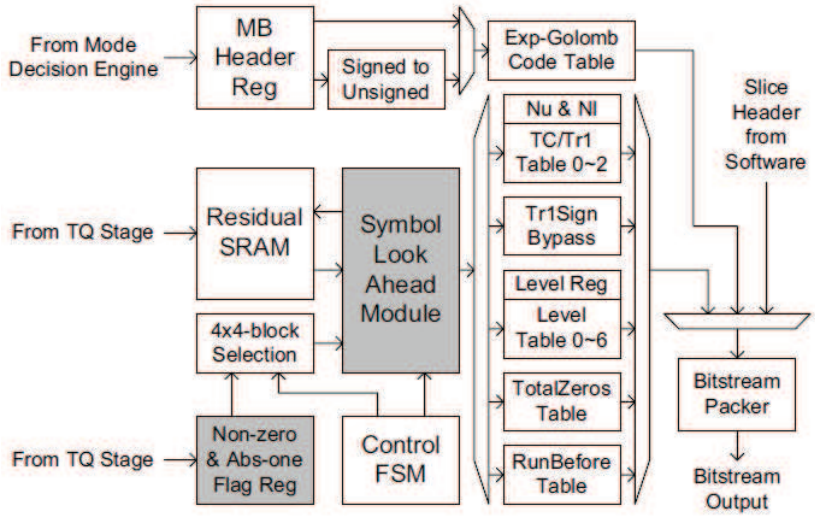

Figure 2.14: Architecture of a low-power Entropy Coder [Tsai06L].

Power consumption reduction is one objective of our work. Thus, now, we discuss power consumption reduction techniques applied in previous EC architectures. C.Y. Tsai et al. [Tsai06L] proposed a low-power EC design that achieves about 69% of power saving in comparison with [Chen05D]. The main features introduced are Side Information Aided (SIA) and Symbol Look Ahead (SLA) to minimize the residual SRAM access, see Figure 2.14. The SIA is composed of non-zero and abs-one flag registers. Each register is a 2-D array of 16×16 flags. Using these two flag registers, the CAVLC encoder can quickly generate the syntax elements to be encoded. With the flags in SIA, the SLA calculates the address of non-zero coefficients to be read. Hence, accessing all coefficients, as usual methods do, is not necessary. The design shows promising entropy coder and data packer power results with $3.7mW$ at $27MHz$ and $1.8\,V$ for CIF video with CMOS $180nm$ technology. However, with these two additional features, the total gate count is higher, that is $26.6Kgates$ in comparison with $23.6Kgates$ in [Chen05D] or $15.86Kgates$ in [Huan08H]. W. J. Lu et al. [Lu08V] also proposed methods to reduce the power consumption. Firstly, the encoding process for SPS, PPS and slice header in HW avoids implementing an

additional processor to encode this data in software. Thus, the power consumption and the HW area of the whole chip are reduced. Secondly, since Exp-Golomb and CAVLC are not working in parallel, the clock is turned off for non-operating engines to consume less energy. However, their power results are not reported. Note that [Huan08H] shows entropy coder and data packer power figures of only $2.5mW$ with CMOS $180nm$ technology, even if it only encodes data at MB level, with a simple data packer.

2.2.2.4 Lessons learned from the previous works

Table 2.7: Comparison of EC designs from the state-of-the-art

	[Chen05D]	[Tsai06L]	[Huan08H]	[Lu08V]	[Chen10T]
Target	High throughput	Low power	Low cost; High throughput	Low cost; low power	CAVLC & Exp-Golomb
Profile	Baseline	Baseline	Baseline	Baseline	N/A
Resolution	CIF 30*fps* to HD1080	CIF	HD1080	QCIF	N/A
Techno. (*nm*)	180 UMC Artisan	180 TSMC Artisan	180 TSMC	180 SMIC	FPGA
Freq. (*MHz*)	100	27	100	200	31.2
Latency (*cycle/MB*)	200-500 depends on QP	N/A	260	1905 in the worst case	N/A
Area (*Kgate*)	23.6	26.598	15.86	9.504	5731 LUTs + 1562REGs
Power	N/A	$3.7mW$	$2.5mW$ ($27MHz$ $1.8V$)	N/A	N/A

Table 2.7 compares the main features of the designs discussed here. Targeting high throughput, both designs [Chen05D] and [Huan08H] have promising latency. Operating at $100MHz$, the latency of $260cycles/MB$ [Huan08H] enables processing 47.5 *HD-frame/s*, which is suitable for real-time applications. However, at the low-power operating frequency of $27MHz$, this design can only encode 12.8 *HD-frame/s*.

In the two low cost designs [Huan08H] and [Lu08V], the latest one has better area result with $9.5Kgates$ compared to $15.86Kgates$ in [Huan08H]. With a slight improvement in the design techniques, the smaller area of [Lu08V] can be explained by its smaller memory blocks for smaller targeted resolution (i.e. QCIF (171×144)) while [Huan08H] deals with HD1080 (1920×1080).

The techniques implemented to reduce the power consumption in [Tsai06L] lead to the highest area cost ($26.6Kgates$). On the other hand, the design in [Lu08V], also targeting low-power, has the smallest area cost, but its latency of 1905 *cycles/MB* in the worst case is high, and its resolution is also smaller. To obtain a small power consumption, [Lu08V] would operate at a frequency lower than $200MHz$. $27MHz$ might be the frequency suitable for low-power operation [Tsai06L, Huan08H].

In our EC-NAL module, we will implement the encoding process for SPS, PPS and slice header in hardware to reduce the tasks of the global H.264 encoder. For the same reason, we built the table selector for **coeff_token** and its reference memory inside the EC-NAL module. This will increase the area cost and tasks of the EC-NAL module. However, we also use pipelining, zero-skipping, arithmetic calculating circuits and VLC tables re-encoding and several techniques to achieve high performance, low area cost and low power consumption [Nguy12A, Nguy13H].

2.3 The Entropy Coder and data packer (EC-NAL) module design for the VENGME platform

This section provides an implementation description of the EC-NAL module for the VENGME platform. The module is designed based on the specification and in regard to the state-of-the-art designs that have been analyzed in section 2.2. We first introduce the overall architecture of the EC-NAL module. Then, the main engines, namely the CAVLC encoder, the Exp-Golomb encoder and the bytestream NAL data packer, will be presented in details.

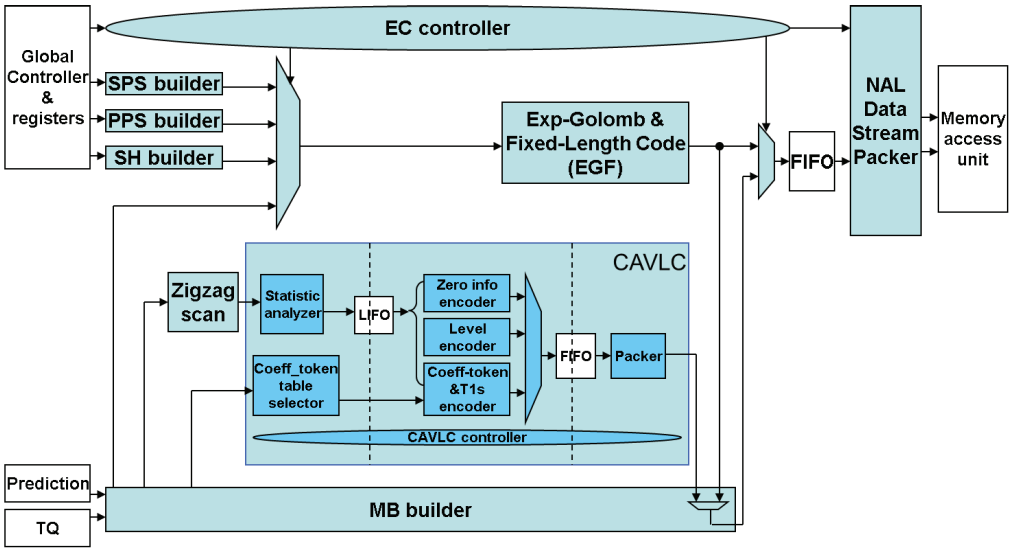

Figure 2.15: Entropy coder and data packer architecture overview.

2.3.1 EC-NAL architecture and main engines

Figure 2.15 shows an overview of the architecture of the entropy coding and data packing engine we proposed. Encoded syntax elements are transferred from the EC to the NAL data packing via the NAL FIFO.

The entropy coding engine consists in two encoders, namely, the Exp-Golomb and Fixed-length code (EGF) engine and the CAVLC engine. SPS, PPS, SH, and MB builders are implemented to collect the "to-be-encoded" data and send it to the EGF module in the specified order.

SPS, PPS and SH information is provided by a global controller via some system registers. The MB header information is received from the Intra prediction, the Inter prediction and the TQ engines. Moreover, the MB builder also sends the residual coefficients from TQ and the related information to the zigzag scanner to encode them by the CAVLC engine.

The NAL data packer engine concatenates the encoded syntax elements into the NALUs.

The role of the EC controller is to synchronize the builders and the NAL data packing engine to generate a video stream in NALUs. The Finite State Machine (FSM) of the EC controller is depicted in Figure 2.16. Before packing and generating the first video slice, a SPS NALU and then a PPS NALU are packed. Each video slice starts with a slice header. The IDR slice starts with an IDR slice header due to the differences in syntax. The final

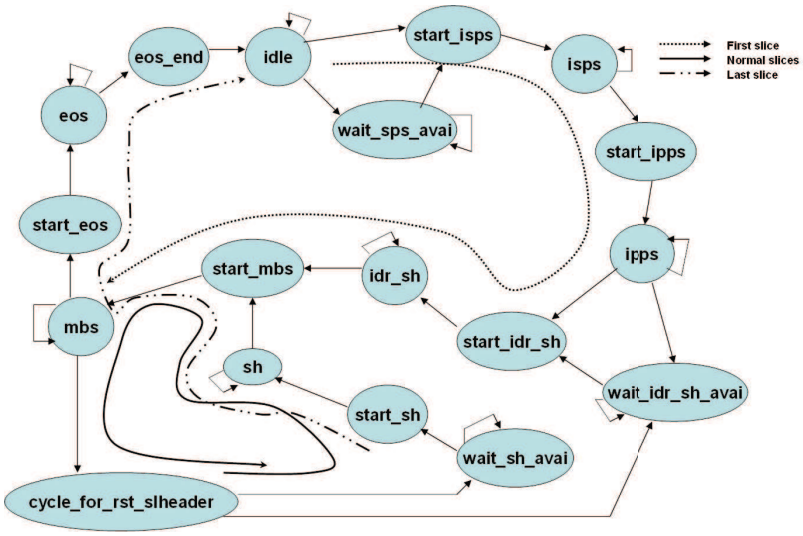

Figure 2.16: Finite state machine of the the entropy coder.

video slice of the stream is followed by an End Of Stream (EOS) NALU.

The main blocks of the design are introduced in details hereafter.

2.3.2 CAVLC encoder

The CAVLC engine is used to encode the residual data of MBs produced by the Transformation-Quantization engine.

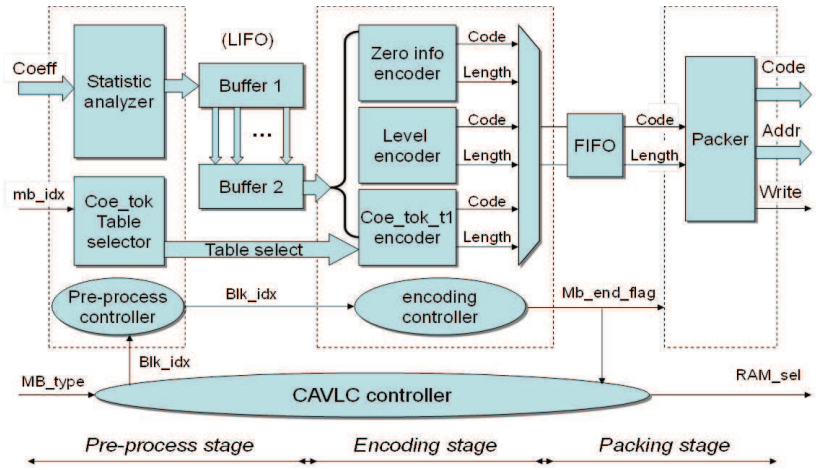

Figure 2.17: Three-stage pipeline architecture of the VENGME CAVLC encoder.

A zigzag scan engine is placed before the CAVLC encoder to re-order

the coefficients in blocks from raster scan order into zigzag order. In the CAVLC encoder, pipelining, zero-skipping, arithmetic calculating circuits and VLC tables re-encoding techniques are used to achieve high performance [Nguy12A]. The three-stage pipelining architecture of our CAVLC engine is provided in Figure 2.17.

The pre-process stage scans the sixteen (resp. four) coefficients of the 4×4 (resp. 2×2) block. Then, it analyzes statistical characteristics of the input block. In this stage, a **coeff_token** table selector (Coe_tok Table selector) is also integrated. It stores the reference information of the neighbouring blocks and calculates the parameter nC to select the coding table for the **coeff_token** encoding. In the encoding stage, various encoders work in parallel to encode the syntax elements of the current block at the same time. Three encoders (coe_tok_t1, level and zero info encoders) generate codewords on a 32-bit bus and the associated code length on a 5-bit bus. The encoding controller (CAVLC controller) synchronizes the three encoders and controls the output data flow according to the syntax structure of the output bitstream imposed by the standard. Finally, the packing stage concatenates the codewords and aligns them into 32-bit words. At first, the CAVLC encoder was designed so that the output words are to be written into an outer memory block. Thus, when it is later implemented in the EC-NAL module, the output data from the CAVLC encoder is already aligned in the form of 32-bit words.

The pipelining mechanism between the pre-process stage and encoding stages is performed at block level. Due to data dependency, the encoders start to encode a block after its scanning process is completed. Two buffers are implemented between the two first stages. By doing this, two blocks can be processed in parallel, one being scanned while the other one is encoded. This choice increases the throughput of the CAVLC encoder. The pipelining mechanism between the two last stages is realized at codeword level. Each codeword generated by the encoder is pushed into the FIFO. Whenever there is a codeword in the FIFO, the packing stage pops it and concatenates it with the previous codes. By doing this, the codewords can be processed in parallel; one is being encoded while the previous one is being packed. The packing stage will stop packing and write out the rest of data of a macroblock into the outer memory when the mb_end_flag is raised, see Figure 2.17. Hence, the architecture of the packing stage is simple.

We also implemented five techniques to improve the throughput and re-

duce the area cost and power consumption of the CAVLC encoder [Nguy12A]. These techniques are now described.

1. *Reduce the number of codewords entering the packing stage to increase the throughput*

   The pre-processing stage scans the input coefficients and stores the statistic information in the first buffer. After the scanning is complete, all the information such as the number of non-zero coefficients, the number of trailing ones, **level**s, **run_before**s..., are copied into the second buffer. During this copy process, the signs of the three first levels are extracted and written into the second buffer (see Figure 2.18). Using the number of trailing ones, the encoding stage can easily construct the syntax element trailing one sign flag (**T1s**) in only one clock cycle.

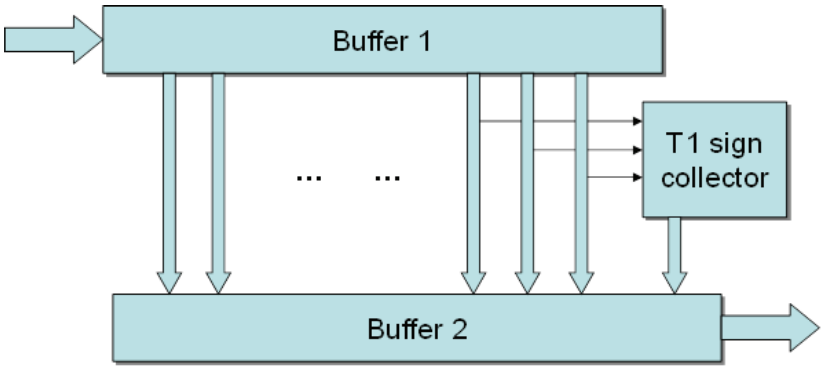

Figure 2.18: Trailing one signs (**T1s**) extraction during buffer copy.

In the encoding stage, the **coeff_token** and **T1s** are 19 bits long at maximum. In a clock cycle, the two syntax elements are merged into only one codeword pushed into the FIFO to reduce the task of the packing stage. This increases the throughput of the CAVLC encoder. Figure 2.19 presents the encoding of the **coeff_token** and the merge of two syntax elements into one (Coe_tok_t1).

In a block, the number of non-zero coefficients and the number of **run_before**s are equivalent. Although **run_before**s and **level**s are encoded simultaneously, the time for the encoding stage to encode and push all the syntax elements into the FIFO still doubles the time for encoding **level**s. Besides, the total number of bits of the **total_zeros**

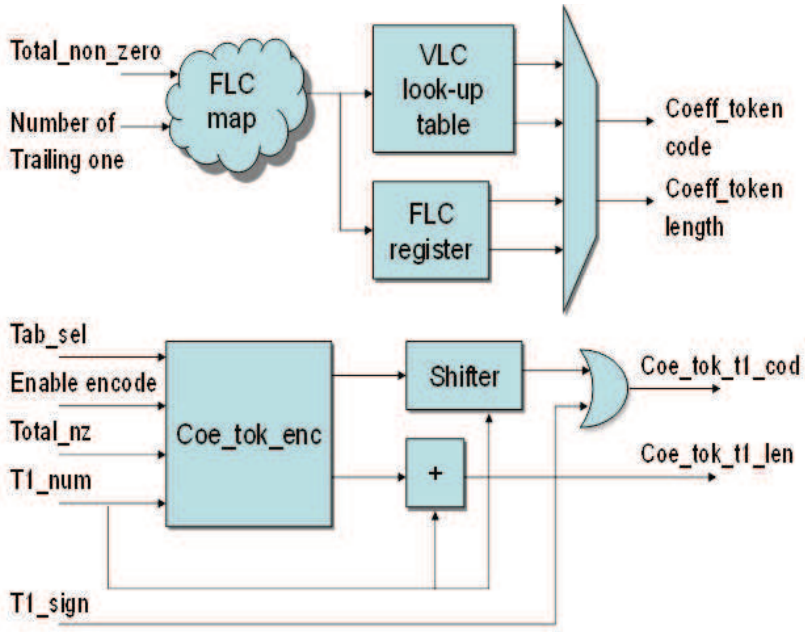

Figure 2.19: **Coeff_token** and **T1** signs encoded into one codeword.

and **run_before**s is less than 32 bits. Thus, we can solve the timing process problem by packing all the zero information of a block into a 32-bit codeword while the **run_before**s are being encoded. With this solution, we reduce the number of zero information codewords entering the packing stage to just one codeword. Figure 2.20 describes the position and architecture of the zero information packer (Zer_info_pkg) in the zero information encoder.

2. *Zero-skipping at block level to increase throughput and to reduce power consumption*

   Another method to enhance the performance of the CAVLC encoder is zero-skipping where all residual zeros are flagged in order to skip the complete encoding process. Chen [Chen06AD] and Chien [Chien06A] adopt zero skipping at 8×8-block level, using Coded Block Pattern. Then, in his later work [Tsai06L], Chen has applied zero skipping at coefficient level to achieve higher throughput and lower power consumption for CAVLC encoding.

   In our design, zero skipping is also used at 8×8-block level. Before being scanned in the pre-processing stage, a 8×8-zero block is detected in the zigzag scan phase. If the flag of a 8×8 zero block is

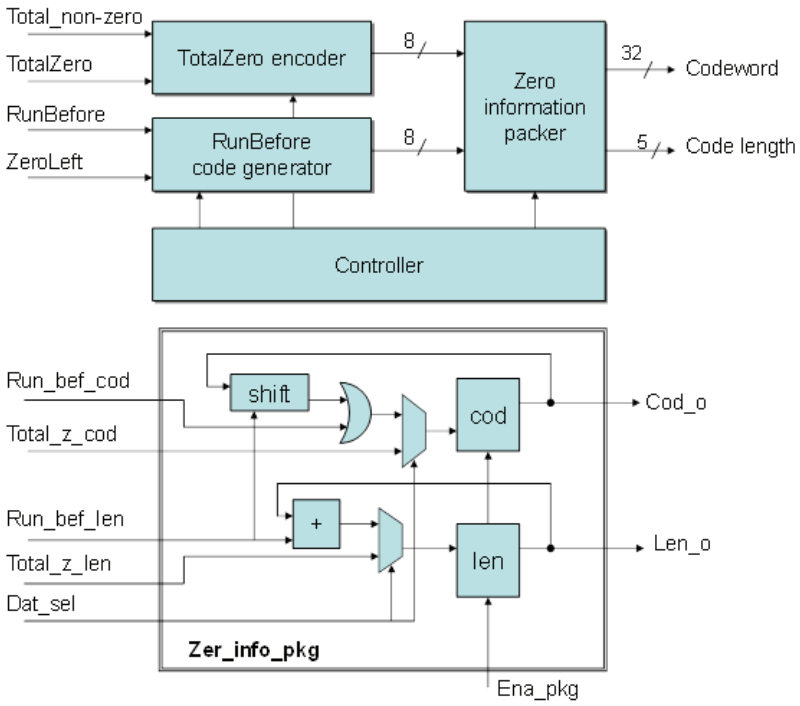

Figure 2.20: Position and architecture of the zero information packer (Zer_info_pkg) in the zero information encoder.

raised, the encoder needs only two clock cycles to store the reference information into the reference memory and all the other encoding process is omitted. At 4×4-block level, zero blocks are also detected and flagged by the zigzag scanner to reduce the tasks of the CAVLC encoder. When a 4×4-block is flagged as a zero one, the pre-processing stage does not have to re-scan it and the **coeff_token** encoder is the only encoder operating in the encoding stage. Reducing scan task of the pre-processing stage and omitting partly or entirely the encoding process in zero-block cases also reduces the power consumption of the two first stages.

3. *VLC table selector for **coeff_token** encoding to increase global throughput and methods to reduce local area cost*

   As mentioned above, the **coeff_token** VLC table selector including the reference memory is implemented in the pre-processing stage. Without this integration, the global processor of the H.264 encoder would have to access the global memory three times per block to read

the nL, nU and write back the number of non-zero coefficients of the current block. Therefore, with the VLC selector integrated in the CAVLC, the global performance of the H.264 encoder is increased.

However, the hardware cost of this table selector is very high. The work presented in [Kim06I] is known as the only design including the nC generator in the CAVLC encoder. The nC generator is reported to occupy more than 50% of the total hardware of the CAVLC encoder.

The significant cost of this table selector leads to optimize the hardware cost of the design. For the blocks whose lower neighbors are in the same macroblock, the reference memory is reused after each macroblock. For the other blocks, the reference memory is reused in a line-by-line way. The data stored in the reference memory is the number of non-zero coefficients in blocks, which is in the range from 0 to 16. According to the CAVLC principle mentioned above, nL (or nU) equal to 16 and 15 yield different nC parameters. However, the VLC table decisions are the same, the FLC table is selected in both cases. To reduce the number of bits stored in the reference memory from five to four bits, we store all data equal to 16 as 15 with no influence on the table selection. The reuse of the reference memory reduces the memory block size and it leads to lower power consumption for the memory.

4. *Codeword arithmetic calculation to reduce area cost and power consumption*

   All the logic/arithmetical relations between the input data and the output codeword are used. In the **coeff_token** encoding, the fixed length coding (FLC) table can be easily constructed by forming the codewords as follows: 4 bits representing the number of non-zero coefficients minus one followed by two bits indicating the number of trailing ones. Hence, we can remove the FLC look-up table. To reduce the Silicon resource for implementing an address generator, the calculated FLC codeword is then used as the address to access the VLC tables.

   In the **level** encoding, the arithmetical relation between the $code-number(cod-num)$ and the codeword is also exploited. The architecture of the **level** encoder is presented in Figure 2.21. The lev_para_cal

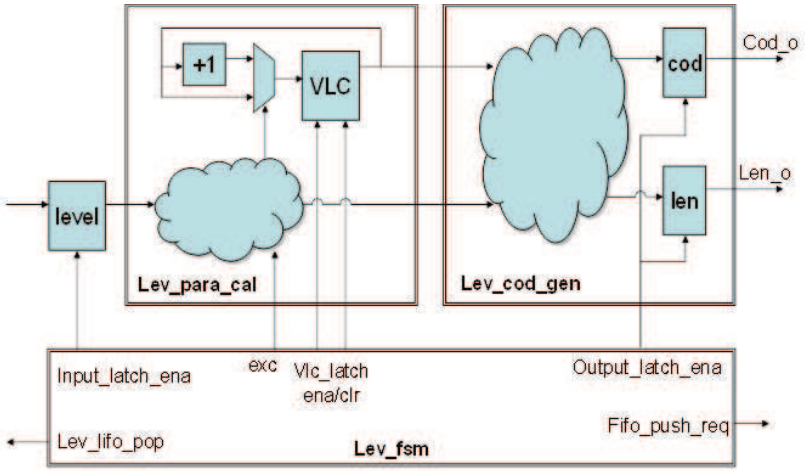

Figure 2.21: Architecture of the **level** encoder.

calculates two parameters, namely, the VLC number and the code-number for each level. The lev_cod_gen generates the codeword and the code length in one of three formats presented above. All the computations are arithmetic and logical. We even optimized some of the mathematical equations to obtain a circuit with minimum resource.

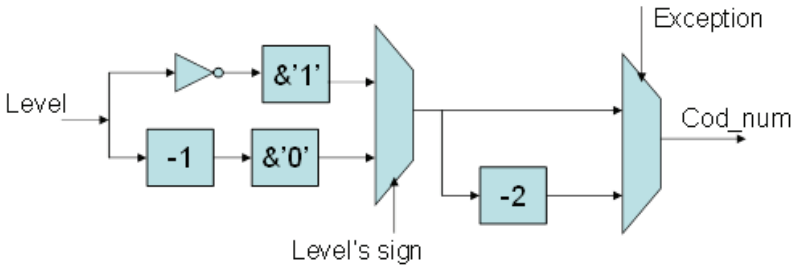

Figure 2.22: $Code-number$ arithmetic expression with optimization.

For example, the circuit in Figure 2.22 calculates the parameter $code-number$. The original relation between the $code-number(cod-num)$ and the **level** is:

$$cod\_num = \begin{cases} |level| \times 2 - 2 \text{ if } level > 0 \text{ and no exception} \\ |level| \times 2 - 1 \text{ if } level < 0 \text{ and no exception} \\ (|level| - 1) \times 2 - 2 \text{ if } level > 0 \text{ and exception} \\ (|level| - 1) \times 2 - 1 \text{ if } level < 0 \text{ and exception} \end{cases} \tag{2.5}$$

The relation is simplified as follows:

- If $level > 0$ and no exception, $cod\_num = level - 1$ concatenates with '0';
- If $level < 0$ and no exception, then $cod\_num = (-level) \times 2 - 1 = (inverse(level) + 1) \times 2 - 2 + 1 = (inverse(level)) \times 2 + 1$. This means that $cod\_num = inverse(level)$ concatenates with '1'.

In **run_before** encoding, the codewords are calculated to remove the look-up table for VLC in the design. Reducing the need for a look-up table implementation leads to power consumption reduction for the **coeff_token**, **level** and **run_before** encoders.

5. *Re-encoding the VLC tables to reduce area cost and power consumption*

   The syntax elements **coeff_token** and **total_zeros** are encoded using look-up tables. However, conventional look-up tables require a large memory size to store the whole codewords and code length. We proposed a simple method to re-encode the **coeff_token** codeword into the format of length and value information as shown in Table 2.8. Because the **coeff_token** codewords have length in the range from 1 to 16 and their values are in the range from 0 to 15, a 8-bit word is enough to store the information of one **coeff_token**. The **total_zeros** coding table is also re-encoded with the same method.

Table 2.8: An example of a 8-bit word in VLC tables

Original codeword	**Proposed codeword**	
	Length - 1	Value
0000000000000010	1111	0010
16 bits	4 bits	4 bits

2.3.3 Exp-Golomb encoder

Due to the exponential expression of the $codeNum$, the challenge of implementing Exp-Golomb encoder in hardware is logarithm operation to calculate M and then the code length ($M = floor(log_2(codeNum + 1))$). To solve this problem, some designers implemented a coding table in their Exp-Golomb [Chen05D, Tsai06L]. Another solution is presented in [Di03A] and then applied in several designs [Huan08H, Lu08V, Chen10T]. This second

solution is based on the fact that the code value is equal to $codeNum + 1$ and the code word is $2M + 1$ bits long, where M is the order of the first one bit in the binary expression of the code value. By this way, output code words are calculated using combinational circuits.

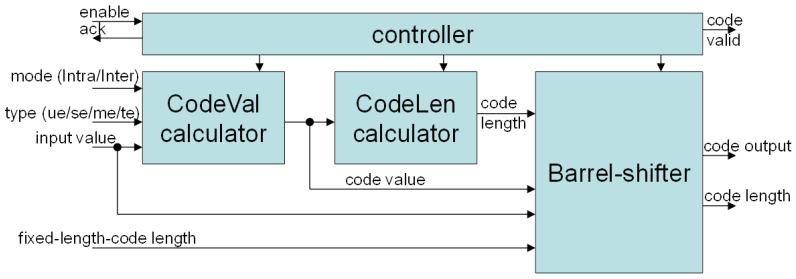

Figure 2.23: Architecture of the Exp-Golomb and Fixed-length coder (EGF).

The architecture of our Exp-Golomb and Fixed-length coder (EGF) engine (see Figure 2.23) also uses this second solution. The input value is processed up to its type (**ue**, **se**, **me** or **te**) in the CodeVal calculator sub-block, to calculate the code value. Then, the order of the first '1' bit in the code value is detected in the CodeLen calculator sub-block. The code length for fixed-length code is given by another input port. The code length and code value are passed through the barrel-shifter sub-block to provide the code output.

2.3.4 BSNAL data packer

According to the specification, we proposed the syntax of the byte stream NALU as illustrated in Figure 2.24. The data packing engine functionalities are:

- generate the first word with **start_code_prefix**;
- select the first byte with corresponding **nal_unit_type** and **nal_ref_idc**;
- insert the **emulation_prevention_three_byte**, if required;
- add final byte with **rbsp_stop_one_bit** then one or more **trailing_zero_8bits** to 32-bit align the output data.

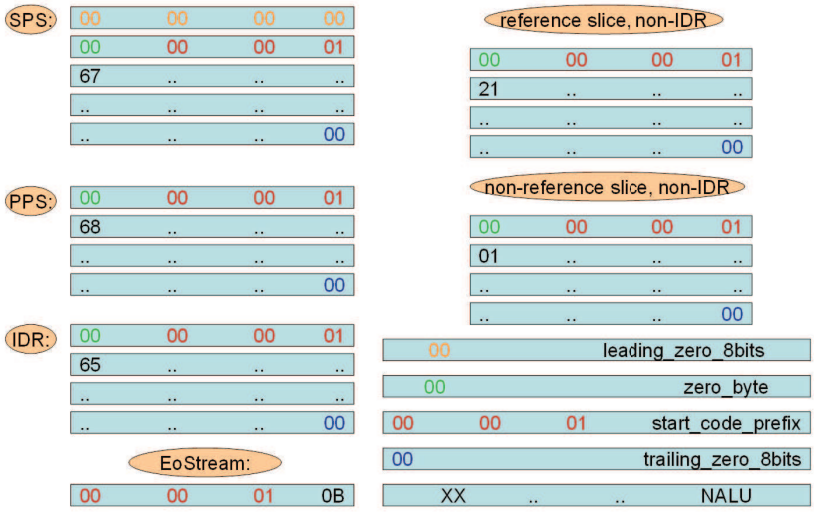

Figure 2.24: Proposed syntax of BSNALU.

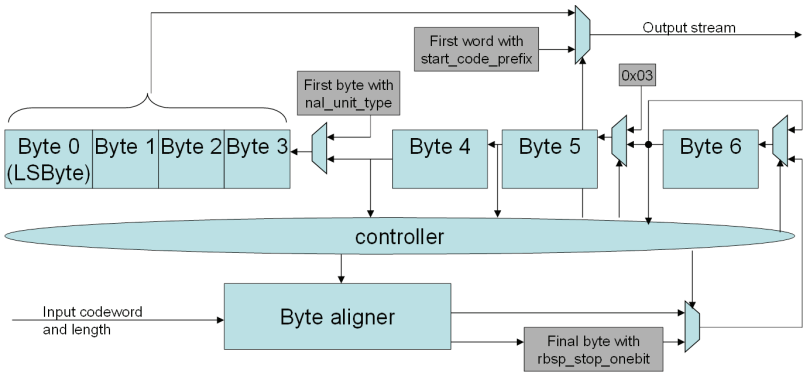

Figure 2.25: Architecture of the NAL packing engine.

Figure 2.25 depicts the proposed architecture for the NAL data packing engine. Because the encoded syntax elements from the FIFO can be at any length from 1 to 64 bits, the first step for packing data is byte aligning the current data and the newly concatenated element. The byte-aligned data is then checked if three continuous bytes emulate the specific code. One byte **emulation_prevention_three_byte** will be inserted to prevent this emulation. The data is then 32-bit aligned and given out at the output stream.

2.4 The VENGME EC-NAL module implementation results and comparison

This section presents the verification and synthesis results of the VENGME EC-NAL module that has been presented in section 2.3. The CAVLC encoder, the largest engine of the Entropy Coder (EC) submodule, was designed and implemented firstly. Hence, the synthesis results of the CAVLC encoder are obtained earlier, in CEA-LETI, using the technology CMOS $65nm$ from ST Microelectronics. These results are presented in subsection 2.4.1. Later on, the entire EC-NAL module is synthesized, in the Vietnam side, using the technology CMOS $180nm$ from AMS. These results are presented in the 2.4.2.

2.4.1 Results for CAVLC encoder

The CAVLC encoder architecture has been modeled in VHDL at RTL level and firstly, synthesized, in CEA-LETI, using low power CMOS $65nm$ technology from STMicorelectronics and DC Compiler Topographical from Synopsys. The verification is done using test data as presented in [Rich10T]. Figure 2.26 presents the relationship between the power consumption (in mW) and the area cost (in μm^2) versus the (synthesized) operating frequency of the design. The design is able to achieve a maximum frequency of about $715MHz$ in the worst case corner (worst-case process, $1.1\,V$, 105°C).

At $550MHz$, the CAVLC encoder consumes approximately $20.7mW$ including $2.4mW$ for the clock tree. It occupies approximately $32.6Kgates$. Due to the data dependency, the processing time per macroblock exhibits a large variability. For a high quality test case, in average, it takes around 450 clock cycles to encode a macroblock. In the case all the coefficients are non-zero, it takes at maximum 540 cycles. In the low-bit rate test case, the number of clock cycles is about 100 cycles. The minimum operating time to process all the zero macroblocks is 52 cycles.

The proposed CAVLC encoder initially targets the CIF video format (resolution: 352×288; frame rate: $30fps$; colour encoded using YCbCr 4:2:0) which is widely used for video teleconferencing. Moreover, at the operating frequency of 550MHz, our CAVLC design can process at least 1019000 macroblocks ($\tilde{2}573$ CIF frames, 126 HD1080p video frames) per

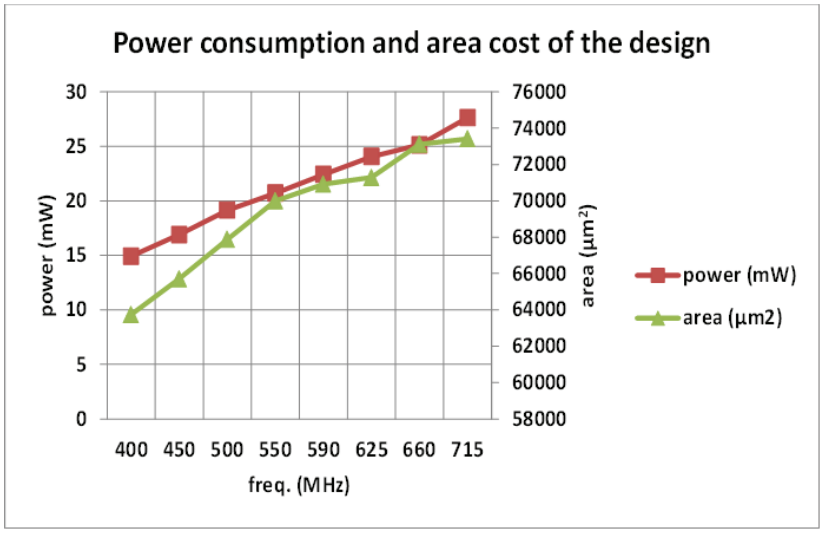

Figure 2.26: Power consumption and area at targeted frequencies of CAVLC implementation.

second. Thus, our design is obviously suitable for real-time applications with HD1080p video format.

In the state-of-the-art designs, some features are not integrated in the CAVLC encoder. Hence, it is not fair to compare only the total gate counts. In Table 2.9, the hardware cost is given for individual parts, to provide a fair comparison.

Only the CAVLC encoder of D. Kim [Kim06I] integrates the **coeff_token** table selector (nC_gen) inside the pre-processing phase. In Table 2.9, the total cost of scanning and **coeff_token** table selector (nC_gen) [Kim06I] and our pre-processing stage (pre-proc) are mostly the same, while the others report only the cost of statistic buffers. Re-encode LUTs and codeword calculation are applied in the proposed encoding stage. Thus, our encoding stage has a fairly low cost, 3611 *gates* compared to 7389 *gates* of [Tian08I] with the same $65nm$ technology.

The packing stage is omitted in some CAVLC designs. In Table 2.9, our packing stage has an equivalent area cost compared to the others. However, the packing stage in [Kim06I] seems to be costly compared to the one we proposed and the one in [Chen06AD]. The reason is that in the design proposed in [Chen06AD], the codewords enter the packing stage in syntax elements order while in [Kim06I] the packing has to classify and order the codewords.

Table 2.9: Hardware cost comparison

Design	Implementation cost (*gates*)					Techno.	Target
	Pre-proc	Encoding	FIFO	Packing	Total		
Ours at 550*MHz*	16734	3611	6248	5747	32636	65*nm*	CIF @30fps 4:2:0
[Chen06AD]	12283 (scan buffer)	5352	N/A	4796	22611	180*nm*	720p 4:2:0
[Chien06A]	5325 (scan buffer)	2614	N/A	N/A	9724	180*nm*	HD 1080p
[Kim10I]	5279 (scan buffer)	N/A	N/A	N/A	12276	180*nm*	N/A
[Kim06I]	17656 (scan + nC_gen)	4472	N/A	9265	31393	FPGA	HD 1080p
[Rahm07C]	N/A	N/A	N/A	-	6850	FPGA	CIF
[Han09A]	-	7389	-	-	-	65*nm*	HD 1080p

Table 2.10: Comparison of CAVLC results

Design	Cycles/MB	Average MBs/sec	Freq. *MHz*	Techno.	Power *mW*
Ours	540-52	5798×10^3	550	65*nm*	20.7
[Lai05A]	N/A	N/A	66	TSMC 350*nm*	21.8
[Chen06AD]	500-200	350×10^3	100	180*nm*	12.0
[Tsai06L]	350-100	174×10^3	27	180*nm*	3.7
[Chien06A]	413-166 300	417×10^3	125	180*nm*	N/A
[Kim06I]	432	231×10^3	100	FPGA	N/A
[Ramo10A]	244	738×10^3	180	FPGA	N/A

Table 2.10 presents other synthesis results for different CAVLC encoders. Zero skipping at block level can reduce the number of cycles per MB only in low-bit rate video. Zero skipping at coefficient level [Tsai06L] and parallel scanning [Ramo10A] break the bottleneck at the scanning phase, hence achieving outstanding throughput. The power consumption of [Tsai06L] is very promising. However, to enhance the throughput, their design uses fewer buffers than the others but this causes longer critical path. Thus, their operating frequency should be low which reduces the performance of the platform. [Tsai06L] also adds non-zero and abs-one flags and SLA which cost totally 14717 *gates*. The two features improve the performance. However they increase the total gate counts to 26598 *gates* (without table selector and residual SRAM).

To summarize, and as presented in Table 2.9 and Table 2.10, our CAVLC encoder has better performances in comparison to other designs while the area cost is slightly higher because the encoder contains a table selector for **coeff_token**.

2.4.2 Results for EC-NAL module

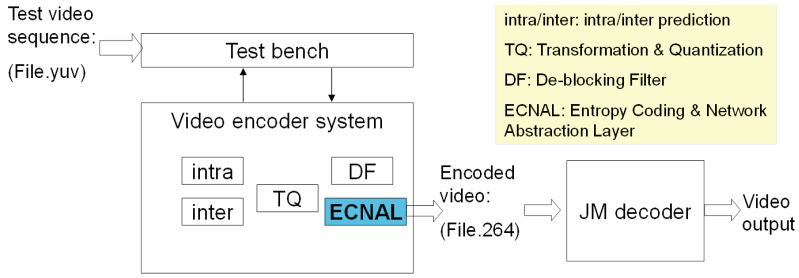

Figure 2.27: Verification method for EC.

Our EC-NAL architecture has been modeled in VHDL at RTL level. For verification purpose, the module has been integrated into the VENGME hardware H.264 video encoding system. Figure 2.27 depicts the verification method. Using the video encoder system with the proposed EC-NAL module, we encode raw test video sequences in the targeted CIF format. The encoded videos are received from the output of the data packing (NAL) submodule. The encoded videos are then decoded successfully by JM decoder, and compared to the initial video.

The simulation is done using ModelSim from Mentor Graphics and the design is synthesized using Design Compiler (DC) from Synopsys. This synthesis is performed in Vietnam when all the modules are integrated to the VENGME platform. At that time, our ASIC library at hand is the CMOS 180nm technology from AMS and the testing frequency is 100MHz.

Some implementation results of our Entropy Coder and byte-stream data packing (EC-NAL) module are presented in Table 2.11.

Table 2.11: Implemantation results of the EC module

Technology	**Cycles/MB**	**Frequency (MHz)**	**Area cost ($Kgates$)**	**Power(mW)**
AMS $0.18\mu m$	691 in the worst case	100	73.5	1.56

Due to the implementation of the table selector for the **coeff_token** syntax element and the full hardware implementation of the EC, including SPS, PPS, slice header data generators, our design occupies a larger silicon area than the others, approximately 73.5$Kgates$. However, in terms of throughput, our EC engine encodes an MB in maximum 691 cycles (151 cycles for the worst case of the MB header plus 540 cycles for the worst case of the residual data [Nguy12A, Nguy13H]). With this speed and at the operating frequency of 100MHz, the design is suitable for 720HD video format. In average cases, the encoding process only requires 25 to 90 cycles for the MB header and 450 cycles for the MB residual [Nguy12A].

In terms of power consumption, our power result at 100MHz is only 1.56mW which is even less than 3.7mW of the low-power design in [Tsai06L] and 2.5mW of the design in [Huan08H], both operating at 27MHz.

2.5 Power simulation for the VENGME EC-NAL module

The power results estimated by using synthesis tools as presented in section 2.4 provide a first idea of the power consumption of the EC-NAL module. However, these synthesis tools usually suppose an average activities of the circuit for power estimation. This leads to inaccuracy. Power estimation from post-synthesis simulation provides more accurate results. However, this method requires long simulation time. Using simulation at RTL level

reduces the simulation time while we can obtain the real activity information of the circuit. For these reasons, we performed power estimation for the EC-NAL module at RTL level to obtain the power results based on the real activity of the circuit with short simulation time. Our purpose is to explore the possibility to decrease the power consumption when we replace the synchronous FIFO embedded in the EC-NAL module by an asynchronous one.

The power simulation at RTL level for the EC-NAL module has been performed during the thesis period in CEA-LETI, France, using technology CMOS $32nm$.

As introduced in subsection 2.1.5, power optimization methods that take advantage of the memory/FIFO communication between submodules can be applied into the EC-NAL module, then extended to the whole VENGME encoder. The power consumption of the EC-NAL module is estimated at RTL level in encoding five frames (IBPBP) video of QCIF resolution. The operating frequency is $50MHz$, based on the operating frequency of the VENGME platform at hand. This result is presented in Table 2.12.

Table 2.12: Power composition of the EC-NAL module using synchronous FIFO with technology CMOS $32nm$, at $50MHz$ and voltage $1V$

Power(μW)	**EC**	**sync. FIFO**	**NAL**	**Total**
Total	793	62.8	39.8	895.1
Leakage	253	47	9.78	309.78
Internal	236	9.57	18.9	264.47
Switching	305	6.26	11.1	322.36

Among several power optimization methods presented in subsection 1.5.1, based on the fact that temporal data transferred from task to task can be written and read at different speeds, Dynamic Frequency Scaling (DFS) is the first possible power optimization solution to be applied to our module. DFS is selected because of its simplicity and independence from the technology. It can be implemented even at RTL level. Besides applying DFS, Dynamic Voltage Scaling (DVS) can be considered to be used later leading to DVFS technique. To make this DFS technique possible, the data communication link in our EC-NAL (the FIFO between EC and NAL submodules) should be bi-synchronous. An asynchronous FIFO is designed,

modeled and implemented in the EC-NAL, in replacing the synchronous one. The details regarding the design of this asynchronous FIFO will be presented in subsection 4.1.1. The power simulation result of the EC-NAL module with an asynchronous FIFO is presented in Table 2.13. In this simulation, the two asynchronous submodules, EC and NAL, use the same operating frequency ($50MHz$). No DFS has been applied yet.

Table 2.13: Power composition of the EC-NAL module using asynchronous FIFO with technology CMOS $32nm$

Power(μW)	**EC**	**async. FIFO**	**NAL**	**Total**
Total	801	119	33.7	953.7
Leakage	253	78.7	9.67	341.37
Internal	237	21.1	15	273.1
Switching	311	19.5	9	339.5

Using an asynchronous FIFO, the total power consumption of the EC-NAL module increases only about 6.5%, from $895.6\mu W$ to $953.7\mu W$. This increase is small enough to apply low power methods, for instance, DFS, in the module with the asynchronous FIFO.

2.6 Conclusion

This chapter has introduced the architecture of the VENGME H.264/AVC video encoder. The VENGME platform targets mobile applications and possibly, extension to other applications. The profile selected is the *Main profile at level 2*. The video encoder is optimized for CIF video at a frame rate of $30fps$. The VENGME modules are designed independently. Different improvement techniques have been reused (from the literature) or proposed to attain specific design objectives, namely scalability, speed and power consumption. Synthesis results show that the VENGME platform consumes about *58.25mW, which is suitable for the mobile applications* the platform is targeting. ($130nm$ CMOS technology, in Vietnam) However, some extra power optimization is expected, for further applications.

The chapter also presented in details the design of the EC-NAL module embedded in the VENGME platform. *This module design is one of the contributions* of the thesis. For this module, several improvements have been reused from the literature and others have been proposed, especially

in the CAVLC encoder which is an important engine of the entropy coder (EC) submodule, so as to increase the throughput, reduce the area cost and the power consumption. The implementation results have shown that our CAVLC encoder has better performances and slightly higher area cost when compared to state-of-the-art designs. Our EC-NAL module exhibits a very low power consumption, only $1.56mW$, at the operating frequency of $100MHz$ ($180nm$ CMOS technology, in Vietnam). The EC-NAL design was initially targeting CIF video format, but it is also suitable for real-time 720HD video format at the operating frequency of $100MHz$.

Another contribution of this work is *the power simulation at platform- and module- levels* to analyze the power distribution over the various modules/submodules and then to seek a possible low-power method that can be applied to the VENGME platform. Power simulation at RTL level using SpyGlass has been performed to obtain early power figures of the platform. In terms of composition, Inter Prediction and SW_MAU are the most consuming modules. However, this part of power consumption can be reduced because these modules operate only on inter predicted frames. The EC-NAL module, which is used in every frame, consumes the third power budget. Hence, decreasing the power consumption of the EC-NAL module is also essential. Indeed, the reduction of the power consumption for the VENGME encoder has been tackled through several axes. The execution of inter prediction and intra prediction separately or in parallel was already a method known for its reduction capability of the power consumption for H.264 encoders. Moreover, the VENGME unbalanced pipeline schedule leaves room for implementing power reduction techniques.

Because the VENGME data transfer links use buffers, which can be implemented as asynchronous ones, among power management approaches[5], the Dynamic Frequency Scaling (DFS) is a simple technique that can be firstly applied to the platform. The EC-NAL module also implements FIFO elements as buffer blocks to transfer data between the EC and NAL submodules. Moreover, this module runs for every video frame, which in turn will induce a quite large power consumption. Lastly, the EC-NAL module has been designed by the thesis author. Therefore, its architecture is deeply understood. Therefore, the EC-NAL module is selected to develop and test a power management approach (namely a DFS technique) that might be applied to the whole VENGME platform.

[5]They have been reviewed in Chapter 1, Section 1.5.1

To apply the DFS technique, we modified the synchronous FIFO data transfer link within the EC-NAL module to an asynchronous one. Power simulations at RTL level using SpyGlass have been performed for both versions of the VENGME EC-NAL module, namely the original version with a synchronous FIFO between EC and NAL submodules and the modified version with an asynchronous FIFO instead. The power simulation results show that this modification increases only by 6.5% the total power consumption of the whole EC-NAL module, which seems small compared to the power gain we expect via the implementation of DFS. The next chapter will introduce our FIFO-based control method to reduce the power consumption of EC-NAL module.

Chapter 3

Power management based on the control of the FIFO/buffer occupancy level and its application to the EC-NAL module

Chapter 2 has presented the VENGME H.264/AVC video encoder platform. Targeting mobile applications, the VENGME platform was designed using low-power design features. Some blocks in the platform, including the Entropy Coder - the bytestream Network Abstraction Layer data packer (EC-NAL) module, present nice power figures and performances, similar to state-of-the-art designs. However, the whole platform power results are not outstanding in comparison to state-of-the-art hardware video encoders. To improve the platform power results, its imbalance workload and different power control techniques can be exploited. This includes applying Dynamic Power Management (DPM) methods, for instance, the Dynamic Voltage and Frequency Scaling (DVFS) approach [Yada12A].

Because of the computational complexity increase, Multi-Processor System-on-Chip (MPSoC) architectures are developed for the design of integrated systems. A general figure of a MPSoC architecture is illustrated in Figure 3.1. In this MPSoC architecture, computing blocks operate in parallel to

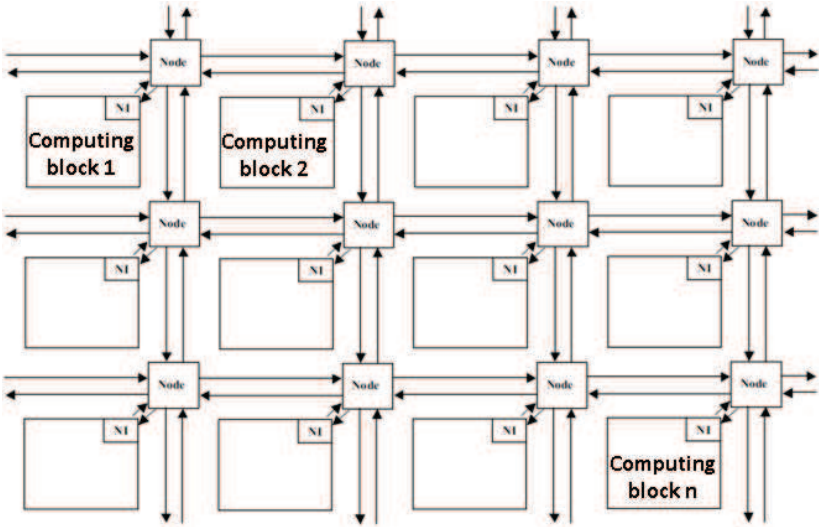

Figure 3.1: General figure of multi-computing architecture, modified from [Beig05A].

perform their tasks as parts of a general job. On-chip communication is constructed to exchange data among these blocks. One of the architecture advantages is the ability to implement DVFS power management method. Indeed, each computing block in the architecture can be implemented as an independent voltage-frequency domain. The voltage and frequency of the computing blocks can be scaled up/down (depending on the workload) to reduce energy and power consumption. To identify the workload status of a domain, one can look on its input data flow. In MPSoC architectures where computing blocks communicate via buffer/FIFO, the voltage and frequency of each domain can be controlled according to the buffer/FIFO occupancy level.

The VENGME platform is designed as a dataflow architecture with modules that communicate via buffers. Therefore, DVFS can be applied on the VENGME platform by controlling the occupancy level of the buffers. The modules of the platform can be implemented as independent voltage-frequency domains. Note that the platform targets mobile applications. Thus, applying DVFS can achieve both energy and power consumption gains. However, due to implementation constraints and the design at hand, currently only the frequency is scaled. The voltage scaling can be then similarly implemented as a future improvement.

The power simulation results on the VENGME platform at RTL level

using SpyGlass[1], see Table 2.3, have shown that the EC-NAL is the third consuming module, after the inter prediction and the Search Window Direct Memory Access (SW_DMA) modules. However, the EC-NAL module functions in every video frame, while inter prediction and SW_DMA can operate only on inter predicted frame. Thus, it is essential to reduce the power consumption of the EC-NAL module. Besides, in the EC-NAL module, a FIFO is implemented to transfer data from EC to NAL submodules. This gives the chance to apply FIFO-level-based DFS control method to the FIFO embedded in the EC-NAL module. Lastly, because the EC-NAL module design is one of the contributions of this work, with a deep understanding of the module architecture, it is easier to modify the design, trace the data during simulations and verify the powerfulness of the power management method discussed hereafter. Applying FIFO-level-based DFS control method to the FIFO between the EC and NAL submodules splits the whole VENGME platform into two parts: the first one contains NAL and NAL SRAM and the other one contains the rest of the platform.

This chapter proposes a FIFO-level-based Dynamic Frequency Scaling (DFS) method applied to the VENGME platform. It can indeed be used for any buffer/FIFO-based system where memory blocks are used to transfer temporal data between modules. The objective is to scale the frequency of the module that receives data to maintain the occupancy level of the FIFO at a given reference value. Since this DFS method is based on control theory, the chapter will present the system modeling for a FIFO link. A Proportional-Integral (PI) controller is chosen and the controlled system is proved to be stable. Some very first simulation results show the effectiveness of the PI control for the FIFO level. All the details related to the implementation and their associated results will be presented in Chapter 4.

3.1 Power management using the FIFO/buffer-level control: an introduction

We propose to scale up and down the frequency according to the occupancy level of a FIFO link between submodules. To introduce the modeling section, few concepts are first provided. We will also review some previous

[1]It has been presented in Chapter 2, Section 2.1.5.

power management works that deal with FIFO/buffer status. While these previous approaches were mostly designed for specific applications and implemented in software, our proposal is full hardware. Moreover, the method proposed can be widely applied to any buffer/FIFO-based system that uses memory blocks to transfer data.

3.1.1 Basic concepts

3.1.1.1 FIFO-based systems

Basically, a buffer is used to store then transfer data between two processes: one process generates data elements while the other one consumes the data elements. FIFO, standing for First-In-First-Out, is a general method for storing and transfering data in a buffer, where the oldest data element in the buffer will be processed firstly. Usually, the buffer is of fixed size and its capacity is bounded. The method can be implemented either in software as an abstract data type or in hardware as shift registers or memory structures. A FIFO-based system is defined as a system where modules communicate via FIFO links.

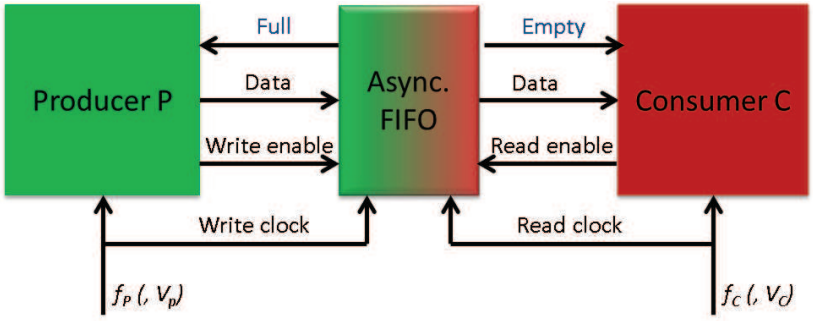

Figure 3.2: FIFO communication between producer and consumer modules.

For each FIFO link, a "producer" module writes data into the FIFO when it is not full. The "consumer" module reads data in the same order that it is written until the FIFO is empty. To avoid losing data, the producer has to wait when the FIFO is full to write a new data element. In the consumer side, if the FIFO is empty, then the consumer has to wait to read new data element. Producer and consumer can operate at the same frequency or at different frequencies. To enable Dynamic Voltage and Frequency Scaling (DVFS), producer and consumer operate with independent supply voltages

V_{dd} (V_P and V_C) and clock frequencies f (f_P and f_C). Hence, the integrated FIFO must be an asynchronous one, with different writing and reading clock signals [Chou09P]. Figure 3.2 illustrates this communication method.

Because technology evolution increased the integration density and complexity of chips, and the distribution of a unique clock tree became difficult, the Globally Asynchronous-Locally Synchronous (GALS) paradigm for System-on-Chip (SoC) appeared. With GALS architecture, locally synchronous modules asynchronously communicate with each other. The architecture can reduce the power consumption and electromagnetic interference (EMI). It also eases the clock distribution [Javi08E]. System-on-chip designers have implemented GALS architectures for several applications, e.g. telecommunications, multimedia, etc... Data transferred between the tasks running on different modules are stored in buffers as asynchronous FIFOs or dual port memory blocks. Hence, a DFS method that takes into account the FIFO occupancy level may target many applications.

3.1.1.2 FIFO-level based DFS method

The DFS method, proposed hereafter, manages the power consumption independently from the application by scaling the frequency according to the occupancy level of a FIFO link between two modules. The DFS method is based on control theory. This control method is implemented in hardware. Thus it can be widely applied. The method will be presented in more details in Section 3.2 dedicated to the control design.

3.1.1.3 Case-study

The FIFO-level-based DFS control method will be embedded in the VENGME hardware H.264 video encoder[2]. As shown in Figure 3.3, buffers (blocks in white) are placed between modules. Data in the current frame is read onto the chip via an AHB/APB interface. Then, it is written in the current macroblock RAM (Cur MB RAM) and search window RAM (SW RAM). The predicted data, from intra prediction and inter prediction, containing integer motion estimation (IME), fractional motion estimation (FME) and motion compensation (MC), are stored in buffers placed inside these modules before being sent to the forward transformation and quantization (FTQ) block. TQIF is a memory block implemented as an interface to

[2]see Chapter 2, Section 2.1, for the presentation of the VENGME platform.

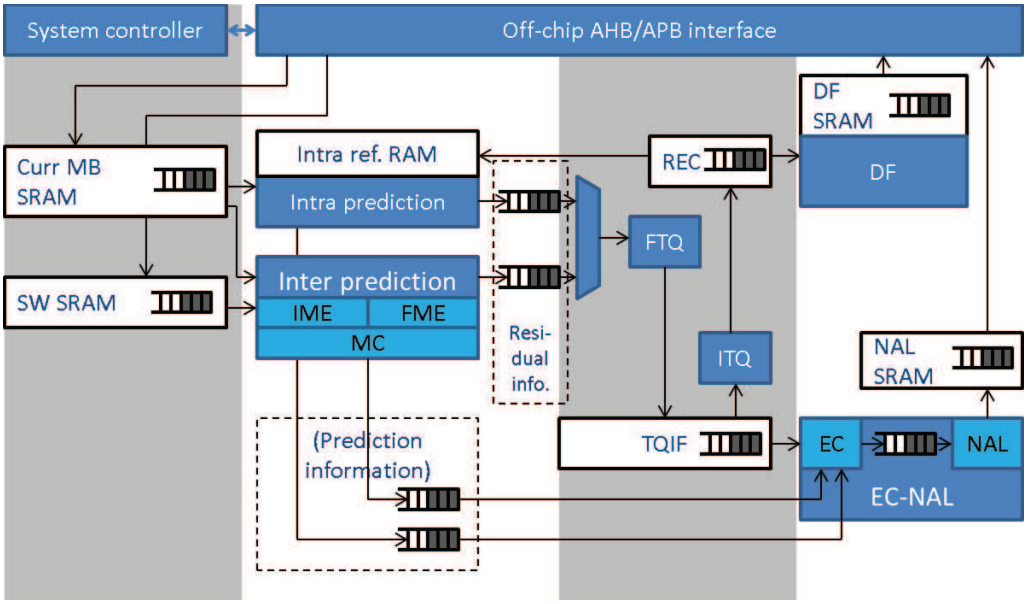

Figure 3.3: The VENGME H.264/AVC encoder platform with FIFO links between modules.

transfer data from FTQ to the de-quantization and inverse transformation (ITQ) block and the entropy coder (EC). The entropy encoded data, after being encapsulated in the Network Abstraction Layer (NAL), are stored in the NAL SRAM before being sent out via the off-chip interface. The same applies to data from the de-blocking filter (DF). In the decoding path, the reconstructed frame (from REC module) is stored in the Intra Ref RAM associated to the Intra prediction block. In each memory block, the reading order is the same as the writing one. In other words, each memory block in the VENGME platform can be considered as a FIFO link between two modules.

Theoretically, we can introduce the FIFO-based DFS control method in all the FIFO links in the VENGME platform. It is mentioned, in chapter 2, that the VENGME imbalance schedule leaves rooms for the implementation of DVFS technique in the platform. The speed of the Transformation-Quantization (FTQ and ITQ), Entropy Coding (EC and NAL) and De-blocking Filter (DF) modules can be adapted with respect to the heavy (inter and intra) prediction modules.

In the case-study here, we implemented the DFS control technique only on the EC module that contains the entropy coder communicating with the Video Data Packer at the Network Abstraction Layer (NAL) via a

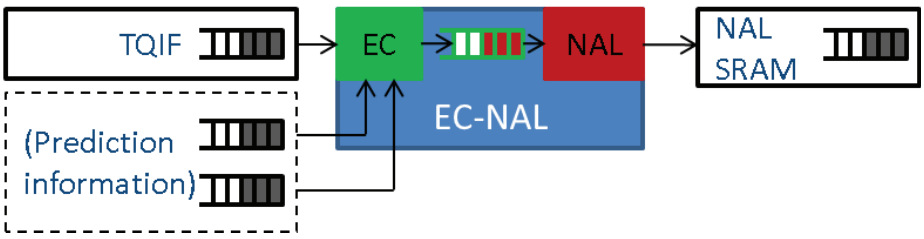

Figure 3.4: EC module and its FIFO link between entropy coder and NAL sub-modules.

FIFO, see Figure 3.4. Indeed, from a deep analysis of the workload and computing speed of the different blocks, it has been seen that having a fine grain adjustment of the clock frequency (and supply voltage) on the NAL block will decrease the power consumption. Actually, the clock frequency of the NAL block can be scaled down when the FIFO is far from full, leading to power consumption decrease.

The original design of the EC-NAL module is synchronous. However, the power simulation results using SpyGlass[3] proved that the implentation of an asynchronous FIFO increases the power consumption of the module by approximately 6.5%. Hence, applying DFS on this EC-NAL module seems reasonable if the global power gain balances this power loss. Besides, the output of the NAL submodule is a memory block with no timing constraint, so scaling down the NAL speed can achieve power reduction without pain. Therefore, it is natural to implement FIFO level control between the EC and NAL blocks. As a consequence, the actual H.264 platform is split in two (voltage-) frequency domains, see Figure 3.5, namely, the NAL block on the one side, and the other blocks on the other side.

3.1.2 Previous works

In the literature, some researchers tried to control the supply voltage V_{dd} and/or the clock frequency f based on a FIFO (or buffer) status [Lu02D, Lu03R, Bae05L, Thie05D, Wu05V, Chou09P]. In these works, the FIFO (or buffer) status can be the FIFO (or buffer) occupancy level, or the filling speed of the FIFO (or buffer) or the status that the producer/consumer has to wait for the utilization of the FIFO, or any derived scheme.

[3]See Section 2.5 in Chapter 2.

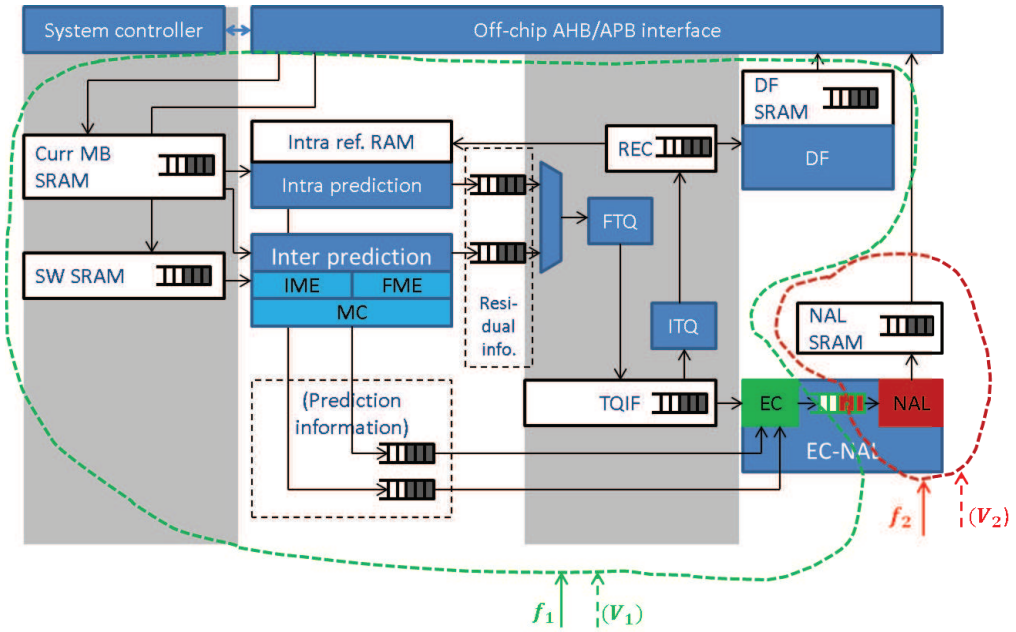

Figure 3.5: The VENGME H.264/AVC encoder platform with 2 power domains.

L. Thiele *et al.* [Thie05D] used the buffer fill-in level that indicates the data stream rate combined with dynamic worst-case bounds based on offline analysis and workload history to adapt the clock rate of the consumer. Note that in Thiele's work, the consumer is a processor. The authors consider processing multimedia streams with restricted buffer size. The restricted buffer size supports just the buffer space for the complete workload average. Hence, if the "consumer" operates at a low rate to reduce the power consumption, a buffer overflow may occur due to a bursty input stream. The method tries to set the consumer rate to obtain the buffer level close to zero during low-load period. Otherwise, when a workload burst arrives, it tries to be as "lazy" as possible to fully exploit the available buffer space. Hence, the consumer rate is set at a rate that it is just sufficient to avoid a buffer overflow in the worst-case.

The worst-case interval-based workload characterization is performed as follows. An application-dependent offline analysis is performed to obtain static worst-case bound, i.e. the worst-case workload only on intervals of length $\Delta(s)$. This infomation is then used along with workload history to compute the dynamic worst-case bound at adaptation timing points t. Suppose that during any time interval $\Delta(s)$, the workload imposed on the

processor is equal to A processor cycles. If at some observational timing point t, the execution demand requested during the time interval $[t-\Delta_p, t]$ is B processor cycles, then it is guaranteed that over the next time interval $[t, t+\Delta-\Delta_p]$, the execution demand will not be more than $A-B$ cycles. The method provides hard Quality of Service (QoS) which is not strictly required in our problem because the output of the consumer (NAL submodule) is a memory block with no timing constraint. The method also has large demand on memory and other implementation resources for the dynamic worst-case workload characterization. Implementation details are not mentioned here.

In [Chou09P], P. Choudhary *et al.* compared the time that the producer and the consumer have to wait due to FIFO fullness or emptiness in a given time interval T_{sample} to decide if it is necessary to match the rate between the frequencies of both domains. Similar to GALS architecture, a VFI-based (Voltage-Frequency Island) system contains Processing Elements (PEs) whose voltage and frequency can be controlled independently. In these systems, mixed-clock FIFOs are used between two communicating PEs. The system can be modelled using a component graph where PEs are modelled as nodes containing a single "source" node and a single "sink" node. [Chou09P] defines the "stall" state as the state where the producer or the consumer has to wait to use the buffer. Figure 3.6 shows an example of the stall state for the producer. It occurs when the FIFO is full and data is available at the output of the producer. A hardware architecture is implemented to monitor the stall state of both producer and consumer sides in a sampling window T_{sample}, see Figure 3.7.

Based on the number of clock cycles where the producer and the consumer are stall, S_f and S_e, during T_{sample}, the *frequency step factor* S is calculated as $S = 1 - |S_e - S_f|/T_{sample}$. The calculated step factor S is used to scale either the frequency of the producer or of the consumer according to the system constraint. If the system is sink constrained, the frequency (and voltage) of the producer will be scaled. Otherwise, if the system is source constrained, the frequency (and voltage) of the consumer will be scaled. Implemented in hardware, the method in [Chou09P] has potential power consumption reduction. However, the method cannot eliminate the stall state. One has to wait until either the producer or the consumer is stall for a time interval long enough before acting on f_C and/or f_P. Besides, selecting the value of T_{sample} is also a problem which is not discussed in the

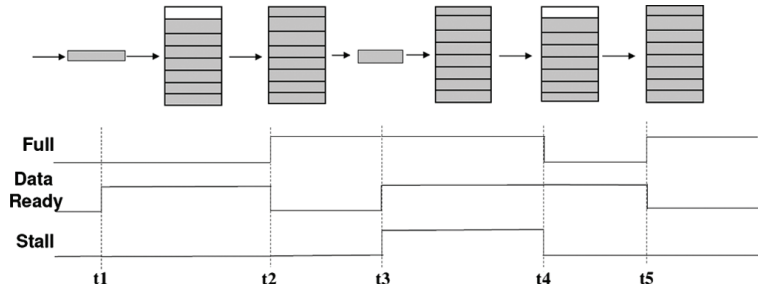

Figure 3.6: Stall state on producer side [Chou09P].

paper. For instance, in the case the producer stalls at the beginning of the first half of the time interval T_{sample} and the consumer stalls during the last part of T_{sample}, there will be no speed scaling for neither the producer nor the consumer. But if we halve the time interval T_{sample}, the DVFS will be applied and the power consumption can be reduced even more.

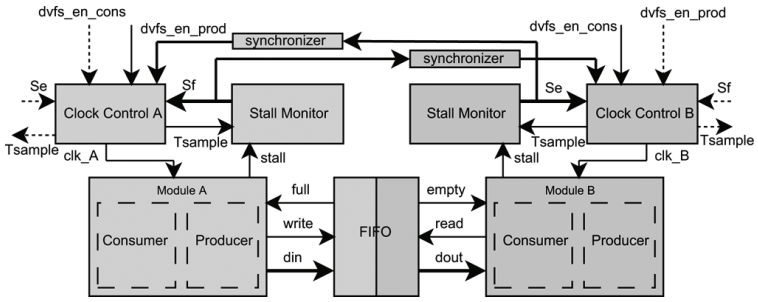

Figure 3.7: Hardware implementation of the method presented in [Chou09P].

Y-H. Lu *et al.* [Lu02D] inserted buffers in a multimedia system for

which a constant output rate is required. A graph-walk algorithm is implemented to assign the processor (producer) frequency based on the buffer state. In the assignment graph, each vertex contains information on the processor frequency, the amount of data in the buffer and the next operation (writing or reading data to or from the buffer). A "walk" is defined as a sequence of assignments of frequencies and executions by the vertices. The frequency change is selected according to the minimum-cost walk in the graph-walk. The frequency is changed only at the beginning of a period of length t. The method to chose the length t is not mentioned in the paper. In the experiment part, it is chosen equal to $1s$.

The methods developed in [Lu03R] and [Bae05L] are applied to a processor operating as a multimedia decoder and communicating to a display device via a buffer. Based on the buffer status, the processor (producer) frequency/voltage is scaled, see Figure 3.8. In [Bae05L], the producer frequency is increased when the buffer is emptier than the target buffer level or running empty. Otherwise, the producer frequency is decreased when the buffer is filled more than the target buffer level or running full. The algorithm can be summarized by the equation

$$\Delta f = c_1 x_i + c_2(x_i - x_{i-1}) \tag{3.1}$$

The controller is then designed using a continuous-time model. The information used in this work is both the buffer occupancy level x_i and its filling speed $x_i - x_{i-1}$.

In [Lu03R], a dead-zone based control algorithm is constructed to adjust the decoder (producer) speed in order to keep the buffer occupancy within a given safe range so as to match the decode and display rates.

In [Wu05V], the system is modeled as a Multiple Clock Domain (MCD) where the DVFS method controls the frequency (and voltage) of each clock domain according to the input queue status, see Figure 3.9. Finite State Machines (FSMs) are implemented to increase/decrease the frequency and voltage by one single step. The voltage/frequency changes are decided based on the queue signal compared to a deviation window after a time delay. The queue signal is either the difference between the queue occupancy and a reference value or the difference between the queue occupancy at two consecutive sampling times (i.e. queue filling speed which is consistent with a flow). The deviation window is the threshold determined for the

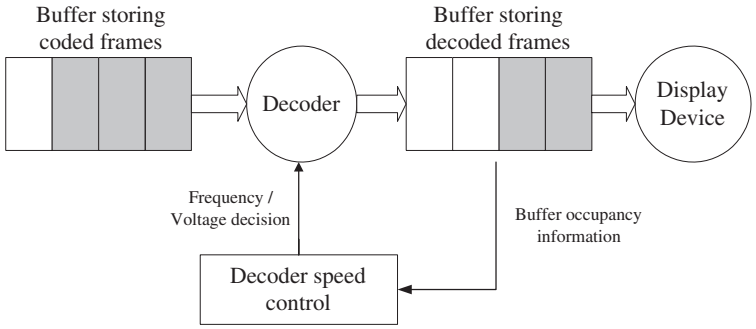

Figure 3.8: Voltage and/or frequency scaling method on a multimedia decode and display system [Lu03R].

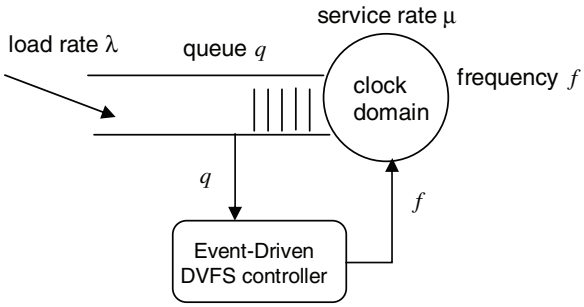

Figure 3.9: Queue-Domain model [Wu05V].

queue signal. For each queue signal, a FSM, as depicted in Figure 3.10, is implemented.

The two queue signals require twice the Silicon resource for an FSM implementation. Then, a *Schedule* state is added to decide the final *Act* operation. The utilized model is continuous-time. The system contains non-linearities, see Figure 3.11. For stability analysis, a linearization is performed by choosing a specific function $h(f)$, which is the effect of the frequency f on the effective time delay, to compensate for the non-linear

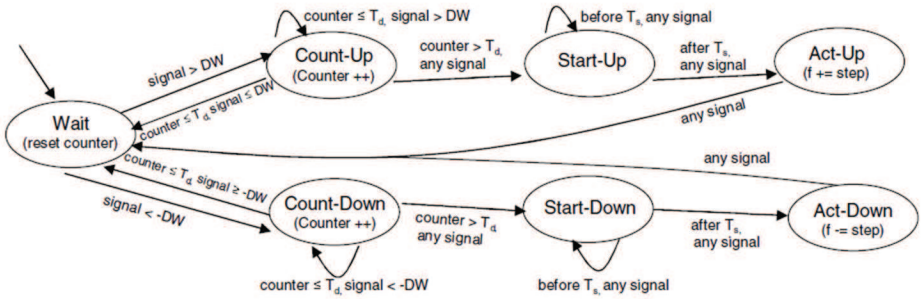

Figure 3.10: Finite State Machine (FSM) implementation for each queue signal [Wu05V].

function of the system model ($h(f)$ is chosen to be proportional to k/f^2). The linearization used in [Wu05V] makes the system a linear one. However, using this linearization, they cannot ensure that the system is always stable in all situations. In [Wu05V], the DVFS method is designed to be adaptive to workload variation and cost-effective. However the time delay, one single step action and the FSMs solution make the implementation more intricate.

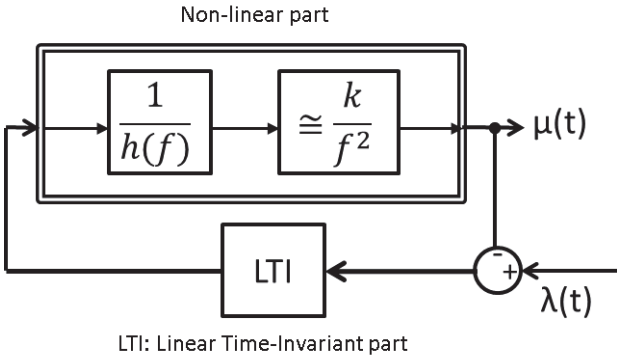

Figure 3.11: Closed-loop system with non-linearities in [Wu05V].

To summarize, the DFS (resp. DVFS) works found in the literature exploit the FIFO (or buffer) status to identify the workload of the module where the DFS (resp. DVFS) method is applied. The FIFO (or buffer) occupancy level is the simplest utilized FIFO (or buffer) status because this information can be directly obtained from the writing and reading indices inside the FIFO. Some FIFO (or buffer) based DFS works that use information

other than FIFO (or buffer) level [Bae05L, Thie05D, Wu05V, Chou09P] require more resources to monitor, collect and then process information. This also leads to a complex decision making with different buffer status information synthesized. Thus, this makes the implementation more intricate. The FIFO (or buffer) based DFS approaches we have presented have been designed for specific applications and implemented in software [Lu02D, Lu03R, Bae05L]. Continuous-time models which seems less accurate than discrete-time model in modeling digital systems, is used in some works [Bae05L, Wu05V]. Only [Wu05V] mentioned the non-linearities of the system. However, their linearization can ensure the stability of the system in a particular case, when $h(f)$ is proportional to k/f^2. Applying a PI controller, the work in [Lu03R] does not analyze the stability of the controlled system.

The work in this thesis implements in hardware a DFS method based only on the FIFO occupancy level. The DFS method is application-independent and suitable for any FIFO-based system. A discrete-time model is used and the controlled system is proved to be stable. More details on the system modeling, the design of the control law and the stability analysis will be presented in Section 3.2.

3.2 FIFO-level based DFS control

As mentioned above, the FIFO-level based control can act on both frequency and voltage (DVFS) to achieve power and energy gains. For the sake of simplicity, the method presented hereafter acts only on the frequency (DFS). This section presents in details the FIFO-level based DFS control method. The method is applied to the FIFO link between the Entropy Coder (EC) and the Network Abstraction Layer (NAL) modules of the VENGME H.264/AVC video encoder platform. This FIFO link splits the whole platform into two power domains, namely, one with the NAL module and one with the other modules. In order not to modify the modules designed by others people involved in the VENGME platform design, the frequency of the "producer" is kept constant and the DFS method controls the frequency of the domain containing the NAL module (i.e. the consumer). Remember that the design of this EC-NAL module is one of our contributions. Thus, their modifications can be easily performed. The control is made according to the FIFO occupancy level of the link between

both (voltage/)frequency domains [Nguy14F].

The next subsections introduce respectively the system modeling, the control design and some remarks related to the implementation and stability analysis of the system under control.

3.2.1 FIFO link modeling

As mentioned above, the control of the FIFO occupancy level has been implemented in the EC module of the VENGME H.264 encoder where the encoder acts as "producer", sending data to the NAL which is the "consumer", see Figure 3.12.

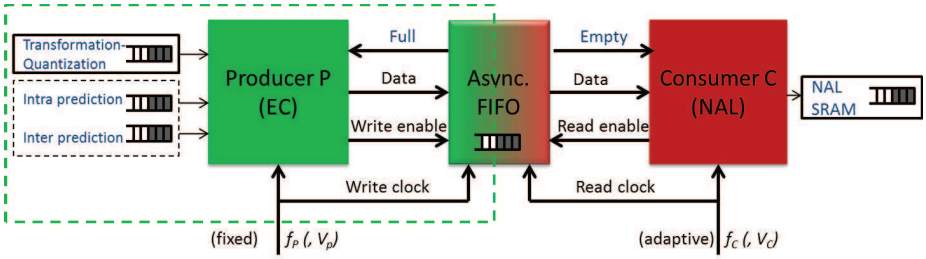

Figure 3.12: Entropy coder - Bytestream NAL data packer (EC-NAL) module where power management method based on FIFO-occupancy level is applied.

Because the EC-NAL is the last module in the encoding path, the so-called producer receives data from the other modules in the VENGME H.264 encoder. Hereafter, f_P (applied to all the blocks in the green dashed rectangle in Figure 3.12) is supposed constant while f_C can be scaled up/down.

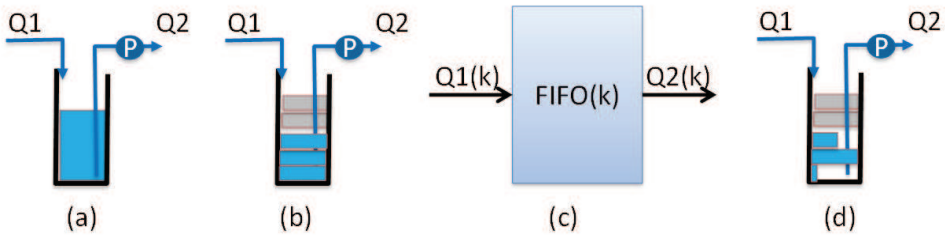

Figure 3.13: FIFO model.

The FIFO can be modeled as a tank with one input $Q1$ and one output $Q2$. Moreover, $Q1$ and $Q2$ are non-negative. Because the output speed is

independent of the level in the tank, the output $Q2$ is extracted using a "pump" **P** (see Figure 3.13-a). However, each writing or reading operation acts only on one data element. Hence, the FIFO level is discrete, given in number of packets (Figure 3.13-b). As a consequence, the FIFO dynamic model is given by (Figure 3.13-c)

$$FIFO(k+1) = FIFO(k) + Q1(k) - Q2(k) \tag{3.2}$$

where $FIFO(k)$ is the occupancy level of the FIFO in number of packets at the k-th sampling time. $Q1(k)$ is the number of data packets entering the FIFO from the producer. $Q2(k)$ is the number of data packets exiting the FIFO to feed the consumer. Due to Silicon area constraints for the H.264 hardware platform, the maximal number of packets in the FIFO is chosen equal to 7. Therefore, $FIFO(k) \in \{0, 1, 2, 3, 4, 5, 6, 7\}$.

Because the functionality of the NAL-consumer is to concatenate data, the actual speed of reading packets in the FIFO depends not only on the clock frequency of the consumer f_C but also on the length of the packets. In the case-study here, packets are of different length, from 1 bit to 32 bits, depending on the data at hand; see Figure 3.13-d. Actually, the packet length depends on the type of data packet written to the FIFO. If the data contains prediction information, e.g. prediction mode, motion vector, etc., the length is usually short. Otherwise, the residual data packet is usually 32 bits long. However, for the sake of simplicity, the FIFO output flowc $Q2$ is supposed proportional to $f_C(k)$:

$$Q2(k) = bf_C(k) \tag{3.3}$$

where b is a positive constant.

To identify the constant b, a simulation is conducted: the VENGME encoder encodes benchmark video frames to trace the data flow and the "FIFO status". The "FIFO status" contains the producer/consumer stall state similar to the one in [Chou09P]. Note that the FIFO status information is extracted only for this simulation and it is used to analyze the data. Indeed, we do not need this status signal in our control method. Hence, in the real implementation presented later on, no block to collects the stall information.

The information we trace contains the producer frequency f_P, the status of the producer input, the status of the producer output (stall of not) $Pwait$, the number of packets written into the FIFO $Q1$, the FIFO level $FIFO$,

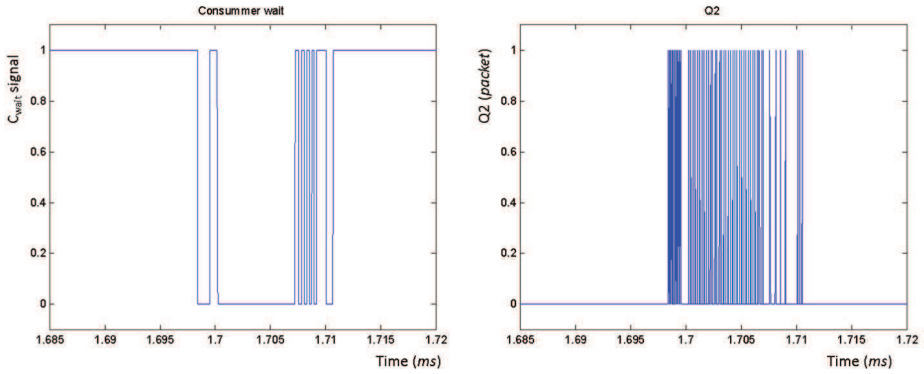

Figure 3.14: Information recorded. It contains the consumer stall ($Cwait$) signal and the $Q2$ value.

the consumer frequency f_C, the status of the consumer input (stall of not) $Cwait$ and the number of packets read from the FIFO $Q2$ at time instants t ($t = k/f_P$). Figure 3.14 illustrates two of the recorded signals, namely, the status of the input of the consumer ($Cwait$), on the left side, and the number of packets read from the FIFO $Q2$ on the right side. From the recorded signal, when the consumer does not have to wait for data available in the FIFO ($Cwait = 0$), i.e. when the consumer is not stall, b is estimated to be equal to $20ns$. Note that b is considered constant, which means that the packet size is not taken into account.

Each time the FIFO is full (resp. empty), and the producer has some data to write into the FIFO (resp. the consumer wants to read data from the FIFO), the producer (resp. consumer) falls into the stall state. This stall state leads to a waste of power consumption. Thus, the control objective is to adapt the consumer frequency to keep the FIFO half-full (neither empty nor full) during normal operation. Of course, at the end of the encoding process, the FIFO will have to be empty.

3.2.2 Controller design

As already said, we suppose that $Q2 = bf_C$, where b is constant. Therefore, we do not take into account different sizes for the data packets in the FIFO. The controller is now designed.

- Performance objective

As mentioned above, the control objective is to adapt the consumer frequency to keep the FIFO half-full during normal operation. The expected time response is about $20\mu s$.

- **FIFO model**

For the FIFO link under study, the input data Q1 is the output of the producer, which is not controlled. Therefore, in the system model, $Q1$ is considered as a disturbance. The FIFO transfer model, without disturbance, in the z-domain, is as follows:

$$\frac{FIFO(z)}{Q2(z)} = G(z) = \frac{-1}{z-1} \tag{3.4}$$

which is actually an integrator. Figure 3.15 shows the controlled system with its controller $C(z)$ that adapts the consumer frequency $f_C(z)$ to keep the FIFO level $FIFO(z)$ near a Reference value $r(z)$. The input of the controller is the error $e(z)$ between the Reference level $r(z)$ and the FIFO level $FIFO(z)$.

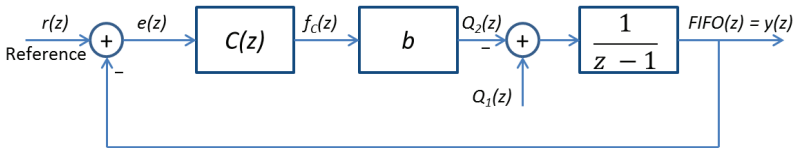

Figure 3.15: Closed-loop control scheme of the FIFO level.

Note that the FIFO saturations (full, empty) are not modeled here.

- **Controller choice**

A discrete-time Proportional-Integral (PI) controller [Astr86C] is selected to reject the "disturbance" $Q1$, to ensure a closed-loop functioning without static error and to tune the closed-loop system response time[4]. The controller is chosen as:

$$\frac{f_C(z)}{e(z)} = C(z) = K_p + K_i \frac{z}{z-1} \tag{3.5}$$

[4]A Proportional (P) controler can also solve the problem but the closed-loop system response will not meet the performance demand

Therefore, the closed-loop transfer function is given by:

$$\frac{FIFO(z)}{r(z)} = \frac{-C(z)b\frac{1}{z-1}}{1+C(z)b\frac{-1}{z-1}} \tag{3.6}$$

$$\frac{FIFO(z)}{r(z)} = \frac{-[(K_p+K_i)z-K_p]b}{z^2-z[2+b(K_p+K_i)]+1+K_pb} \tag{3.7}$$

where $r(z)$ is the Reference for the FIFO level.

The poles z_1, z_2 determine the system dynamics in closed-loop. Their numerical values depend on K_p and K_i values. Once z_1 and z_2 have been chosen, K_p and K_i are computed with:

$$K_p = \frac{z_1z_2-1}{b} = \frac{z_1z_2-1}{0.02} \tag{3.8}$$

$$K_i = \frac{z_1+z_2-z_1z_2-1}{b} = \frac{z_1+z_2-z_1z_2-1}{0.02} \tag{3.9}$$

where $b = 0.02\mu s$ according to its identification given above.

- **Step response of the closed-loop system**

Figure 3.16 shows the step response of the closed-loop system with the PI controller. The time response is equal to $20\mu s$, as expected in the performance objectives. The poles in closed-loop are equal to 0.75 and 0.5.

3.2.3 Implementation constraints and remarks

The output of the controller designed in the previous section can have any real value. However, in our application, the frequency f_C is limited in a given range. For instance, there is no negative frequency. To adapt the frequency from the output of the controller to its "implementable" value, either a saturation, a relay, or a non-uniform quantizer can be used, see Figure 3.17 where this non-linear block is added.

With a saturation, the admissible frequency f_C is any value from $0MHz$ to a maximum frequency, e.g. $100MHz$, see Figure 3.18. If the controller demands a negative frequency, the output will be equal to $0MHz$. Similarly, if the demand is a frequency exceeding $100MHz$, the frequency applied to the consumer will be $100MHz$. The case of using a relay can be implemented as a clock-gating cell. The usual clock frequency is $100MHz$ and when the controller demands a zero or negative action, the clock signal is gated.

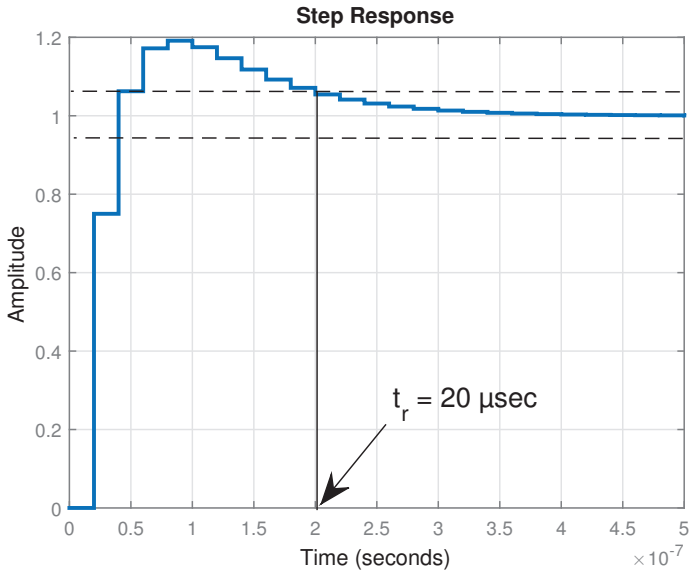

Figure 3.16: Step response of the linear closed-loop system. The poles in closed-loop are equal to 0.75 and 0.5

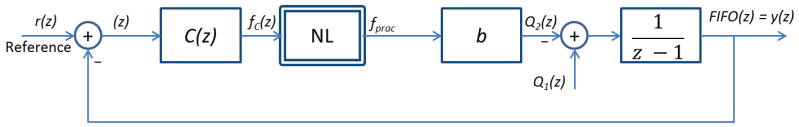

Figure 3.17: Closed-loop control scheme of the FIFO level with non-linearity (NL) added.

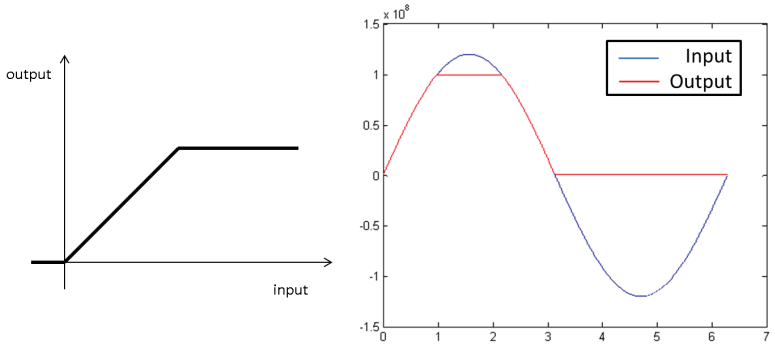

Figure 3.18: Input and output of a saturation block.

With a relay, the admissible frequency f_C is either $0MHz$ or $100MHz$, see Figure 3.19. For instance, if the controller demands a negative frequency,

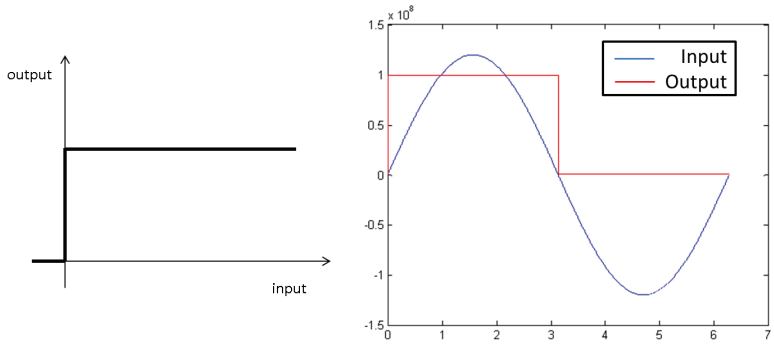

Figure 3.19: Input and output of a relay block.

the frequency f_C applied to the consumer will be equal to $0MHz$. Otherwise if the demand is a positive frequency, it will be $100MHz$.

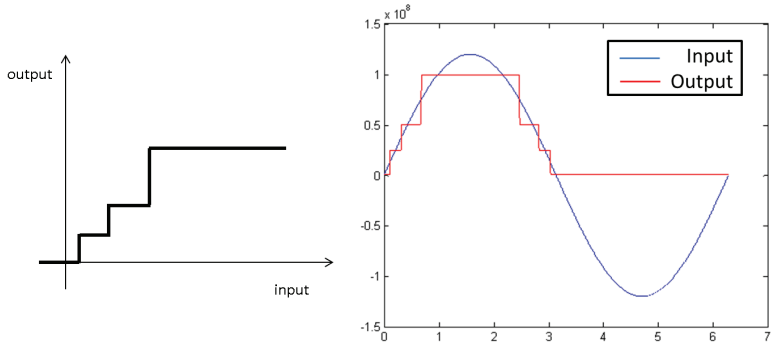

Figure 3.20: Input and output of a quantizer block.

In the case of a non-uniform quantizer, a discrete set of frequency values is admissible, see Figure 3.20. In the present case-study, the set of possible frequency values is chosen equal to $\{0, 12.5, 25, 50, 100\}$ MHz.

The additional block and the integral part of the controller can cause windup phenomena[5]. It occurs when the PI controller keeps integrating the error even when the input is saturating. An anti-windup mechanism is added in the controller to avoid the windup effect. Figure 3.21 shows an example of applying an anti-windup scheme when a saturation is used. Note that when the controller is not saturating, the anti-windup scheme

[5]Every actuator has its own natural limitations. Windup phenomena occurs when an error during actuator saturation keeps being accumulated by the integral part inside the closed-loop system

has no effect. An anti-windup will be used in the MATLAB modeling and simulation results, see Section 3.3.

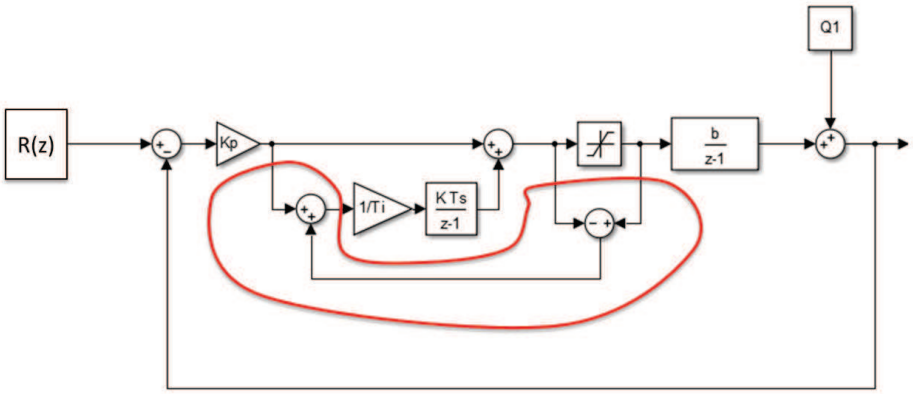

Figure 3.21: Anti-windup scheme in the controller using a saturation.

Because the VENGME platform does not embed any PLL to generate frequency values, we can only implement a frequency divider to generate a dicrete set of possible values for f_C. The highest frequency signal that the VENGME platform supports is $100MHz$. One simple way to implement a frequency divider is using a binary counter. In this way, the frequency is divided by two. As a consequence, possible values for the frequency f_C can be 100, 50, 25, 12.5, etc. MHz. Thus, a non-uniform quantizer is selected to perform DFS. It adds a non-linearity in the system. To simplify the implementation, the anti-windup will not be implemented. In the next subsection, the stability is analyzed for the system using a non-uniform quantizer and without anti-windup mechanism.

3.2.4 Stability analysis

This section analyzes the stability of the closed-loop system. The PI controller is as designed in Section 3.2.2. A non-uniform quantizer is applied to map the frequency computed by the controller to the frequency applied in practice on the consumer. No anti-windup is used in the system.

Because the non-uniform quantizer adds a non-linearity feature in our system, the stability analysis using the pole values is insufficient. Hence, we introduce here two approaches to analyze the stability of a system containing non-linearities. The first approach is based on describing function, using Nyquist stability criterion. The second one uses the topological separation

condition. After a short summary of each approach, we analyze the stability of our system.

3.2.4.1 Nyquist stability criterion: summary

1. **Basic principles of stability analysis for a linear system using the Nyquist stability criterion**

   Consider a linear system where $e(t)$ is the input and $s(t)$ is the output. The transfer function can be determined by applying the sinusoidal input:

   $$e(t) = e_0 \sin(\omega t) \tag{3.10}$$

   the output being, in steady state:

   $$s(t) = s_0 \sin(\omega t + \phi) \tag{3.11}$$

   where $A = s_0/e_0$ is the amplitude ratio and ϕ is the output phase shift. It is well known that A and ϕ are, on the one hand, functions of the input frequency ω but, on the other hand, they are independent from the amplitude e_0. The conclusion is that for linear systems, the frequency response can be described only by the frequency ω.

   A Nyquist plot is a plot of the frequency response of a system. For a function $F(s)$, where $s = j\omega$ is the Laplace operator, the Nyquist plot $F(j\omega)$ is the mapping of the imaginary axis $s = j\omega$ on the s-plane to the $F(s)$-plane.

   Consider a closed-loop system whose open-loop is a transfer function $G(s)$, see Figure 3.22. The closed-loop transfer function of the system is $\frac{G(s)}{1+G(s)}$. It is well known from the Nyquist stability criterion that the closed-loop system is stable if and only if the net number of counter-clockwise encirclements of the critical point $(-1, 0)$ for the contour evaluation of $G(j\omega)$ on the $G(s)$-plane in the sense of growing frequency on the plane equals to the number of unstable poles of $G(s)$. Unstable poles are the poles on the right half plane (RHP) of the s-plane including the imaginary axis $s = j\omega$. If the point is on the Nyquist plot, the system oscillates at the corresponding frequency ω [Fran97D].

   Consider now a proportional gain K to be applied on the system $G(s)$, see Figure 3.23. The same stability criterion is valid: if the

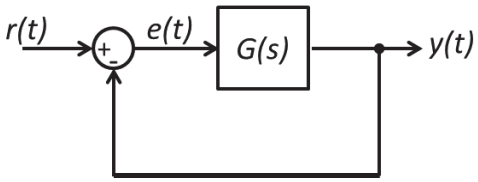

Figure 3.22: Closed-loop system with the open-loop transfer function $G(s)$.

net number of counter-clockwise encirclements of the critical point $(-1/K, 0)$ for $G(j\omega)$ in the sense of growing frequencies, the closed-loop system is stable.

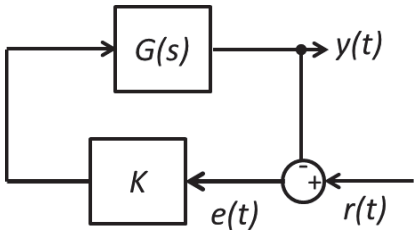

Figure 3.23: Closed-loop system with a proportional gain K.

Figure 3.24 shows two examples of applying the gain K on $G(s)$. Suppose that the plant $G(s)$ has no unstable pole, the proportional gain K whose critical point $-1/K$ is not counter-clockwise encircled by $G(j\omega)$ in the sense of the growing frequencies, i.e. the green one, makes the system stable. Otherwise, the point $-1/K$ in red makes the system unstable. As shown in their corresponding step responses on the left part of the figure, the green one is stable while the red one diverges.

In the case that the open-loop $G(s)$ has pole(s) on the imaginary axis of the s-plane, we still can apply the Nyquist criterion. Trying to exclude the pole from the unstable region, we create a small "detour" which is a semicircle in the counterclockwise direction around the pole on the right half of the s-plane. The radius of the detour is infinitesimally small so that the detour does not exclude any part from the RHP. The Nyquist criterion is then applied using $G(j\omega)$ plot which is mapped from the imaginary axis $s = j\omega$, excluding the poles but including the detours. The poles on the imaginary axis $s = j\omega$ will not be considered in the unstable region. Figure 3.25 illustrates

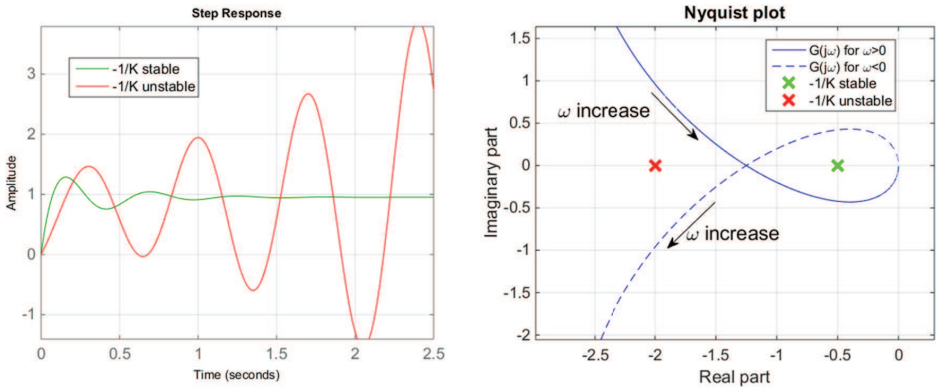

Figure 3.24: Closed-loop stability criteria using the open loop transfer function $G(s)$ and the critical point $-1/K$.

an example of mapping the Nyquist plot when the open-loop system transfer function is $L(s) = \frac{s^2+s-2}{s^2+16}$. In this Nyquist plot, each detour around the pole $s = \pm j4$ causes a semicircle of $L(j\omega)$ with $|L(s)| = \infty$.

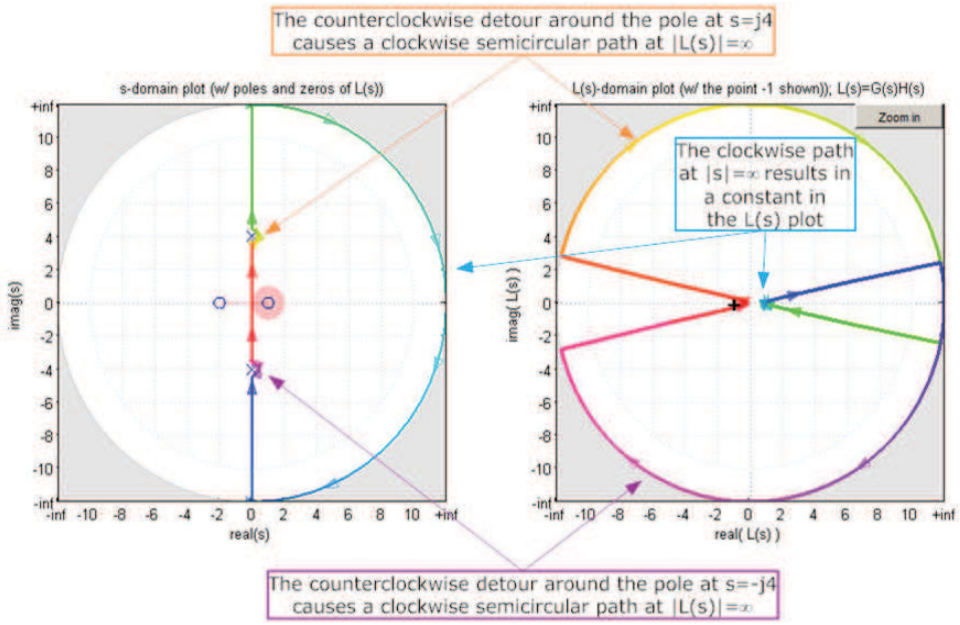

Figure 3.25: An example of mapping the Nyquist plot when the open-loop system has poles on the imaginary axis [lpsa].

2. **Basic principles of stability analysis for a non-linear system**

using the Nyquist stability criterion

For a non-linear system, we assume the same procedure as above. Consider the following input:

$$x = x_1 \sin(\omega t) \tag{3.12}$$

The output $s(t)$ is generally periodical, but not necessarily sinusoidal. As an approximation, one can define a sinusoidal function as:

$$w(t) = w_1 \sin(\omega t + \psi) \tag{3.13}$$

which can be used to define an "equivalent transfer function" by taking the first harmonic of $s(t)$, where the amplitude is w_1/x_1 and the phase is ψ.

The main difference with respect to the linear case is that the equivalent transfer function depends not only on the frequency ω but also on the input amplitude x_1:

$$\begin{array}{c} \frac{w_1}{x_1} = B(x_1, \omega) \\ \psi = \psi(x_1, \omega) \end{array} \tag{3.14}$$

With the Nyquist plot, one has now a family of frequency responses in terms of frequency and amplitude.

- *Equivalent transfer functions - Describing Functions*

A describing function can be interpreted as a transfer function adapted to the non-linear case. The describing function method was first studied by [Duti50T, Ecar50E, Koch50AE, Koch50AA] and is has been widely used to demonstrate the effects of non-linearity on the feedback loop [Gelb68M, Gill75S].

We briefly discuss here the application principle of these approximations, but without details. For further information, the reader should refer to [Gelb68M, Gill75S] and the references therein.

Suppose that the input in equation (3.12) is applied on a non-linearity N, with period $T = 2\pi/\omega$. The output (3.13) generally is a periodic function, whose period is T. Thus, it can be decomposed in a Fourier series:

$$s(t) = w_1 \sin(\omega t + \psi(x_1, \omega)) + w_2 \sin(2\omega t + \psi_2) + w_3 \sin(3\omega t + \psi_3) + \ldots \tag{3.15}$$

By assuming the first harmonics $w_1 \sin(\omega t + \psi)$, the *Describing Function* N is given by:

$$N(x_1, \omega) = \frac{w_1}{x_1} e^{j\psi(x_1,\omega)} \tag{3.16}$$

where the amplitude is $B(x_1, \omega) = w_1/x_1$ and the phase $\psi(x_1, \omega)$.

This procedure can be applied to determine the Describing Function of non-linearities, e.g. dead-zone, backlash, saturation, relay, quantizer, etc. Some examples can be found in [Gelb68M, Gill75S].

The advantage of (3.16) is the possibility to use classical Control theory tools, mainly the Nyquist plot. A particular case is when N only depends on x_1, i.e. a frequency-independent non-linearity. In this case, $N(x_1)$ can be seen as an "equivalent gain" in the Linear Time-Invariant (LTI) sense, function of x_1 and ψ, described by:

$$N(x_1) = B(x_1) e^{j\psi} \tag{3.17}$$

- *Stability analysis with a Describing Function*

Consider the closed-loop scheme given in Figure 3.26, where a non-linearlity N is applied on $G(s)$.

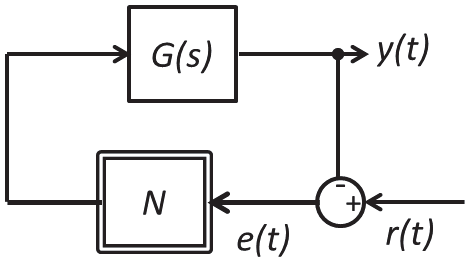

Figure 3.26: Closed-loop with a non-linearity N.

The Nyquist stability criterion can be applied if instead of K one uses the equivalent gain $N(x_1)$, e.g. given in (3.17), if the Describing Function is frequency independent. In the Nyquist plot, the crictical point becomes a "crictical curve" $-1/N(e(t))$. If the curve $-1/N(e(t))$ does not intersect $G(j\omega)$ and the number of counter-clockwise encirclements of the whole curve $-1/N(e(t))$ for $G(j\omega)$ in the sense of growing frequency equals to the number of unstable poles of $G(s)$, the

closed-loop system is stable. If $G(j\omega)$ intersects $-1/N(e(t))$ at the point where $\omega = \omega_1$ and $e(t) = e_1$, the effect on the system is some oscillations at the corresponding frequency ω_1 and with amplitude e_1. The stability of the system at the intersection point is determined using perturbation check. If when the amplitude is slightly increased $e(t) > e_1$, the critical point on $-1/N(e(t))$ is moved to the point where the system is stable, then the system at the intersection point is stable.

Even if the analysis above is made for continuous-time systems, the same reasoning holds for discrete-time systems, by taking $z = e^{j\omega T_e}$, where T_e is the sampling period.

3.2.4.2 Application of the Nyquist criterion approach to our system

The stability analysis using a describing function has been applied in [Murt90N, Hu08N, Romb13A].

In the framework of the present thesis, only a set of values can be applied on the system by the frequency actuator. This means that there is a non-linearity between the controller and the system (see Figure 3.27), which changes the frequency to be applied in practice f_{prac}.

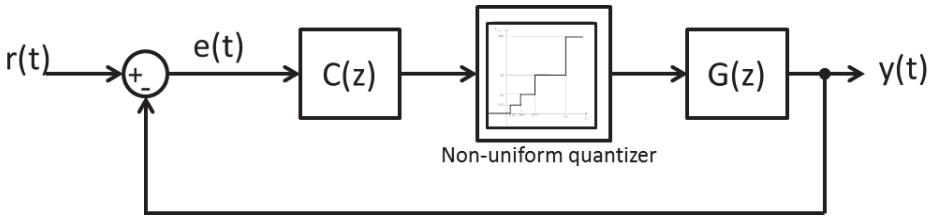

Figure 3.27: Closed-loop system with the controller, the system and a non-uniform quantizer.

This non-linearity is known as a "general odd quantizer" or a "non-uniform quantizer". It is depicted on Figure 3.28. Actually, it is a frequency independent non-linearity.

The describing function for the non-linearity in Figure 3.28 is given by [Gelb68M]:

$$\begin{aligned} A < d_1 \quad &\rightarrow \quad N(A) = 0; \\ d_n < A < d_{n+1} \quad &\rightarrow \quad N(A) = \frac{4}{\pi A} \sum_{i=1}^{n} (D_i - D_{i-1}) \sqrt{1 - \left(\frac{d_i}{A}\right)^2} \end{aligned} \tag{3.18}$$

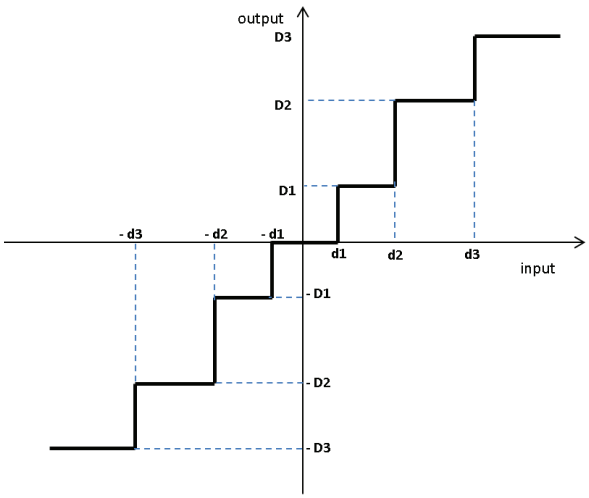

Figure 3.28: General odd quantizer.

In fact, the actuator contains a frequency divider, where the admissible frequencies are $f_{prac} \in \{0, 12.5, 25, 50, 100\}$ MHz (Figure 3.29). The quantizer function is not odd. If the quantizer output is positive, i.e. the input $f_C > 6.25MHz$, it is similar to the "non-uniform quantizer" presented in Figure 3.28. If the input $f_C < 6.25$ MHz, the output is always zero.

Thus, we will consider two cases for the calculated frequency f_C to analyze the stability of the system.

- When the calculated frequency $f_C > 6.25MHz$, the applied frequency f_{prac} is positive.

Because with a positive output, our quantizer function is similar to the "general odd quantizer" function, we use the Describing Function for the "general odd quantizer" to analyze the stability of the system in this case.

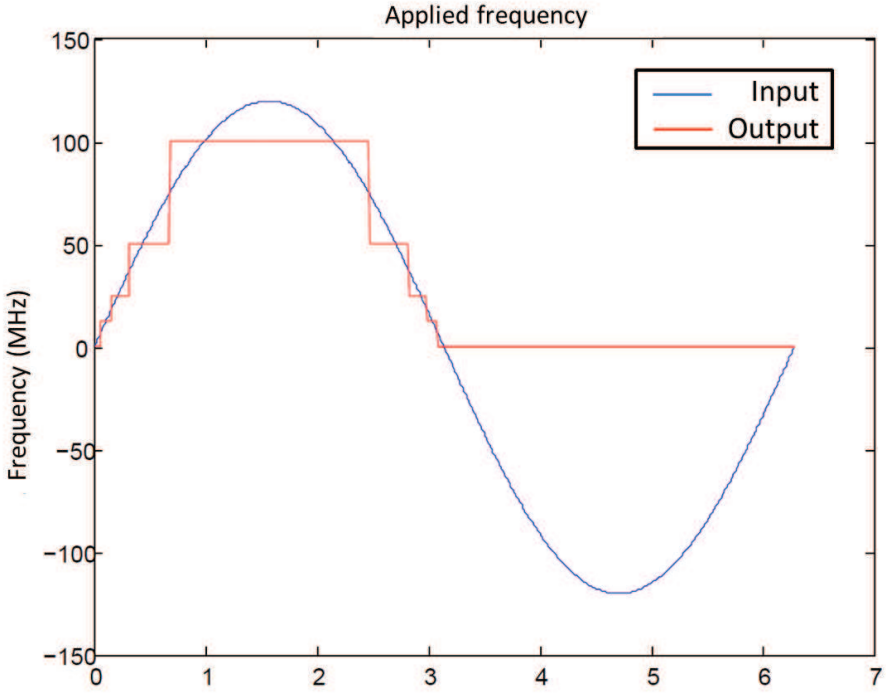

Figure 3.29: Admissible applied frequencies.

Applied to a frequency divider, Equation (3.18) becomes:

$$
\begin{aligned}
f_C < 6.25\ [MHz] \rightarrow \quad N(f_C) = \quad & 0; \\
6.25 < f_C < 18.75\ [MHz] \rightarrow \quad N(f_C) = \quad & \frac{4}{\pi f_C}\left(12.5\sqrt{1-\left(\frac{6.25}{f_C}\right)^2}\right); \\
18.75 < f_C < 37.5\ [MHz] \rightarrow \quad N(f_C) = \quad & \frac{4}{\pi f_C}\left(12.5\sqrt{1-\left(\frac{18.75}{f_C}\right)^2}+12.5\sqrt{1-\left(\frac{6.25}{f_C}\right)^2}\right) \\
37.5 < f_C < 75\ [MHz] \rightarrow \quad N(f_C) = \quad & \frac{4}{\pi f_C}\left(25\sqrt{1-\left(\frac{37.5}{f_C}\right)^2}+12.5\sqrt{1-\left(\frac{18.75}{f_C}\right)^2}\right. \\
& \left.+12.5\sqrt{1-\left(\frac{6.25}{f_C}\right)^2}\right) \\
75\ [MHz] < f_C \rightarrow \quad N(f_C) = \quad & \frac{4}{\pi f_C}\left(50\sqrt{1-\left(\frac{75}{f_C}\right)^2}+25\sqrt{1-\left(\frac{37.5}{f_C}\right)^2}\right. \\
& \left.+12.5\sqrt{1-\left(\frac{18.75}{f_C}\right)^2}+12.5\sqrt{1-\left(\frac{6.25}{f_C}\right)^2}\right)
\end{aligned}
\tag{3.19}
$$

Graphically by using (3.19), Figure 3.30 is obtained.

One can see by using Figure 3.30 that the maximum value of $N(f_C)$ is 1.27. Because we do not consider $f_C < 6.25$ here, $N(f_C) > 0$. In fact, when $f_C \rightarrow +\infty$, $N(f_C) \rightarrow 0$ but in practice, f_C is a finite number, so $N(f_C) \neq 0$. This means that the Nyquist plot $-1/N(f_C)$ of the Describing

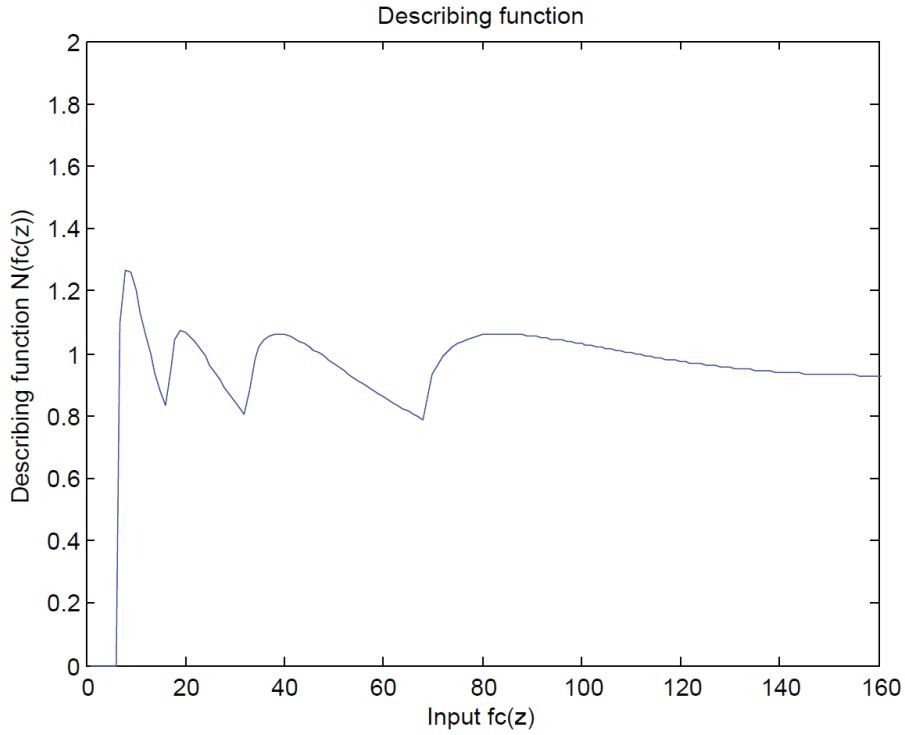

Figure 3.30: Describing function for Equation (3.19).

Function $N(f_C)$ is in the finite interval $[-1/N_{min}(f_C); -1/1.27]$ on the left half of the real axis.

The stability analysis can be done by using the Nyquist plot of $C(z)bG(z)$ under a frequency divider [Tlib05C].

In fact, the FIFO $G(z)$ is an integrator and the controller $C(z)$ is a PI controller. We have a double-integrator in $C(z)bG(z)$, meaning that there are two poles on the unit circle at $(1, 0)$.

In order to demonstrate the properties of our system we use :

1. exact variable change $z = e^{sT_e}$ for the explanation of high frequency behaviour ;

2. approximation $z = e^{sT_e} \approx 1 + sT_e$ for the explanation of low frequency behaviour.

The first point is treated with $s = j\omega$, i.e. $s \in]-j\infty, j0^-]$ and $s \in [j0^+, +j\infty[$, the mapping to $C(z = e^{sT_e})bG(z = e^{sT_e})$-plane is illustrated in Figure 3.31, plotted using MATLAB[6].

[6]Matlab "nyquist()" function uses the variable change $z = e^{j\omega T_e}$ in order to obtain

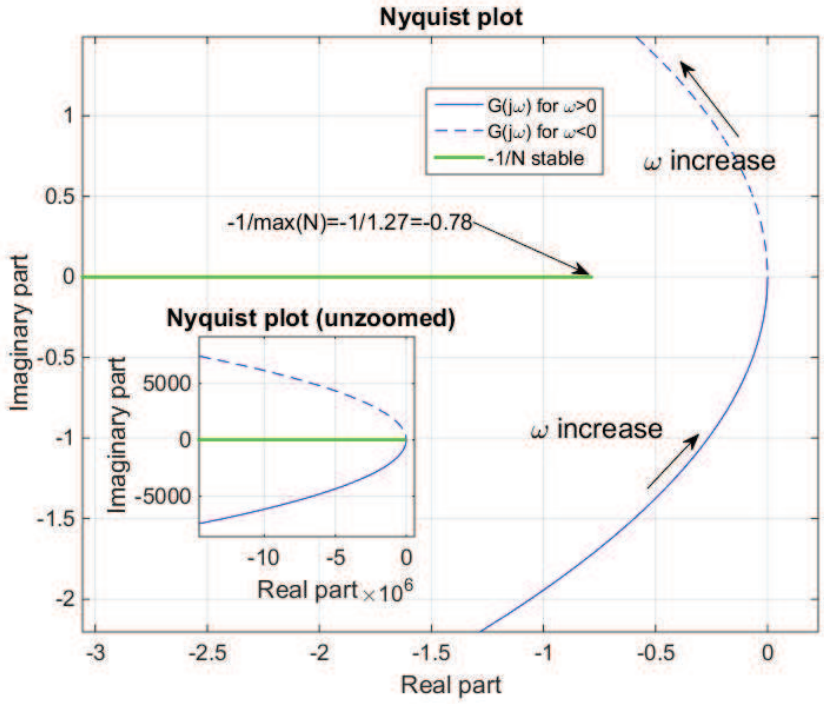

Figure 3.31: Nyquist stability criterion considering the describing function (3.19).

The latter point is treated with the mapping of the detour $s = \rho e^{j\theta}$ (ABC) in s-plane with $\rho << 1$ and $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$. For our case of $L(z)$ with $z \approx 1 + sT_e$ gives $L(z) = C(z)bG(z) \approx L(s)$ in the form :

$$L(s) = \frac{k(s + \omega_0)}{s^2} \tag{3.20}$$

where $k = \frac{b(K_p+K_i)}{T_e}$, $\omega_0 = \frac{K_i}{T_e(K_i+K_p)}$ are from the discrete time representation. Putting $s = \rho e^{j\theta}$ into (3.20) gives:

$$L(\rho e^{j\theta}) = \frac{k(\rho e^{j\theta} + \omega_0)}{\rho^2 e^{2j\theta}} = \frac{k\omega_0}{\rho^2} e^{-2j\theta}. \tag{3.21}$$

For θ from $-\pi/2$ to $\pi/2$, the phase of $L(j\omega)$ is -2θ from π to $-\pi$. The mapping from the detour $s = \rho e^{j\theta}$ (ABC) in s-plane to the $L(s)$-plane is a circle A'B'C' in the clockwise direction. The radius of this cirle is $|L(j\omega)| = \frac{k\omega_0}{\rho^2}$.

nyquist plot.

Remark *The approximation of z-transform variable* $z = e^{sT_e} \approx 1 + sT_e$ *is very precise for* $s = j\omega$ *with* $\|\omega\| << 1$*, thus conserving the static gain of the original discrete time transfer function. High frequency behaviour has considerable residuals in the frequency response between continous and discrete tranfer functions since the consideration* $z = e^{sT_e} \approx 1 + sT_e$ *is an approximation of* z *about* $s = j0$*. That is why we use exact variable change* $z = e^{sT_e}$ *for high frequency studies.*

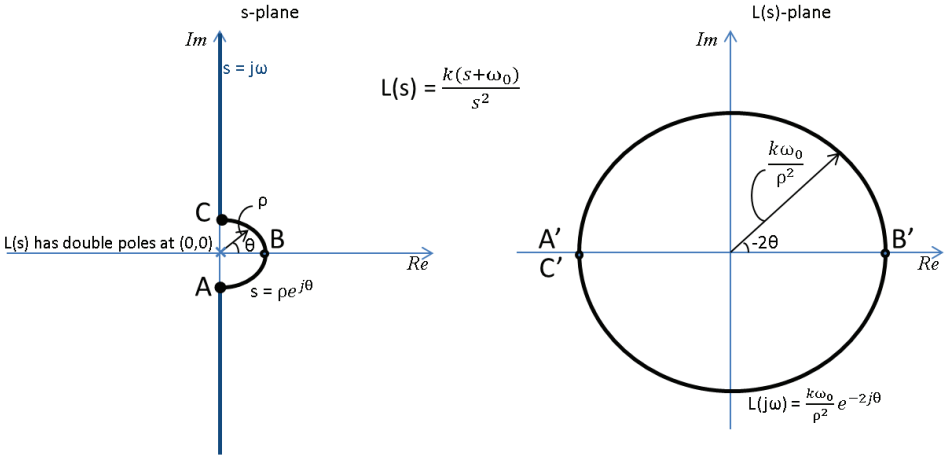

Figure 3.32: Nyquist plot of $L(s) = \frac{k(s+\omega_0)}{s^2}$ for the detour $s = \rho e^{j\theta}$.

Because the radius ρ of the detour is infinitesimal ($\rho \ll 1$), when $\rho \to 0$, the radius of the mapping $L(s) = \frac{k(s+\omega_0)}{s^2}$ is $|L(s)| = \frac{k\omega_0}{\rho^2} \to \infty$.

As concluded above, the Nyquist plot $-1/N(f_C)$ is the finite interval $[-1/N_{min}(f_C); -1/1.27]$ on the left half of the real axis. The Figure 3.32 and Figure 3.31 show that the Nyquist plot of $C(j\omega)bG(j\omega)$ and $-1/N(f_C)$ do not intersect and the number of encirlements of $C(j\omega)bG(j\omega)$ around $-1/N(f_C)$ is zero.

The conclusion is that the frequency divider does not affect the stability of the closed loop system when the calculated frequency is $f_C > 6.25\ MHz$.

- When the calculated frequency $f_C \leq 6.25 MHz$, the applied frequency f_{prac} is zero.

In this case, with whatever $f_C \leq 6.25\ MHz$ value of the calculated frequency input, the applied output will be zero, i.e. $f_{prac} = 0Hz$. This means that $-1/N(f_C)$ goes to $-\infty$. $C(j\omega)bG(j\omega)$ intersects $-1/N(f_C)$ at

the point $(-\infty, 0)$. This point is the limitation of the segment $-1/N(f_C)$. In other words, whatever change we make on f_C, the critical point can only be moved on the real axis, from $(-\infty, 0)$ to $(-1/N_{max}(f_C), 0)$. In this region, the system is stable as already proved above. We conclude that even at the intersection point $(-\infty, 0)$, the system is stable.

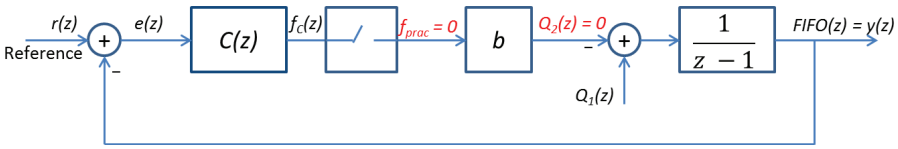

Figure 3.33: Open-loop system when the calculated $f_C < 6.25$.

Indeed, this means that the consumer stops consuming data, in other words, $Q2 = 0$. The closed-loop system becomes an open-loop one, see Figure 3.33. The input of the FIFO only depends on the disturbance $Q1$. This disturbance is integrated in the FIFO plant and it has effects on the input of the controller. Since the disturbance is non-negative, the magnitude of the error signal $e(z)$ will be gradually increased. This increases the controller output f_C. Thus, f_{prac} will certainly be positive. When f_{prac} is positive, the loop is closed, our system is stable as proved above.

To conclude, when the calculated frequency is non-positive, our system is conditionally stable. The condition for stability is that the disturbance signal $Q1$ is non-negative. In the FIFO model, the disturbance signal $Q1$ is the data written to the FIFO, whose value is either 0 or 1. This will turn into the case of positive f_{prac}, where the stability is already proved. Hence, our system is stable.

Using the Nyquist criterion **we prove the stability of our system** that contains a cascade combination of a PI controller and an integrator in the forward part and a non-linearlity in the feedback part. However, in our application system, in order to apply the Nyquist criterion, many approximations are required. Indeed, with the hypothesis that the linear part is low-pass, the describing function, which is the first harmonic approximation, is used to represent the non-linear part of the system. Moreover, to demonstrate properties of the linear part $C(z)bG(z)$ of low frequency behaviour, we use the approximation $z = e^{sT_e} \approx 1 + sT_e$.

Therefore, we introduce another approach for system stability analysis using the topological separation condition, which is more general than the

Nyquist criterion. This approach allows to consider worst case behaviour of our non-linear system. This is a strong method that can be used independently from the system characteristic (continuous, discrete, linear, non-linear, etc.). The topological separation stability condition can be directly applied to our system containing discrete LTI part and the non-linearity. The mathematical sufficient stability conditions are found and their graphical interpretation allow to prove the stability of our system.

3.2.4.3 Topological separation condition: summary

The topological separation presented hereafter is a stability condition, applied to an interconnection between a stable LTI open-loop system and a feedback which may be linear, non-linear, uncertain, etc. To ease the application of this stability condition, one may need to use the loop shifting technique that is now explained.

Notice that the topological separation stability condition already exists. We provide here a summary of the method and its application to our system.

- **Loop shifting technique**

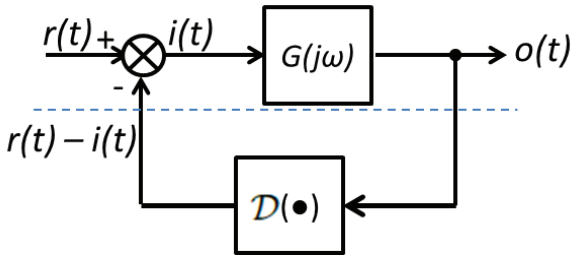

Figure 3.34: The closed-loop system contains a LTI plant $G(j\omega)$ and a negative feedback $\mathcal{D}(\bullet)$.

Consider a system containing a LTI (Linear Time-Invariant) plant $G(j\omega)$ and a negative feedback with block $\mathcal{D}(\bullet)$, see Figure 3.34.

By performing a loop shifting with $\delta > 0$, see Figure 3.35, an equivalent system containing $\tilde{G}(j\omega)$ and $\tilde{\mathcal{D}}(\bullet)$ is obtained. The new plant $\tilde{G}(j\omega) = \frac{G(j\omega)}{1+\delta G(j\omega)}$ can be represented as $D + C(e^{j\omega T_s} I - A)^{-1} B)$ which is stable with $\lambda(A) < 1$. The output $\tilde{i}(t)$ from $\tilde{\mathcal{D}}(\bullet)$ with the input $o(t)$ can be represented as follows: $\tilde{i}(t) = \tilde{\mathcal{D}}(o(t)) = \delta o(t) - \mathcal{D}(o(t))$. The stability of the system in Figure 3.34 is equivalent to the stability of the one in Figure 3.35.

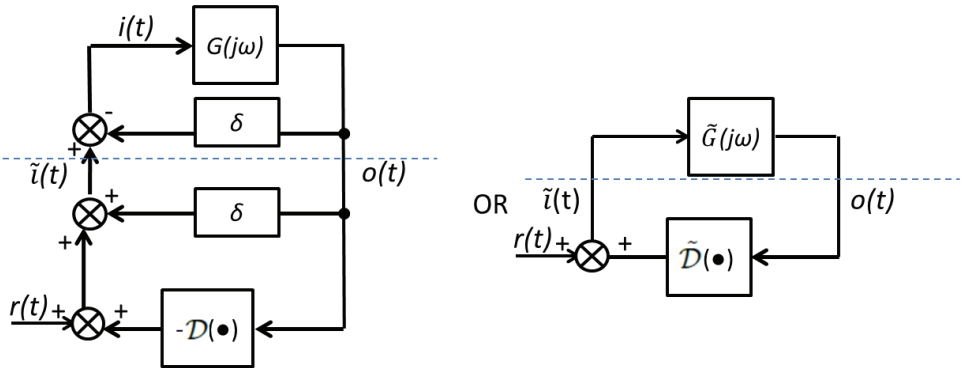

Figure 3.35: The closed-loop system with loop shifting.

- **The topological separation condition**

The topological separation condition of the stability of a system [Safo80S] can be abstractedly stated as follows. The system containing a stable LTI part $\tilde{G}(j\omega)$ and a static feedback $\tilde{\mathcal{D}}(\bullet)$ can be proved to be stable if there exists a topological seperation between the graphs of $\tilde{G}(j\omega)$ and $\tilde{\mathcal{D}}(\bullet)$.

- **The quadratic constraint approach**

The quadratic constraint approach [Hill76T, Goh95R] can be used to construct the graph seperator. Firstly, it is used to find a characterization for $\tilde{\mathcal{D}}(o)$:

$$\int_{-\infty}^{+\infty} \begin{bmatrix} \tilde{i}(t) \\ o(t) \end{bmatrix}^T \begin{bmatrix} x & y \\ y & z \end{bmatrix} \begin{bmatrix} \tilde{i}(t) \\ o(t) \end{bmatrix} dt \geq 0, \forall t, \forall \tilde{i}, o. \tag{3.22}$$

where $\tilde{i}(t) = \tilde{\mathcal{D}}(o(t))$.

Using this quadratic constraint, one can define the $\{x, y, z\}$ triplet such that the Equation (3.22) is verified. The found $\{x, y, z\}$ triple is the parameter of the graph seperator where the graph of $\tilde{\mathcal{D}}(o)$ is outside of the $\{x, y, z\}$ seperator.

The stability of the system in Figure 3.35 can be proved if the graph of $\tilde{G}(j\omega)$ verifies the $\{x, y, z\}$ seperator:

$$\int_{-\infty}^{+\infty} \begin{bmatrix} \tilde{i}(t) \\ o(t) \end{bmatrix}^T \begin{bmatrix} x & y \\ y & z \end{bmatrix} \begin{bmatrix} \tilde{i}(t) \\ o(t) \end{bmatrix} dt < 0 \tag{3.23}$$

for every t, where $o(t)$ is the output of $\tilde{G}(j\omega) = \frac{G(j\omega)}{1+\delta G(j\omega)}$ with the input $\tilde{i}(t)$.

Based on Parseval-Plancherel equality [Megr97S], (3.23) is equivalent to:

$$\int_{-\infty}^{+\infty} \begin{bmatrix} \tilde{i}(j\omega) \\ o(j\omega) \end{bmatrix}^* \begin{bmatrix} x & y \\ y & z \end{bmatrix} \begin{bmatrix} \tilde{i}(j\omega) \\ o(j\omega) \end{bmatrix} d\omega < 0 \tag{3.24}$$

for every frequency ω and signal $\tilde{i}(j\omega)$, where $o(j\omega) = \tilde{G}(j\omega)\tilde{i}(j\omega) = \frac{G(j\omega)}{1+\delta G(j\omega)}\tilde{i}(j\omega)$.

The inequality (3.24) is equivalent to

$$\int_{-\infty}^{+\infty} i(j\omega)^* \begin{bmatrix} I \\ G(j\omega) \end{bmatrix}^* \begin{bmatrix} x & y \\ y & z \end{bmatrix} \begin{bmatrix} I \\ G(j\omega) \end{bmatrix} i(j\omega)d\omega < 0, \forall\omega, \forall i(j\omega). \tag{3.25}$$

$$\Leftrightarrow \begin{bmatrix} I \\ G(j\omega) \end{bmatrix}^* \begin{bmatrix} x & y \\ y & z \end{bmatrix} \begin{bmatrix} I \\ G(j\omega) \end{bmatrix} < 0, \forall\omega. \tag{3.26}$$

The problem of verifying the inequality (3.26) for every frequency ω is NP-hard [Safo09R]. To reduce the complexity, the Kalman-Yakubovich-Popov (KYP) lemma can be used.

- **The Kalman-Yakubovich-Popov (KYP) lemma**

The KYP lemma for discrete-time system is stated as follows [Rant96O, Dinh05P].

Given a linear discrete dynamical system whose transfer function is $\hat{T}(j\omega) = \hat{C}\left(e^{j\omega T_s}I - \hat{A}\right)^{-1}\hat{B}+\hat{D}$ *where* $\hat{A} \in \mathbb{R}^{n\times n}$ *so that* $\det\left(e^{j\omega T_s}I - \hat{A}\right) \neq 0$, $\hat{B} \in R^{n\times l}$ *and the pair* $(\hat{A}, \hat{B})$ *is controllable and the matrices* $\hat{C} \in R^{k\times n}$, $\hat{D} \in R^{k\times l}$ *and given* $\mathcal{M} = \mathcal{M}^T \in R^{k\times k}$, *the following two statements are equivalent:*

a) There exists a matrix $P = P^T \in \mathbb{R}^{n\times n}$ *such that:*

$$\begin{bmatrix} \hat{A}^T P\hat{A} - P & \hat{B}^T P\hat{A} \\ \hat{A}^T P\hat{B} & \hat{B}^T P\hat{B} \end{bmatrix} + \begin{bmatrix} \hat{C}^T \\ \hat{D}^T \end{bmatrix} \mathcal{M} \begin{bmatrix} \hat{C} & \hat{D} \end{bmatrix} \leq 0 \tag{3.27}$$

is verified.

b) The transfer function $\hat{T}(j\omega)$ *verifies:*

$$\forall\ \omega \in \mathbb{R}^+, \quad \hat{T}(j\omega)^*\mathcal{M}\hat{T}(j\omega) \leq 0, \tag{3.28}$$

For strict inequalities, the lemma is correct even if $(\hat{A}, \hat{B})$ *is not controllable.*

If such matrix $P = P^T > 0$ can be found then the LTI system $\hat{T}(j\omega)$ is stable and (3.28) is verified for every ω. Find the matrix $P = P^T > 0$ is an optimization problem and it can be solved efficiently using the Linear Matrix Inequality (LMI) formulation [Boyd94L]

To summarize, the system in Figure 3.35 is proved to be internally stable if one can find a $\{x, y, z\}$ seperator that verifies the two following conditions:

1. The graph of $\tilde{\mathcal{D}}$ verifies the quadratic separator, meaning that

$$\int_{-\infty}^{+\infty} \begin{bmatrix} \tilde{i}(t) \\ o(t) \end{bmatrix}^T \begin{bmatrix} x & y \\ y & z \end{bmatrix} \begin{bmatrix} \tilde{i}(t) \\ o(t) \end{bmatrix} dt \geq 0, \forall t, \forall \tilde{i}, o \tag{3.29}$$

   where $\tilde{i}(t) = \tilde{\mathcal{D}}(o(t))$.

2. Using the same $\{x, y, z\}$, the graph of $\tilde{G}(j\omega)$ verifies:

$$\int_{-\infty}^{+\infty} \begin{bmatrix} \tilde{i}(t) \\ o(t) \end{bmatrix}^T \begin{bmatrix} x & y \\ y & z \end{bmatrix} \begin{bmatrix} \tilde{i}(t) \\ o(t) \end{bmatrix} dt < 0 \tag{3.30}$$

$$\Leftrightarrow \int_{-\infty}^{+\infty} \begin{bmatrix} \tilde{i}(j\omega) \\ o(j\omega) \end{bmatrix}^* \begin{bmatrix} x & y \\ y & z \end{bmatrix} \begin{bmatrix} \tilde{i}(j\omega) \\ o(j\omega) \end{bmatrix} d\omega < 0, \forall \omega, \forall \tilde{i}, o \tag{3.31}$$

   for every frequency ω, where $o(t)$ is the output of $\tilde{G}(j\omega) = \frac{G(j\omega)}{1+\delta G(j\omega)}$ with the input $\tilde{i}(t)$.

The procedure to prove the closed-loop system stability is as follows:

- Use quadratic constraint to characterize $\tilde{\mathcal{D}}$ by finding the $\{x, y, z\}$ seperator that verifies the first condition.
- With the same $\{x, y, z\}$ triplet, find a matrix $P = P^T > 0$ so that

$$\begin{bmatrix} \hat{A}^T P \hat{A} - P & \hat{B}^T P \hat{A} \\ \hat{A}^T P \hat{B} & \hat{B}^T P \hat{B} \end{bmatrix} + \begin{bmatrix} \hat{C}^T \\ \hat{D}^T \end{bmatrix} \begin{bmatrix} x & y \\ y & z \end{bmatrix} \begin{bmatrix} \hat{C} & \hat{D} \end{bmatrix} \leq 0 \tag{3.32}$$

  is verified.

- **Maximum tolerable non-linear gain (Robust stability)**

For a closed-loop system proved to be stable as above, one can find the maximum non-linear gain that the system can tolerate by solving the optimization problem with given y, z and $\{\hat{A}\hat{B}\hat{C}\hat{D}\}$-LTI system description:

$$\begin{aligned} &\max \quad x, \\ &x \in \mathbb{R}, \\ &q \in \mathbb{R}, \\ &P \in \mathbb{R}^{n \times n} \end{aligned}$$

$$\textit{such that} \quad \begin{bmatrix} \hat{A}^T P \hat{A} - P & \hat{B}^T P \hat{A} \\ \hat{A}^T P \hat{B} & \hat{B}^T P \hat{B} \end{bmatrix} + \begin{bmatrix} \hat{C}^T \\ \hat{D}^T \end{bmatrix} \begin{bmatrix} x & y \\ y & z \end{bmatrix} \begin{bmatrix} \hat{C} & \hat{D} \end{bmatrix} < 0$$

$$\begin{aligned} &x < 0, \\ &x > -q, \\ &q > 0, \end{aligned} \tag{3.33}$$

3.2.4.4 Application of the topological separation condition to our system

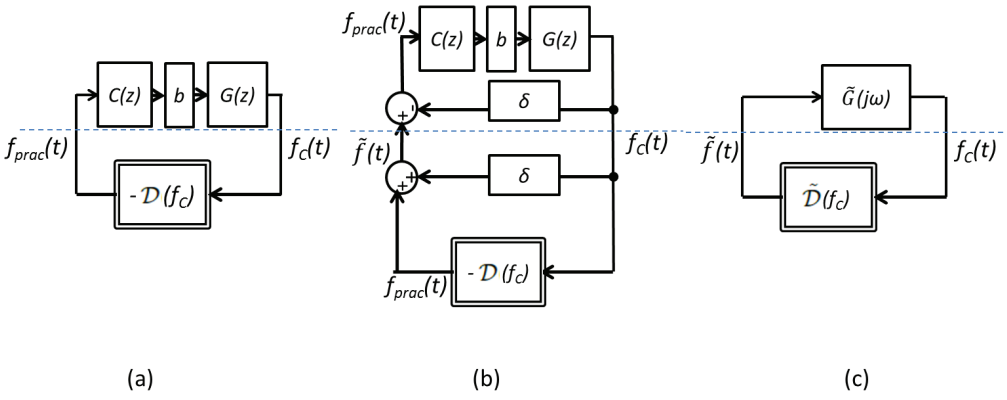

Figure 3.36: System to which the topological separation stability condition is applied.

As presented above, our system can be considered as a Linear Time-Invariant (LTI) $C(z) \cdot b \cdot G(z)$ plant and a non-linear feedback $\mathcal{D}(f_C)$, see

Figure 3.36-a, where the non-linearity comes from the quantizer used to map the calculated frequency f_C to the practical frequency f_{prac} applied.

The LTI part is a cascade combination of a PI controller and an integrator. The transfer function is:

$$C(z) \cdot b \cdot G(z) = \frac{0.75z - 0.625}{z^2 - 2z + 1} \tag{3.34}$$

To ease the application of the topological separation condition, firstly a loop shifting is performed to our system, with $\delta = 2.5$, see Figure 3.36-b,c. The transfer function of the loop shifted plant $\tilde{G}(z)$ is:

$$\tilde{G}(z) = \frac{0.75z - 0.625}{z^2 - 0.125z - 0.5625} \tag{3.35}$$

which is stable.

The minimal state space representation of the loop shifted transfer function is:

$$\begin{aligned} A &= \begin{bmatrix} 0.125 & 0.5625 \\ 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \\ C &= \begin{bmatrix} 0.75 & -0.625 \end{bmatrix}, \quad D = 0. \end{aligned} \tag{3.36}$$

The non-linear part is a quantizer whose input and output are presented in Figure 3.37. In the case where the non-linear function is situated in a sector, it is sufficient to use equivalent gain $-\mathcal{D}(f_C) = \frac{f_{prac}}{f_C}$ for representing this static frequency-independent non-linearity. Note that $\mathcal{D}(f_C) \in [\mathcal{D}_{min}(f_C), \mathcal{D}_{max}(f_C)]$ where $\mathcal{D}_{min}(f_C) = 0$ and $\mathcal{D}_{max}(f_C) = 2$. Then, the loop shifted non-linear block $\tilde{\mathcal{D}}(f_C)$ is represented using the gain $\tilde{\mathcal{D}}(f_C) = \frac{\tilde{f}}{f_C} = \frac{f_{prac}+\delta f_C}{f_C} = \frac{-\mathcal{D}(f_C)f_C+\delta f_C}{f_C} = \delta - \mathcal{D}(f_C)$. The plot of the loop shifted non-linearity $\tilde{\mathcal{D}}(f_C)$ is, hence, a finite segment $[\tilde{\mathcal{D}}_{min}(f_C), \tilde{\mathcal{D}}_{max}(f_C)] = [0.5, 2.5]$ on the complex plane (see the red segment in the Figure 3.38).

Using the quadratic constraint approach, we can find the $\{x, y, z\}$ characterization of the graph of $\tilde{\mathcal{D}}(f_C)$. The graph of $\{x, y, z\}$ is the separator that isolates the two graphs $\tilde{\mathcal{D}}(f_C)$ and $\tilde{G}(z)$ in two separated parts of the plane. We consider here two applications as follows.

In the first application, the $\{x, y, z\}$ separator is a line that leaves the graph of $\tilde{\mathcal{D}}(f_C)$ on the right semiplane. With $\{x, y, z\} = \{-2.76, 3, 0\}$, the separator is the vertical line $c = 0.46$. Figure 3.38 shows that the blue line is the separation where the finite segment $[\tilde{\mathcal{D}}_{min}(f_C), \tilde{\mathcal{D}}_{max}(f_C)]$ is on

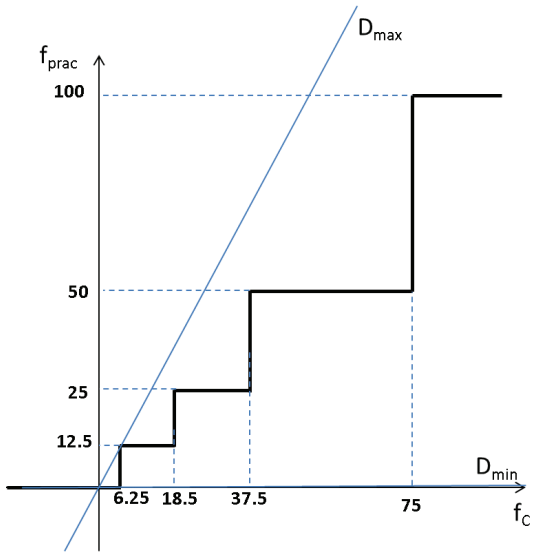

Figure 3.37: Input f_C and output f_{prac} of the non-linear block which is a quantizer.

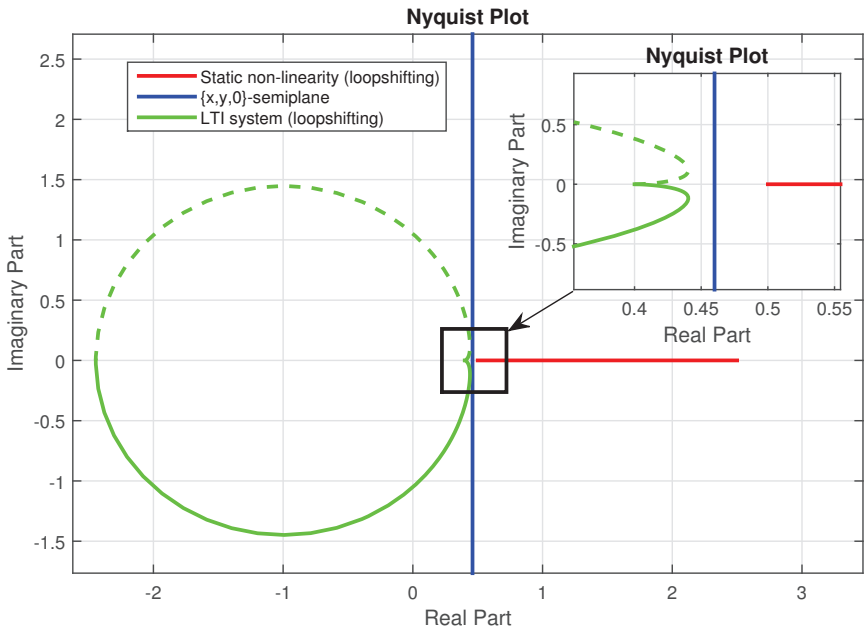

Figure 3.38: Nyquist plot with the semiplane separation.

the right of the line and the Nyquist plot $\tilde{G}(j\omega)$ is on the left. Using the Robust control toolbox, we can efficiently solve the LMI problem to find matrix $P = P^T > 0$ which verifies the KYP lemma on page 131 with:

$\hat{A} = A, \hat{B} = B, \hat{C} = \begin{bmatrix} 0 & C^T \end{bmatrix}^T, \hat{D} = \begin{bmatrix} 1 & D^T \end{bmatrix}^T$. In this case, the matrix P is:

$$P = \begin{bmatrix} 2.40 & -1.91 \\ -1.91 & 1.61 \end{bmatrix} \tag{3.37}$$

with eigenvalues $\lambda_1 = 5.22 \times 10^{-2}$ and $\lambda_2 = 3.96$.

An optimization problem can be solved to find the maximum possible gain for the non-linearity of the original closed-loop system $\frac{G \cdot C \cdot b}{1 + G \cdot C \cdot b \cdot \mathcal{D}(\bullet)}$. In the current $\{x, y, 0\}$-semiplane application, the maximum x that verifies the robust stability condition on page 132 is $x_{opt} \approx -2.64$. The separator is the vertical line $c = 0.44$. The matrix P_{opt} is:

$$P_{opt} = \begin{bmatrix} 2.25 & -1.9 \\ -1.9 & 1.66 \end{bmatrix} \tag{3.38}$$

with eigenvalues $\lambda_1 = 3.62 \times 10^{-2}$ and $\lambda_2 = 3.88$. For this optimization case, the $\mathcal{D}_{max} = \delta - c = 2.5 - 0.44 = 2.06$ is the maximum non-linear gain that the system can tolerate under particular δ, y, z.

In the second application, the $\{x, y, z\}$ separator is a circle that encircles $\tilde{G}(j\omega)$ and leaves the graph of $\tilde{\mathcal{D}}(f_C)$ outside. With $\{x, y, z\} = \{-2.3, 2, 2\}$, the centre is $(c, 0)$ with $c = -\frac{y}{z} = -1$ and the radius is $r = \sqrt{\frac{y^2}{z^2} - \frac{x}{z}} \approx 1.47$. In the Figure 3.39, the blue circle is the graph separator where the finite segment $[\tilde{\mathcal{D}}_{min}(f_C), \tilde{\mathcal{D}}_{max}(f_C)]$ is outside of the circle and the Nyquist plot $\tilde{G}(j\omega)$ is inside. The LMI problem is found to be feasible with $P = P^T > 0$

$$P = \begin{bmatrix} 2.25 & -1.79 \\ -1.79 & 1.52 \end{bmatrix} \tag{3.39}$$

which verifies the KYP lemma on page 131. The eigenvalues of P are $\lambda_1 = 6.02 \times 10^{-2}$ and $\lambda_2 = 3.70$.

Similarly, an optimization problem can be solved to find the maximum tolerable non-linear gain of the system $\frac{G \cdot C \cdot b}{1 + G \cdot C \cdot b \cdot \mathcal{D}(\bullet)}$. In the current $\{x, y, z\}$-circle application, the maximum x that verifies the robust stability condition on page 132 is $x_{opt} \approx -2.21$. The separator is the circle with the same centre $(-1, 0)$ and the radius $r = \sqrt{\frac{y^2}{z^2} - \frac{x}{z}} \approx 1.45$. The matrix P_{opt} is:

$$P_{opt} = \begin{bmatrix} 2.21 & -1.78 \\ -1.78 & 1.49 \end{bmatrix} \tag{3.40}$$

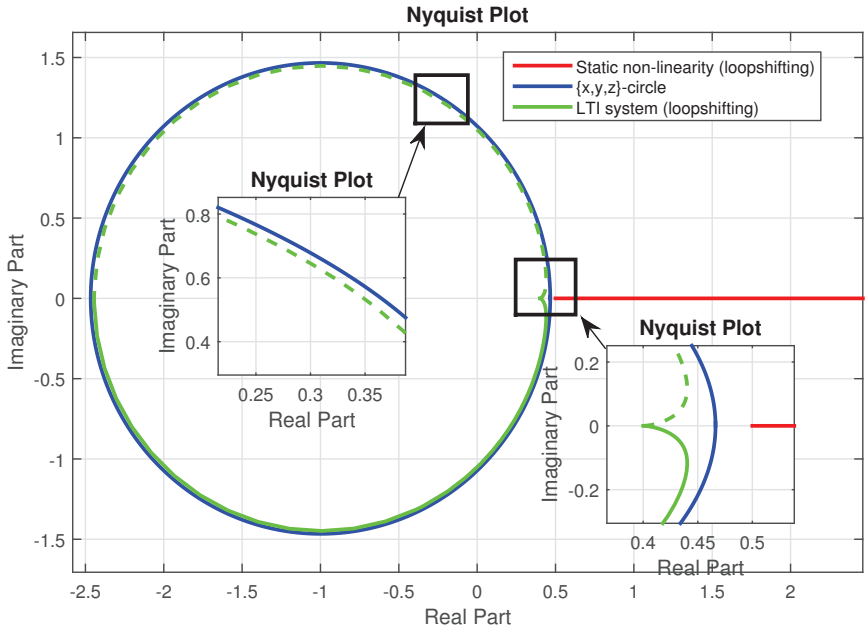

Figure 3.39: Nyquist plot with the circle separation.

with eigenvalues $\lambda_1 = 3.59 \times 10^{-2}$ and $\lambda_2 = 3.66$. For this optimization case, the value $\mathcal{D}_{max} = \delta - r - c = 2.5 - 1.45 + 1 = 2.05$ is the maximum non-linear gain that the system can tolerate under particular δ, y, z.

To conclude, in two applications of the $\{x, y, z\}$ separator, namely $\{x, y, 0\}$-semiplane and $\{x, y, z\}$-circle, the LMI feasibility problem is solved by finding the matrix $P = P^T > 0$ that verifies the KYP lemma. The closed-loop system with interconnected $\tilde{G}(j\omega)$ and $\tilde{\mathcal{D}}(\bullet)$ is stable. Equivalently, **the original system containing the Linear Time-Invariant (LTI)** $C(z) \cdot b \cdot G(z)$ **plant and the non-linear feedback** $-\mathcal{D}(\bullet)$ **is also stable**.

In this application, the topological separation approach is used only for stability analysis. However, it is also useful for performance analysis, and furthermore, robust performance analysis, when the consuming speed of the consumer module is constrained. Moreover, the implementation of the DFS method may contain some uncertainties, especially in hardware implementation. Characterization of these uncertainties in finding some $\{x, y, z\}$ can be used to define the worst-case bound of the uncertainties that the system can tolerate. Inversely, one can find what is the worst-case performance for a known uncertainty. With whatever uncertainties $\mathcal{D}(\bullet)$ whose graph is maintained outside the $\{x, y, z\}$ separator, the system is

still stable. These analysis problems can be considered as a perspective work of the thesis.

3.3 MATLAB simulation results

As mentioned in the FIFO link modeling in Subsection 3.2.1, we performed the simulation of the VENGME platform encoding benchmark video frames to trace data flow in the FIFO link. The traced data contains the producer frequency f_P, the status of the producer input, the status of the producer output (stall or not), the number of packets written into FIFO $Q1$, the FIFO level $FIFO$, the consumer frequency, the status of the consumer input (stall of not) and the number of packets read from FIFO $Q2$ at time instants t.

From the data in the simulation trace, the system and controller have been modeled using the MATLAB environment. In this simulation, the maximal FIFO level is 7, due to silicon resource limitation, and the reference level is chosen equal to 2. As explained in Section 3.2.2, a non-uniform quantizer is used to map the calculated frequency from the controller to the frequency applied on the consumer. The consumer frequency is taken in the set of values $\{0, 12.5, 25, 50, 100\}$ *MHz*. This discrete set of values for f_C adds non-linearities in the system. For getting the simplicity, these non-linearities are not considered in the simulation. However, the anti-windup mechanism is applied.

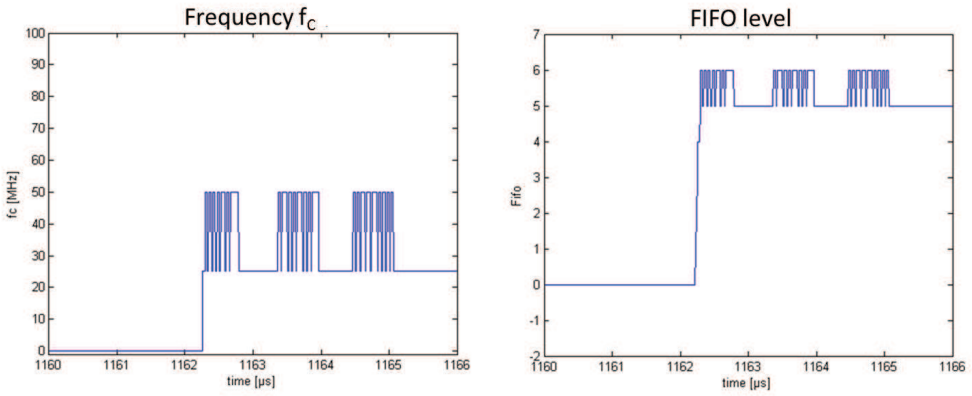

Figure 3.40: Simulation results for closed-loop poles $z_{1,2} = 0.5 \pm 0.2i$ (left: clock frequency f_C, right: FIFO level).

The choice of the closed-loop poles is highly important because they will impose the closed-loop dynamics. Firstly, a fast closed-loop dynamics

response is considered and the poles $z_{1,2} = 0.5 \pm 0.2i$ are chosen as an example. Figure 3.40 shows the closed-loop behavior of the FIFO level (right) together with the clock frequency value (left). It is seen that even if the FIFO level is high, the FIFO is not full. The consumer frequency is mapped to $25MHz$ or $50MHz$. Even if the behavior is satisfactory (taking into account the disturbance Q1), from an implementation point-of-view, this pole choice will imply strong constraints on the clock frequency engine (i.e. actuator) that delivers f_C. Actually, this latter will react as soon as a change in the system is detected, leading to many changes in the clock frequency engine output. Note that these fast changes will add extra power consumption.

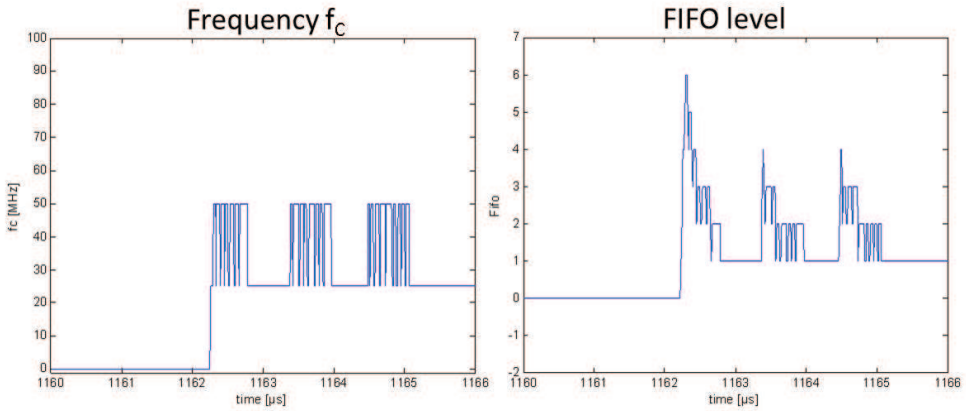

Figure 3.41: Simulation results for closed-loop poles $z_{1,2} = 0.9$ (left: clock frequency f_C, right: FIFO level).

As a consequence, another pole selection is proposed, $z_{1,2} = 0.9$, in order to slow down the closed-loop dynamics and impose less constraints on the clock frequency engine. The simulation results are shown in figure 3.41. For this pole selection, the FIFO level is changed gradually, in comparation to the first pole selection. As expected, the behavior is consistent with the control requirements while the clock frequency engine supports less dynamics constraints.

Both simulation cases show the effectiveness of the PI control for the FIFO level. The controller must now be implemented in hardware (i.e. in Silicon, not in Software). This Silicon implementation will impose extra constraints in order to limit the Silicon area of the controller. The implementation details and related results will be presented in Chapter 4.

3.4 Conclusion

In this chapter, a method to dynamically scale the clock frequency based on control theory is proposed. The control input is the FIFO occupancy level. A FIFO-based system containing FIFO links, where each link connects producer and consumer, was firstly modeled. A PI controller scheme that is simple, possibly implemented in hardware and independent from the application is designed.

The control system stability is analyzed. Because the non-uniform quantizer for frequency mapping adds non-linearlities in the system, the stability analysis using poles position is not sufficient. The method of stability analysis using the describing fucntion approach is applied. According to the Nyquist stability criterion, our system is proved to be stable. A more general approach, namely, the topological separation condition, is also applied to our system. The advantage is that this approach can be used directly to our system without any approximation. The system is also proved to be stable under the topological separation condition.

MATLAB simulations are performed. The results also give a first idea on how the system will behave according to the pole value selection. The pole selection will be consider carefully later, from the implementation constraints view point. The details on implementation and verification and the validation results will be presented in Chapter 4.

Chapter 4

Implementation and validation of the FIFO-level based DFS method

As analyzed in Chapter 2, the VENGME platform has nice but not outstanding power results. We can improve its power characteristics by implementing Dynamic Power Management (DPM) methods. Dynamic Voltage and Frequency Scaling (DVFS) is an efficient method that reduces both energy and power consumption. Moreover, DVFS can be applied after the design phase to dynamically decrease the power during operation. Due to the implementation constraints and the design at hand, DFS is firstly applied. Then, DVS can be developed to perform DVFS.

Chapter 3 has introduced the FIFO-based DFS method to reduce power consumption for every FIFO-based system, as the VENGME video encoder. The method is expected to be implemented for buffers embedded in the VENGME platform. However, in the first step, the FIFO/buffer link to transfer data between the Entropy Coder (EC) and the bytestream Network Abstraction Layer data packer (NAL) of the EC-NAL module is selected to implement this FIFO-based DFS method because the EC-NAL module has a large power consumption in comparison to the other VENGME modules that work for all video frames. Another reason is that the EC-NAL module design is one of the contributions of this work, so that, deep understanding of the module architecture enables modification and data tracing during

simulation to verify the DFS method.

In Chapter 3, the system modeling, the control design and stability analysis have been presented. The MATLAB simulation results give some early idea on how the system will behave according to the pole value selection. The method is then implemented in hardware. The details of implementation and its verification/validation results are presented hereafter.

4.1 Hardware design and implementation

In order to implement the FIFO-level-based DFS method to the FIFO link embedded in the EC-NAL module, the original EC-NAL module has to be split into two frequency domains, namely the one contains EC module and the one contains NAL module. The domain containing EC module operates at $50MHz$frequency. The other domain operates at a frequency that is selected by the DFS method actuator, i.e. the PI controller. Figure 4.1 illustrates an overview of these modifications: the original architecture of the EC-NAL module in the higher part and the modified one in the lower part. In comparison to the original EC-NAL module, besides the FIFO modification, some blocks (in blue) are newly added.

Firstly, to implement the interface between two clock domains, in other words, the Clock Domain Crossing (CDC) interface, one must ensure the synchronization of signals to avoid metastability. Synchronizer is used for single-bit signals and asynchronous FIFO is used for multi-bit data transferring. This synchronization will be discussed more in Subsection 4.1.1.

Secondly, to perform frequency scaling, different frequencies must be generated. Because the platform at hand does not contain any phase-locked loop (PLL), at architecture level, we implement some frequency dividers to obtain clock signals of different frequencies. The details of the frequency divider implementation in the EC-NAL module will be presented in Subsection 4.1.2.

The frequency divider in the module EC-NAL generates four clock signals of frequencies 12.5, 25, 50 and 100 *MHz*. A multiplexer for clock signal is implemented to select the operating frequency for the NAL module. To stop the clock, a clock gating cell is used. Both multiplexer and clock gating cell are provided in the cell library.

Finally, the PI controller designed in Chapter 3 is modeled in VHDL and then implemented. To limit the Silicon usage, some constraints affect

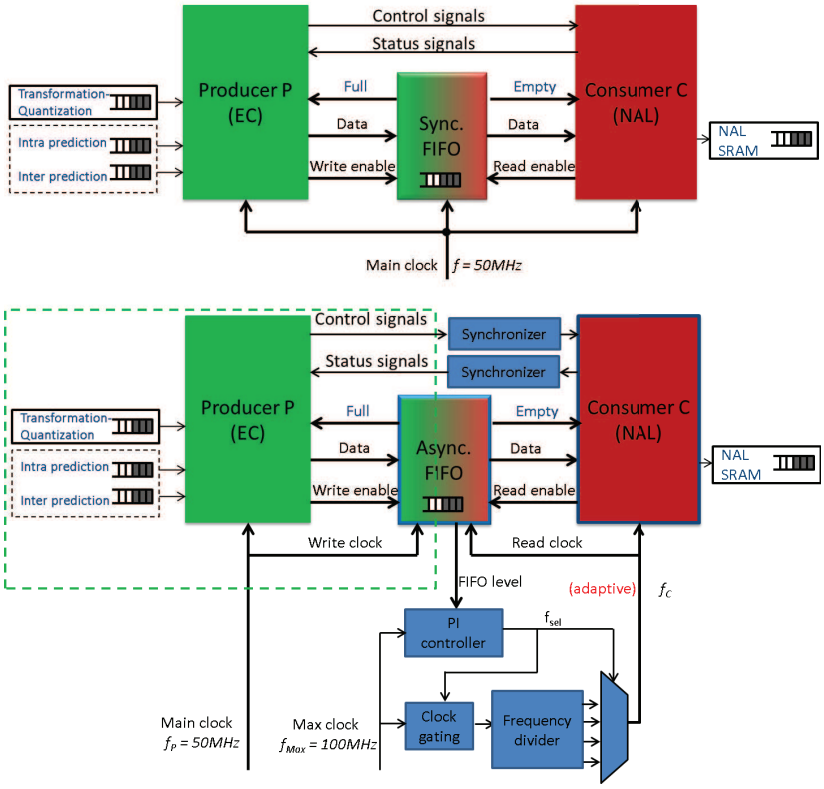

Figure 4.1: Architecture of the original and modified EC-NAL modules to apply the FIFO-level-based DFS method.

the poles selection and the PI implementation. These constraints and the implementation details will be explained in Subsection 4.1.3.

All the implementation details are now presented for the application of the FIFO-level-based DFS method to the EC-NAL module, from the synchronization in the CDC interface to the implementation of the frequency divider and the PI controller.

4.1.1 CDC interface implementation in EC-NAL module

Firstly, we explore metastability in interfacing two clock domains and the use of synchronization techniques for transferring signal and data. Then the implementation of synchronizers and asynchronous FIFO to the CDC interface in the EC-NAL module is explained.

4.1.1.1 Metastability and synchronization techniques

In digital systems, a signal can be in an indeterminate state, i.e. it cannot be determined to be stable at '0' or '1' logic level within the time required for proper circuit operation. In synchronous systems, a clock signal is used to sample signals, if set up and hold times for flip-flop are satisfied, this metastable state can occur without causing a problem.

In CDC interface, the design becomes asychronous at the boundary of two domains, metastability is inherent. Figure 4.2 shows an example of the signal **adat** crossing two clock domain and being sampled, the result is the signal **bdat** that falls into metastable state. The signal **bdat** will be then propagated into the receiving domain. It causes unreliable data transfer. Thus, in CDC design, synchronization techniques [Cumm08C, Jain14S] presented hereafter are required.

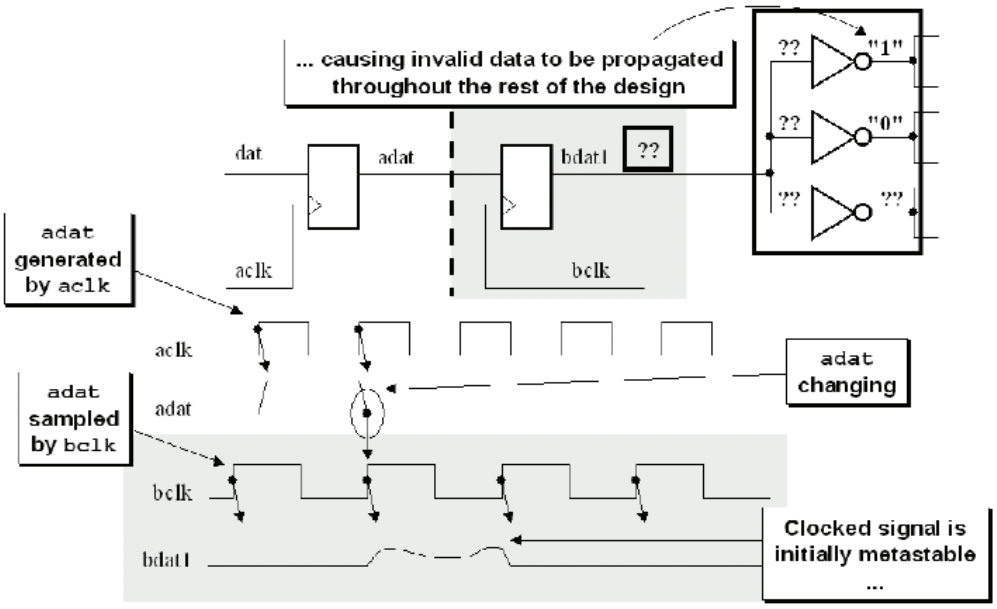

Figure 4.2: Metastable state of CDC signal and the propagation of the signal in the design [Cumm08C].

Synchronizers are used to sample an asynchronous single bit level signal. The synchronizer output is a version of the signal that has transitions synchronized to the local clock.

The conventional two flip-flop synchronizer is the most commonly used by designers. As shown in Figure 4.3, the output of the first flip-flop **B1-q**

may go into a metastable state. However, during one clock cycle of the clock **CLK_B** in the domain **B**, it will be settled to some stable value, thus the output **B2-q** is dertermined. If **B1-q** does not settle into a stable value, then **B2-q** goes to metastable. The Mean Time Before Failure (MTBF) is calculated for CDC signals. Failure means that the CDC signal is still in metastable state after passing the two stages flip-flop synchronizer. Larger MTBF is preferred. It indicates that the probability for the synchronizer output signal to be metastable is close to zero. The MTBF is inversely proportional to the synchronizing clock frequency and the data changing frequency. For high-speed designs, a larger number of stages might be used to increase the MTBF.

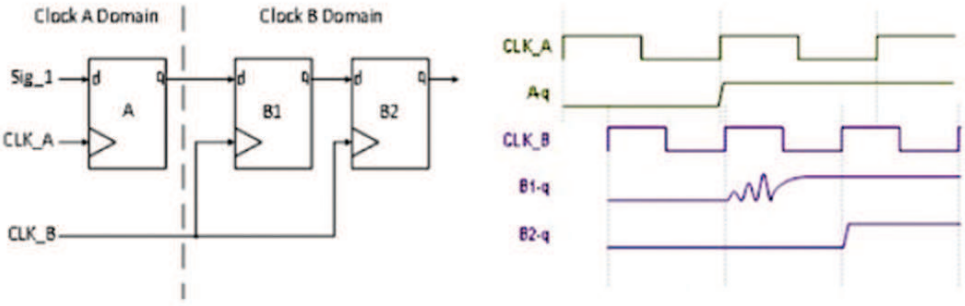

Figure 4.3: Conventional synchronizer with two flip-flops [Jain14S].

The toggle synchronizer, see Figure 4.4, is used for pulse synchronization, i.e. to generate a pulse in the receiving domain when there is a pulse in the sending domain.

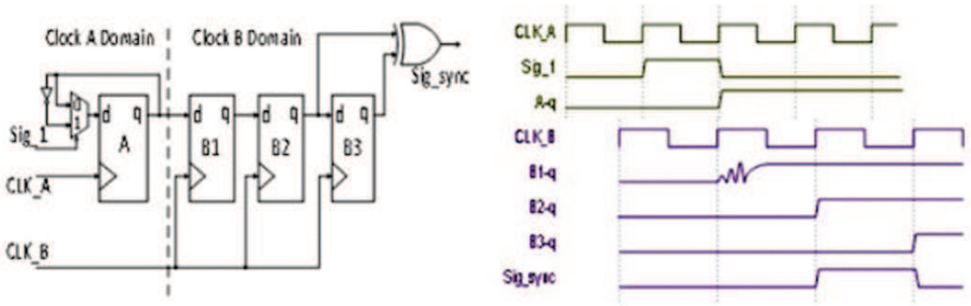

Figure 4.4: Toggle synchronizer for pulse synchronization [Jain14S].

For a multi-bit signal, each bit is sampled using a conventional synchronizer with two flip-flops. However, the outputs of these synchronizers may

not settle to correct value at the same clock. Gray encoding is used to avoid this data incoherency. It ensures that there is only one single bit change at a clock cycle. The use of Gray encoding is illustrated in Figure 4.5 where a Binary to Gray converter is employed in the sending domain and a Gray to Binary converter is used in the receiving domain.

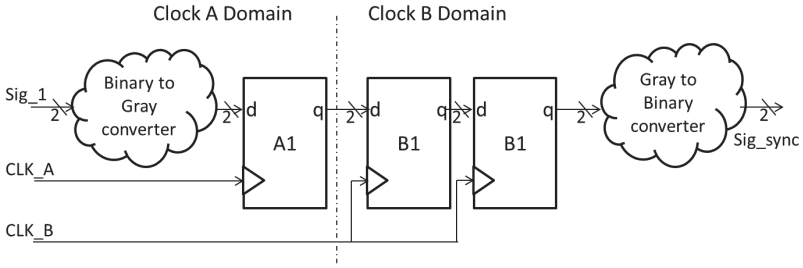

Figure 4.5: Gray encoding used for multi-bit signals [Jain14S].

Asynchronous FIFO implementation is interesting to address our multi-bit CDC signals. A FIFO contains a memory block (or registers) which data can be written into until it is full and read from until it is empty.

A synchronous FIFO employs binary counters for writing and reading pointers. The write pointer points to the next words to be written while the read pointer points to the current word to be read. At the beginning, both pointers are reset at zero, the FIFO is empty. Then, the pointer will be increased after each writing (or reading) operation. The FIFO is considered full when the last writing operation increases the write pointer by one and then it is equal to the read pointer. The FIFO is considered empty when the last reading operation increases the read pointer by one and then it is equal to the write pointer.

For a CDC FIFO, see architecture in Figure 4.6, writing and reading clocks are asynchronous which complicates the pointers comparison and thus, empty/full decision. Because both pointers are asynchronous, each one must be synchronized into the other clock domain for comparison. Hence, Gray encoding is used and Gray code counters are implemented in asynchronous FIFO. These important details in asynchronous FIFO design are explained in [Cumm02S].

For gate-level simulation, metastability in the first stage flip-flop of the CDC synchronizer can cause errors. Then, the signal of value X is propagated into the design. There are many ways, dependent or not on the simulation tools, to address the problem. Using synchronizers from the ASIC

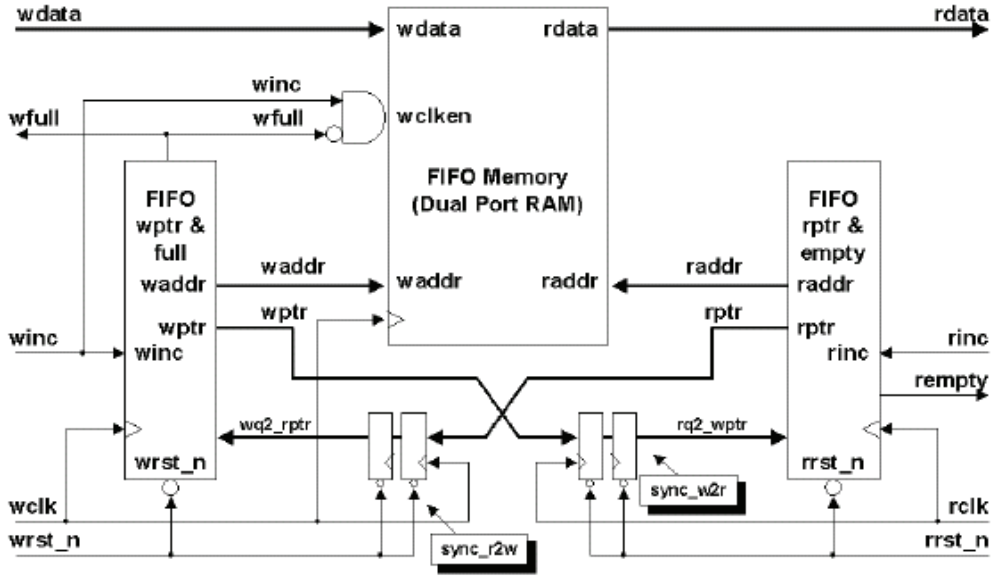

Figure 4.6: Architecture of an asynchronous FIFO [Cumm02S].

library cell is one solution that is independent from the simulation tools. The use of ASIC library cell synchronizers can avoid this X-propagation by annotating the setup and hold time on a synchronizer cell to 0 [Cumm08C].

4.1.1.2 Implementation of the synchronizer

As shown in the original architecture, there are some signals from the Entropy coder to control the NAL data packer and from the NAL data packer to inform its status to the Entropy coder. When these signals pass through two different clock domains, synchronizers must be used. For the gate-level simulation of the EC-NAL module, we directly use the synchronizers provided in the ASIC library cell.

4.1.1.3 Implementation of an asynchronous FIFO

In the EC-NAL module, the entropy encoded data is transferred from the entropy coder (EC) submodule to the NAL data packer submodule via a FIFO. Simulations at RTL level help to decide the appropriate FIFO size so that the data transfer is kept fluently while it does not consume too much Silicon resource. The FIFO size between EC and NAL is chosen equal to 7. To verify the FIFO functionality, it is implemented in the EC-NAL module inside the VENGME platform. During the system simulations, the platform

encodes standard reference videos [VidLib]. These encoded videos are then decoded successfully using the reference software video decoder JM18 [JM].

When the EC-NAL module is split into two clock domains, the FIFO must be an asynchronous one. The FIFO is designed with the details presented above, i.e. using Gray code counters for write and read pointers, synchronizers for pointer comparison and full (empty) decision.

The modified FIFO is also embedded into the VENGME EC-NAL module, replacing the original synchronous FIFO. Simulations at $50MHz$ show that this asynchronous FIFO operates properly.

Simulation results at RTL level, i.e. the Value Change Dump (VDC) file, are then used to estimate the FIFO power consumption using SpyGlass. The results, presented in Section 2.5, page 89, show that the FIFO modification increases only about 6.5% of the total EC-NAL module power consumption. This increment is expected to be small enough compared to the power gain expected by the DFS method implemented in the EC-NAL module.

4.1.2 Implementation of a frequency divider

The NAL operating frequency is selected from a set of available frequencies to perform the DFS. A frequency divider is used to generate different clock frequencies for the NAL module.

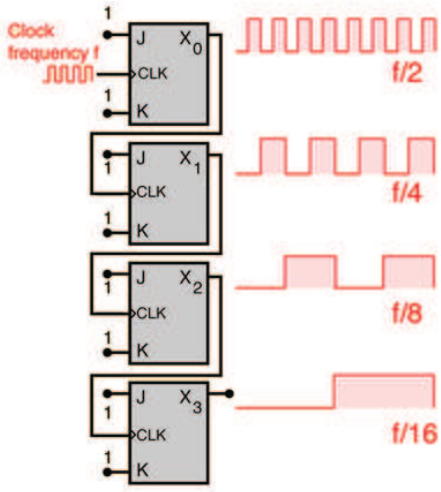

Figure 4.7: A four-bit binary counter used as frequency divider [HyPh].

A frequency divider takes as input a frequency f_{in} and provides as output a frequency f_{out} where f_{out} is a divisor of f_{in}. In digital design, a simple

binary counter can be used for power-of-two integer division. Figure 4.7 shows an example of a four-bit binary counter which is used to divide the input frequency f to obtain output frequencies $\frac{f}{2}$, $\frac{f}{4}$, $\frac{f}{8}$ and $\frac{f}{16}$.

In the VENGME platform, there are two clock signals, namely the main clock of frequency $50MHz$ and the max clock of frequency $100MHz$. The max clock is chosen to be the frequency divider input. A three-bit counter is implemented to perform frequency division. The set of output frequencies is {12.5, 25, and 50} *MHz*.

4.1.3 Implementation of the PI controller

The hardware implementation of the controller which has been designed in Chapter 3 is now presented. The pole selection has effect on the controlled system behaviour. Due to hardware constraints and dynamic constraints in pole selection, the controller coefficients are computed.

4.1.3.1 Implementation of the PI controller

From the controller transfer function (in the z-domain)

$$\frac{f_C(z)}{E(z)} = C(z) = K_p + K_i \frac{z}{z-1} \tag{4.1}$$

$$zf_C(z) - f_C(z) = (K_p + K_i)zE(z) - K_pE(z) \tag{4.2}$$

the recurrence equation is derived:

$$f_C(k+1) - f_C(k) = (K_p + K_i)E(k+1) - K_pE(k) \tag{4.3}$$

which is equivalent to:

$$f_C(k) = f_C(k-1) + (K_p + K_i)E(k) - K_pE(k-1) \tag{4.4}$$

A calculating circuit can be constructed from (4.4) to calculate the consumer frequency $f_C(k)$ from $f_C(k-1)$ and from the errors $E(k)$ and $E(k-1)$. This calculating circuit contains two multipliers with constants $(K_p + K_i)$ and K_p.

To avoid implementing multipliers with too large coefficients, the calculating circuit is modified as follows:

$$f_{sel}(k) = f_{sel}(k-1) + \frac{K_p + K_i}{12.5}E(k) - \frac{K_p}{12.5}E(k-1) \tag{4.5}$$

where f_{sel} is defined as the value to select the corresponding consumer frequency, which is equal to $\frac{1}{12.5} f_C$. Figure 4.8 illustrates the calculating circuit.

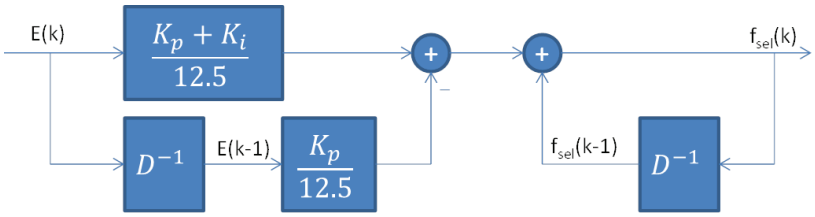

Figure 4.8: Calculating circuit to be implemented.

4.1.3.2 Controller coefficients requirements and selection

The hardware implementation becomes easier if the closed-loop poles are chosen so that the coefficients of the multipliers are multiples of $\frac{1}{8}$. In this way, it will require at most only three bits to represent the fractional part of the binary expression.

The closed-loop poles must also ensure the expected performances for the closed-loop system (including the stability). As shown in the simulations performed in the MATLAB environment, having poles with small absolute values will introduce strong dynamic constraints on the clock frequency engine. Therefore, the poles that will be selected have to limit the overload (in terms of fast changes) on the frequency engine. Moreover, real poles will be preferred in order to avoid pseudo-periodic behaviours for the output system.

Taking into account the range $[0.5, 1[$ for the closed-loop poles, 21 different pairs of poles that satisfy all the constraints above can be selected. All the 21 corresponding controllers have been modeled in VHDL and then embedded in the EC-NAL module for a simulation at RTL level. Simulation results verify the PI controller behaviour in the EC-NAL module. More details on the simulation is presented in Subsection 4.2.2. The preferred controller is the one that keeps the FIFO not full nor empty for a long period while the rate of change of the frequency f_C is low.

Figure 4.9 presents the architecture of such a controller, whose poles are $z_1 = 0.75$ and $z_2 = 0.5$. Hence $K_p = -31.25$ and $K_i = -6.25$ and f_{sel} is given by

$$f_{sel}(k) = f_{sel}(k-1) - \frac{6}{2}E(k) + \frac{5}{2}E(k-1) \tag{4.6}$$

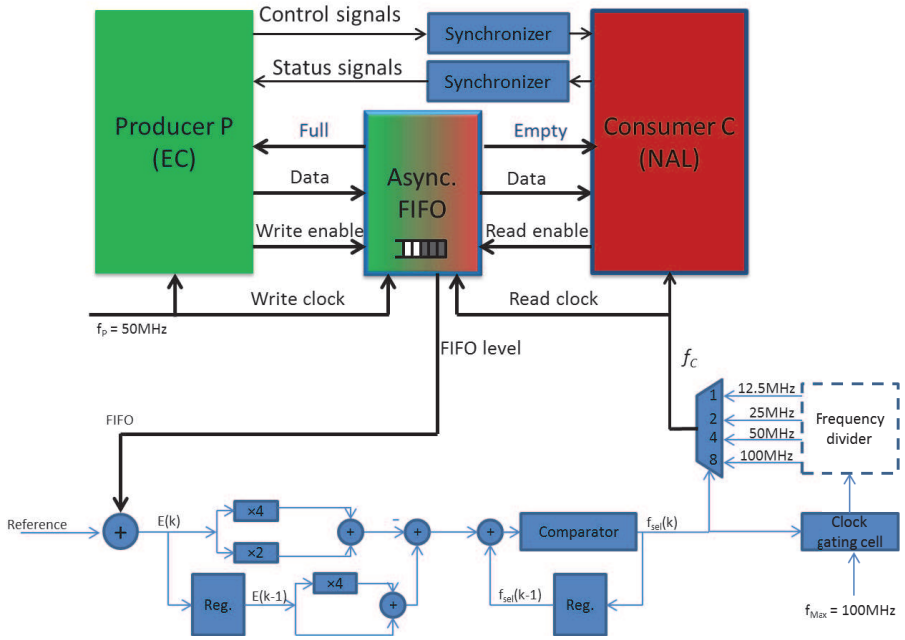

Figure 4.9: The controller with $z_1 = 0.75$ and $z_2 = 0.5$ is integrated into the EC-NAL module.

The implemented calculating circuit contains only four adders and two registers. The multiplications by 2 and 4 are simply done by bit-shifting. To perform frequency non-uniform quantizer, a comparator and multiplexer are added to select the corresponding frequency from the available set of clock signals. These signals are delivered by a frequency divider implemented using a four-bit counter. Note that to save Silicon area, it is preferable for the calculating circuit to have small fractional part in binary expression and few adders.

4.2 Verification of the FIFO-level-based DFS method

To apply the DFS method presented in Chapter 3, the EC-NAL module has been modified. Simulation is done, at RTL level, to verify that the modifications do not change the functionality of the EC-NAL module and of the VENGME platform. Moreover, simulations demonstrate that the implemented DFS method changes the frequency of NAL module. This section presents how the verification is performed and the simulation demonstration.

4.2.1 Verification methodology

The modified version of the EC-NAL module is proved to operate correctly if it can generate the same output encoded video data as the original EC-NAL module from the same input.

- **Control method integration**

The PI controller is then integrated into the VENGME video encoder in order to evaluate its performance within the encoder, when a benchmark of videos is run. The integration of the controller is described in Figure 4.10.

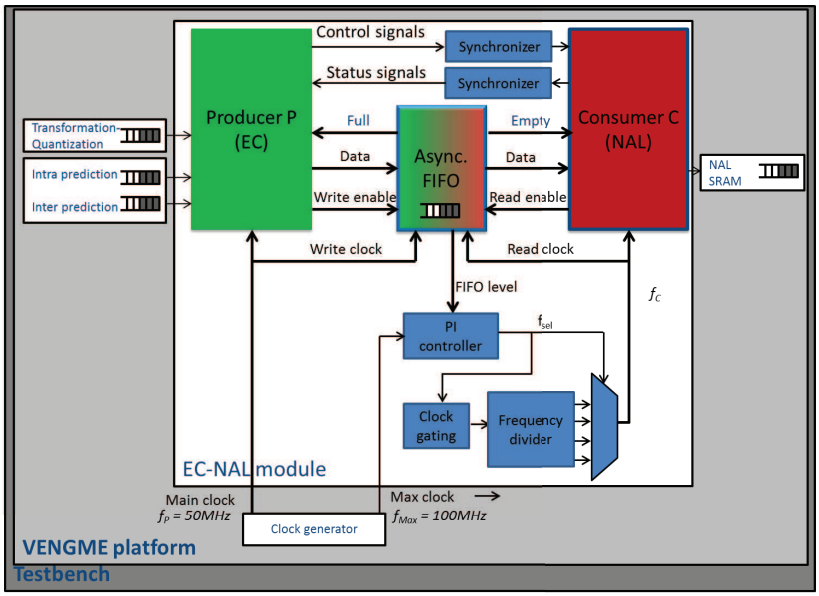

Figure 4.10: Integration of the DFS controller in the H.264 encoder.

The main clock ($50MHz$)which is used for most of the modules in the H.264 encoder is used for the EC submodule.

Moreover, in the VENGME H.264 encoder, a clock generator whose maximal frequency is $100MHz$ is already implemented. The clock signal after the gating clock cell is used as the input of the frequency divider. According to the controller decision, this clock signal can be gated if the selected frequency is $0Hz$. The four output clock signals of the frequency divider (12.5, 25, 50 and $100MHz$) can be selected. The clock for the NAL submodule and the reading clock of the asynchronous FIFO is selected using the f_{sel} signal delivered by the PI controller.

4.2.2 Simulation demonstration

When the FIFO occupancy level reference value is changed, a trade-off between the power consumption and the computing performance is performed [Wu05V]. In this hardware simulation, the reference value is chosen equal to 3, i.e. where the FIFO is half-full. The simulation results prove again the impact of the closed-loop pole values. For controllers with the module of the associated closed-loop poles close to 1, the consumer frequency and FIFO level change gradually. For instance, for closed-loop poles equal to $z_1 = 0.875$ and $z_2 = 0.75$, the maximal frequency of $100MHz$ is not reached, even when the FIFO is full for a long time. The consumer frequency jumps up one step at once. With the controller whose associated poles are $z_{1,2} = 0.5$, the FIFO is rarely full and the consumer frequency can jump from $0MHz$ to $50MHz$ in just one sampling time.

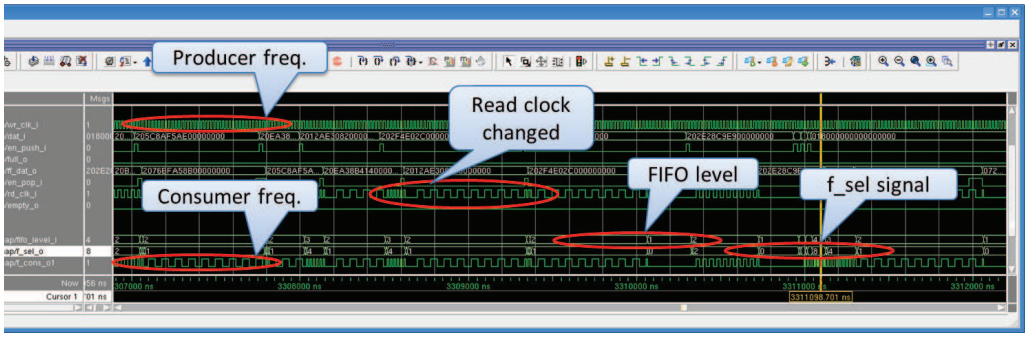

Figure 4.11: Waveform of the selected PI controller applied on the EC-NAL module.

Figure 4.11 shows simulation results when the controller reacts in a smooth way and the FIFO is rarely full. This selection is also appropriate in terms of hardware implementation. It costs only four adders and one bit for the fractional part. The closed-loop pole values are $z_1 = 0.75$ and $z_2 = 0.5$. Hence $K_p = -31.25$ and $K_i = -6.25$. The simulation is performed using ModelSim from Mentor Graphics. The simulation waveforms, see Figure 4.11 show that while the clock frequency on the "write side" of the FIFO (i.e. producer frequency) keeps unchanged, the clock frequency on the "read side" (consumer frequency) varies according to the control decision. The control decision can be seen on the figure as the values of the signal $f_{sel}(0, 1, 2, 4, 8)$ that are related to the frequency values 0, 12.5, 25, 50 and $100MHz$.

4.3 Validation of the control of the FIFO occupancy level for DFS

This section presents how we validate that the FIFO-level control reduces the power consumption. The validation method is presented in Subsection 4.3.1, including the modeling and simulation process descriptions. In each experimental step, i.e. modeling at behavioural level, synthesis and post-synthesis simulations and gate level power estimation, power reduction is estimated. Subsection 4.3.4 discusses the results.

4.3.1 Validation methodology

Figure 4.12 illustrates the experimental steps to finally estimate the power gain when applying the proposed DFS method. After the system modeling, the controller is designed and modeled in MATLAB. These steps at behavioural level, including simulation in MATLAB, were presented in Chapter 3. The DFS control method, i.e. the PI controller, the frequency divider and all the modifications on the EC-NAL module, is modeled in VHDL as presented in Section 4.1. The controller is then integrated into the VENGME EC-NAL module, with modifications as discussed above. The module is synthesized with the $28nm$ FDSOI technology from STM using RC synthesis tool from Cadence. The post-synthesis simulation using the module EC-NAL netlist in the VENGME platform shows the control operations at gate level. The power consumption of the EC-NAL module is estimated using PrimeTime from Synopsys.

For comparison, the original EC-NAL module is also synthesized, simulated at gate level and its power consumption is estimated.

Each step enables an estimation of the power reduction but the lower level power estimation has more precise results. The results are presented hereafter.

4.3.2 Power gain estimation at behavioural model level

At behavioural level, the power reduction gain of the DFS method proposed in Chapter 3 is estimated using MATLAB simulations. These simulations are performed before the controller is modeled in VHDL, see Figure 4.12, that is, before the final values of the controller coefficients are decided for

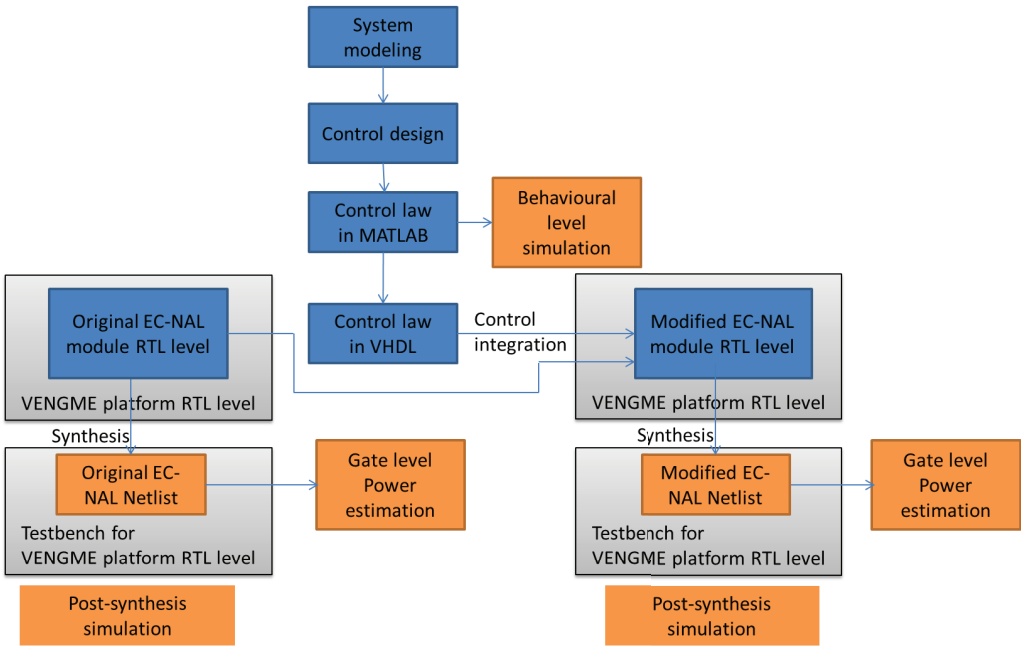

Figure 4.12: Validation flow of the DFS control based on FIFO occupancy level.

implementation. Therefore, for the MATLAB result presented hereafter, the closed-loop pole values $z_{1,2} = 0.5 \pm 0.2i$ are selected as an example of the PI controller to be implemented.

In the MATLAB simulations presented in Section 3.3, the controlled frequency is recorded, as shown in Figure 3.40, page 138. This gives an idea on the power consumption figure during the simulation, because $P_{dyn} \sim f$. The **dynamic part** of the energy possible gain on the consumer side, is evaluated in the MATLAB environment thanks to an integration of the consumer clock frequency f_C over a time period of $0.1s$. When compared to the solution without FIFO level control, the gain is equal to 96.8%. Note that this gain is **only on the dynamic part** of the energy of the consumer, i.e. the NAL submodule. In real life, it will be smaller, this result is very optimistic because it does not take into account extra consumption related to the control part and the asynchronous FIFO. Again, applying dynamic voltage scaling (DVS) will improve this gain.

4.3.3 Power gain estimation at gate level

To accurately estimate the power reduction reachable by the proposed control method, power estimations using the netlist of both the original and the modified versions of the EC-NAL module are performed.

4.3.3.1 Synthesis results

The proposed control method requires the implementation of an asynchronous FIFO and the PI controller that will add extra Silicon and power consumption. For FD-SOI $28nm$ technology, the RC synthesis tool from Cadence shows a Silicon area overhead of 10.8% when compared to the original EC-NAL module. Note that in our application, DFS is applied only on the NAL submodule, that is the so-called consumer. Because it is a very small block, the power consumption gain on the total EC-NAL module is only 1.6%. This latest result is obtained without V_{dd} scaling. Thus with DVFS, the power consumption can be decreased even more because $P_{dyn} \sim fV_{dd}^2$.

In both versions of the EC-NAL module, the EC submodule is unchanged. Thus, from now on, only the results on the other parts of the EC-NAL module, i.e. the FIFO and NAL submodule, are presented and analyzed.

Table 4.1 illustrates the synthesis results of the original "consumer" part, containing the synchronous FIFO and the orignal NAL submodules. Note that in this table, the orignal NAL submodule is in the same order of magnitude to the synchronous FIFO regarding area and power consumption. Indeed, the original NAL submodule uses only 1164 cells, slightly more than the 906 cells of the synchronous FIFO.

Table 4.1: Synthesis results of the original NAL submodule and of the synchronous FIFO

Core	**sync. FIFO**	**original NAL**	**Total**
Silicon resource ($cell$)	906 (43.8%)	1164 (56.2%)	2070 (100%)
Power consumption ($\times 10^{-03} mW$)	348.924 (53.1%)	307.653 (46.9%)	656.577 (100%)

Table 4.2 illustrates the synthesis results of the modified consumer, containing the PI controller, the frequency divider, the asynchronous FIFO and the modified NAL submodule. The results show that the additional Silicon

for the PI controller and the frequency controller is small, only 75 cells, in comparison to the NAL submodule (1175 cells). However, the asynchronous FIFO requires more Silicon resource.

Table 4.2: Synthesis results of the modified NAL submodule and the asynchronous FIFO

Core	**PI controller**	**frequency divider**	**async. FIFO**	**modified NAL**	**Total**
Silicon resource ($cell$)	66 (2.7%)	9 (0.4%)	1184 (48.6%)	1175 (48.3%)	2434 (100%)
Power consumption ($\times 10^{-03} mW$)	41.176 (3.3%)	6.601 (0.5%)	566.037 (45.6%)	629.065 (50.6%)	1242.879 (100%)

Because the RC synthesis tool estimates the power consumption based on the assumption on average activity of the circuit, this number is not yet accurate and only for reference.

4.3.3.2 Post-synthesis simulation results

Post-synthesis simulations are performed on the netlist of the EC-NAL module operating in the RTL VENGME platform with the same testbench for both the original and modified versions of the EC-NAL module. For this testbench, the VENGME platform encodes the first video frame in the reference video Foreman [VidLib]. The test video is in QCIF resolution due to space limitation.

The simulations show the operations of the DFS control method and its effect on the frequency of the NAL submodule. Table 4.3 shows the percentage of time spent in each frequency value of the NAL frequency in both simulation cases.

This result provides a rough estimation of the power reduction by using the DFS FIFO-level control method, see Figure 4.13.

Because the dynamic power, which is the dominate part of the power consumption, is proportional to the operating frequency, the ratio between the average power consumption of the original NAL submodule and of the modified NAL submodule is calculated as follows:

Table 4.3: Frequency for the NAL submodule during simulation time og VENGME test video

Frequency	**origininal NAL time(%)**	**modified NAL time(%)**
100*MHz*	-	5
50*MHz*	100	15
25*MHz*	-	10
12.5*MHz*	-	10
0*MHz*	-	60

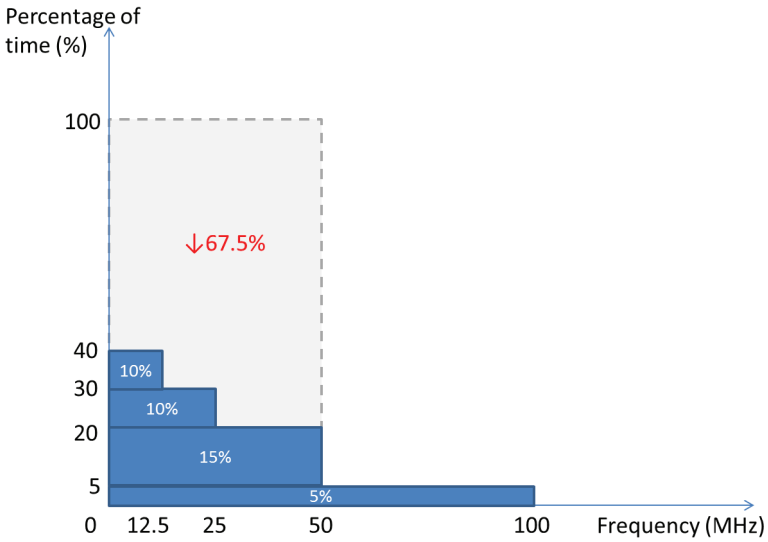

Figure 4.13: DFS control method: estimated power reduction from post-synthesis simulation results.

$$\begin{aligned} &\frac{\text{Power consumption of the modified NAL}}{\text{Power consumption of the origininal NAL}} \\ \approx &\frac{\text{Dynamic power of the modified NAL}}{\text{Dynamic power of the origininal NAL}} \\ = &\frac{5 \times 100 + 15 \times 50 + 10 \times 25 + 10 \times 12.5 + 60 \times 0}{100 \times 50} \\ = &\frac{32.5}{100} \end{aligned} \tag{4.7}$$

Thus, the power reduction is approximately equal to

$$\frac{100 - 32.5}{100} = 67.5\% \tag{4.8}$$

These simulations at gate level also enable estimation of power consumption based on the real activities of the circuit which is recorded in a Value Change Dump (VCD) file.

4.3.3.3 Power estimation results at gate level

Power estimation using PrimeTime from Synopsys is performed based on the activity of the circuit during post-synthesis simulation which is recorded in the VCD file. Power estimation is done on both the original and the modified versions of the EC-NAL module.

The results in Table 4.4 are the power consumption of the origininal version, which contains the synchronous FIFO and the origininal NAL submodules. The two submodules consume in average $2.126 \times 10^{-04} mW$, which is obviously smaller than the number estimated based on the assumption of the average activity of the circuit in the synthesis phase.

Table 4.4: Power results of the synchronous FIFO and of the origininal NAL submodules

Core	Synchronous FIFO	original NAL	Total
Total power consumption (mW)	1.338×10^{-04}	7.876×10^{-05}	2.126×10^{-04}
Dynamic power (mW)	1.309×10^{-04}	6.9243×10^{-05}	2.001×10^{-04}
Leakage power (mW)	2.952×10^{-06}	9.515×10^{-06}	12.467×10^{-06}

Table 4.5 presents the power consumption of the modified version, which contains the controller, the frequency divider, the asynchronous FIFO and the modified NAL subemodules.

Table 4.5: Power results of the controller, frequency divider, asynchronous FIFO and modified NAL submodules

Core	Controller	Frequency divider	Async. FIFO	modified NAL	Total
Total power (mW)	7.621×10^{-06}	2.019×10^{-06}	1.594×10^{-04}	3.208×10^{-05}	2.011×10^{-04}
Dynamic power (mW)	6.482×10^{-06}	1.787×10^{-06}	1.541×10^{-04}	2.241×10^{-05}	1.848×10^{-04}
Leakage power (mW)	1.139×10^{-06}	2.323×10^{-07}	5.267×10^{-06}	9.671×10^{-06}	16.309×10^{-06}

To demonstrate the effect of the DFS control method on the power consumption, the average total power of both versions are compared. It shows that the **power reduction that the proposed method achieves when applied in the FIFO between the EC and NAL submodules is only 5.3%**, from $2.123 \times 10^{-04} mW$ to $2.011 \times 10^{-04} mW$. This result is not equivalent to the power reduction of 67.5%, estimated in the post-synthesis simulation. Indeed, the estimation in the post-synthesis simulation based on the NAL frequency is equivalent to the estimation on dynamic power of the NAL submodule only. Basically, the reduction of the dynamic power of the NAL submodule is:

$$\begin{aligned} &\frac{\text{Dynamic power of the modified NAL}}{\text{Dynamic power of the origininal NAL}} \\ =&\frac{2.241 \times 10^{-05}}{6.9243 \times 10^{-05}} \\ =&32.4\% \end{aligned} \tag{4.9}$$

Thus, the **dynamic power reduction for the NAL submodule is equal to 100 - 32.4 = 67.6%**.

When taking into account the leakage power, the power reduction on the NAL submodule is:

$$\begin{aligned} &\frac{\text{Power consumption of the modified NAL}}{\text{Power consumption of the origininal NAL}} \\ =&\frac{3.208 \times 10^{-05}}{7.876 \times 10^{-05}} \\ =&40.7\% \end{aligned} \tag{4.10}$$

Thus, the **total power reduction for the NAL submodule is equal to 100 - 40.7 = 59.3%**.

The reason for a power reduction of only 5.3% is not due to the implementation cost of the PI controller and frequency divider. Synthesis results have shown that these additional submodules are small in comparison to the already existing submodules. Power estimation results confirm this fact, because the total consumption of the additional submodules is only $9.64 \times 10^{-06} mW$, about 4.8% of the total modified version and **only 12.2% of the power consumption of the origininal NAL submodule**.

However, the payment due to the asynchronous FIFO is rather high. In fact, the asynchronous FIFO is the most consuming submodule in the modified consumer part. **The modification from the synchronous FIFO to**

the asynchronous one increases per 19.1% of the power consumption, from $1.338 \times 10^{-04} mW$ to $1.594 \times 10^{-04} mW$. The **power reduction achievement is 47% without taking into account the part of the synchronous and asynchronous FIFO submodules**. Indeed, the ratio between both power consumptions is calculated as follows.

$$\begin{aligned} &\frac{\text{Power consumption of (modified NAL + PI controller + frequency divider)}}{\text{Power consumption of the origininal NAL}} \\ =&\frac{3.208 \times 10^{-05} + 7.621 \times 10^{-06} + 2.019 \times 10^{-06}}{7.876 \times 10^{-05}} \\ =&\frac{4.172}{7.876} \approx 53\% \end{aligned} \tag{4.11}$$

Thus, the power reduction is equal to 100 - 53 = 47%.

In fact, even in the original version, **the synchronous FIFO consumes 1.7 times more than the NAL submodule**, i.e. $1.338 \times 10^{-04} mW$ in comparison to $7.876 \times 10^{-05} mW$. Moreover, in the modified version, while the NAL frequency is controlled by the DFS controller, the asynchronous FIFO always has another clock input signal, the writing clock, which is the main clock of $50 MHz$. Thus, the dominate power consumption of the asynchronous FIFO reduces the power reduction effect of the DFS method. For a clearer explanation, the power in the period of time that the NAL frequency has a certain value is estimated as follows.

During 5% of the simulation time, the NAL frequency is equal to $100 MHz$. The corresponding power results are presented in Table 4.6.

Table 4.6: Power results of the modified version when the NAL frequency is equal to $100 MHz$

Core	Controller	Frequency divider	Async. FIFO	modified NAL
Total power (mW)	1.836×10^{-05}	5.535×10^{-06}	2.021×10^{-04}	18.630×10^{-05}
Dynamic power (mW)	17.287×10^{-06}	5.294×10^{-06}	1.968×10^{-04}	17.672×10^{-05}
Leakage power (mW)	1.075×10^{-06}	2.420×10^{-07}	5.222×10^{-06}	9.572×10^{-06}

During 15% of the simulation time, the NAL frequency is equal to $50 MHz$. The corresponding power results are presented in Table 4.7.

Table 4.7: Power results of the modified version when the NAL frequency is equal to $50MHz$

Core	Controller	Frequency divider	Async. FIFO	original NAL
Total power (mW)	1.156×10^{-05}	5.706×10^{-06}	1.771×10^{-04}	12.100×10^{-05}
Dynamic power (mW)	10.395×10^{-06}	5.464×10^{-06}	1.721×10^{-04}	11.137×10^{-05}
Leakage power (mW)	1.163×10^{-06}	2.422×10^{-07}	5.068×10^{-06}	9.616×10^{-06}

During 10% of the simulation time, the NAL frequency is equal to $25MHz$. The corresponding power results are presented in Table 4.8.

Table 4.8: Power results of the modified version when the NAL frequency is equal to $25MHz$

Core	Controller	Frequency divider	Async. FIFO	modified NAL
Total power (mW)	1.023×10^{-05}	5.686×10^{-06}	1.652×10^{-04}	6.329×10^{-05}
Dynamic power (mW)	9.047×10^{-06}	5.445×10^{-06}	1.599×10^{-04}	5.368×10^{-05}
Leakage power (mW)	1.179×10^{-06}	2.416×10^{-07}	5.314×10^{-06}	9.622×10^{-06}

During 10% of the simulation time, the NAL frequency is equal to $12.5MHz$. The corresponding power results are presented in Table 4.9.

During 60% of the simulation time, the NAL frequency is equal to $0Hz$. The corresponding power results are presented in Table 4.10.

These results detailed prove that the writing frequency has an important role in the power consumption of the FIFO. The writing clock frequency of the asynchronous FIFO is always $50MHz$. When the reading clock frequency is $100MHz$, the FIFO power consumption is $2.021 \times 10^{-04}mW$. Even when the reading clock frequency is $0MHz$, the FIFO power consumption is still high, namely $1.577 \times 10^{-04}mW$.

Table 4.9: Power results of the modified version when the NAL frequency is equal to $12.5MHz$

Core	**Controller**	**Frequency divider**	**Async. FIFO**	**modified NAL**
Total power (mW)	1.008×10^{-05}	5.706×10^{-06}	1.659×10^{-04}	3.566×10^{-05}
Dynamic power (mW)	8.989×10^{-06}	5.464×10^{-06}	1.605×10^{-04}	2.612×10^{-05}
Leakage power (mW)	1.089×10^{-06}	2.431×10^{-07}	5.352×10^{-06}	9.540×10^{-06}

Table 4.10: Power results of the modified version when the NAL frequency is equal to $0Hz$

Core	**Controller**	**Frequency divider**	**Async. FIFO**	**modified NAL**
Total power (mW)	5.631×10^{-06}	2.277×10^{-07}	1.577×10^{-04}	9.832×10^{-06}
Dynamic power (mW)	4.498×10^{-06}	0	1.522×10^{-04}	7.318×10^{-11}
Leakage power (mW)	1.133×10^{-06}	2.277×10^{-07}	5.401×10^{-06}	9.832×10^{-06}

Thus, to fairly evaluate the method effect when it is applied to the EC-NAL module, the power consumption of the asynchronous FIFO should be considered separately.

4.3.4 Discussion

The power estimation enables to evaluate the DFS control method on the EC-NAL module. Through experimental steps, the method is proven to be efficient. It requires low Silicon overhead and power cost for implementing the PI controller. It reduces per about 67% the dynamic power and about 59% the total power consumption on the consumer side.

However, the proposed method reduces only per 5.3% the power consumption of the particular circuit, i.e. the consumer part. The reason is that the chosen consumer, that is the original NAL submodule, is small,

even in comparison to the FIFO link. If the DFS method is applied to another FIFO link, with the same FIFO, the same PI controller and frequency divider, but for a larger consumer submodule, the efficiency of the method can be more evidently proven. The following normalization illustrates this argument.

According to the results presented above, consider that:

- the power consumption of the original NAL submodule is 1;
- the power consumption of the synchronous FIFO is 1.7 times larger than the original NAL submodule.

Then:

- the power consumption of the PI controller and of the frequency divider is equal to 0.12× the power consumption of the original NAL submodule;
- the power consumption of the asynchronous FIFO is 19% higher than the one of the synchronous FIFO;
- our DFS method reduces per 59.3% the total power consumption of the consumer, i.e. of the NAL submodule.

Thus, the power consumption of the original version is equal to $1+1.7 = 2.7$.

The power consumption of the modified version is equal to $1 \times (1 - 59.3\%) + 1.7 \times 119\% + 0.12 = 2.567$.

The power gain is therefore:

$$\frac{2.7 - 2.567}{2.7} = 4.9\% \tag{4.12}$$

Now, suppose that the method is applied to a consumer which is 100 times larger than the original NAL submodule, i.e. 100 times more consuming than the original NAL. One can calculate the power gain as follows:

- the power consumption of the original version is equal to $100 + 1.7 = 101.7$.
- the power consumption of the modified version is equal to $100 \times (1 - 59.3\%) + 1.7 \times 119\% + 0.12 = 42.86$.

Thus, the power gain of the method is:

$$\frac{101.7 - 42.86}{101.7} = 57.9\% \tag{4.13}$$

This power gain of 57.9% is closer to the power gain estimated in simulation step. With a submodule which is 100 times more consuming than the NAL submodule, the proposed method efficiency is proven.

4.4 Conclusion

The PI controller was designed and implemented in hardware to adapt the consumer (i.e. the NAL submodule) frequency according to the FIFO occupancy level. This chapter has presented how the module EC-NAL was modified and new submodules were added to implement the DFS control method designed in Chapter 3 to the FIFO link inside the VENGME EC-NAL module. Simulations at RTL level have verified the operation of the implemented controller in adapting the NAL frequency based on the FIFO level. The coefficients of the controller were selected to meet the required performances for the closed-loop system, taking also into account hardware implementation constraints.

Post-synthesis power estimation results obtained using PrimeTime are analyzed to evaluate at a finer grain the efficiency of the FIFO control method proposed.

The method does not efficiently reduce the power consumption of the VENGME EC-NAL module, only 5% for the power consumption part excluding the EC submodule. However, the control method is well designed and implemented. It reduces per about 59.3% the power consumption of the consumer submodule, i.e. the NAL submodule. Its efficency will be better in the case that the consumer submodule is larger (larger power consumption).

This result is very appealing because it has been obtained without voltage scaling: with a complete DVFS scheme, the power gain will be even larger. Future works include the extension of this FIFO control approach to the whole VENGME architecture. Voltage scaling techniques will be as well integrated in the platform.

Conclusion

Because of technology evolution, energy and power consumption have become a tremendous problem in System-on-Chip design. They affect the system reliability, cooling cost, and battery lifetime. These aspects are important, especially for mobile applications and for environmental considerations. Therefore, power consumption reduction is strongly required when designing complex System-on-Chips.

Hardware video encoders are now widely used in mobile devices. The H.264 Advanced Video Coding (H.264/AVC) is one of the latest and most efficient standards for video applications. It adopts and improves many advanced methods and algorithms for video compressing to achieve high compression efficiency. Even though the H.264 successor, H.265/High Efficiency Video Coding (HEVC), that enables Ultra High Deffinition Television (UHTV), has been released, the H.264/AVC has still an important role and its codecs are widely used.

The H.264/AVC specification increases the complexity of hardware implementation and also the chip power consumption. Thus, to bring the H.264/AVC standard into commercial products, especially for hand-held devices, designers need to apply power techniques to the hardware when designing the video codec. Targeting mobile applications, the VENGME platform, a hardware video encoder from a research project in Vietnam, was designed using low-power design features. Some blocks in the platform present nice power figures and performances, similar to state-of-the-art designs. However, the whole platform power results are not outstanding in comparison to state-of-the-art hardware video encoders.

In this thesis, we have investigated the H.264 specification and the overall H.264 encoding architecture. Several hardware implementations of the H.264 video encoding were reviewed. H.264/AVC encoders are de-

signed based on the basic pipelining architecture with different improvements added to serve specific design objectives, namely scalability, speed and power consumption. This is presented in the publication "An overview of H.264 hardware encoder architectures including low-power features" [Nguy14A]. Many power consumption reduction methods have been applied during the design phase of the VENGME platform. Moreover, traditional power reduction mechanisms can be also used.

One contribution of this thesis work (presented in Chapter 2) is **the design of the EC-NAL module in the VENGME video encoder**. For this module, several improvements from the literature have been reused to increase the throughput, reduce the area cost and the power consumption. The VENGME EC-NAL module exhibits a very low power consumption, only $1.56mW$, at the operating frequency of $100MHz$, using the 180nm CMOS technology. The EC-NAL design was initially targeting CIF video format, but it is also suitable for real-time 720HD video format at the operating frequency of $100MHz$. This contribution has led to the following publications:

- "An effcient Context Adaptive Variable Length coding architecture for H.264/AVC video encoders" [Nguy12A] presents the CAVLC encoder, one of the main engines of the Entropy Coder (EC).

- "Hardware implementation for entropy coding and byte stream packing engine in H.264/AVC" [Nguy13A] presents the design and implementation of the whole VENGME EC-NAL module.

We have also presented the design, architecture, implementation and validation of the VENGME H.264/AVC hardware video encoder. The VENGME platform consumes about $58.25mW$ with the 130nm CMOS technology, which is suitable for the targeted mobile applications. However, this power consumption result is not outstanding, compared to the state-of-the-art H.264 video encoders. Some extra power optimizations are expected.

Power estimation at RTL level using SpyGlass, another contribution of the thesis, presented in Chapter 2, **has been done for the whole VENGME platform**. This helps to analyze the power characteristics of the VENGME platform. In terms of composition, the Inter Prediction and the SW MAU are the most consuming modules. However, this part of power

consumption can be reduced because these modules operate only on inter predicted frames. The EC-NAL module, which is used in every frame, consumes the third power budget. The VENGME architecture and this power estimation result are presented in the publication: "H.264/AVC Hardware Encoders and Low-Power Features" [Nguy14H].

We had to embrace all general techniques used for low-power optimization. Traditional mechanisms were analyzed with the VENGME power characteristics to propose a suitable low power solution for this H.264 encoding System-on-Chip.

Because the VENGME platform is designed as a data flow architecture with modules that communicate via buffers, DVFS can be applied on the VENGME platform by controlling the occupancy level of the buffers. Applying DVFS can achieve both energy and power consumption gains. However, due to implementation constraints and the design at hand, currently only the frequency is scaled (DFS). The VENGME EC-NAL module consuming the most among the modules that operate for every video frame. Power consumption of the EC-NAL module must be reduced. Besides, the EC-NAL module has been designed by the thesis author. Therefore, its architecture is deeply understood. Lastly, in the EC-NAL module, a FIFO is implemented to transfer data from EC to NAL submodules. Therefore, **the EC-NAL module is selected to develop and test our buffer-occupancy-level-based DFS control method** that might be applied to the whole VENGME platform. This is presented in Chapter 3. We proposed that the frequency of the NAL submodule will be scaled based on the occupancy level of the FIFO between EC and NAL submodules.

In order to apply the DFS control method to the FIFO embedded in the EC-NAL module, the FIFO link was modeled and a PI controller was selected. **Behavioral modeling and simulations** have shown the method effect. Moreover, the controled system is proven to be stable.

Based on the control proposed, **we have designed the hardware controller for low-power in the EC-NAL module**. The controller, developed using control theory, was designed with respect to hardware implementation constraints. **Simulations at RTL level** verified that the controller changed the NAL submodule frequency and that the EC-NAL module still functions properly in the sense that its generates the same output bytestream for the same input data. These simulations also helped in choosing the controller parameters.

The controller is then implemented in the EC-NAL module. This work is presented in the publication "FIFO-level-based Power Management and its Application to a H.264 Encoder" [Nguy14F].

Finally, **power estimation at gate level** using the $28nm$ FDSOI technology enabled to evaluate the DFS control efficiency. All the hardware design, implementation and results have been presented in Chapter 4.

The control method is well designed and implemented. The controller implemented reduces only 5% the power consumption of the part including the FIFO and NAL submodule. However, it reduces per about 59.3% the power consumption of the consumer submodule only, i.e. the NAL submodule. The reason for the small global power gain on the EC-NAL module is that the consumer, i.e. the NAL submodule, is too small compared to the FIFO link. Its effcency will be better in the case that the consumer submodule is larger (larger power consumption). This result is very appealing because it has been obtained without voltage scaling: with a complete DVFS scheme, the power gain will be even larger.

The continuation of our work will be the extension of this FIFO control approach to the whole VENGME architecture. This includes choosing the consuming module to become "the consumer" and arbiter algorithm required when there are many controlled FIFOs in the platform.

Another perspective is the integration of voltage scaling (DVS) techniques in the platform. The control law will be designed again for DVS additional constraints, e.g. the delay due to power supply switching.

Lastly, evaluation on the Silicon platform must be conducted. Especially, power figure obtained in simulation will be compared to the state-of-the-art methods.

Bibliography

[Astr86C] K. J. Astrom and Björn Wittenmark. *Computer controlled systems: theory and design, 3rd edition.* Prentice Hall, 1986.

[Atre] Atrenta Inc. *SpyGlass Power tool.* http://www.atrenta.com/.

[Bae05L] Geuntae Bae, Jaesub Kim, Daewon Kim, and Daeyeon Park. Low-power multimedia scheduling using output pre-buffering. In *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2005. 13th IEEE International Symposium on*, pages 389–396, Sept 2005.

[Baker08P] Michael A. Baker, Viswesh Parameswaran, Karam S. Chatha, and Baoxin Li. Power reduction via macroblock prioritization for power aware h.264 video applications. In *Proceedings of the 6th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, CODES+ISSS '08, pages 261–266, New York, NY, USA, 2008. ACM.

[Beig05A] E. Beigne, F. Clermidy, P. Vivet, A. Clouard, and M. Renaudin. An asynchronous noc architecture providing low latency service and its multi-level design framework. In *Asynchronous Circuits and Systems, 2005. ASYNC 2005. Proceedings. 11th IEEE International Symposium on*, pages 54–63, March 2005.

[Beni00A] L. Benini, A. Bogliolo, and G. De Micheli. A survey of design techniques for system-level dynamic power management. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 8(3):299–316, June 2000.

[Bjon02C] G. Bjontegaard and K. Lillevold. *Context-adaptive VLC (CVLC) Coding of Coefficients*. JVT document JVT-C028, Fairfax, VA, 2002.

[Boyd94L] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*, volume 15 of *Studies in Applied Mathematics*. SIAM, Philadelphia, PA, June 1994.

[Bui11H] Duy-Hieu Bui. *Hardware architecture for intra prediction module in H.264/AVC*. Master thesis, University Paris Sud XI, November 2011.

[Bui12A] Duy-Hieu Bui, Van-Huan Tran, Van-Mien Nguyen, Duc-Hoang Ngo, and Xuan-Tu Tran. A hardware architecture for intra prediction in h.264/avc encoder. In *Proceeding of the 2012 IEICE International Conference in Integrated Circuits and Devices in Vietnam (ICDV)*, August 2012.

[Chan09A] Hsiu-Cheng Chang, Jia-Wei Chen, Bing-Tsung Wu, Ching-Lung Su, Jinn-Shyan Wang, and Jiun-In Guo. A Dynamic Quality-Adjustable H.264 Video Encoder for Power-Aware Video Applications. *Circuits and Systems for Video Technology, IEEE Transactions on*, 19(12):1739–1754, Dec 2009.

[Chan11S] Che-Wei Chang and Rainer Dömer. *System level modeling of a h.264 video encoder*. Technical report, University of California, Ivrine, CA, USA, June 2011.

[Chand92L] A.P. Chandrakasan, S. Sheng, and R.W. Brodersen. Low-power cmos digital design. *Solid-State Circuits, IEEE Journal of*, 27(4):473–484, Apr 1992.

[Chen05D] Tung-Chien Chen, Yu-Wen Huang, Chuan-Yong Tsai, Bing-Yu Hsieh, and Liang-Gee Chen. Dual-block-pipelined vlsi architecture of entropy coding for h.264/avc baseline profile. In *VLSI Design, Automation and Test, 2005. (VLSI-TSA-DAT). 2005 IEEE VLSI-TSA International Symposium on*, pages 271–274, April 2005.

[Chen05F] Chao-Chung Cheng and Tian-Sheuan Chang. Fast three step intra prediction algorithm for 4 times;4 blocks in h.264. In *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, pages 1509–1512 Vol. 2, May 2005.

[Chen06AA] Tung-Chien Chen, Shao-Yi Chien, Yu-Wen Huang, Chen-Han Tsai, Ching-Yeh Chen, To-Wei Chen, and Liang-Gee Chen. Analysis and architecture design of an HDTV720p 30 frames/s H.264/AVC encoder. *Circuits and Systems for Video Technology, IEEE Transactions on*, 16(6):673–688, June 2006.

[Chen06AD] Tung-Chien Chen, Yu-Wen Huang, Chuan-Yung Tsai, Bing-Yu Hsieh, and Liang-Gee Chen. Architecture design of context-based adaptive variable-length coding for h.264/avc. *Circuits and Systems II: Express Briefs, IEEE Transactions on*, 53(9):832–836, Sept 2006.

[Chen07F] Tung-Chien Chen, Yu-Han Chen, Sung-Fang Tsai, Shao-Yi Chien, and Liang-Gee Chen. Fast algorithm and architecture design of low-power integer motion estimation for h.264/avc. *Circuits and Systems for Video Technology, IEEE Transactions on*, 17(5):568–577, May 2007.

[Chen08A] Liang-Gee Chen, Tzu-Der Chuang, Yu-Jen Chen, Chung-Te Li, Chia-Jung Hsu, Shao-Yi Chien, and Liang-Gee Chen. An H.264/AVC scalable extension and high profile HDTV 1080p encoder chip. In *VLSI Circuits, 2008 IEEE Symposium on*, pages 104–105, June 2008.

[Chen09A] Yu-Han Chen, Tung-Chien Chen, Chuan-Yung Tsai, Sung-Fang Tsai, and Liang-Gee Chen. Algorithm and Architecture Design of Power-Oriented H.264/AVC Baseline Profile Encoder for Portable Devices. *Circuits and Systems for Video Technology, IEEE Transactions on*, 19(8):1118–1128, Aug 2009.

[Chen10T] Jing Chen, Yongliang Chen, Hantao Zhu, Chunchun Sui, PuFeng Wu, and Xixin Cao. The hardware design and implementation for cavlc and exp-golomb in h.264/avc. In *Advanced Communication Technology (ICACT), 2010 The 12th International Conference on*, volume 2, pages 1610–1613, Feb 2010.

[Chien06A] Chih-Da Chien, Keng-Po Lu, Yi-Hung Shih, and Jiun-In Guo. A high performance cavlc encoder design for mpeg-4 avc/h.264 video coding applications. In *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, pages 4 pp.–3841, May 2006.

[Chio06T] De-Shiuan Chiou, Shih-Hsin Chen, Shih-Chieh Chang, and Chingwei Yeh. Timing driven power gating. In *Design Automation Conference, 2006 43rd ACM/IEEE*, pages 121–124, 2006.

[Choi05F] 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.

[Chou09P] P. Choudhary and D. Marculescu. Power management of voltage/frequency island-based systems using hardware-based methods. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 17(3):427–438, March 2009.

[Cumm02S] Clifford E. Cummings. Simulation and synthesis techniques for asynchronous fifo design, 2002.

[Cumm08C] Clifford E. Cummings. World class verilog & systemverilog training clock domain crossing (cdc) design & verification techniques using systemverilog, 2008.

[DE2] Altera Corporation. *Development and Education board.* http://www.altera.com/education/univ/materials/boards/de2/unv-de2-board.html.

[Dang13A] Nam-Khanh Dang, Van-Mien Nguyen, and Xuan-Tu Tran. A vlsi implementation for inter-prediction module in h.264/avc encoders. In *Proceeding of the 2013 IEICE International Conference in Integrated Circuits and Devices in Vietnam (ICDV)*, November 2013.

[Deva95A] Srinivas Devadas and Sharad Malik. A survey of optimization techniques targeting low power vlsi circuits. In *Design Automation, 1995. DAC '95. 32nd Conference on*, pages 242–247, 1995.

[Di03A] Wu Di, Gao Wen, Hu Mingzeng, and Ji Zhenzhou. An exp-golomb encoder and decoder architecture for jvt/avs. In *ASIC, 2003. Proceedings. 5th International Conference on*, volume 2, pages 910–913 Vol.2, Oct 2003.

[Dinh05P] M. Dinh. *Parameter dependent design by finite dimensional LMI optimisation: application to the design of trade-off dependent controllers.* PhD thesis, Université de Caen, December 2005.

[Duti50T] J. Dutilh. Theorie des servomecanismes a relais. *L'onde electrique*, 30:438–445, 1950.

[Ecar50E] C. Ecary. Etude graphique du regime transitoire d'un systeme non-lineaire. *C.E.M.V.*, 1950.

[Etoh05A] M. Etoh and T. Yoshimura. Advances in wireless video delivery. *Proceedings of the IEEE*, 93(1):111–122, Jan 2005.

[Fell11T] C. Feller, J. Wuenschmann, T. Roll, and A. Rothermel. The vp8 video codec - overview and comparison to h.264/avc. In *Consumer Electronics - Berlin (ICCE-Berlin), 2011 IEEE International Conference on*, pages 57–61, Sept 2011.

[Fran97D] G. F. Franklin, M. L. Workman, and D. Powell. *Digital control of dynamic systems*. Addison-Wesley Longman Publishing Co., Inc., 1997.

[Gelb68M] A. Gelb and W. E. Vander Velde. *Multiple input describing function and nonlinear system design*. McGraw-Hill, 1968.

[Gill75S] J.-Ch. Gille, P. Decaulne, and M. Pelegrin. *Systemes asservis non-lineaires, tome 1*. Dunod Automatique, 1975.

[Goh95R] Keat-Choon Goh and M.G. Safonov. Robust analysis, sectors, and quadratic functionals. In *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*, volume 2, pages 1988–1993 vol.2, Dec 1995.

[Goog] Google. *Google Inc. Website*. www.google.com.

[H.120] ITU-T Study Group XV. Recommendation H.120, March 1993.

[H.261] CCITT. Rec.H.261, November 1988.

[H.262] ITU-T. Recommendation and International Standard of Joint Video Specification. *ITU-T Rec. H.262/ISO/IEC 13818-2*, July 1995.

[H.263] ITU-T. Rec.H.263, March 1996.

[H.264] ITU-T. Recommendation and International Standard of Joint Video Specification. *ITU-T Rec. H.264/ISO/IEC 14496-10 AVC*, April 2013.

[H.265] ITU-T. Recommendation and International Standard of Joint Video Specification. *ITU-T Rec. H.265/HEVC*, April 2013.

[Han09A] Chang Su Han and Jae-Hun Lee. Area efficient and high throughput cavlc encoder for 1920×1080 @30p h.264/avc. In *Consumer Electronics, 2009. ICCE '09. Digest of Technical Papers International Conference on*, pages 1–2, Jan 2009.

[Herb12E] S. Herbert, Siddharth Garg, and D. Marculescu. Exploiting process variability in voltage/frequency control. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 20(8):1392–1404, Aug 2012.

[Hill76T] D. Hill and P. Moylan. The stability of nonlinear dissipative systems. *Automatic Control, IEEE Transactions on*, 21(5):708–711, Oct 1976.

[Hu08N] Haitao Hu, V. Yousefzadeh, and D. Maksimovic. Nonuniform a/d quantization for improveddynamic responses of digitally controlled dc-dc converters. *Power Electronics, IEEE Transactions on*, 23(4):1998–2005, July 2008.

[Huan05A] Yu-Wen Huang, Bing-Yu Hsieh, Tung-Chien Chen, and Liang-Gee Chen. Analysis, fast algorithm, and vlsi architecture design for h.264/avc intra frame coder. *Circuits and Systems for Video Technology, IEEE Transactions on*, 15(3):378–401, March 2005.

[Huan05A1] Yu-Wen Huang, Tung-Chien Chen, Chen-Han Tsai, Ching-Yeh Chen, To-Wei Chen, Chi-Shi Chen, Chun-Fu Shen, Shyh-Yih Ma, Tu-Chih Wang, Bing-Yu Hsieh, Hung-Chi Fang, and Liang-Gee Chen. A 1.3tops h.264/avc single-chip encoder for hdtv applications. In *Solid-State Circuits Conference, 2005. Digest of Technical Papers. ISSCC. 2005 IEEE International*, pages 128–588 Vol. 1, Feb 2005.

[Huan08H] Feng-Min Huang and S.-F. Lei. High performance and low cost entropy encoder for h.264 avc baseline entropy coding. In *Communications, Circuits and Systems, 2008. ICCCAS 2008. International Conference on*, pages 675–678, May 2008.

[HyPh] Department of Physics and Georgia State University Astronomy. *HyperPhysics*. http://hyperphysics.phy-astr.gsu.edu/.

[Iwat09A] K. Iwata, S. Mochizuki, M. Kimura, T. Shibayama, F. Izuhara, H. Ueda, K. Hosogi, H. Nakata, M. Ehama, T. Kengaku, T. Nakazawa, and H. Watanabe. A 256 mW 40 Mbps Full-HD H.264 High-Profile Codec Featuring a Dual-Macroblock Pipeline Architecture in 65 nm CMOS. *Solid-State Circuits, IEEE Journal of*, 44(4):1184–1191, April 2009.

[JM] Joint Video Team. *H.264 Reference Software, JM.* iphome.hhi.de/suehring/tml.

[Jaco07A] Marco Jacobs and Jonah Probell. A brief history of video coding. *ARC International PLC*, 2007.

[Jain14S] Tejas DaveAmit Jain and Divyanshu Jain. Synchronizer techniques for multi-clock domain socs & fpgas, 2014.

[Javi08E] Javier D. Garcia-Lasheras. Efficient implementation of GALS systems over commercial synchronous fpgas: a new approach. *CoRR*, abs/0802.3441, 2008.

[Joch02P] A. Joch, F. Kossentini, H. Schwarz, T. Wiegand, and G.J. Sullivan. Performance comparison of video coding standards using lagrangian coder control. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 2, pages II–501–II–504 vol.2, 2002.

[Khan08T] A.K. Khan and H. Jamal. The intra prediction in h.264. In T. Sobh, K. Elleithy, A Mahmood, and M.A. Karim, editors, *Novel Algorithms and Techniques in Telecommunications, Automation and Industrial Electronics*, pages 11–15. Springer Science+Business Media B.V., 2008.

[Khan99A] Zia Khan and Gaurav Mehta. Automatic clock gating for power reduction, 1999.

[Kim06I] Daeok Kim, E. Jung, Hyunho Park, Hosoon Shin, and Dongsoo Har. Implementation of high performance cavlc for h.264/avc video codec. In *System-on-Chip for Real-Time Applications, The 6th International Workshop on*, pages 20–23, Dec 2006.

[Kim10I] Yong-Jun Kim, Kyu-Yeul Wang, Sang-Seol Lee, Byung-Soo Kim, Bo-Keun Choi, and Duck-Jin Chung. Implementation of high efficient cavlc encoder for h.264/avc. In *Pervasive Computing Signal*

Processing and Applications (PCSPA), 2010 First International Conference on, pages 912–915, Sept 2010.

[Kim11P] Hyun Kim, Chae-Eun Rhee, Jin-Sung Kim, Sunwoong Kim, and Hyuk-Jae Lee. Power-aware design with various low-power algorithms for an H.264/AVC encoder. In *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*, pages 571–574, May 2011.

[Koch50AA] R. Kochenburger. A frequency method for analyzing and synthesizing contactor servomechanisms. *Transaction A.I.E.E.*, 69:270–280, 1950.

[Koch50AE] RJ. Kochenburger. Analyzing contacter servomechanisms by frequency response methods. *Electrical Engineering*, 69:687–692, 1950.

[Lai05A] Yeong-Kang Lai, Chih-Chung Chou, and Yu-Chieh Chung. A simple and cost effective video encoder with memory-reducing cavlc. In *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, pages 432–435 Vol. 1, May 2005.

[Le07A] Stephen The Uy Le. *A fine grained many-core H.264 video encoder*. Master thesis, University of California, March 2007.

[LecNENTS1] Mirko Loghi. Power estimation and optimization in VLSI digital systems. Ecole Doctorale E.E.A.T.S. Lecture, 2013.

[Lee04V] Jae Hun Lee and Nam Suk Lee. Variable block size motion estimation algorithm and its hardware architecture for h.264/avc. In *Circuits and Systems, 2004. ISCAS '04. Proceedings of the 2004 International Symposium on*, volume 3, pages III–741–4 Vol.3, May 2004.

[Li07A] De-Wei Li, Chun-Wei Ku, Chao-Chung Cheng, Yu-Kun Lin, and Tian-Sheuan Chang. A 61mhz 72k gates 1280x720 30fps h.264 intra encoder. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 2, pages II–801–II–804, April 2007.

[Lin08A1] Yu-Kun Lin, De-Wei Li, Chia-Chun Lin, Tzu-Yun Kuo, Sian-Jin Wu, Wei-Cheng Tai, Wei-Cheng Chang, and Tian-Sheuan Chang. A 242mW 10mm2 1080p H.264/AVC High-Profile Encoder Chip.

In *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, pages 314–615, Feb 2008.

[Lin08A2] Yu-Kun Lin, Chia-Chun Lin, Tzu-Yun Kuo, and Tian-Sheuan Chang. A hardware-efficient h.264/avc motion-estimation design for high-definition video. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 55(6):1526–1535, July 2008.

[Liu07A] Zhenyu Liu, Yang Song, Ming Shao, Shen Li, Lingfeng Li, S. Ishiwata, M. Nakagawa, S. Goto, and T. Ikenaga. A 1.41W H.264/AVC Real-Time Encoder SOC for HDTV1080P. In *VLSI Circuits, 2007 IEEE Symposium on*, pages 12–13, June 2007.

[Lu01C] Yung-Hsiang Lu and G. De Micheli. Comparing system-level power management policies. *Design Test of Computers, IEEE*, 18(2):10–19, Mar 2001.

[Lu02D] Yung-Hsiang Lu, L. Benini, and G. De Micheli. Dynamic frequency scaling with buffer insertion for mixed workloads. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 21(11):1284–1305, Nov 2002.

[Lu03R] Zhijian Lu, J. Lach, M. Stan, and K. Skadron. Reducing multimedia decode power using feedback control. In *Computer Design, 2003. Proceedings. 21st International Conference on*, pages 489–496, Oct 2003.

[Lu08V] WeiJun Lu, Ying Li, Dunshan Yu, and Xing Zhang. Vlsi implementation of an entropy encoder for h.264/avc baseline. In *Industrial Electronics and Applications, 2008. ICIEA 2008. 3rd IEEE Conference on*, pages 1422–1425, June 2008.

[MP4Apps] ITU. *MPEG-4 Applications. http* : $//web.itu.edu.tr/\ pazarci/mpeg4/MPEG4_{A}pplications_{w}2195.htm.$

[Ma10E] Dongsheng Ma and R. Bondade. Enabling power-efficient dvfs operations on silicon. *Circuits and Systems Magazine, IEEE*, 10(1):14–30, First 2010.

[Megr97S] A. Megretski and A. Rantzer. System analysis via integral quadratic constraints. *Automatic Control, IEEE Transactions on*, 42(6):819–830, Jun 1997.

[Meng03E] B. Meng, O.C. Au, Chi-Wah Wong, and Hong-Kwai Lam. Efficient intra-prediction mode selection for 4.times.4 blocks in h.264. In *International Conference on Multimedia and Expo*, volume 3, pages III–521–4, July 2003.

[Ment] Mentor Graphics Corp. *ModelSim tool*. http://www.mentor.com/.

[Micr] Microsoft Corporation. *Microsoft Corporation Website*. www.microsoft.com.

[Moch07A] S. Mochizuki, T. Shibayama, M. Hase, F. Izuhara, K. Akie, M. Nobori, R. Imaoka, H. Ueda, K. Ishikawa, and H. Watanabe. A low power and high picture quality H.264/MPEG-4 video codec IP for HD mobile applications. In *Solid-State Circuits Conference, 2007. ASSCC '07. IEEE Asian*, pages 176–179, Nov 2007.

[Murt90N] S. Murtuza. Non-uniform error-sample control system. In *In Proceedings of the 29th Conference on Decision and Control*, Dec 1990.

[Nero] nero AG. *Nero Multimedia Suite*. http://www.nero.com/eng/downloads/previous-versions/nero-multimedia-suite.php.

[Nguy12A] Ngoc-Mai Nguyen, Xuan-Tu Tran, P. Vivet, and S. Lesecq. An efficient context adaptive variable length coding architecture for h.264/avc video encoders. In *Advanced Technologies for Communications (ATC), 2012 International Conference on*, pages 158–164, Oct 2012.

[Nguy13A] Viet-Thang Nguyen, Xuan-Tu Tran, and Ha Vu Le. An efficient algorithm of inter-prediction coding for h.264/avc video encoders. In *International Conference on Green and Human Information Technology (ICGHIT)*, March 2013.

[Nguy13H] Ngoc-Mai Nguyen, E. Beigne, S. Lesecq, P. Vivet, Duy-Hieu Bui, and Xuan-Tu Tran. Hardware implementation for entropy coding and byte stream packing engine in h.264/avc. In *Advanced Technologies for Communications (ATC), 2013 International Conference on*, pages 360–365, Oct 2013.

[Nguy14A] N.M. Nguyen, E. Beigne, D.H. Bui, N.K. Dang, S. Lesecq, P. Vivet, and Tran. X.T. An overview of h.264 hardware encoder architectures including low-power features. *REV Journal on Electronics and Communications (JEC)*, 4(1-2):8–17, January - June 2014.

[Nguy14F] Ngoc-Mai Nguyen, W. Lombardi, E. Beigne, S. Lesecq, and Xuan-Tu Tran. Fifo-level-based power management and its application to an h.264 encoder. In *Industrial Electronics Society, IECON 2014 - 40th Annual Conference of the IEEE*, pages 158–163, Oct 2014.

[Nguy14H] N.-M. Nguyen, E. Beigne, S. Lesecq, D.-H. Bui, N.-K. Dang, and X.-T. Tran. H.264/avc hardware encoders and low-power features. In *In Proceeding of the IEEE Asia Pacific Conference on Circuits And Systems, APCCAS 2014*, pages 77–80, Nov. 2014.

[Port11A] Marcelo Schiavon Porto, Gustavo Sanchez, Diego Noble, Luciano Agostini, and Sergio Bampi. An efficient me architecture for high definition videos using the new mpds algorithm. In *Proceedings of the 24th Symposium on Integrated Circuits and Systems Design*, SBCCI '11, pages 119–124, New York, NY, USA, 2011. ACM.

[Qu07P] Gang Qu. Power management of multicore multiple voltage embedded systems by task scheduling. In *Parallel Processing Workshops, 2007. ICPPW 2007. International Conference on*, pages 34–34, Sept 2007.

[QuTime] QuickTime. *QuickTime codec*. www.apple.com.

[Rahm07C] C.A. Rahman and W. Badawy. Cavlc encoder design for real-time mobile video applications. *Circuits and Systems II: Express Briefs, IEEE Transactions on*, 54(10):873–877, Oct 2007.

[Ramo10A] F.L.L. Ramos, B. Zatt, T.L. Silva, A. Susin, and S. Bampi. A high throughput cavlc hardware architecture with parallel coefficients processing for hdtv h.264/avc enconding. In *Electronics, Circuits, and Systems (ICECS), 2010 17th IEEE International Conference on*, pages 587–590, Dec 2010.

[Rant96O] A. Rantzer. On the Kalman—Yakubovich—Popov lemma. *Systems & Control Letters*, 28(1):7–10, June 1996.

[Rich03H] Iain E. Richardson. *H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia*. John Willey & Sons, New York, NY, USA, 2003.

[Rich10T] Iain E. Richardson. *The H.264 Advanced Video Compression standard, 2nd edition*. John Willey & Sons, New York, NY, USA, 2010.

[Rodr13P] D. Zegarra Rodriguez and G. Bressan. Performance assessment of high efficiency video coding - hevc. In *Global High Tech Congress on Electronics (GHTCE), 2013 IEEE*, pages 110–111, Nov 2013.

[Romb13A] P. Rombouts, M. De Bock, J. De Maeyer, and L. Weyten. A describing function study of saturated quantization and its application to the stability analysis of multi-bit sigma delta modulators. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 60(7):1740–1752, July 2013.

[Safo09R] Michael George Safonov. Robust control: Stability margin. In Christodoulos A. Floudas and Panos M. Pardalos, editors, *Encyclopedia of Optimization*, pages 3300–3305. Springer US, 2009.

[Safo80S] Michael George Safonov. *Stability and Robustness of Multivariable Feedback Systems*. MIT Press, Cambridge, MA, USA, 1980.

[Sanc12D] Gustavo Sanchez, Felipe Sampaio, Marcelo Porto, Sergio Bampi, and Luciano Agostini. Dmpds: A fast motion estimation algorithm targeting high resolution videos and its fpga implementation. *International Journal of Reconfigurable Computing*, 2012.

[Scha95D] R. Schafer and T. Sikora. Digital video coding standards and their role in video communications. *Proceedings of the IEEE*, 83(6):907–924, Jun 1995.

[Stan13E] C.S. Stangaciu, M.V. Micea, and V.I. Cretu. Energy efficiency in real-time systems: A brief overview. In *Applied Computational Intelligence and Informatics (SACI), 2013 IEEE 8th International Symposium on*, pages 275–280, May 2013.

[Sula08U] D.R. Sulaiman. Using clock gating technique for energy reduction in portable computers. In *Computer and Communication Engineering, 2008. ICCCE 2008. International Conference on*, pages 839–842, May 2008.

[Sull05O] G. Sullivan's presentation. Overview of International Video Coding Standards. *ITU-T VICA Workshop*, July 2005.

[Sylv07A] Miermont Sylvain, Pascal Vivet, and Marc Renaudin. A power supply selector for energy- and area-efficient local dynamic voltage scaling. In *In Proceedings of PATMOS 2007, Goteborg, Sweden*, pages 556–565, September 2007.

[Syno] Synopsys Inc. *Design Compiler tool*. http://www.synopsys.com/.

[Thie05D] L. Thiele, S. Chakraborty, and A. Maxiaguine. Dvs for buffer-constrained architectures with predictable qos-energy tradeoffs. In *Hardware/Software Codesign and System Synthesis, 2005. CODES+ISSS '05. Third IEEE/ACM/IFIP International Conference on*, pages 111–116, Sept 2005.

[Tian08I] X.H. Tian, T.M. Le, X. Jiang, and Y. Lian. Implementation strategies for statistical codec designs in h.264/avc standard. In *Rapid System Prototyping, 2008. RSP '08. The 19th IEEE/IFIP International Symposium on*, pages 151–157, June 2008.

[Tlib05C] S. Tliba, M. Jungers, and Y. Chitour. Commande des processus asservissements numeriques. Note de cours, 2005.

[Tran11C] Xuan-Tu Tran and Van-Huan Tran. Cost-efficient 130nm tsmc forward transform and quantization for h.264/avc encoders. In *Design and Diagnostics of Electronic Circuits Systems (DDECS), 2011 IEEE 14th International Symposium on*, pages 47–52, April 2011.

[Tran13B] Tran. et al. *VENGME QGDA.10.02*. VNU Hanoi, 2013. Project summary report.

[Tsai06L] Chuan-Yung Tsai, Tung-Chien Chen, and Liang-Gee Chen. Low power entropy coding hardware design for h.264/avc baseline profile encoder. In *Multimedia and Expo, 2006 IEEE International Conference on*, pages 1941–1944, July 2006.

[VidLib] Arizona State University. *Video Trace Library*. http://trace.eas.asu.edu/index.html.

[Vive10O]

[Wang03A] Shih-Hao Wang, Wen-Hsiao Peng, Yuwen He, Guan-Yi Lin, Chen-Yi Lin, Shih-Chien Chang, Chung-Neng Wang, and Tihao Chiang. A platform-based mpeg-4 advanced video coding (avc) decoder with block level pipelining. In *Information, Communications and Signal Processing, 2003 and Fourth Pacific Rim Conference on Multimedia. Proceedings of the 2003 Joint Conference of the Fourth International Conference on*, volume 1, pages 51–55 Vol.1, Dec 2003.

[Watk04T] John Watkinson. *The MPEG Handbook, 2nd edition.* Taylor & Francis, UK, 2004.

[Wen11D] Mei Wen, Ju Ren, Nan Wu, Huayou Su, ChangQing Xun, and Chunyuan Zhang. Data parallelism exploiting for h.264 encoder. In *Multimedia and Signal Processing (CMSP), 2011 International Conference on*, volume 1, pages 188–192, May 2011.

[Weng03A] Li-Chuan Weng, XiaoJun Wang, and Bin Liu. A survey of dynamic power optimization techniques. In *System-on-Chip for Real-Time Applications, 2003. Proceedings. The 3rd IEEE International Workshop on*, pages 48–52, June 2003.

[Wieg03O] T. Wiegand, G.J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h.264/avc video coding standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(7):560–576, July 2003.

[Wu05V] Q. Wu, P. Juang, M. Martonosi, and D.W. Clark. Voltage and frequency control with adaptive reaction time in multiple-clock-domain processors. In *High-Performance Computer Architecture, 2005. HPCA-11. 11th International Symposium on*, pages 178–189, Feb 2005.

[Xiao11A] Zhibin Xiao and B.M. Baas. A 1080p h.264/avc baseline residual encoder for a fine-grained many-core system. *Circuits and Systems for Video Technology, IEEE Transactions on*, 21(7):890–902, July 2011.

[Yada12A] M.K. Yadav, M.R. Casu, and M. Zamboni. Dvfs based on voltage dithering and clock scheduling for gals systems. In *Asynchronous Circuits and Systems (ASYNC), 2012 18th IEEE International Symposium on*, pages 118–125, May 2012.

[Yada12D] M.K. Yadav, M.R. Casu, and M. Zamboni. Dvfs based on voltage dithering and clock scheduling for gals systems. In *Asynchronous Circuits and Systems (ASYNC), 2012 18th IEEE International Symposium on*, pages 118–125, May 2012.

[Yi08H] Yongseok Yi and Byung Cheol Song. High-speed cavlc encoder for 1080p 60-hz h.264 codec. *Signal Processing Letters, IEEE*, 15:891–894, 2008.

[Yin10A] HaiBing Yin, Huizhu Jia, Honggang Qi, Xianghu Ji, Xiaodong Xie, and Wen Gao. A hardware-efficient multi-resolution block matching algorithm and its vlsi architecture for high definition mpeg-like video encoders. *Circuits and Systems for Video Technology, IEEE Transactions on*, 20(9):1242–1254, Sept 2010.

[Zhur13S] S. Zhuravlev, J.C. Saez, S. Blagodurov, A. Fedorova, and M. Prieto. Survey of energy-cognizant scheduling techniques. *Parallel and Distributed Systems, IEEE Transactions on*, 24(7):1447–1464, July 2013.

[Zrid09H] H.K. Zrida, A. Jemai, A.C. Ammari, and M. Abid. High level h.264/avc video encoder parallelization for multiprocessor implementation. In *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09.*, pages 940–945, April 2009.

[Zuo12A] Shikai Zuo, Mingjiang Wang, and Liyi Xiao. A cache hardware design for h.264 encoder. In *Instrumentation, Measurement, Computer, Communication and Control (IMCCC), 2012 Second International Conference on*, pages 922–925, Dec 2012.

[bbcrd] BBC R&D. *BBC Research & Development Website.* www.bbc.co.uk/rd.

[iec] ISO/IEC. *International Electrotechnical Commission Website.* www.iec.ch.

[iso] ISO. *International Organization for Standardization Website.* www.iso.org.

[itu] ITU. *International Telecommunication Union Website.* www.itu.int.

[lpsa] Swarthmore College. *Linear Physical Systems Analysis.* http://lpsa.swarthmore.edu/.

[mpeg-1] ISO/IEC MPEG. *ISO/IEC 11172, Coding of moving pictures and associated audio at up to about 1.5 Mbit/s.* mpeg.chiariglione.org/standards/mpeg-1, 1993.

[mpeg-4] ISO/IEC MPEG. ISO/IEC 14496, Coding of audio-visual objects. *mpeg.chiariglione.org/standards/mpeg-4*, 1998.

[smpte] SMPTE. *Society of Motion Picture and Television Engineers Website.* www.smpte.org.

[stm] STMicroelectronics. *CMOS technology.* http://www.st.com/.

[x264] x264 team. *x264 codec.* www.videolan.org/developers/x264.html.

RÉSUMÉ DE THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

préparée dans le cadre d'une cotutelle entre l'*Université Grenoble Alpes* et l'*Université Nationale du Vietnam, Hanoi*

Spécialité : **Automatique - Productique**

Arrêté ministériel : le 6 janvier 2005 - 7 août 2006

Présentée par

Ngoc-Mai NGUYEN

Thèse dirigée par **Xuan-Tu TRAN** et **Suzanne LESECQ**
co-encadrée par **Edith BEIGNÉ**

préparée au sein des **Laboratoire d'Electronique et des Technologies de l'Information, CEA Grenoble et Laboratoire SIS, VNU-UET, Hanoi**

dans **l'École Doctorale Electronique, Electrotechnique, Automatique et Traitement du Signal (EEATS) et l'UET du VNU**

Stratégies d'optimisation de la consommation pour un système sur puce encodeur H.264

Thèse soutenue publiquement le **29/06/2015**,
devant le jury composé de :

M. Nicolas MARCHAND
Laboratoire GIPSA-Lab, Président
M. Alain MERIGOT
Université Paris XI, Rapporteur
Mme. Nadine AZEMARD
LIRMM, Rapporteur
M. Xuan-Tu TRAN
Université Nationale du Vietnam, Directeur de thèse
M. Amara AMARA
ISEP, Membre
M. Huu-Duc NGUYEN
Université Nationale du Vietnam, Membre
Mme. Edith BEIGNÉ
CEA, Invitée
Mme. Suzanne LESECQ
CEA, Co-directeur de thèse, Invitée

Introduction

Les circuits dédiés aux codecs vidéo ont été récemment utilisés dans diverses applications, par exemple les conférences vidéo, les systèmes vidéo de surveillance, ou bien encore les applications grand public de divertissement. Pour répondre aux besoins des applications mobiles, le codec vidéo est de préférence mis en œuvre via une implémentation matérielle plutôt que logicielle, ce qui lui garantit de meilleures performances en terme d'efficacité énergétique et de traitement temps-réel. L'une des normes les plus récentes et les plus efficaces pour les applications vidéo est la norme H.264 qui offre un codage vidéo avancé (H.264/AVC) avec une meilleure qualité vidéo et un débit plus faible que les standards précédents. Il a été spécifié dans le cadre d'une coopération entre le groupe d'experts en codage vidéo de l'UIT-T et le groupe d'experts en images animées de l'ISO/IEC. La spécification H.264/AVC adopte de nombreuses méthodes et approches pour la compression vidéo avancée et elle les améliore également. Cela rend la norme plus efficace, mais nécessite la mise en œuvre de matériel plus complexe, et une consommation accrue

Récemment, le successeur de la norme H.264, à savoir la norme H.265/High Efficiency Video Coding (HEVC) a été publiée. Cette nouvelle norme promet une bande passante accrue, et le traitement de formats avancés tels que la Télévision Ultra Haute Définition (UHDTV). Cependant, cette nouvelle norme présente également une complexité de calcul encore plus élevée que la complexité de calcul de la norme H.264, ce qui conduit à une plus grande consommation en énergie et donc à une durée de vie de la batterie des équipements mobiles qui diminue. Avec ces coûts plus élevés (en termes calculatoire et consommation énergétique), le passage à la nouvelle norme d'encodage/décodage vidéo doit être soigneusement pesé et il est fort à parier que les codecs vidéo H.264/AVC seront encore en utilisation dans les années à venir.

Le projet VENGME (Video Encoder for the Next Generation Multimedia Equipment) est un projet de recherche accordé et financé par l'Université Nationale du Vietnam, Hanoi (VNU). Il vise à concevoir et à mettre en œuvre un encodeur matériel H.264/AVC ciblant les plateformes mobiles. La conception actuelle est optimisée pour la vidéo CIF. Cependant, l'architecture de la plateforme VENGME peut être utilisée également pour de plus grandes résolutions en élargissant la mémoire de référence et la fenêtre de recherche.

Pour amener la norme H.264/AVC dans les produits commerciaux, en particulier pour les appareils mobiles, les concepteurs ont besoin d'appliquer des techniques de conception pour circuits de faibles consommations. Ciblant les applications mobiles, la plateforme

VENGME a été conçue en utilisant les techniques de conception de circuits de faible consommation. Certains blocs de la plateforme présentent des chiffres de consommation et des performances très intéressants, semblables à ceux des conceptions de l'état de l'art. Toutefois, l'ensemble des résultats de consommation de la plateforme pourraient être encore améliorés, en comparaison avec les encodeurs vidéo matériels de l'état de l'art. Pour améliorer les résultats en puissance de la plateforme, sa charge de travail déséquilibrée peut être exploitée via différentes techniques de contrôle de la puissance. Cela comprend l'application de technique de type DPM (Dynamic Power Management), par exemple des méthodes de variation dynamique de la tension d'alimentation et de la fréquence d'horloge (Dynamic voltage Frequency Scaling, DVFS).

L'objectif de cette thèse est d'abord de concevoir le module EC-NAL de l'encodeur vidéo de la plateforme VENGME. D'autre part, l'architecture de la plateforme VENGME est analysée pour extraire une cartographie de la consommation. Nous proposons également de nouveaux mécanismes pour contrôler la consommation d'énergie pour un encodeur matériel H.264, en particulier pour la plateforme VENGME. Ensuite, une solution de commande de bas niveau matériel est proposée pour réduire la consommation d'énergie du système. Enfin, la mise en œuvre matérielle et la validation sont réalisées pour prouver la fonctionnalité et l'intérêt de la solution de commande proposée et implantée en matériel.

Pour répondre aux objectifs, cette thèse apporte trois contributions principales :

- **Concevoir le module EC-NAL (Entropy Coder and Network Abstraction Layer)**

  Cette contribution nécessite une compréhension complète des techniques de l'encodeur H.264. Les implémentations H.264 de l'état de l'art sont examinées et les implémentations disponibles sur le matériel sont classées suivant différents critères, en particuliers les conceptions orientées basse consommation.

  Le module EC-NAL est le module final dans le chemin de codage d'un codeur vidéo. Son rôle est d'éliminer la redondance statique dans les données vidéo du signal encodé. Il reçoit les données de la plupart des modules de la plateforme (en fait, de tous les modules dans le chemin de codage) et compresse ces données en utilisant une certaine technique de codage entropique. Notons qu'il existe trois techniques principales possibles de codage entropique pour la norme H.264, à savoir, Exp-Golomb, CAVLC et CABAC. Dans notre

encodeur, les techniques implémentées sont Exp-Golomb et CAVLC, utilisés dans le profil principal de la norme H.264.

Pour l'encodage H.264 implanté dans l'encodeur vidéo H.264 VENGME, nous avons proposé, conçu et mis en œuvre le module EC-NAL. Le module a été intégré et validé dans la plateforme complète ;

- **Estimer la puissance au niveau RTL pour la plateforme VENGME d'encodeur H.264**

  Une estimation de la puissance est effectuée sur la plateforme VENGME complète pour obtenir une pré-évaluation de la cartographie de la consommation, une analyse des charges calculatoires des différents modules et sous-modules, et une analyse de l'opportunité d'appliquer une méthode de gestion de l'alimentation. L'objectif pour ce dernier point est de savoir si il est possible de réduire la consommation en ajoutant des mécanismes de contrôle des tension d'alimentation et fréquence d'horloge des différents modules et sous-modules ;

- **proposer une méthode pour la réduction de la consommation d'énergie sur la base du niveau d'occupation de FIFO**

  On propose une méthode pour optimiser la consommation d'énergie en modifiant certains des facteurs qui influent sur la consommation d'énergie des circuits CMOS numériques. Idéalement, cela nécessite un moyen de surveiller l'état de la charge de travail du circuit. Des techniques basées sur le contrôle de l'état d'un tampon mémoire de type FIFO (First In First Out) ont été étudiées.

  La méthode proposée dans notre travail est basée sur la théorie du contrôle. La décision de commande est réalisée en fonction du niveau d'occupation de la FIFO placée dans la liaison entre deux (sous-)modules de calcul. En fonction de ce niveau, la fréquence d'horologe du second sous-module (consommateur) est adaptée.

  Le système à contrôlé a été modélisé. La loi de contrôle proposée a été vérifiée dans l'environnement Matlab. On a vérifié que le système bouclé est stable. La méthode proposée est mise en œuvre et validée dans Matlab pour une FIFO implantée dans le module EC-NAL de la plate-forme de VENGME. Le contrôleur a ensuite été modélisé en VHDL et mis en œuvre dans le matériel. Son fonctionnement est vérifié par l'intermédiaire de l'intégration dans le module EC-NAL de la plate-forme de VENGME. L'intérêt de la méthode a

également été validé via une estimation de la puissance avant et après l'implémentation de la technique de contrôle du niveau dans la FIFO.

Ce travail de thèse est menée dans trois laboratoires, à savoir le laboratoire des Systèmes Intelligents Intégrés (SIS) à l'Université nationale du Vietnam, Hanoi (VNU), Vietnam, le Vietnam, le Laboratoire Intégration Silicium et Architecture Numérique (LISAN) et le Laboratoire Infrastructures et Ateliers Logiciel pour Puces (LIALP) du CEA-LETI, France. L'architecture de la plateforme VENGME est conçue dans le laboratoire SIS. Les techniques de conception de circuits à faible puissance sont étudiées dans le laboratoire LISAN. La partie contrôle est étudiée avec le laboratoire LIALP. Cette variété de lieu et d'équipes de recherches a conduit l'auteur à travailler avec différentes technologies, y compris les plus avancées.

En dehors de cette introduction, ce résumé étendu contient quatre chapitres et une conclusion. Le premier chapitre introduit la norme de codage vidéo H.264 et la positionne dans le paysage des autres normes de codage vidéo utilisées dans le passé ou toujours en cours d'utilisation. Certains concepts de base du codage vidéo, et en particulier du codage vidéo H.264, sont également présentés. En outre, à la fin du chapitre, un état de l'art de H.264 des implémentations matérielles est fourni et analysé. Ces travaux de l'état de l'art posent certaines questions quant à l'application de la norme de codage vidéo H.264, et à l'implémentation matérielle qui en est faite.

Le deuxième chapitre présente la plateforme VENGME. Cette dernière est un encodeur matériel H.264/AVC ciblant les applications mobiles. La plateforme VENGME a été conçue par le groupe de recherche au Vietnam dans le cadre du projet VENGME. L'architecture, la modélisation et la validation du module EC-NAL, l'une des principales contributions de cette thèse, sont également présentées dans ce chapitre. Les simulations de puissance au niveau RTL pour l'ensemble de la plateforme VENGME et pour le module EC-NAL sont menées. L'objectif est d'analyser s'il est possible de diminuer la consommation de la plateforme en implémentant des techniques de contrôle, en particulier en contrôlant le niveau de remplissage des FIFOs.

Dans le troisième chapitre, la gestion de la consommation via le contrôle du niveau d'une FIFO est étudiée. Ensuite, une méthode pour contrôler le niveau de la FIFO pour diminuer la consommation est proposée et appliquée à la mémoire FIFO intégrée dans le module EC-NAL. Toutes les étapes de modélisation du système, la conception du contrôle,

l'analyse de la stabilité du système bouclé et des résultats en simulation dans l'environnement MATLAB sont présentés.

Le quatrième chapitre présente la mise en œuvre matérielle de la méthode de contrôle du niveau de la FIFO proposée dans le chapitre précédent. Les résultats en simulation d'estimation de la puissance permettent de vérifier le fonctionnement du contrôle et de valider de la méthode.

Enfin, le dernier chapitre résume les travaux effectués au cours de cette thèse et il dresse quelques perspectives à court et moyen terme.

Chapitre 1: La norme de codage vidéo H.264/AVC et ses implémentations

Les objectifs de ce chapitre sont d'examiner tout d'abord différents formats de codage vidéo et des normes anciennes, et de présenter leur gamme d'applications. Ensuite, les concepts de base des techniques de codage des normes vidéo H.264 et de l'Advanced Video Coding (H.264/AVC) seront présentés. Enfin, les principales difficultés et les tendances en matière de mise en œuvre matérielle de la norme H.264/AVC seront discutées avec un aperçu et une analyse de différentes familles d'implémentation.

La norme H.264/AVC de codage vidéo, qui fait suite à la norme H.263, est une norme de compression vidéo très largement utilisée. Elle a été recommandée par le Joint Video Team (JVT) qui a été formé par l'UIT-T VCEG et les groupes ISO/IEC MPEG en 2001. Elle utilise un codage par blocs. Ses objectifs techniques primaires sont d'améliorer de manière significative l'efficacité du codage, à avoir une meilleure robustesse face aux erreurs/pertes, à obtenir une faible latence pour une utilisation en temps réel, à améliorer la flexibilité pour l'utilisateur sur différents systèmes et réseaux. La première recommandation a été finalisée en 2003.

La norme H.264/AVC contient un large éventail d'outils de codage vidéo pour soutenir une variété d'applications allant des services mobiles, la vidéoconférence, la diffusion numérique pour l'IPTV, la HDTV et les supports de stockage numérique. Par rapport aux normes antérieures telles que MPEG-4, H.263, MPEG-2, la norme H.264/AVC atteint respectivement 39%, 49%, et 64% de réduction du débit binaire.

L'architecture générale du codeur H.264/AVC est construite à partir de différents blocs fonctionnels, voir la Figure 1.

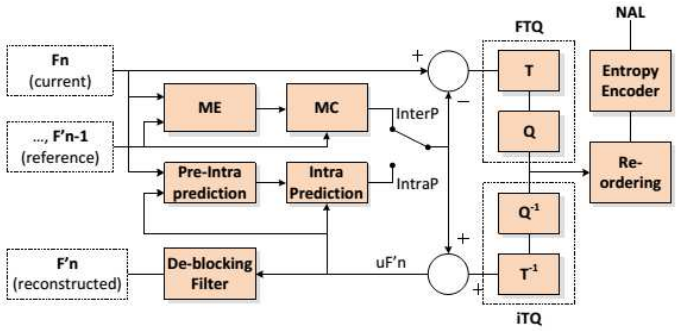

Figure 1: Schéma de l'encodeur H.264.

Le chemin de codage comprend le bloc de prédiction Intra, la prédiction Inter contenant l'estimation de mouvement (ME) et les blocs de Compensation de Mouvement (MC), le bloc Forward Transform et Quantification (FTQ), le codeur d'entropie. Le bloc de prédiction Intra prédit le « macrobloc » courant (MB, un bloc de 16×16 pixels) sur la base des pixels précédemment encodées, à supprimer les redondances spatiales des données vidéo. Afin d'éliminer les redondances temporelles des données vidéo, le bloc de prédiction inter estime les mouvements des MB actuels basés sur les pixels précédemment encodées dans les images différentes. Les données résiduelles, à savoir la différence entre le MB original courant et celui prédit, est ensuite transformé et quantifié. Les coefficients post-quantification sont alors reclassés et un codage entropique est appliqué. La vidéo encodée peut être encapsulé dans la Couche d'abstraction de Réseau (Network Abstraction Layer, NAL). Un chemin de décodage qui contient la transformée inverse et de dé-quantification (ITQ) et le filtre de déblocage est également intégré dans l'encodeur vidéo pour générer des données de référence pour les procédés de prédiction. La prédiction intra utilise directement les données du bloc ITQ, tandis que la prédiction inter fait référence à des trames reconstruites à partir du filtre de déblocage.

Chaque norme définit différents profils comme sous-ensembles d'outils ou des algorithmes qui peuvent être utilisés lors du codage. Des niveaux, qui sont les contraintes sur les paramètres clés du bitstream, sont également définis. Initialement, la norme H.264/AVC a défini trois profils, à savoir, le profil de base, le profil principal et des profils étendus.

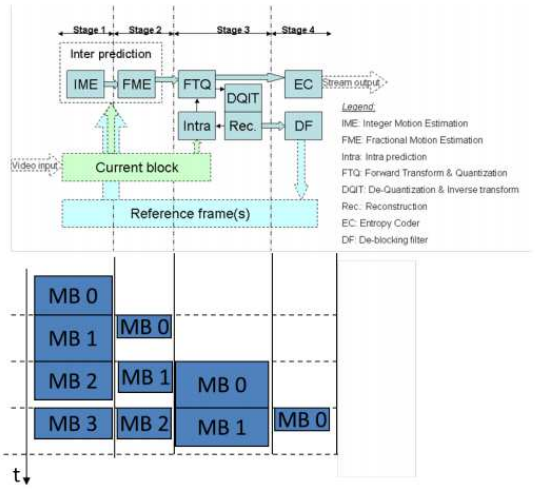

Figure 2: Architecture de pipeline classique d'un encodeur H.264 encodeur matériel.

En raison du flot de codage long de la norme H.264, des architectures en pipeline au niveau MB sont habituellement mises en œuvre. La Figure 2 montre les principaux modules d'un encodeur H.264 en quatre étapes (ou niveaux). Le bloc (ou module) Motion Estimation

(ME), fonctionnant avec la Motion Compensation (MC) pour effectuer une prédiction inter, est un outil de codage potentiel, mais avec une complexité de calcul énorme. Le module ME en plein travail peut compter pour plus de 90% du coût calcul global. Par conséquent, dans les architectures en pipeline, la tâche ME est coupée en deux sous-tâches (entier (IME) et fractionnaire (FME)). Pour obtenir une charge plus équilibrée, la prédiction intra (IntraP) est placée dans la troisième étape. La décision de mode intra nécessite la transformation directe de quantification/de-quantification et de transformation inverse (FTQ/ITQ) et les résultats de la reconstruction du MB courant dans le même stade que le bloc IntraP. La dernière étape comprend deux modules indépendants, à savoir Entropy Coder (EC) et filtre de blocage (DF). Afin de réduire la taille de la mémoire tampon entre les étages, le pipeline est généralement prévu pour fonctionner au niveau MB plutôt qu'au niveau de la trame. L'architecture en pipeline à quatre étages coupe le flot de codage d'une manière équilibrée, ce qui facilite la planification des tâches mais elle augmente également le temps de latence globale de l'encodeur.

Des encodeurs H.264 adaptatifs ont été mis en œuvre en intégrant des fonctions supplémentaires. Ils exigent un coût matériel élevé, ainsi qu'une consommation accrue.

D'autre encodeurs H.264 sont conçus avec un souci d'améliorer leur vitesse de traitement Les concepteurs peuvent alors modifier l'architecture du pipeline pour obtenir plus de parallélisation du traitement. Par exemple, l'encodeur H.264 proposé par Iwata et al. [Iwat09A], voir la Figure 3, utilise les deux structures de pipelines CE1 et CE0, pour encoder deux lignes de MB en même temps.

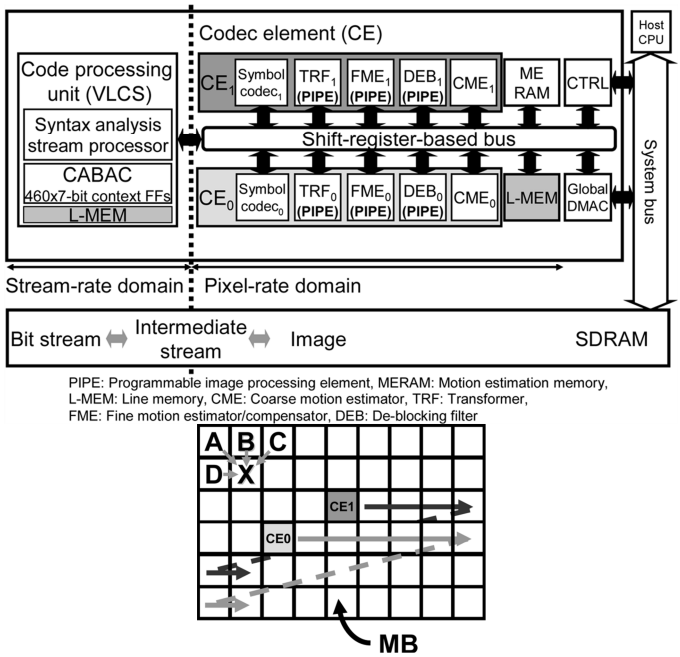

Figure 3: Encodeur H.264 orienté vitesse [Iwat09A].

Dans la conception d'un encodeur H.264 qui s'intéresse en premier lieu à la consommation de puissance, on peut utiliser des techniques spécifiques de conception de circuits faible puissance, ou appliquer des techniques de réduction de l'accès de mémoire au cours de la phase de conception, ou encore développer une certaine évolutivité de la plateforme par le paramétrage. Par exemple, la Figure 4 illustre les deux techniques spécifiques pour désactiver le signal d'horloge (clock gating) pour les modules ou étages moins chargés. Dans la figure, les segments noirs représentent les périodes de temps pendant lesquelles le module fonctionne. Les segments en orange représentent les périodes où la désactivation de l'horloge est appliquée. Les segments bleus sont pour les arrêts dynamiques de l'horloge (DCSS). Alors que le DCSS est appliqué uniquement lorsque chaque module de la même étape ne fonctionne pas, à grain fin, le clock gating est appliqué au niveau du module.

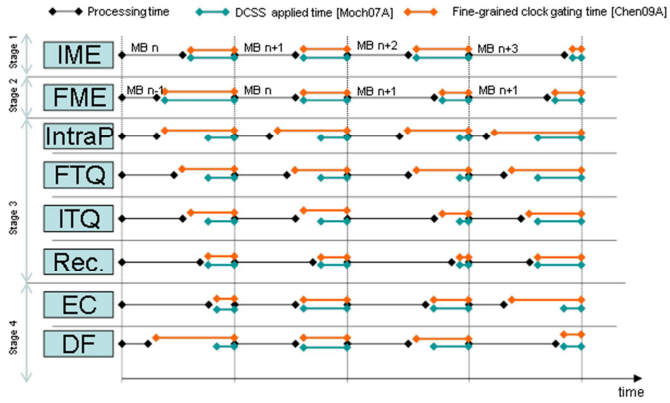

Figure 4: Dynamic arrêt de tension d'horloge (DCSS) et grain fin clock gating.

Dans ce chapitre, un aperçu des normes de codage vidéo et la norme H.264/AVC a été présenté. Des solutions d'état de l'art sur la conception et la mise en œuvre matérielle des encodeurs vidéo H.264/AVC ont été discutées.

La norme H.264/AVC adopte de nombreuses techniques de codage vidéo avancé issues de normes précédentes. Il est largement appliqué dans de nombreuses applications. Même si la nouvelle norme H.265 est maintenant disponible, il est difficile de dire quoi que ce soit quant à son succès dans un futur proche.

Pour les plateformes mobiles, la tendance est à mettre en œuvre l'encodeur H.264/AVC en matériel dédié. Comme présenté ci-dessus, de nombreuses techniques peuvent alors être appliquées pour améliorer l'architecture du pipeline de base en termes d'évolutivité, de vitesse et de consommation d'énergie.

Chapitre 2: La plateforme VENGME et son module EC-NAL

Ce chapitre est dédié à la conception d'un encodeur H.264/AVC matériel qui cible les applications mobiles. Cette conception a été menée dans le cadre du projet de recherche VENGME (n° QGDA.10.02), financé par l'Université Nationale du Vietnam, Hanoi (VNU). La plateforme VENGME, qui est un encodeur vidéo H.264/AVC, a été conçue en appliquant des techniques d'amélioration dans tous les modules (autrement dit, chaque module, individuellement a été optimisé en vue de diminuer sa consommation d'énergie).

Une des contributions de cette thèse est la conception et la mise en œuvre du module de Codage Entropique et Network Abstraction Layer Data Packer (EC-NAL) de la plateforme VENGME.

Les estimations de puissance, au niveau RTL, sur l'ensemble de la plateforme VENGME et sur le module EC-NAL sont également réalisées pour analyser la répartition de la consommation de puissance de la plateforme, en vue de l'optimisation de la consommation.

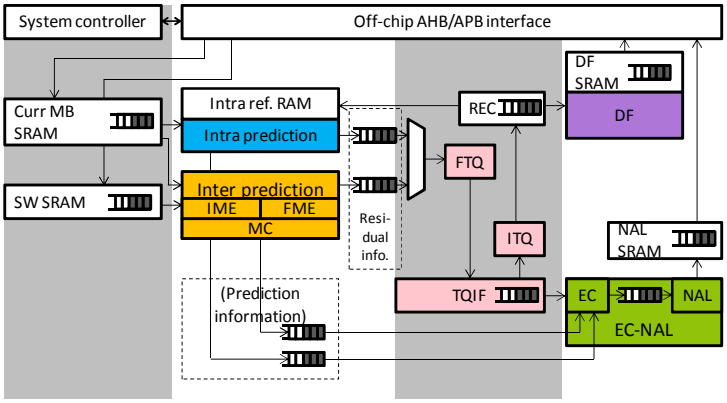

Figure 5: L'architecture du codeur H.264 VENGME.

L'architecture matérielle de l'encodeur VENGME H.264/AVC est représentée sur la Figure 5. Il contient tous les blocs qui apparaissent dans une architecture d'encodeur H.264 standard, à savoir, la prédiction Intra (IP), la prédiction inter constituée de Motion Estimation (ME) et de Motion Compensation (MC), la transformation et la quantification regroupés dans le module FTQ, la dé-quantification et transformée inverse (ITQ), le codeur entropique (EC), la couche d'abstraction de réseau (NAL), la reconstruction d'un Macroblock (Recons. MB), le filtre de déblocage (DF), et le taux de distortion (RDO).

L'encodeur vidéo communique avec le système externe via un bus Advanced Microcontroller Bus Architecture (AMBA). La configuration des paramètres de l'encodeur se fait via le Bus Périphérique Avancé (APB), une interface de bus esclave, tandis que les

données vidéo sont transférées via le Bus Avancé haute performance (AHB), une interface maître de bus. Après la phase de configuration, le codeur commence le procédé de codage H.264. Les entrées de la source vidéo, les données de référence et les données codées sont stockées dans des mémoires hors puce. Par conséquent, l'encodeur vidéo a également besoin d'accéder à ces blocs mémoire via le bus maître AHB pendant le processus d'encodage. Un bus interne est utilisé pour communiquer entre les Unités Memory Access (MAU) et l'Interface Maître de Bus AHB. Dans l'architecture VENGME, il y a quatre unités d'accès de la mémoire (MAUs):

- l'unité d'accès de la mémoire de MB actuelle (CMB MAU) est utilisée pour récupérer les données de la source ;
- l'unité d'accès de la mémoire de la Fenêtre de recherche (SW MAU) est utilisée pour accéder à des données de référence ;
- l'unité d'accès de la mémoire de filtre De-blocage (DF MAU) est utilisée par le module DF ;
- l'unité d'accès de la mémoire de la couche d'abstraction de réseau (NAL MAU) est utilisée pour écrire les données codées.

Parmi ces unités, le NAL MAU a la plus haute priorité pour éviter un goulot d'étranglement dans le flux de données.

Tous les modules de l'encodeur H.264/AVC ont été modélisés en VHDL avec certains modèles RAM/ROM en Verilog. Il est important de vérifier et de valider l'ensemble du système intégré. Un CPU ou un noyau IP peuvent contrôler l'encodeur VENGME en accédant à ses registres de contrôle via une communication du système de bus AMBA AHB/APB. Le modèle de DDR2 de bloc de mémoire est utilisé pour stocker les données vidéo d'entrée et de référence. Les vidéos d'entrée de la source sont les vidéos de référence standard. Le bloc AHB_VIP lit le fichier vidéo YUV et charge son contenu dans le bloc de mémoire. Les données codées sont ensuite décodées par le décodeur logiciel de référence JM18.

Il a été prouvé que le codeur VENGME peut coder avec succès les données vidéo au format H.264. La Figure 6 illustre les résultats obtenus. L’image originale (Suzie.yuv) est donnée sur la gauche et la vidéo encodée avec la plateforme VENGME (puis décodée avec JM18) est donnée sur la droite.

Figure 6: Figures extraites de la vidéo originale (gauche) et encodée avec l'encodeur VENGME (et décodée avec JM18).

Avant la phase de fabrication, le codeur doit être validé pour éviter des erreurs inattendues. L'outil DE2 de Altera est utilisé pour mettre en œuvre le système sur puce (SoC) pour la phase de validation. La synthèse est réalisée au Vietnam par des personnes du projet de VENGME. Certains modules de VENGME présentent des performances et/ou de consommation en puissance intéressants, au moins équivalentes aux conceptions de l'état de l'art. La Table 1 illustre les résultats de la plateforme VENGME pour la technologie CMOS 130nm de Global Fonderie (GF). Ces résultats de synthèse pour l'ensemble de la plateforme de VENGME ont été obtenus en utilisant l'outil Design Compiler de Synopsys.

Table 1: Résultats de VENGME

	Intra prediction	Inter prediction	Transform Quantization	DF	EC-NAL	Mémoire et Contrôle	VENGME en totale
Ressource (*Kgate*)	30.448	550.415	90.778	151.314	90.648	27.953*Kbyte*	913.604 (sauf de la mémoire)
Consommation (*mW*)	1.58	14.77	1.46	2.66	3.72	34.06	58.25

Le codeur VENGME, ciblant la résolution CIF, nécessite une ressource de 913,604 Kgate pour les blocs fonctionnels et 27,953KB pour les blocs mémoire. Il consomme 58,25mW, ce qui est tout à fait approprié pour une application mobile.

En comparaison avec d'autres encodeurs vidéo H.264/AVC implantés en matériel, l'encodeur VENGME a des résultats dans l'état de l'art (pour une technologie similaire). Le coût en consommation et la taille de la plateforme VENGME sont inférieurs à la plupart des implémentations de l'état de l'art, mais ces deux chiffres sont légèrement plus élevés que ceux des meilleures implémentations, avec la même cible technologique. Par exemple, en ciblant également la résolution CIF et avec une synthèse en technologie CMOS 130nm, la conception

dans [Chan09A] consomme de 7mW à 25mW. Le circuit dans [Lin08A1] consomme 6,74mW. La conception dans [Chen09A], visant également à la résolution CIF et utilisant la technologie CMOS 180nm, consomme de 9,8mW à 40,3 mW en fonction du niveau de puissance. Cependant, les conceptions dans [Lin08A1, Chan09A et Chen09A] qui ciblent le profil de base, tandis que la plateforme VENGME vise le profil principal. Cela rend la comparaison pas tout à fait correcte. La conception dans [Kim11P] vise également la résolution CIF, mais sa consommation en énergie est plutôt élevée. La plateforme décrite dans cette publication consomme de 238,38mW à 359,89mW selon le niveau de puissance.

En conclusion, les résultats de l'encodeur VENGME vidéo peuvent être considérées comme équivalents à ceux des conceptions de l'état de l'art. L'application de certaines méthodes de gestion de l'alimentation pourrait aider à réduire sa consommation d'énergie.

La conception du module EC-NAL, qui est une des contributions de cette thèse, est maintenant présentée.

Le codeur entropique (EC) dans l'encodeur VENGME implémente les solutions Exp-Golomb et CAVLC, pour toutes les informations vidéo H.264, à savoir, le mode de prédiction, le vecteur de mouvement, les données résiduelles, etc. Les processus les données du codeur entropique EC dans les éléments de syntaxe, à savoir, les éléments de données sont représentées dans le flux binaire. Les éléments codés sont ensuite « emballés » dans la syntaxe spécifiée dans la norme H.264/AVC afin que le décodeur puisse les «comprendre».

La Figure 7 montre la structure d'un flux vidéo encodé. Les données vidéo de la couche de codage de la vidéo (VCL) se composent de tranches. Les premières tranches d'une séquence vidéo sont toujours les tranches Instantaneous Decoder Actualiser (IDR), formant une trame IDR. Les autres tranches IDR dans le flux démarrent de nouvelles séquences vidéo. Chaque tranche est composée d'un en-tête de tranche (en SH) et des données de tranche. Le SH contient des informations qui aident le processus de décodage de tranche. Les données de tranche sont une séquence de macro-blocs échangés (MBS) et MB indicatif. Chaque Mo contient un en-tête de Mo et des données résiduelles. L'en-tête MB rassemble les informations de prédiction de la prédiction intra et des modules de prédiction Inter, et d'autres informations comme le paramètre de quantification (QP) et le bloc codé Motif (CBP). Les données résiduelles sont encodées en coefficients d'entropie résiduelle post-quantification.

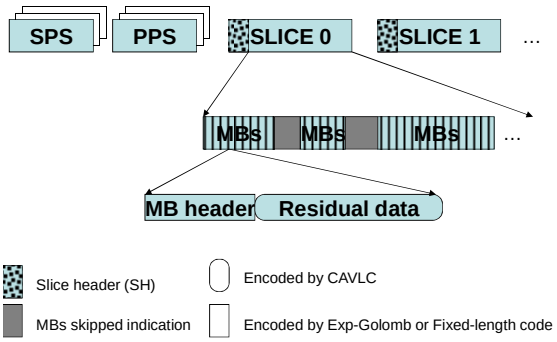

Figure 7 : Structure d'un flux vidéo encodé.

Étant encapsulé au niveau du Network Abstraction Layer (NAL), les données de sortie vidéo se composent d'unités NAL (NALUs). Chaque NALU VCL contient une tranche. Les VCL non-NALUs sont des Sequence Parameter Set (SPS), Image contient un ensemble de paramètres (PPS), la fin de la séquence (EOSeq), la fin du flux (EOS), etc. Le SPS est un ensemble de paramètres pour décoder une séquence vidéo. Le PPS regroupe les paramètres pour les images, à savoir soit des cadres ou des champs. Il y a au moins un SPS et un SPA au début d'un flux vidéo. EOSeq, EOS, et beaucoup d'autres non-VCL NALUs sont facultatifs.

Chaque NALU commence par un octet spécial suivi par des octets Raw Octet Séquence charges utiles (TBA).

Dans le profil principal, le codage Golomb exponentiel (Exp-Golomb) peut être utilisé avec l'Adaptive Variable Length Coding à base de contexte (CAVLC), ou basé sur le contexte arithmétique binaire adaptatif de codage (CABAC). Compte tenu des objectifs de la plateforme VENGME, nous avons seulement implantés les codages Exp-Golomb et CAVLC dans notre profil principal de codage entropique. Comme montré dans la Figure 7, les données résiduelles sont codées en utilisant CAVLC et les autres données sont codées en utilisant Exp-Golomb ou un code de longueur fixe (FLC).

Dans la littérature, il y avait plusieurs techniques pour augmenter la vitesse du codeur H.264 et du packer des données entropiques. Habituellement, l'architecture de pipeline est utilisée, soit pour le codeur entropique ou seulement pour le codeur CAVLC. Par exemple, Chen et al. ont proposé un pipeline à 5 étages pour l'ensemble du codeur d'entropie, voir la Figure 8.

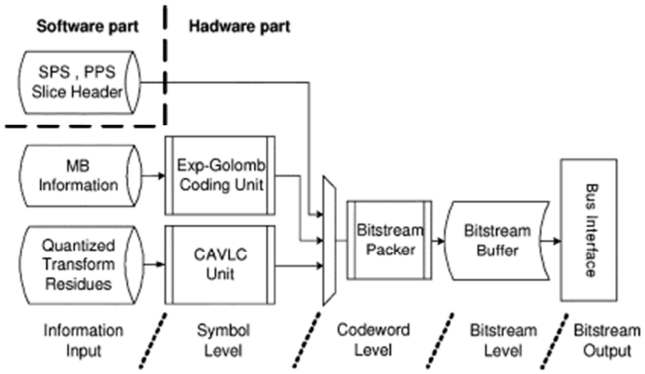

Figure 8: Exemple d'un codeur entropique en pipeline.

Les symboles de codage en parallèle peuvent rendre l'encodage plus rapide mais ils doublent le coût de la région symbole de l'encodeur. En outre, un circuit supplémentaire est nécessaire en raison de la dépendance de données. En repérant les coefficients nuls, on peut également réduire le temps de traitement.

Pour réduire les coûts de la zone, on peut optimiser la table de codage pour réduire sa taille. Une autre méthode consiste à remplacer les tables de codage par des circuits de calcul dédiés.

Pour réduire la consommation d'énergie, Tsai a proposé d'effectuer un pré-traitement des données avant d'entrer dans le codeur CAVLC. Ainsi, le codeur doit accéder à la mémoire que pour les éléments de données nécessaires. On peut également désactiver l'horloge pour les sous-modules ne fonctionnant pas. Traiter l'encodage SPS, PPS et SH dans le module matériel permet de réduire la charge de travail et la consommation d'énergie pour l'ensemble de l'encodeur H.264.

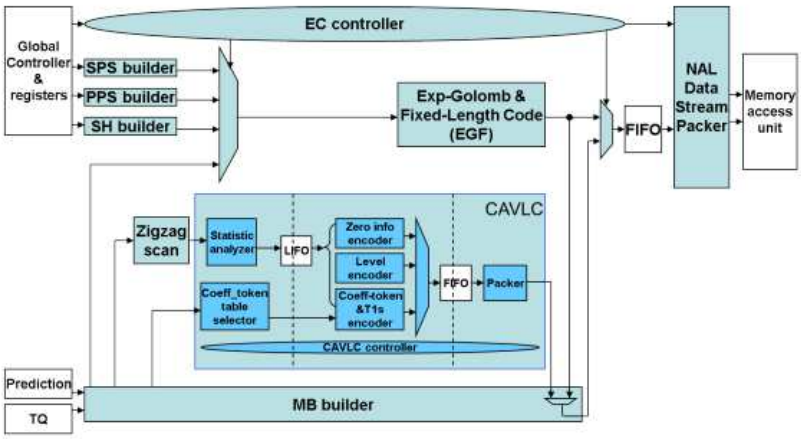

Figure 9 : Architecture du module EC-NAL proposé.

La Figure 9 montre un aperçu de l'architecture du module EC-NAL que nous avons proposé. Les éléments syntaxiques codés sont transférés du sous-module EC au sous-module NAL via la NAL FIFO.

Le moteur de codage entropique consiste en deux codeurs, à savoir, l'Exp-Golomb et le moteur de code de longueur fixe (FEM) et le moteur CAVLC. Les sous-blocs de construction de SPS, PPS, SH, et MB sont mis en œuvre pour collecter les données devant être codées et les envoyer au module EGF dans un ordre spécifié.

SPS, PPS et l'information SH sont fournis par un contrôleur global par l'intermédiaire de certains registres du système. Les informations d'en-tête du MB sont reçues de la prédiction Intra, la prédiction Inter et des moteurs de TQ. En outre, le constructeur MB envoie également les coefficients résiduels de TQ et les informations relatives au scan en zigzag pour les encoder par le moteur CAVLC. Le moteur de compression des données NAL concatène les éléments de syntaxe codés dans le NALUs. Le rôle du contrôleur CE est de synchroniser les constructeurs et le moteur du NAL « d'emballage » des données pour générer un flux vidéo en NALUs.

Cette conception de module EC-NAL et la mise en œuvre proposée permettent d'obtenir un module complet matériel tandis que certaines solutions de l'état de l'art mettent en œuvre une partie logicielle. La mise en œuvre complète du module en matériel réduit la tâche et la consommation électrique de l'ensemble de la plateforme.

Pour augmenter le débit, nous avons proposé d'utiliser des « pré-drapeaux » des données d'entrée non seulement pour les coefficients, mais aussi pour des blocs de coefficients 4x4. De plus, le codage parallèle et la concaténation des données sont utilisés. Lorsque les niveaux sont encodés et « emballés », l'information total_zero et la séquence des run_befores sont également codées et concaténées en un seul code avant d'entrer dans la phase d'emballage, voir la Figure 10.

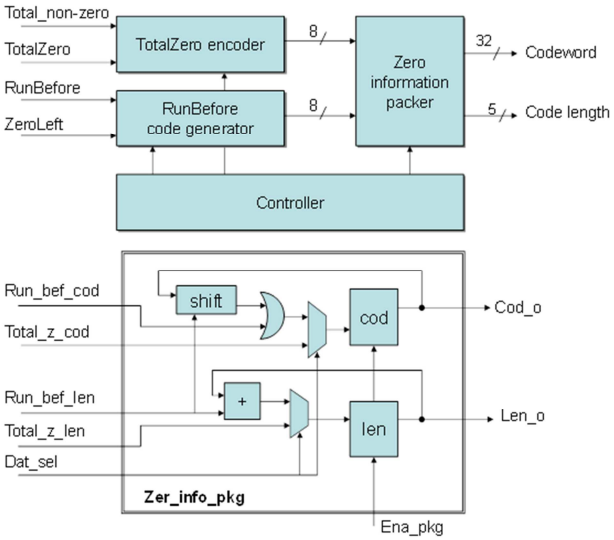

Figure 10: Encodage et concaténation des informations de zéro.

Pour réduire les coûts de la zone, dans le codage de coeff_token, des tables de codage sont ré-encodées pour stocker des mots de code plus courts. Dans le codage de niveau, un circuit de calcul est utilisé en remplacement des tables de codage.

Notre architecture EC-NAL a été modélisée en VHDL au niveau RTL. A des fins de vérification, le module a été intégré dans la plateforme VENGME de codage vidéo H.264. En utilisant le système de codage vidéo avec le module EC-NAL proposé, nous codons des séquences vidéo de test tout d'abord dans le format CIF ciblé. Les vidéos codées sont reçues à partir de la sortie du sous-module NAL. Les vidéos codées sont ensuite décodées avec succès par le décodeur JM, et comparées à la vidéo initiale.

La simulation se fait en utilisant ModelSim de Mentor Graphics et le design est synthétisé en utilisant Design Compiler de Synopsys. Cette synthèse est réalisée au Vietnam lorsque tous les modules sont intégrés à la plateforme. Lors de ce travail, la bibliothèque disponible est la technologie CMOS 180nm d'AMS. La fréquence de test est de 100MHz, voir la Table 2.

Table 2:Résultats du module EC-NAL de la plateforme VENGME

	T.-C. Chen	**W. Lu**	**C-Y. Tsai**	**F-M. Huang**	**Proposé**
Fréquence (*MHz*)	100	200	27	100	100
Latence (*cycle/MB*)	200-500	1905 dans le pire des cas	-	260	87-691
Ressources (*Kgate*)	23.6	9.504	26.598	15.86	73.5
Consommation (*mW*)	-	-	3.7	2.5	1.56

Le module EC-NAL proposé dans cette thèse nécessite 87 à 691 cycles d'horloge pour traiter un MB. Cette vitesse est non seulement appropriée pour la résolution de format CIF mais aussi HD. Le coût matériel est relativement haut mais le module matériel développé est complet (y compris les tables de codage), ce qui, au global, réduit la consommation de la plateforme complète. La consommation de ce bloc est prometteuse, avec des résultats inférieurs à ceux des plateformes de l'état de l'art qui visent la basse consommation.

Pour ce qui est de la plateforme VENGME, l'estimation de la consommation au niveau RTL est effectuée pour obtenir rapidement une cartographie de la consommation pour les différents modules, sur la base d'une activité réaliste du circuit. La Table 3 résume les résultats obtenus.

Table 3 : Estimation de la consommation au niveau RTL pour la plateforme VENGME

Intra prediction	0.32 mw
Inter prediction	7.86 mw
Transformation-Quantization	0.82 mw
De-blocking filter	0.71 mw
EC-NAL	0.9 mw
SW Direct Memory Access	6.39 mw
VENGME	19.1 mw

Notons que le module EC-NAL fonctionne à la fois en intra et en Inter prédiction. Aussi, sa consommation ne peut pas être négligée.

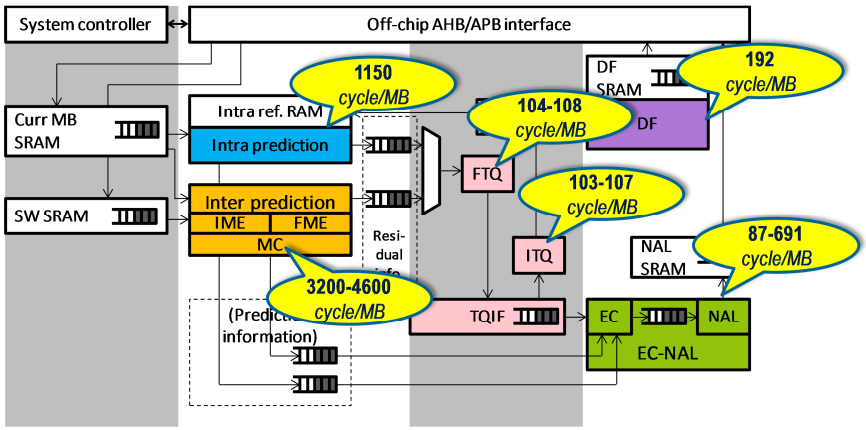

Figure 11 : Déséquilibre de la charge de travail pour la plateforme VENGME.

En outre, dans le fonctionnement de la plateforme VENGME, la charge de travail entre les différents modules est fortement déséquilibrée, voir la Figure 11. Ce déséquilibre laisse de la place pour plus d'optimisation de la puissance consommée. En effet, les modules rapides peuvent travailler plus lentement pour attendre les modules lents, et vice-versa. Cette réduction de la vitesse (en pratique de la fréquence d'horloge) a une influence directe sur la baisse de la consommation. Pour identifier la charge de travail actuelle d'un module, on peut regarder ses flux de données d'entrée. Il faut noter que l'architecture VENGME est de type flot de données (data flow). Elle utilise des tampons pour transférer les données entre les modules. Ainsi, l'état du tampon peut indiquer la charge de travail actuelle d'un module, ou des modules de chaque côté du tampon (producteur en entrée du tampon, consommateur en sortie du tampon). En exploitant ce déséquilibre, on peut réduire la consommation de la plateforme VENGME.

Chapitre 3: Gestion de la consommation basée sur le contrôle du niveau d'occupation du tampon/FIFO entre deux (sous-)modules. Application au module EC-NAL

La plateforme VENGME possède une architecture de type flot de données avec des modules qui communiquent via des tampons. Par conséquent, une approche de type DVFS (Dynamic Voltage and Frequency Scaling, i.e. contrôle dynamique de la tension d('alimentation et de la fréquence d'horloge d'un domaine de consommation pour un circuit sur puce) peut être appliquée à la plateforme VENGME. Le choix des points de fonctionnement tension-fréquence peut être fait en contrôlant le niveau d'occupation des tampons. Chaque module de la plateforme VENGME peut être vu comme un domaine tension-fréquence indépendant : en pilotant la fréquence d'horloge appliquée à ce domaine (et la tension d'alimentation), on peut venir modifier la consommation du domaine, sous contrainte que la charge de travail attribuée à ce domaine soit satisfaite. Rappelons que la plateforme VENGME cible les applications mobiles. Ainsi, en appliquant une approche de type DVFS, on peut atteindre des gains non seulement en puissance, mais aussi en énergie.

Toutefois, compte tenu de contraintes de mise en œuvre, actuellement, seule la fréquence est modifiée, en fonction de niveau d'occupation des tampons, ce qui permet de gérer le déséquilibre de charge de travail entre les modules. En effet, si le tampon si vide plus vite qu'il ne se remplit, la fréquence du module situé après le tampon, appelé consommateur, peut théoriquement (cela dépend aussi des modules suivants) être diminuée, ce qui conduit à une baisse de la consommation pour ce module consommateur. Inversement, si le tampon se remplit plus vite, la fréquence du module consommateur peut être augmentée. La modification de la tension d'alimentation, en concordance avec la modification de la fréquence, pourra être mise en œuvre dans une future version de la plateforme.

La technique DFS (Dynamic Frequency Scaling) est tout d'abord appliquée à la FIFO (qui est en fait un tampon) située entre les sous modules EC et NAL dans le module EC-NAL pour trois raisons. Tout d'abord, le module contient une FIFO pour transférer les données entre les sous-modules. Deuxièmement, ce module fonctionne pour chaque image vidéo ; il semble donc intéressant de diminuer sa consommation dans le budget global. Enfin, le module EC-NAL est l'une des contributions de cette thèse. Ainsi, la compréhension profonde du module permet une modification aisée et un traçage des modifications beaucoup plus facile.

L’application de la méthode de contrôle du niveau de remplissage de la FIFO située entre les sous-modules EC et NAL permet, par action sur la fréquence du sous-module NAL, de se placer dans une approche de type DFS. Ce choix conduit à diviser la plateforme VENGME en deux parties. La première contient les éléments NAL et NAL SRAM tandis que l’autre partie contient le reste de la plateforme.

Ce chapitre propose une méthode de mise à l'échelle dynamique de la fréquence en exploitant le niveau de remplissage de la FIFO. La méthode proposée est en fait générique et s’applique à des systèmes intégrant un tampon mémoire (par exemple FIFO) utilisé pour transférer les données entre des modules effectuant un traitement de ces données. Le choix réalisé ici est de mettre à l'échelle la fréquence du (sous-)module qui reçoit les données ((sous-)module consommateur) pour maintenir le niveau d'occupation de la FIFO à une valeur de référence donnée.

Pour chaque lien FIFO, un (sous-)module «producteur» écrit des données dans la FIFO, jusqu’à ce que le lien mémoire soit plein, auquel cas il doit attendre que la FIFO se vide un peu pour insérer de nouvelles données. Le (sous-)module «consommateur» lit les données dans le même ordre que celui dans lequel elles ont été écrites jusqu'à ce que la FIFO soit vide. Lorsque la FIFO est vide, le consommateur doit attendre pour lire un nouvel élément. Les blocs « producteur et consommateur » peuvent fonctionner à la même fréquence ou à des fréquences différentes, cette dernière situation permettant de contrebalancer des différences de charge de travail entre les deux (sous-)modules. Pour implanter une approche de contrôle dynamique des tensions et fréquences (technique de DVFS), les blocs « producteur et consommateur » fonctionnent avec des tensions d’alimentation différentes (V_P et V_C) et des fréquences d'horloge f différentes (f_P et f_C). Par conséquent, la FIFO intégrée entre ces deux blocs doit être « asynchrone », avec des signaux d'horloge de lecture et d’écriture différents. La Figure 12 illustre ce mode de communication.

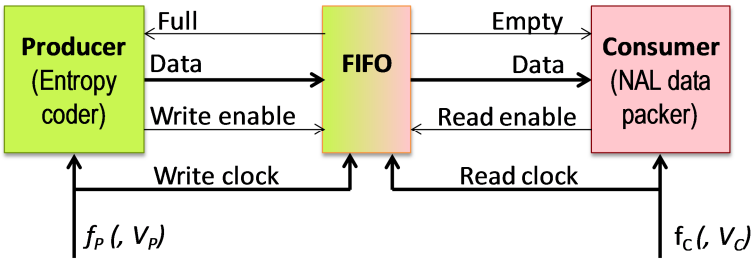

Figure 12: Lien FIFO entre deux domaines tension-fréquence

Dans la littérature, certains auteurs ont essayé de contrôler la tension d'alimentation Vdd et/ou la fréquence d'horloge f en surveillant l’état de la FIFO (ou tampon) placée entre

deux blocs. Dans ces travaux, l'information sur la FIFO utilisée pour le contrôle peut être le taux d'occupation de la FIFO (ou tampon), ou la vitesse de remplissage de la FIFO ou le statut que le producteur/consommateur doit attendre pour l'utilisation de la FIFO , ou toute association de ces inducateurs.

L. Thiele et al. [Thie05D] ont utilisé le niveau de remplissage du tampon, combiné au taux de flux de données dynamique, ainsi qu'une information fondée sur l'analyse hors ligne de la charge de travail afin d'adapter la vitesse de l'horloge. Dans le travail de Thiele et al. [Thie05D], le consommateur est un processeur. Les auteurs considèrent le traitement des flux multimédia avec une taille de la mémoire tampon restreinte. Le taux de consommation du consommateur (lié à la fréquence de l'horloge f_C) est fixé de manière à éviter un débordement du tampon dans le pire cas d'utilisation.

P. Choudhary et al. [Chou09P] exploitent le temps que le producteur et/ou le consommateur doivent attendre en raison d'une FIFO pleine ou vide, dans un intervalle de temps donné $T_{échantillon}$. Ils décident à partir de cette information s'il est nécessaire d'adapter les fréquences des deux domaines. Choudhary et al. [Chou09P] définissent l'état de « décrochage » comme l'état où le producteur ET/ou le consommateur doivent attendre pour utiliser le tampon. La Figure 13 montre un exemple de l'état de « décrochage » pour le producteur. Cet état se produit lorsque la FIFO est pleine et que des données sont disponibles à la sortie du producteur.

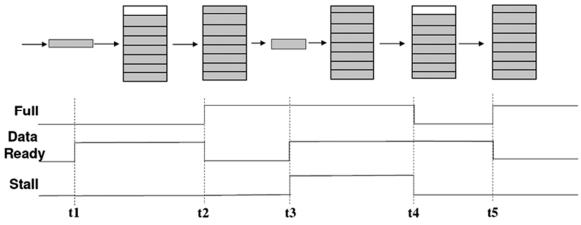

Figure 13 : État de « décrochage » du producteur.

Une architecture matérielle est mise en œuvre pour surveiller l'état des « producteur et consommateur », voir la Figure 14.

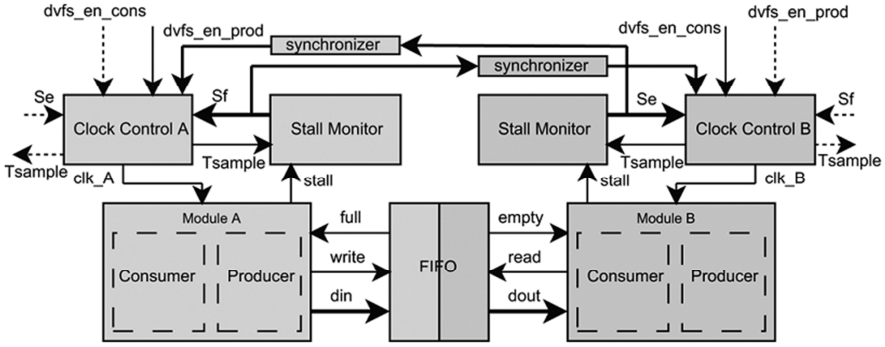

Figure 14: L'architecture matérielle pour surveiller l'état de décrochage.

En exploitant le nombre de cycles d'horloge où le producteur et le consommateur sont en « décrochage », Sf et Se, au cours de $T_{échantillon}$, le facteur S est calculé

$S = 1 - | Se – Sf | / T_{échantillon}$.

Ce facteur S est utilisé pour la mise à l'échelle soit de la fréquence du producteur ou du consommateur, selon la contrainte du système. Dans la mise en œuvre matérielle, il est montré que la méthode de P. Choudhary et al. a un réel potentiel de réduction de la consommation. Cependant, la méthode ne peut pas éliminer l'état de décrochage. Par ailleurs, la sélection de la valeur de $T_{échantillon}$ est un problème qui n'est pas discuté par les auteurs.

Y-H. Lu et al. [Lu02D] s'intéressent à des tampons insérés dans un système multimédia pour lesquels un taux de données de sortie constant est nécessaire. Un algorithme est mis en œuvre pour attribuer au processeur (producteur) la fréquence permettant de garantir l'état optimal de la mémoire tampon.

Les méthodes mises au point dans [Lu03R] et [Bae05L] sont appliquées à un processeur fonctionnant en tant que décodeur multimédia et communiquant avec un dispositif d'affichage par l'intermédiaire d'un tampon mémoire. Sur la base de l'état du tampon, les « fréquence et tension » du processeur (producteur) sont mises à l'échelle. Le contrôleur est conçu en utilisant un modèle à temps continu. L'information utilisée dans ce travail est à la fois le niveau d'occupation du tampon mémoire et sa vitesse de remplissage.

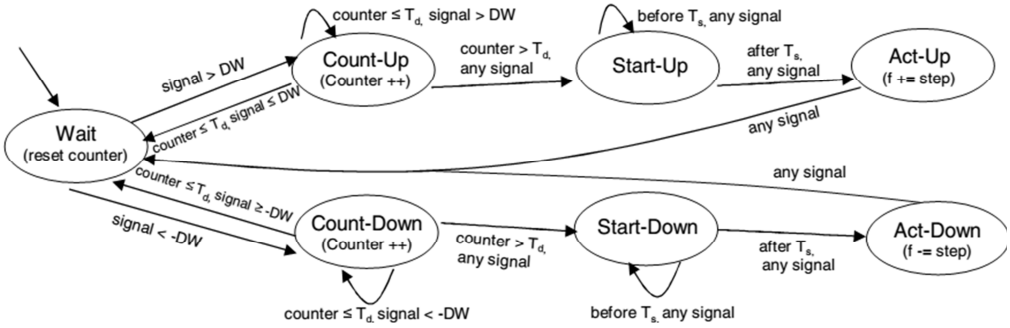

Figure 15: FSM pour augmenter/diminuer la fréquence.

Dans [Wu05V], le système est modélisé sous la forme de domaines d'horloge multiples (MCD). La méthode DVFS contrôle la fréquence (et la tension) de chaque domaine d'horloge selon l'état de la file d'attente d'entrée. Des machines à états finis (FSM) sont mises en œuvre pour augmenter/diminuer la fréquence et la tension. Pour chaque signal de file d'attente, un FSM, tel que présenté sur la Figure 15, est mis en œuvre. Les changements de tension/fréquence sont déterminés en fonction du signal de file d'attente, par rapport à une fenêtre de déviation après un retard prédéterminé. Le signal de file d'attente est soit la différence entre le taux d'occupation de la file d'attente et une valeur de référence ou la

différence entre le taux d'occupation de la file d'attente à deux périodes d'échantillonnage consécutives (c'est-à-dire la vitesse de remplissage).

Les deux signaux de la file d'attente nécessitent deux fois plus de surface Silicium pour une mise en œuvre de la FSM. Ensuite, l'état Schedule est ajouté pour décider de l'opération finale. Le modèle utilisé est à temps continu. Le procédé DVFS est conçu pour être adapté à une charge de travail variable. Toutefois, la mise en œuvre de chaque FSM rend cette solution complexe à implanter.

Pour résumer, l'approche DFS (resp. DVFS) des travaux de la littérature exploite l'état de la FIFO (ou tampon mémoire) pour analyser la charge de travail du module où la méthode DFS (resp. DVFS) est appliquée. Le niveau d'occupation de la FIFO (ou tampon) est la plus simple information pour surveiller le statut de la FIFO parce que cette information peut être obtenue directement à partir de l'écriture et de la lecture des indices à l'intérieur de la FIFO. Les approches DFS qui exploitent le statut d'une FIFO, et que nous avons présentées, ont été conçues pour des applications spécifiques et mises en œuvre en logiciel [Lu02D, Lu03R, Bae05L]. Des modèles à temps continu qui nous semblent moins pertinents qu'un modèle à temps discret dans la modélisation des systèmes numériques, sont utilisés dans certains travaux [Bae05L, Wu05V].

Comme mentionné ci-dessus, le contrôle du taux d'occupation de la FIFO a été mis en œuvre dans le module EC de l'encodeur H.264 de la plateforme VENGME, où le codeur agit comme «producteur», envoie des données au NAL qui est le «consommateur», voir la Figure 16.

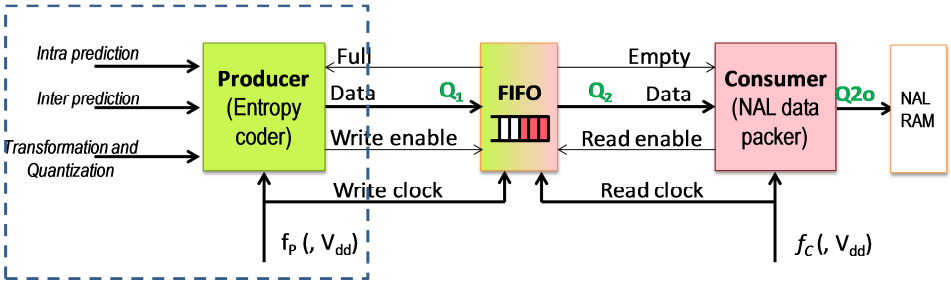

Figure 16 : Cas d'étude de la méthode de contrôle de la FIFO proposée.

Comme le sous-module EC-NAL est le dernier dans le chemin de codage, le producteur reçoit des données à partir des autres modules du codeur H.264 VENGME. Ci-après, f_P (appliqué à tous les blocs dans le rectangle bleu en pointillés dans la Figure 16) est supposée constante, tandis que f_C peut être diminuée/augmentée.

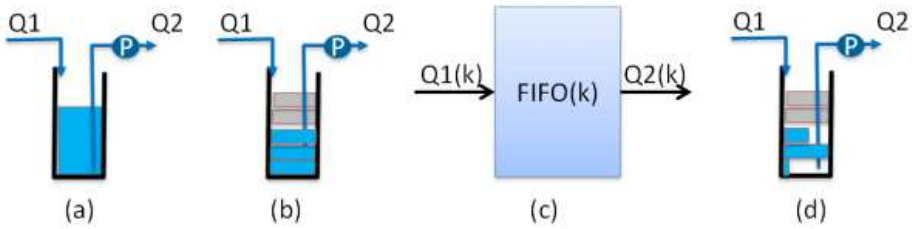

Figure 17 : Modélisation de la FIFO.

La FIFO peut être modélisée comme un réservoir avec une entrée Q1et une sortie Q2. En outre, Q1 et Q2 sont non-négatives. Comme la vitesse de sortie est indépendante du niveau dans le réservoir, la sortie Q2 est extraite à l'aide d'une «pompe» P (voir la Figure 17-a). Chaque opération d'écriture ou de lecture agit uniquement sur un élément de données. Par conséquent, le niveau dans la FIFO est discret, compte tenu du nombre de paquets (Figure 17-b). En conséquence, le modèle dynamique est donné (Figure 17-c) :

$$FIFO(k + 1) = FIFO(k) + Q1(k) - Q2(k)$$

où FIFO(k) est le niveau d'occupation de la FIFO en nombre de paquets au $k^{ième}$ instant d'échantillonnage. Q1(k) est le nombre de paquets de données entrant dans la FIFO ; ces paquets proviennent du producteur. Q2(k) est le nombre de paquets de données sortant de la FIFO pour alimenter le consommateur. En raison de contraintes d'implémentation Silicium (liées à la surface) pour la plateforme matérielle H.264, le nombre maximal de paquets dans la mémoire FIFO est choisi égal à 7. Par conséquent, FIFO(k) ϵ {0, 1, 2, 3, 4, 5 , 6, 7}.

Étant donné que la fonctionnalité du sous-module NAL est de concaténer des données, la vitesse réelle des paquets en lecture depuis la FIFO dépend non seulement de la fréquence d'horloge du consommateur f_C, mais également de la longueur des paquets. Cependant, pour des raisons de simplicité, la sortie de la FIFO Q2 est censée proportionnelle à $f_C(k)$:

$$Q2(k) = bf_C(k)$$

où b est une constante positive. La constante b est identifiée à 20ns, en utilisant des techniques d'identification classiques.

Chaque fois que la FIFO est pleine (resp. vide), et le producteur a des données à écrire dans la FIFO (resp. le consommateur veut lire les données depuis la FIFO), le producteur (resp. consommateur) tombe dans l'état de « décrochage » (il doit attendre). Cet état de décrochage entraîne un gaspillage de la consommation d'énergie. Aussi, l'objectif du contrôle est d'adapter la fréquence du consommateur en gardant la FIFO à moitié pleine (ni vide ni pleine), pendant le fonctionnement normal. Bien sûr, à la fin du processus d'encodage, la FIFO devra être vidée. La conception de la loi de contrôle est maintenant résumée.

Pour la liaison FIFO à l'étude, Q1 est la sortie du producteur et elle n'est pas contrôlée. Par conséquent, dans le modèle du système, Q1 est considéré comme une perturbation. Le modèle de la FIFO, sous forme de fonction de transfert, sans perturbation, dans le domaine z, est la suivante :

$$\frac{FIFO(z)}{Q2(z)} = G(z) = \frac{-1}{z-1}$$

qui est en fait un intégrateur. La Figure 18 montre le système contrôlé avec son contrôleur *C(z)* qui adapte la fréquence du consommateur $f_C(z)$ pour maintenir le niveau de la FIFO *FIFO(z)* près d'une valeur de référence *R(z)*, L'entrée du contrôleur est l'erreur *E(z)* entre le niveau de référence *R(z)* et le niveau de la FIFO *FIFO(z).*

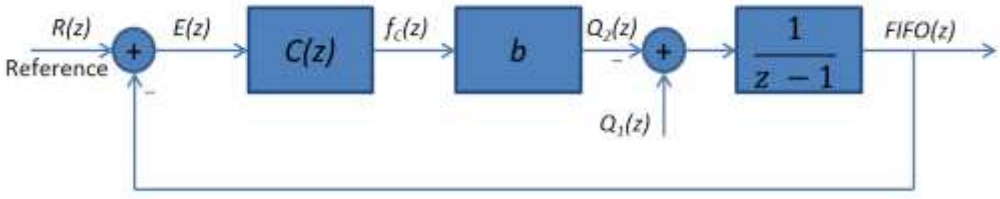

Figure 18: Schéma de contrôle.

Un contrôleur Proportionnel-Intégral (PI) à temps discret est sélectionné afin de rejeter la « perturbation » Q1, d'assurer un système en boucle fermée sans erreur statique et afin de régler le temps de réponse du système en boucle fermée. Le contrôleur est choisi comme :

$$\frac{f_C(z)}{E(z)} = C(z) = K_p + K_i \frac{z}{z-1}$$

Par conséquent, la fonction de transfert en boucle fermée est donnée par:

$$\frac{FIFO(z)}{R(z)} = \frac{-C(z)b\dfrac{1}{z-1}}{1+C(z)b\dfrac{-1}{z-1}}$$

$$\frac{FIFO(z)}{R(z)} = \frac{-[(K_p+K_i)z-K_p]b}{z^2-z[2+(K_p+K_i)b]+1+K_pb}$$

où *R(z)* est la référence pour le niveau de la FIFO.

Les pôles z_1, z_2 déterminent la dynamique du système en boucle fermée. Leurs valeurs numériques dépendent de valeurs de K_p et K_i. Une fois que z_1 et z_2 ont été choisis, K_p et K_i sont calculés avec :

$$K_p = \frac{z_1z_2-1}{b} = \frac{z_1z_2-1}{0{,}02} \qquad K_i = \frac{z_1+z_2-z_1z_2-1}{b} = \frac{z_1+z_2-z_1z_2-1}{0{,}02}$$

où b = 0,02µs.

La plate-forme VENGME n’intègre pas de PLL pour générer les valeurs de la fréquence horloge. Nous allons donc mettre en œuvre un diviseur de fréquence pour générer un ensemble de valeurs possibles pour f_C. Cet ensemble de valeur constituera donc un ensemble de valeurs discrètes. Le signal le plus élevé de fréquence que la plateforme VENGME prend en charge est 100MHz. Une façon simple de mettre en œuvre un diviseur de fréquence est d'utiliser un compteur binaire. De cette manière, la fréquence est divisée par deux. En conséquence, les valeurs possibles pour la fréquence f_C peuvent être 100, 50, 25, 12.5, etc. MHz. Ainsi, une quantification non uniforme est sélectionnée pour effectuer le contrôle de la fréquence (DFS). Elle ajoute une non-linéarité dans le système.

Malgré cette non-linéarité, on peut montrer que le système en boucle fermée est stable, en utilisant le critère de Nyquist. En pratique la méthode du 1^{er} harmonique (pour la partie non-linéaire du système) est utilisée pour montrer la stabilité. En utilisant la condition de séparation topologique, une autre méthode qui permet de prouver la stabilité de systèmes non-linéaires, le système contrôlé est également prouvé être stable.

Chapitre 4: Mise en œuvre et la validation de la méthode DFS fondée sur le niveau de FIFO

Afin de mettre en œuvre la méthode DFS, sur la base du contrôle du niveau de la FIFO pour le lien FIFO intégré dans le module EC-NAL, le module EC-NAL original doit être scindé en deux domaines de fréquence, à savoir celui contenant le sous-module EC et celui contenant le sous-module NAL. Le domaine contenant le sous-module EC fonctionne à la fréquence de 50MHz. L'autre domaine fonctionne à une fréquence qui est sélectionnée par le procédé DFS, plus précisément par le régulateur PI. La Figure 19 illustre ces modifications. En comparaison avec le module EC-NAL d'origine, en plus de la modification de la FIFO, certains blocs (en bleu) sont ajoutés.

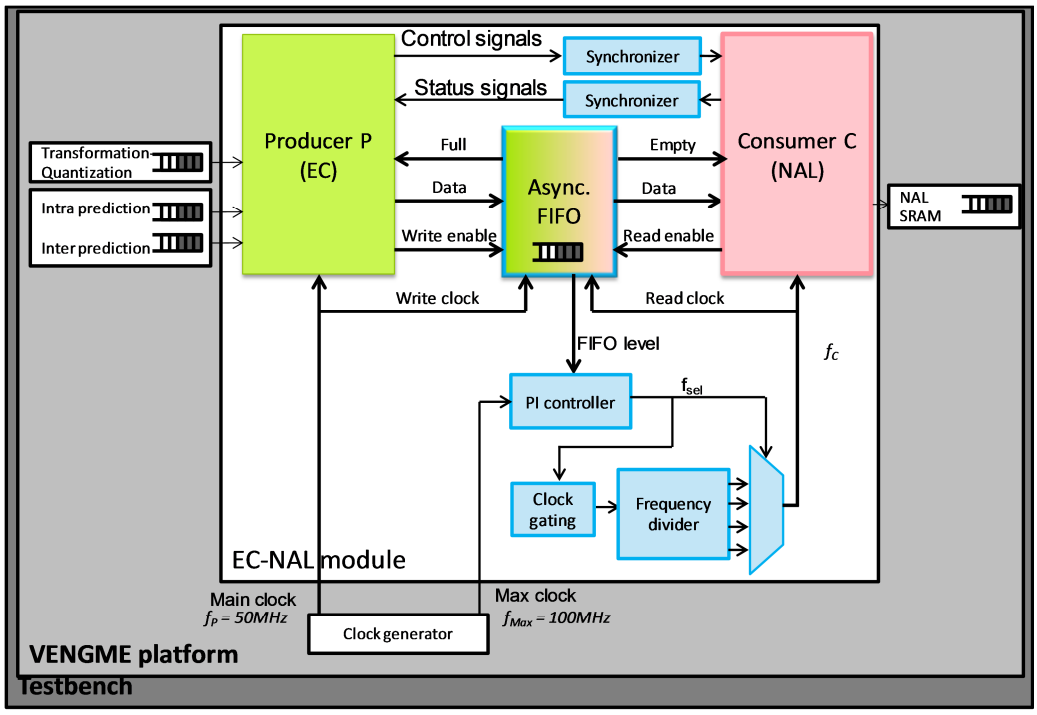

Figure 19: Modification de la plateforme VENGME pour intégrer le contrôleur de niveau de la FIFO.

Tout d'abord, il faut mettre en œuvre l'interface entre les deux domaines d'horloge, en d'autres termes, à l'interface des deux domaines d'horloge, il faut assurer la synchronisation des signaux pour éviter la métastabilité. Le bloc « Synchronizer » est utilisé pour les signaux sur un seul bit et une FIFO asynchrone est utilisée pour les données à transférer multi-bits.

Ensuite, pour effectuer le contrôle de la fréquence, des fréquences différentes doivent être générées. Pour cela, nous mettons en œuvre des diviseurs de fréquence pour obtenir des signaux d'horloge de fréquences différentes.

Le diviseur de fréquence dans le module EC-NAL génère quatre signaux d'horloge de fréquences 12,5, 25, 50 et 100 MHz. Un multiplexeur de signal d'horloge est mis en œuvre pour sélectionner la fréquence de fonctionnement pour le module NAL.

Enfin, le régulateur PI conçu est modélisé en VHDL et ensuite intégré dans la plateforme. Pour limiter le coût en surface silicium, des contraintes complémentaires influencent sur le choix des pôles et de fait la valeur des paramètres du correcteur PI. Ces contraintes et les détails de mise en œuvre sont maintenant expliqués.

De la fonction de transfert du dispositif de commande (dans le domaine z), l'équation de récurrence est obtenue :

$f_C(k) = f_C(k\text{-}1) + (K_p + K_i) \times E(k) - K_p \times E(k\text{-}1)$

Un circuit de calcul peut être construit à partir de cette équation de récurrence pour calculer la fréquence $f_C(k)$ du consommateur, à partir de $f_C(k\text{-}1)$ et des erreurs E(k) et E(k-1). Ce circuit de calcul contient deux multiplicateurs avec les constantes $(K_p + K_i)$ et K_p.

Pour éviter la mise en œuvre des multiplicateurs avec de trop grands coefficients, le circuit de calcul est modifié comme suit:

$$f_{sel}(k) = f_{sel}(k-1) + \frac{K_p + K_i}{12{,}5} E(k) - \frac{K_p}{12{,}5} E(k-1)$$

où f_{sel} est défini comme la valeur pour sélectionner la fréquence du bloc consommateur correspondant, qui est égale à $f_C/12{,}5$. La Figure 20 illustre le circuit de calcul. La sélection des pôles a un effet direct sur le comportement du système contrôlé. A partir des contraintes matérielles et des contraintes dynamiques, les coefficients du contrôleur sont calculés.

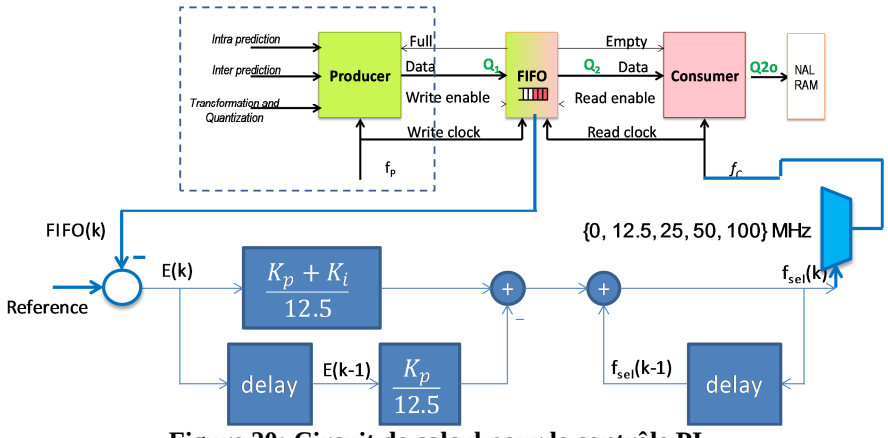

Figure 20: Circuit de calcul pour le contrôle PI.

La mise en œuvre matérielle devient plus aisée si les pôles en boucle fermée sont choisis de telle sorte que les coefficients des multiplicateurs soient des multiples de 1/8. De

cette manière, il faudra tout au plus trois bits pour représenter la partie fractionnaire de l'expression binaire.

Les pôles en boucle fermée doivent également assurer les performances attendues pour le système en boucle fermée (y compris la stabilité). Des pôles avec de petites valeurs absolues vont introduire de fortes contraintes dynamiques sur le générateur de fréquence d'horloge. En effet, on aura une sur-réactivité du système bouclé au moindre changement du niveau dans la FIFO. Par conséquent, les pôles sélectionnés doivent limiter la surcharge (en termes de changements rapides) sur le générateur de fréquence. En outre, des pôles réels sont privilégiés afin d'éviter les comportements pseudo-périodiques pour la sortie du système.

En tenant compte de l'intervalle [0.5, 1[pour les pôles en boucle fermée, 21 paires de pôles différents qui satisfont toutes les contraintes ci-dessus peuvent être sélectionnés. Les 21 contrôleurs correspondants ont été modélisés en VHDL, puis intégrés dans le module EC-NAL pour une simulation au niveau RTL. Les résultats de simulation permettent de vérifier le comportement du contrôleur PI dans le module EC-NAL. Le contrôleur préféré est celui qui maintient le niveau de la FIFO dans un état intermédiaire pendant une longue période de fonctionnement de l'encodeur tout en minimisant le taux de variation de la fréquence f_C.

La Figure 21 présente l'architecture d'un tel contrôleur, dont les pôles sont $z_1 = 0,75$ et $z_2 = 0,5$. Ainsi $K_p = -31,25$ et $K_i = -6,25$ et f_{sel} est donnée par :

$$f_{sel}(k) = f_{sel}(k-1) - \frac{6}{2}E(k) + \frac{5}{2}E(k-1)$$

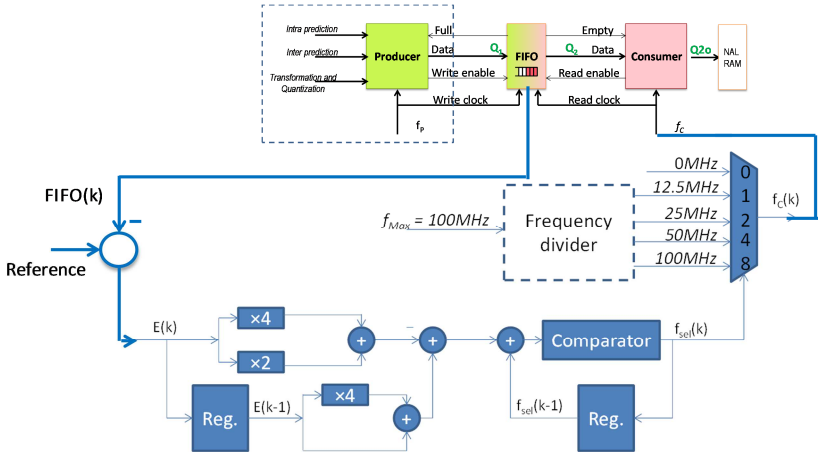

Figure 21: Architecture du contrôleur PI pour des pôles en boucle fermée $z_1 = 0,75$ et $z_2=0,5$.

Le circuit de calcul mis en œuvre ne contient que quatre additionneurs et deux registres. Les multiplications par 2 et 4 sont faites simplement par décalage binaire. Pour effectuer le quantificateur de fréquence non uniforme, un comparateur et le multiplexeur sont ajoutés pour sélectionner la fréquence correspondant à partir de l'ensemble de signaux d'horloge disponibles. Ces signaux sont délivrés par un diviseur de fréquence mis en œuvre en

utilisant un compteur à quatre bits. Notons que pour diminuer de la surface silicium, il est préférable que le circuit de calcul ait une petite partie fractionnaire en expression binaire et quelques additionneurs.

Pour appliquer la méthode DFS, le module EC-NAL a été modifié. Les simulations sont faites au niveau RTL, pour vérifier que les modifications ne changent pas la fonctionnalité du module EC-NAL et de la plateforme de VENGME, voir la Figure 22. En outre, on constate sur les simulations que la méthode DFS mise en œuvre modifie bien la fréquence du module NAL.

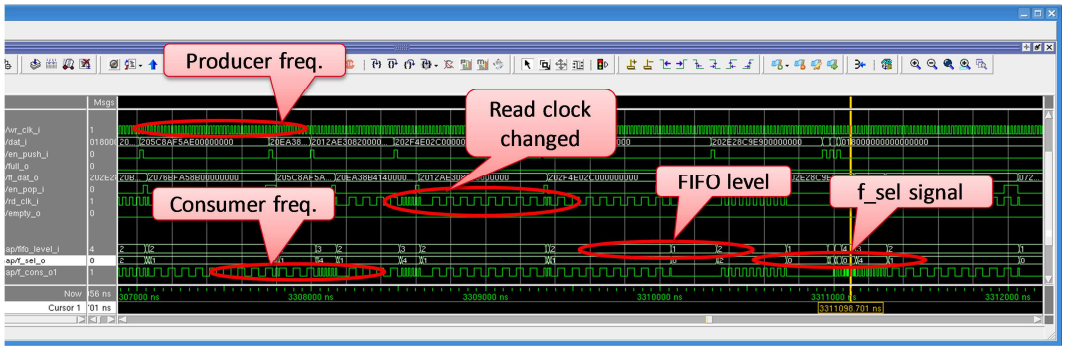

Figure 22: Résultat de la simulation.

La Figure 23 illustre les étapes expérimentales pour finalement estimer le gain en puissance lors de l'application de la méthode de contrôle proposée. Après la modélisation du système, le dispositif de commande est conçu et modélisé dans MATLAB. Le procédé de commande, à savoir le régulateur PI, le diviseur de fréquence et toutes les modifications sur le module EC-NAL, est modélisé en VHDL. Le contrôleur est alors intégré dans le module EC-NAL de la plateforme VENGME.

Le module est synthétisé avec la technologie 28nm FDSOI de STM en utilisant l'outil de synthèse RC de Cadence. La simulation post-synthèse en utilisant la netlist du module EC-NAL dans la plateforme VENGME montre les opérations de contrôle au niveau de la porte. La consommation d'énergie du module EC-NAL est estimée en utilisant PrimeTime de Synopsys.

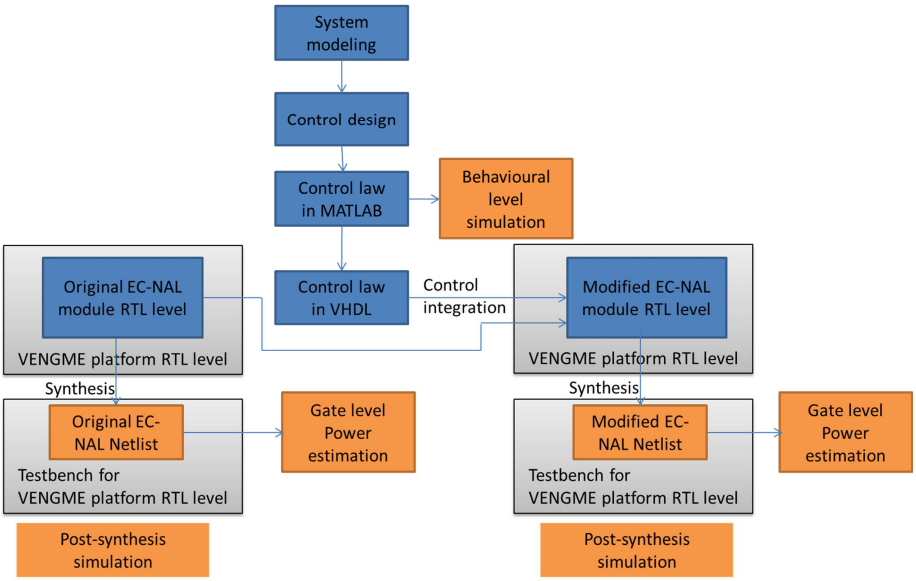

Figure 23: Étapes de modèle, mis en oeuvre et validation.

A titre de comparaison, le module EC-NAL original est également synthétisé, simulé au niveau de grille et sa consommation d'énergie est estimée. Les résultats électriques sont présentés dans la Table 4.

Table 4: Comparaison des deux versions du module EC-NAL (sans et avec contrôle de la FIFO)

mW	Blocs ajoutés (PI, fréquence diviseur)	FIFO	NAL	Totale (sauf EC)
EC-NAL original	-	$13{,}38\times10^{-5}$	$7{,}876\times10^{-5}$	21.26×10^{-5}
EC-NAL modifié	$0{,}964\times10^{-5}$	15.94×10^{-5}	3.208×10^{-5}	20.11×10^{-5}
		↑19%	↓59.3%	↓5.3%

L'estimation de la puissance permet d'évaluer l'influence du bloc de commande sur le module EC-NAL. Grâce à ces simulations à bas niveau, la méthode proposée montre sa capacité à réduire la consommation. Elle nécessite un faible surcoût en surface silicium et le coût énergétique pour mettre en œuvre le régulateur PI est acceptable. En effet, on réduit par environ 59% la consommation totale du côté consommateur.

Cependant, la méthode proposée ne réduit que de 5,3% la consommation du circuit complet. La raison en est que le consommateur choisi, qui est le sous-module NAL initial, a une consommation faible, même lorsqu'il est comparé à la consommation de la liaison FIFO. Si la méthode est appliquée à un lien FIFO, avec un consommateur consommant plus que le sous-module NAL, avec la même FIFO asynchrone, le même régulateur PI et le même diviseur de fréquence, l'efficacité de l'approche DFs proposée peut être prouvée. La normalisation suivante illustre cet argument.

Selon les résultats présentés ci-dessus, considérons que:

- la consommation du sous-module d'origine est celle du NAL;

- la consommation de la FIFO synchrone est 1,7 fois plus grande que le sous-module initial NAL.

Alors :

- la consommation du régulateur PI et du diviseur de fréquence est égale à 0,12 fois la consommation du sous-module initial NAL;
- la consommation de la FIFO asynchrone est 19% plus élevée que celle de la FIFO synchrone;
- notre méthode DFS réduit par 59,3% la consommation totale d'énergie du consommateur, à savoir le sous-module NAL.

Ainsi, la consommation de puissance de la version originale est égale à 1 + 1,7 = 2,7. La consommation d'énergie de la version modifiée est égale à :

1 x (1 - 59,3%) + 1,7 x 119 = 0,12% + 2,567.

Le gain de puissance est donc: (2,7 à 2,567) ÷ 2,7 = 4,9%

Maintenant, supposons que le procédé est appliqué à un consommateur qui consomme 100 fois plus que le sous-module NAL d'origine. On peut calculer le gain en puissance comme suit:

- la consommation de la version originale est égale à 100 + 1,7 = 101,7.
- la consommation de la version modifiée est égale à

100 x (1 - 59,3%) + 1,7 x 119 = 0,12% + 42,86.

Ainsi, le gain en puissance de la méthode est la suivante: (101,7 à 42,86) ÷ 101,7 = 57,9%.

mW	Blocs ajoutés (PI, fréquence diviseur)	FIFO	NAL	Totale (sauf EC)
EC-NAL original	-	1,7	1	2.7
EC-NAL modifié	0,12	1,7×119%	1-59,3%	2.567
			↓59,3%	↓4,9%
↓				
mW	Blocs ajoutés (PI, fréquence diviseur)	FIFO	Consommateur	Totale
Version originale	-	1,7	100	101,7
Version modifiée	0,12	1,7×119%	40,7	42,86
			↓59,3%	↓57,9%

Avec un sous-module qui est 100 fois consommant que le sous-module NAL, le gain en puissance est de 57,9%. L'efficacité de la méthode proposée est donc prouvée.

Conclusions et perspectives

Ce travail de thèse s’est intéressé à l'optimisation des performances de la plateforme VENGME, un encodeur vidéo matériel H.264. En pratique, le module EC-NAL de la plateforme a été conçu, optimisé et mis en oeuvre. Le module EC-NAL a été optimisé afin d’augmenter les performances en vitesse tout en réduisant sa taille et sa consommation électrique.

Ensuite, la consommation des différents modules de la plateforme VENGME, sur la base d’estimation de la puissance au niveau RTL a été étudiée. Ces résultats aident à comprendre les caractéristiques en puissance de la plateforme. Avec les résultats de puissance, le déséquilibre de la charge de calcul entre les modules et l'architecture en flot de données de la plateforme VENGME ont été exploitées pour trouver une solution d'optimisation de la puissance pour l'ensemble de la plateforme.

Une méthode DFS de gestion de la consommation via le contrôle du niveau de remplissage d’une FIFO, basée sur la théorie du contrôle, a été proposée. Cette méthode vise en premier lieu la plateforme VENGME, mais aussi les plateformes qui utilisent des tampons mémoire / FIFOs pour transférer des données entre des modules. La méthode proposée est mise en œuvre tout d'abord dans le module EC-NAL. Un dispositif de commande est conçu pour commander la fréquence du sous-module NAL en fonction du niveau de remplissage de la FIFO placée entre les sous-modules CE et NAL. Le contrôleur a été mis en œuvre en matériel. Les résultats d'estimation de la puissance consommée au niveau porte valident l’approche proposée.

La poursuite naturelle de ce travail est l'extension de l’approche de contrôle du niveau de remplissage de la FIFO à l'ensemble de l'architecture de la plateforme VENGME. Cela consiste à choisir les modules « consommateur » et l'algorithme d'arbitrage nécessaire quand il y a beaucoup de FIFOs contrôlées dans la plateforme.

Une autre perspective est l'intégration de la mise à l’échelle de la tension d’alimentation pour étendre l’approche DVS en une technique DVFS. La loi de commande devra être conçue en tenant compte de contraintes imposées par le matériel, par exemple le retard dû à la commutation d'une alimentation.

Enfin, l'évaluation sur la plateforme silicium doit être effectuée. En particulier, la consommation des blocs de la plateforme réelle devra être comparée aux simulations et aux valeurs de l’état de l’art.