LOW DELAY VIDEO CODING

THÈSE N^{O} 3453 (2006)

PRÉSENTÉE À LA FACULTÉ SCIENCES ET TECHNIQUES DE L'INGÉNIEUR

Institut de traitement des signaux

SECTION DE GÉNIE ÉLECTRIQUE ET ÉLECTRONIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Graziano VAROTTO

Laurea in ingegneria elettronica, Università di Pisa, Italie
et de nationalité italienne

acceptée sur proposition du jury:

Prof. D. Mlynek, directeur de thèse
Dr B. Hochet, rapporteur
Prof. G. Iannaccone, rapporteur
Prof. M. Kunt, rapporteur

Lausanne, EPFL
2006

> Aerodynamically, the bumble bee should not be able to fly, but the bumble bee does not know it so it goes on flying anyway.

Mary Kay Ash

Acknowledgments

I would like to acknowledge that persons that either directly either indirectly have supported me in these years of work. First of all I would like to thank Prof. Daniel Mlynek who gives me the opportunity to join the LTS3 laboratory, where I have spent these years learning the importance to work in a group and to be strong motivated to obtain the best results. I also thank Dr. Marco Mattavelli for his guidance and encouragement during this long period. I am grate to Professors Murat Kunt, Bertrand Hochet and Giuseppe Iannaccone who accepted to be members of the jury, for their suggestions and useful comments to improve the quality of this work.

I would like to thank my colleagues Mauro Lattuada and Jean-Françoise Csomo for the close collaboration and for their help and suggestion during the difficult development of this work. A great thank to Renzo Posega for his friendship during these years of sharing a considerable part of problems and solutions, and not least his help in reading and reviewing this work.

A double thank to Massimo Ravasi for his frequent suggestions on C++ and further programming issues and to his wife Ester, who has accepted to correct my written English. A special thank to the old and present LTS3 collegues, friends and all Livetools staff for the professional experience and pleasant moments we have shared.

Finally, I would like to express my love and gratitude to Saviana for her everlasting support and encouragement.

Lausanne 19^{th} Jannuary 2006.

Contents

List of Figures

List of Tables

Glossary

4:2:0 (sampling)	Cr, Cb components have half the horizontal and vertical resolution of Y component
4:2:2 (sampling)	Cr, Cb components have half the horizontal resolution of Y component
4:4:4 (sampling)	Cr, Cb components have same resolution as Y component
ADC	Analog to Digital Converter
arithmetic coding	Coding method to reduce redundancy
artefact	Visual distortion in an image
ASIC	Application-Specific Integrated Circuit
ASO	Arbitrary Slice Order
B-picture (slice)	Coded picture (slice) predicted using bidirectional motion compensation
BAB	Binary Alpha Block, Indicates the boundary of a region (MPEG-4 Visual)
Block	Region of macroblock (8 x 8 or 4 x 4 samples)
Block matching	Motion estimation carried out on rectangular picture areas
Blocking	Square or rectangular distortion areas in an image
CABAC	Context-based Adaptive Binary Arithmetic Coding
CAE	Context-based Arithmetic Encoding
CAVLC	Context Adaptive Variable Length Coding
Chrominance	Color difference component
CIF	Common Intermediate Format
CODEC	COder / DECoder pair
COFDM	Coded-OFDM
Color space	Method of representing color images
DAC	Digital to Analog Converter
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DPCM	Differential Pulse Code Modulation
DSCQS	Double Stimulus Continuous Quality Scale
DSP	Digital Signal Processor
DV	Digital Video. International standard for a consumer digital video format introduced by Sony Corp. in 1995
DVB	Digital Video Broadcasting
DVB-C	DVB-Cable

DVB-S	DVB-Satellite
DVB-T	DVB-Terrestrial
DWT	Discrete Wavelet Transform
Entropy coding	Coding method to reduce redundancy
FFT	Fast Fourier Transform
Field	Odd or even numbered lines from an interlaced video sequence
FIFO	First-In, First-Out
FIR	Finite Impulse Response
FPGA	Field-Programmable Gate Array
FSM	Finite State Machine
GOP	Group Of Pictures, a set of coded video images
H.261	A video coding standard
H.263	A video coding standard
H.264	A video coding standard
HDTV	High Definition TeleVision
Huffman coding	Coding method to deduce redundancy
HVS	Human Visual System, the system by which the humans perceive and interpret the images
I-picture (slice)	Coded picture (slice) without reference to any other frame
IEC	International Standard Commission
IEEE	Institute of Electrical and Electronics Engineers
IFFT	Inverse Fast Fourier Transform
IIR	Infinite Impulse Response
Inter (coding)	Coding of video frames using temporal prediction or compensation
Interlaced (video)	Video data represented as a series of fields
Intra (coding)	Coding of video frames without temporal prediction
ISO	International Standards Organization
ITU	International Telecommunication Union
JPEG	Joint Photographic Expert Group (also an image coding standard)
JPEG2000	An image coding standard
Level	A set of conformance parameters (applied to a Profile)
LSB	Less Significant Bit
Macroblock	A region of frame (usually 16 x 16 pixels) formed of blocks
Motion compensation	Video frame prediction method, based on a motion model
Motion estimation	Method to estimate relative motion between two or more video frames
Motion vector	Vector indicating a displacement block or region to be used in motion compensation
MPEG	Moving Picture Experts Group
MPEG-1	A multimedia coding standard
MPEG-2	A multimedia coding standard

MPEG-3	A multimedia coding standard
MSB	Most Significant Bit
NAB	National Association of Broadcasters
NAL	Network Abstraction Layer
OFDM	Orthogonal Frequency Division Multiplexing
P-picture (slice)	Coded picture (slice) using motion-compensated prediction from one reference frame
PAL	Phase-Alternating Line
Profile	A set of functional capabilities (related to a video CODEC)
Progressive (video)	Video data represented a series of consecutive completes frames
PSNR	Peak Signal to Noise Ratio, an objective quality measure
QAM	Quadrature Amplitude Modulation
QCIF	Quarter Common Intermediate Format
QPSK	Quaternary Phase Shift Keying
Quantize	Reduce the precision of a scalar or vector quantity
Rate-distortion	Measure of CODEC performance (distortion at a range of coded bit rates)
RGB	Red Green Blue color space
RTL	Register Transfer Logic
Scalable coding	Coding a signal into a number of layers
SDRAM	Synchronous Dynamic Random Access Memories
SDTV	Standard Definition TeleVision
SECAM	Systme Sequentiel Couleur A Mmoire
SNR	Signal to Noise Ratio
Studio quality	Lossless or near-lossless video quality
TS	Transport Stream
VALIDATE	Verification And Launch of Integrated Digital Advanced Television in Europe
VCEG	Video Coding Expert Group (committee of ITU)
VHDL	Very High level Design Language
VLC	Variable Length Code
VLSI	Very Large Scale Integration
VO	Video Object
VOP	Video Object Plane
VQEG	Video Quality Expert Group
YCrCb	Luminance, Blue chrominance, Red chrominance color space
YUV	A color space

Abstract

Analogue wireless cameras have been employed for decades, however they have not become an universal solution due to their difficulties of set up and use. The main problem is the link robustness which mainly depends on the requirement of a line-of-sight view between transmitter and receiver, a working condition not always possible. Despite the use of tracking antenna system such as the Portable Intelligent Tracking Antenna (PITA [1]), if strong multipath fading occurs (e.g. obstacles between transmitter and receiver) the picture rapidly falls apart.

Digital wireless cameras based on Orthogonal Frequency Division Multiplexing (OFDM) modulation schemes give a valid solution for the above problem. OFDM offers strong multipath protection due to the insertion of the guard interval; in particular, the OFDM-based DVB-T standard has proven to offer excellent performance for the broadcasting of multimedia streams with bit rates over 10 Mbps in difficult terrestrial propagation channels, for fixed and portable applications. However, in typical conditions, the latency needed to compress/decompress a digital video signal at Standard Definition (SD) resolution is of the order of 15 frames, which corresponds to $\simeq 0.5$ sec.

This delay introduces a serious problem when wireless and wired cameras have to be interfaced. Cabled cameras do not use compression, because the cable which directly links transmitter and receiver does not impose restrictive bandwidth constraints. Therefore, the only latency that affects a cable cameras link system is the on cable propagation delay, almost not significant. when switching between wired and wireless cameras, the residual latency makes it impossible to achieve the audio-video synchronization, with consequents disagreeable effects.

A way to solve this problem is to provide a low delay digital processing scheme based on a video coding algorithm which avoids massive intermediate data storage. The analysis of the last MPEG based coding standards puts in evidence a series of problems which limits the real performance of a low delay MPEG coding system. The first effort of this work is to study the MPEG standard to understand its limit from both the coding delay and implementation complexity points of views.

This thesis also investigates an alternative solution based on HERMES codec, a proprietary algorithm which is described implemented and evaluated. HERMES achieves better results than MPEG in terms of latency and implementation complexity, at the price of higher compression ratios, which means high output bit rates. The use of HERMES codec together with an enhanced OFDM system [2] leads to a competitive solution for wireless digital professional video applications.

Key words: *digital, video, algorithm, MPEG, codec, delay, latency, complexity, FPGA.*

Version Abrégée

Les caméras analogiques sans fils ont été utilisées pendant des années sans devenir une solution universelle car elles sont difficiles à metre en place et à utiliser. Le problème majeur se situe au niveau de la robustesse de la liaison RF, qui dépend principalement de la présence d'une vue directe entre le transmetteur et le récepteur, ce qui n'est pas toujours possible dans la pratique. Bien que des systèmes de poursuite d'antenne aient été développés coment le Portable Intelligent Tracking Antenna (PITA [1]), une perte significative d'images apparaît en présence de fortes atténuations et de chemins multiples, due par exemple à des obstacles entre le transmetteur et le récepteur.

Les caméras numériques sans fils basées sur le multiplexage par répartition orthogonale de la fréquence (OFDM) représentent une solution appropriée au problème décrit auparravant. L'OFDM offre une protection efficace en cas de chemins multiples rencontrés en transmission hertzienne terrestre grâce, entre autres, à l'insertion d'un intervalle de garde. Plus particulièrement, la norme DVB-T, basée sur la technologie OFDM, s'est avérée excellente pour la distribution des flux multimédias avec des débits de plus de 10 Mbps, dans des canaux terrestres difficiles, pour des applications fixes ou portables. Néanmoins dans des conditions typiques de travail, la latence nécessaire pour compresser et décompresser un signal vidéo numérique de résolution SD (Standard Definition) est de l'ordre de 15 images, ce qui correspond à environ 0.5 secondes.

Ce délai introduit un problème important dans le cas où ces caméras sans fil doivent être interfacées avec des caméras classiques (avec fils). Ce type de caméras n'utilisent pas de compression, vu que la connection via un câble entre le transmetteur et le récepteur ne donne pas de restrictions sur la bande passante. La seule latence sur une connexion câblée est due à la propagation des signaux dans le câble, qui peut être considérée comme négligeable. Lors de la commutation entre différentes sources composées de caméras avec et sans fil, la latence résiduelle ne permet pas de maintenir la synchronisation entre l'audio et la vidéo.

Une solution à ce problème est l'utilisation d'un algorithme de compression vidéo à très bas délai, basé sur un schéma qui ne requiert pas le stockage d'une grande quantité de donnés intermédiaires. L'analyse de la dernière méthode de codage vidéo, basée sur le

standard MPEG, met en évidence une série de problèmes qui limite l'efficacité d'un tel codage pour le bas délai. Nous aborderons tout d'abord l'étude du standard MPEG afin de mettre en évidente ses limites du point de vue du délai d'encodage et de la complexité à être implémenté.

Ensuite, une solution alternative basée sur HERMES, un algorithme propriétaire, sera décrite, évaluée et implémentée. HERMES donne des meilleurs résultats que MPEG sur la latence d'encodage et la complexité d'implémentation, mais HERMES ne peut pas produire des taux élevés de compression, ce qui se traduit par un haut débit binaire en sortie. L'utilisation du codec HERMES, conjointement avec un système OFDM amélioré [2], offre une solution compétitive pour des applications professionnelles de transmission vidéo numérique sans fil.

Mots clef: *numérique, vidéo, algorithme, MPEG, codec, retarde, latence, complexité, FPGA.*

Summary

The first part of this thesis introduces some basic concepts which are required to understand the basic principles of digital video compression techniques. The discussion explains the generic motion-compensated DPCM/DCT model, which represents the core of all the MPEG compression algorithm, giving a further description of the rate-controlled codec model and some related synchronization techniques, which are employed to develop constant bit rates applications. The introduced concepts are useful to understand MPEG-4 Visual and H.264, the latter and most advanced video coding methods, which are the state of the art of video compression.

Chapter 3 gives an overview on the evolution of video compression techniques, starting from MPEG-1 till MPEG-4. The description become more detailed on both MPEG-4 Visual and H.264, giving some information about profiles & levels and latter developed features which increase the compression performances. What comes out from this discussion is that since MPEG-1, the standard focuses on the development of more and more efficient compression schemes to achieve better quality at lower bit rates, whereas the processing latency has not taken into account. Therefore, MPEG does not provide yet a real low delay coding profile which can satisfy the requirements of an emerging group of applications.

The only low delay mode presented by MPEG achieves about 150 ms using a restricted subset of coding tools, whereas broadcast and studio-production applications request latencies lower than 40 ms. Chapter 4 starts from this basic configuration, carrying out a description of the main MPEG low latency tools, which puts in evidence a loss of compression efficiency due to a simpler processing scheme, which is in contrast with the increasing complexity of latter MPEG standards evolutions. The VBV buffer control strategy at encoder side is a further critical point that affects the design of a coding scheme for CBR applications. The best known MPEG-2 low delay codec proposes a very inefficient solution based on a bandwidth overhead over 66%, which is the expensive price to achieve the end to end processing delay of 54 ms at SD resolution and PAL format. However, the existing solution is affected by further implementation problems related to both encoding and decoding process, which denounces the limit of existing standard MPEG platforms in supporting a real low latency. Therefore, the achievement of a low delay MPEG-based encoding scheme is not only a theoretical problems, some implementation issues have to be taken into account as well.

Chapter 5 introduces HERMES algorithm, which is a proprietary codec based on a DPCM predictive intra-coding scheme. The described method seems to be a good candidate to achieve a very short latency, as its serial processing does not requires intermediate data storage. Furthermore, a low implementation complexity is expected. In a HERMES rate-controlled scheme, the main issue to achieve a low coding/deconding delay is the VBV buffer size, which depends on the slice-coding feature, the control strategy and the available quality scale. Some methods are

COMPANY	CODEC	MODULATION	LINK DELAY (ms)	BIT RATES (Mbps)
Tandberg	MPEG-2	DVB-T	180	–
Kingswood Warren Ventures	DVCPRO	2 × DVB-T	80	28
Atlantic Digital Limited	MPEG-2	DVB-T	80	–
Thomson/Grass Valley	WAVELET	DVB-T	60	22
LiveTools	MPEG-2	Proprietary	60	5 ÷ 20
LINK-Research	MPEG-2	DVB-T	50	–
Global Microwave Systems	MPEG-2	Proprietary	40	–
Gigawave	WAVELET	Proprietary	20	–

proposed and some estimations confirm the possibility to achieve a delay less than few milliseconds at medium high encoding quality. On the other hand, the simpler HERMES coding scheme, cannot achieve high compression ratios, therefore high output bit rates are expected.

HERMES is proposed as alternative to MPEG for applications which require a very short encoding/decoding latency and that have access to data links at bit rates $R > 22$ Mbps. The latter development of an enhanced OFDM modulation scheme, makes possible the integration of HERMES into LiveTools DVB-T system, which can provides a valid solution for wireless studio production applications.

Chapter 6 proposes a complete hardware architecture for HERMES codec. Each VHDL block has been described and a global VBR verification version has been implemented on a Xilinx FPGA platform. However, the estimation of the real processing latency requires a CBR implementation, which will be realized in next verification step. Finally, a resource requirements comparison between HERMES and a MPEG-4 core, puts in evidence the advantages of HERMES low complex design.

The CBR buffer control method which has been presented in chapter 5, has been fully evaluated using a C++ simulator of HERMES codec. The previous estimations about buffer size and delay are confirmed by the simulation results. Furthermore, comparison tests show that HERMES outperforms MPEG-2 coding quality at bit rates $R > 36$ Mbps, employing a redundant buffer size $B = 400$ Kbit and achieving a delay $D < 12$ ms. A further analysis of the coding impairments revels that the slice-independency condition may cause artifacts. However, the problem can be solved using minimal processing modifications at the price of a slightly increased encoding bit rate.

Chapter 1

Introduction

1.1 DIGITAL COMMUNICATIONS

The demands for multimedia services has reached a level never seen before and the quality expected by customers is becoming higher and higher. To attain the highest possible quality all these services use digital technology: analog signals such as speech, audio, image and video, are sampled and digitized as binary data for transmission or storage purposes and reconstructed at the receiving ends in order to be free from noise and waveform distortion which affect similar analog applications.

However, these digitized data are often voluminous. Even though the technology is continuously progressing at pushing up the bandwidth limit and reducing the transmission/storage cost, but the channels bandwidth and storage capabilities are still limited and expensive in comparison to the volume of these *raw* digital data. To make all the digital service feasible and low cost, data compression is essential.

Because of its importance in digital communications and applications, compressibility of digital signals has been the object of continuous research and study over decades. The performance of a source coder can be represented into a *five* dimensions space [3] as referred in figure 1.1.

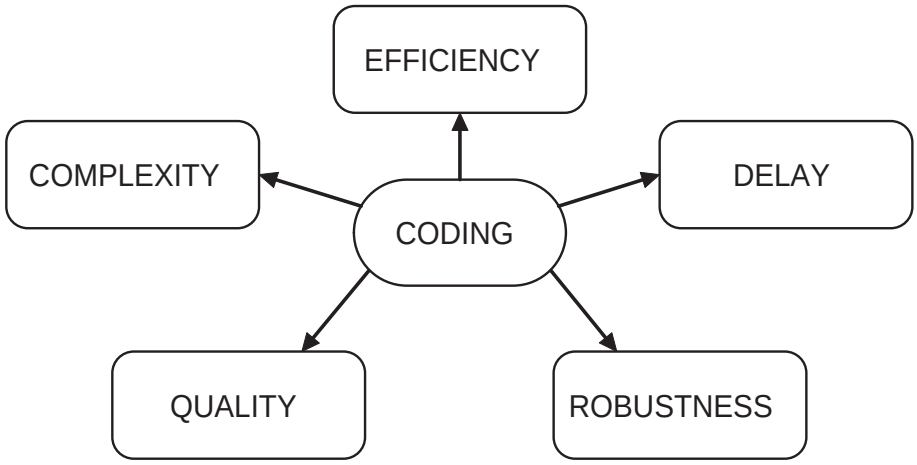


Figure 1.1: Coding Space

- **Coding efficiency** - It is measured in bit per sample or bit per second and it is limited by the source entropy (information content). Sources with bigger entropy are more difficult to compress.
- **Coding complexity** - It is the computational effort needed to implement the encoder and decoder functionalities. It can be measured in terms of hardware requirements (memory, BUS width, clock frequency, etc..) and number of arithmetic operations per second (MOPS).
- **Coding delay** - A sophisticated coding-decoding algorithm often leads to increased coding/decoding delays. In many applications coding delay has to be constrained and this forces

the designer to use a simpler algorithm.

- **Coding quality** - With this term we characterize the signal at the decoder output. It is measured using SNR or PSNR, but these methods are not fully correlated to the perceptual quality, for this reason the mean opinion score is still used: very annoying, annoying, slightly annoying, perceptible but not annoying, imperceptible.
- **Coding robustness** - This comes from a typical telecommunication terminology. It is the ability of the algorithm to react when in critical situations, e.g. the lost of synchronization with the source, a fatal error due to a mismatch in the parameters setting or whatever can deviate the CODEC from the regular working.

All the digital multimedia applications take place in the space defined by these variables, however there are still unexplored zones as for example the areas related to a very low encoding delay and medium-high quality.

In the field of digital data compression, still images and audio/video sequences are the main research topics in the technical community due to immediate and potential applications:

- **Video-Conferencing**
- **Videophony**
- **Multimedia**
- **HDTV**
- **Interactive TV**
- **Telemedicine**

Many different systems and algorithms for compression and decompression have been conceived and developed in the last 20 years [4, 5, 6]. In order to encourage interworking and competition and to allow intercommunication within products from different manufactures, standard methods of compression (coding and decoding) have been proposed [7]. A number of key international standards for image and video compression have been developed, including the JPEG, MPEG and H.26x series.

1.2 WIRELESS DIGITAL TELEVISION PRODUCTION

In the past years, great interest was raised for applications aimed to wireless digital television productions. The technology evolution made it possible to compress a digital audio/video stream and reduces the required bit rate down to some Megabit-per-second (Mbps). For example, standard-definition, 50 Hz interlaced video generates around 216 Mbps of uncompressed digital video

stream, while each digital audio channel adds $\simeq$ 1 Mbps more. Employing MPEG2 encoding and exploiting both spatial and temporal redundancy, the bit rate can be lowered down to an aggregate 5Mbps (video + 2 audio channels) still preserving a suitable quality for broadcasting. The definition of the DVB-T [8] standard made it possible to implement digital wireless links capable of carrying the compressed multimedia stream over difficult terrestrial propagation channels.

There is a strong demand for such systems: there are applications where wired systems simply cannot be deployed, as bicycle race, sailing race and marathon just to cite some examples, but even in other scenarios the possibility to use wireless links is greatly appreciated. Wired systems require specialized and very expensive cables that have to be deployed before and must be collected after the event; the cost in terms of material bill and time spent can be very important. For example, football matches require a large amount of wiring, in the order of several kilometers, and a couple of days must be accounted for system setup and de-mounting. A wireless system would be easier to setup and would require less time to deploy.

The requirements of these applications can vary over a wide range [2]; a summary of most common features is listed below:

- **Long distance** - The wireless links can be as long as 50km. The RF power should be sufficient and the modulation should be robust to gaussian noise in order to overcome the attenuation due to strong propagation loss.

- **High mobility** - The system should withstand a high degree of mobility; transmitter and/or receiver motion can be expected, and the speed can be as high as 300km/h.

- **Multipath protection** - In this case of a terrestrial channel, multiple echoes reaches the reception antenna and experience different attenuations and delays. This is the cause of the so-called *multipath fading*. The employed modulation scheme should be able to overcome this effect with less or no performance degradation.

- **Shadowing protection** - Another impairment deriving from mobility is called *shadow fading*, and is caused by the interposition of obstacles in the line-of-sight between transmission and reception antennas. For power-budget limited wireless links, this translates in short signal loss. The modulation and the error-correction scheme should overcome flawlessly this type of impairment.

- **High bit rate** - The available bit rate should be as high as possible to reduce the compression ratio and allow transmission of high-quality and high-resolution audio/video streams.

- **Low latency** - The modulation and error-correction scheme should exhibit a latency as low as possible (typically $<$ 10 ms) to allow mixing of wired and wireless system without noticeable delay.

- **Backward channel** - A backward channel (broadcasted from the regia to all the cameras) can be used to convey the live feed to each camera along with camera control data.
- **Compact size** - The system should be compact and light-weight to allow mounting on the back of a professional camera.
- **Low power consumption** - Since the system will be mostly powered by batteries, power consumption is still a major issue.

Even without further in-depth discussions, it is clear that some contradictory requirements are given in the above list. For example, a low latency encoding at broadcast picture quality calls for higher encoding bit rates which are supported by less robust modulation schemes (16QAM and 64QAM) that are more affected by shadow and multipath fading. Nevertheless, there are several typical scenarios which can be accounted for; of course, for each one some compromises are needed and some priority are given to the above requirements.

1.2.1 Studio Production

The encoding/decoding latency becomes a critical point for applications which require strong interactions between wired and wireless systems, for example in studio production. In this case, wireless cameras will not experience high mobility; distance and power are usually limited and shadow fading is not expected. However, if the latency of the wireless system is not kept at minimum, the following tasks may be affected:

- **Video Mixing** - The problem occurs when some wireless cameras are mixed with wired ones. When switching between the different sources, we may expect latencies in the range $300 \div 600$ ms, which means $7 \div 14$ frames. This is not acceptable in professional applications as it generates discontinuities in the video sequence.
- **Audio-Video synchrony** - Audio and video signals are completely separated sources; they are captured by different equipments and then mixed together after a preprocessing stage. If the delay is higher than 80ms (2 PAL frames) the phase shift starts to be perceivable as the well known delay effect typical in some satellite telecommunications.
- **Control camera feedback** - A centralized camera control and the live feed can be implemented using a backward channel, which gives the studio director the possibility to modify certain video parameters on remote cameras (colors, darkness, brightening etc.) in order to achieve the optimal image equalization. If the delay between the parameters setting and their effect on video is too high, an instability feedback could happen.

Actually, most of existing digital wireless systems are MPEG-2 based, but their latencies are not low enough to satisfy the application requirements. On the other hand, MPEG was conceived

to mainly cope with storage and transmission applications; consequently most efforts have been spent to get the best encoding quality in minimum space, while the processing time was not considered as an important constraint. Since the MPEG-1, the compression performances and algorithms flexibility have been increasing, but also development and implementation costs have grown up and low delay coding applications have not been yet taken into account.

Most of low delay non-MPEG systems are based on wavelets and DV codec [9, 10]. These solutions give better results on the latency point of view and require less complex hardware with the further advantage of lower power consumption. On the other hand, both wavelets and DV methods produce higher output bit rates, as they perform simpler compression schemes than MPEG.

An ideal coding algorithm for applications of wireless studio production should meet the following constraints:

- **Low delay** - With *low delay* we mean the overall encoding/decoding latency, which should be < 40 ms to satisfy the already mentioned requirements. Actually DV and wavelets methods achieve latencies around 20 ms, while MPEG best result is close to 54 ms.

- **High quality** - The difference between the original frame and the reconstructed one is due to the lossy nature of the coding method. Since these differences or artifacts are not perceivable by the human visual system (HVS), a high quality is achieved. There is a direct relation between quality and bit rate: The higher the bit rate, the higher the video sequence quality. Both wavelets and DV exploit high coding quality between $20 \div 30$ Mbps, which in wireless applications is usually considered a high bit rates range.

- **Low complexity** - In a wireless system the encoding device is located on the camera, so it is directly powered by the on system battery. It is obvious that an algorithm which requires a more complex hardware (in terms of on chip resources and clock frequency) leads to a higher power consumption. The complexity can be also evaluated in terms of development costs. If the algorithm complexity is lower, a shorter design time and a lower implementation cost can be expected.

1.3 GOAL OF THIS WORK

The first part of this work investigates the MPEG latency limits correlated with the implementation complexity. For this purpose, a complete analysis of MPEG features is discussed, starting from the algorithm basis towards the most advanced developments of recent years, which are represented by MPEG-4 and H.264 standards. Then a subset of MPEG features is identified which is required to achieve a low coding latency and a real MPEG-2 low delay system is kept as reference in order to define the MPEG trade-off between latency and complexity.

In the second part of this thesis HERMES is introduced; a proprietary low delay coding algorithm which intends to be an alternative to the existing methods. HERMES is fully described and then a corresponding FPGA architecture implementation is proposed. Using a C++ simulator comparisons between HERMES and MPEG are finally presented.

1.4 MAIN CONTRIBUTIONS

This thesis addresses a wide spectrum of practical and theoretical tasks, which involve hardware development implementation and debug as well as algorithm conception and study. The main contributions of this work can be summarized as follow:

- The integration of further tools in a reference C++ simulation software used to study and enhance HERMES performances. The main task has been the study and implementation of a Constant Bit Rate (CBR) buffer control strategy, in order to obtain a rate controlled encoding system.

- The study of the lowest coding delay supported by an hardware MPEG-2 codec integrated on the Livetools Technology [11] wireless system, presented at National Association of Broadcasters (NAB) in Las Vegas as the first integrated encoder/modulator ready to be mounted on the back of professional cameras. This equipment was developped during a collaboration project between Livetools Technology and the Ecole Polytechnique Federale de Lausanne (EPFL) [12].

- The study of complexity and performances of the MPEG video coding algorithms (MPEG-2, MPEG-4 and H.264) has been carried out using standard C++ software verification models which are: *MSYS Toolkit v1.1 (1994)* for MPEG-2 and the *JM Reference Software v8.2 (2001)* for AVC/H.264. Further data on existing MPEG-4 FPGA encoder core have been made available thanks to Xilinx [13].

- The development and implementation of a Variable Bit Rate (VBR) version of HERMES on FPGA. This first HERMES version has been integrated on the Xilinx multimedia evaluation board, which is a hardware platform equipped with input/output audio/video chip interfaces, 5 fully-independent banks of 512K $\times$32 ZBT RAM and a Virtex-II XC2V2000-FF896^{TM} Xilinx FPGA.

1.5 STRUCTURE OF THIS THESIS

This thesis is structured as follows. Chapter 2 describes the basis concepts and techniques in video coding, which are required to understand the main features of MPEG coding standards. The following Chapter 3 gives an idea about the complexity of MPEG-4 and AVC/H.264, the

most advanced video coding algorithms which represent the state of the art in video coding. The analysis of MPEG minimum coding delay is carried out in Chapter 4, where a subset of MPEG coding features is selected to define the structure of a low delay MPEG codec. Then a real MPEG-2 low delay system is fully described and analyzed to be used as reference for comparisons and optimizations.

In Chapter 5 HERMES video coding algorithm is presented, while a CRB buffer control strategy is proposed, which makes it possible to achieve predictions about latency and performances. Chapter 6 is fully dedicated to HERMES VHDL architecture for targeting Xilinx FPGA platforms. The last section presents a hardware complexity analysis based on a further comparison between HERMES implementation results and a MPEG-4 VHDL core.

Chapter 7 presents many data coming from software simulations on a rate controlled HERMES version, which gives results on HERMES coding efficiency and minimum latency. Finally Chapter 8 contains the conclusion and future work.

Chapter 2

Video Coding Concepts

2.1 INTRODUCTION

As shown in table 2.1, the volume of digital video data is notoriously huge, despite communication channels have limited transmission bandwidths. However networks bit-rates continue to increase and the storage capabilities of hard disks, flash memories and optical media (CD DVD etsc.) is greater than ever before; with the price per transmitted or stored bit continually falling. In this scenario, it is perhaps not immediately obvious why video compression is necessary.

FORMAT	RESOLUTION	RAW BIT-RATE	REMARK
QCIF	176 ppl ×144 lfp ×25 fps	10.1 Mbps	digital video
CIF	352 ppl ×288 lfp ×25 fps	40.5 Mbps	digital video
SIF	360 ppl ×288 lfp ×25 fps	41.4 Mbps	digital video
SDTV	720 ppl ×576 lfp ×25 fps	165.8 Mbps	digital TV
HDTV	1920 ppl ×1080 lfp ×25 fps	829.4 Mbps	digital TV

Table 2.1: Uncompressed digital video sources

Video coding is the process of compressing and decompressing a digital video signal. There are two main reasons why video coding is still used and why there is such a significant effort to make it better:

- It makes it possible to use the digital video in transmission and storage environment that would not support uncompressed *raw* video because of a lower bit-rate bandwidth or storage capability. For example, current internet throughput rates are insufficient to handle a compressed video in real time, even at low bit-rates and resolutions. A DVD (Digital Versatile Disk), can store 17 Gbytes if both layers of 4.7 Gbyte are used on both disk sides, that means that we could play only 13.6 minutes of video in SDTV raw format, so DVD-Video storage would be useless without audio/video compression.

- Video compression allows to exploit transmission and storage resources in a more efficient way. For example, If a high bit rate transmission channel is available, it is a more attractive solution to send a high-resolution compressed video or multiple compressed video sub-channels rather than sending a single, low resolution, uncompressed bitstream.

Even with constant advances in storage and transmission capacity, compression is likely to be an essential component of multimedia services for many years to come.

This chapter introduces concepts such as sampling formats, space-time redundancy reduction and quality metrics which are the base of any video coding algorithm. Then the description is more and more focused on MPEG, starting from its basic coding features until the generic motion-compensated DPCM/DCT model. Finally, the rate-controlled codec model is explained.

2.2 NATURAL VIDEO SEQUENCES

A scene is sampled at a point in time to produce a frame, which became the digital representation of that visual scene at that point in time. A frame is composed of two fields, consisting of odd or even numbered lines of spatial samples. A video sequence is a collection of frames sampled at different points in time. Over a rate of 15 frames per second (fps), the Human Visual System (HVS) is not capable to distinguish each single frames and it gives us the illusion of a fluid continuity in the movements of objects represented in the scene. PAL and NTSC worldwide TV standards use rates of 25 and 30 frame per second respectively.

A typical *real world* or *natural* scene is composed of multiple objects each one with their own shape, depth, texture and illumination. The color and brightness can change in different ways depending on the subject, its motion and the camera motion. We can identify two main features that are relevant for video processing and compression:

- **Spatial** - Texture variations, number and shape of objects, colors, brightness etc.
- **Temporal** - Objects motion, changes in illumination, camera or viewpoint movement etc.

2.3 VIDEO DATA SAMPLING

A natural video scene is spatially and temporally continuous. The digital video is a representation of the scene in the digital domain, where we perform a spatial-temporal sampling as shown in figure 2.1. Usually a rectangular grid is used (frame) and each point on the grid (picture element or pixel) is characterized by a set of levels that represent brightness (luminance) and color. The temporal sampling is performed collecting new still frames at regular intervals in time.

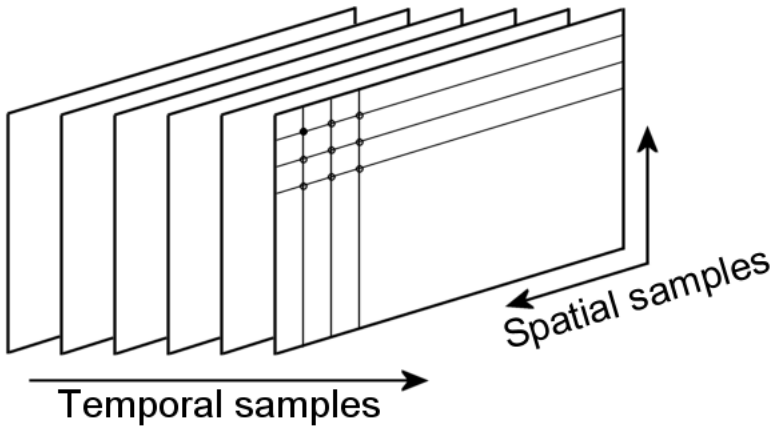


Figure 2.1: Spatial and temporal sampling of a natural video sequence

The process of video digitizing involves the three basic operations of filtering, sampling and quantization. The filtering stage is applied to avoid aliasing artifacts to the following up-sampling

process. In the case of color image capture, each color component (Red, Green and Blue) is obtained by a separate optical filter and projected onto an image sensor, as for example a Charge Coupled Device (CCD) matrix.

2.3.1 Spatial Sampling

The output of a CCD matrix is an analog signal, a continuous electrical signal that represents a video image. An anti-aliasing filter is then applied to restrict the spectral characteristic. Sampling the filtered signal means to collect a set of numeric values (samples) corresponding to regular positions on the picture. A rectangle is the most common format for a sampled image, with the sampling points positioned on a grid. The sample rate (number of samples collected per second) determines the grid resolution. Choosing a *coarse* sampling grid produces a low-resolution sampled image, while increasing the sampling rate increases the resolution of the sampled image. Another important parameter is the number of bits used to represent each sample. This latter value is known as the quantization levels, which for consumer video applications is set to 8 bits according to ITU-R Recommendation 601 [14].

2.3.2 Temporal Sampling

A moving video scene is captured by taking rectangular *snapshot* of the signal at regular time intervals. The illusion of motion is given by playing back the series of collected frames. A higher temporal sampling rate (frame rate) gives better results in term of motion fluidity. We can distinguish within four frame rate categories:

- **Very low** Frames rate below 10 fps are sometimes used for very low bit-rate application as slow motion video. The main advantage is that the amount of data is linearly reduced, but motion is clearly unnatural at this rate.
- **Low** In video conference applications we use a rate between 10 and 20 frames per second. The sequence is fluid, but in the case of fast movements, unnatural motion may occurs.
- **Standard** Sampling at 25 or 30 frames per second is the standard for television pictures for PAL and NTSC respectively.
- **High** In HDTV applications, rates of 50 or 60 frames per second can be used. The motion appears very smooth and fluid, but at the expense of a very high data rate.

2.3.3 Frames and Fields

A video frame is a collection of lines. Half of the data in a frame, corresponding to odd-numbered lines, are called *top field*, while the remaining even-numbered lines form the *bottom field*. This leads to two different ways of sampling and displaying a video scene:

- **Progressive**: Since video is a series of still images, it makes sense to just display each full image consecutively, one after the other. This is the basic technique of progressive, or non-interlaced, displays which leads to higher vertical resolution. However, the higher refresh time can cause flicker noise in large regions of constant color.

- **Interlaced**: In the early days of television, a technique called *interlacing* was employed to reduce the amount of information sent for each image. By transferring the top field, followed by the bottom field (as shown in 2.2), the amount of information to capture and send at the same time for each image is halved. For the about same cost as a 25 or 30 Hz progressive display, the interlaced display can double its refresh rate (to 50 or 60 Hz) in an attempt to avoid flickering.

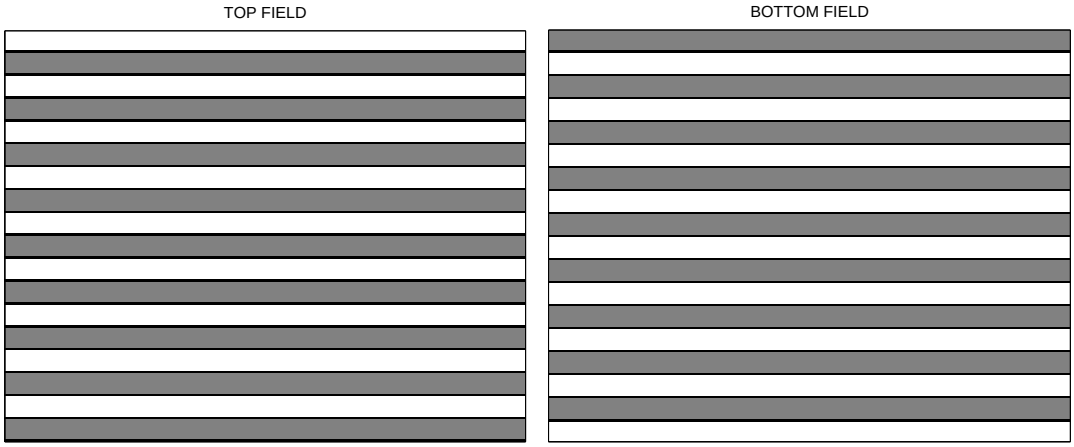


Figure 2.2: Interlaced video sequence. Each gray strip is an active video line

The advantage of the interlacing sampling method is that it is possible to send twice as many fields per second as the number of frames in an equivalent progressive sequence with the same data rate, giving the appearance of smoother motion. For example, a PAL video sequence consists of 50 fields per second and, when played back, motion can appears smoother than in an equivalent progressive video sequence at 25 frames per second.

2.4 COLOR SPACES

Most of digital video applications make intensive use of color images and thus need methods to capture and represent color information. Every single color can be obtained by a linear combination of three main color components, so three numbers per pixel position are required to accurately represent color images. This means that each color image can be decomposed into three monochrome pictures having same sizes and resolution, as in the well known RGB color

space. The method chosen to represent brightness and color is described as a color space. All color spaces are three-dimensional normal coordinate systems, meaning that there are three axes (red, green, and blue color intensities in RGB) that are perpendicular to one another.

2.4.1 RGB

The RGB color space is an additive color space. Its origin starts at black, and all other colors are achieved by linear combinations of primary colors. It is a natural choice for Color Cathode Ray Tubes (CRTs) and Liquid Crystal Displays (LCDs) where black, or no light intensity, is the starting point, and the increasing intensity of the red, green, and blue components provides the range of *true* colors.

The red, green and blue intensities start at zero at the origin and increase along one of the axes. Because each color only gets values between zero and a maximum intensity (255 for 8 bit depth), the resulting structure is the cube shown in 2.3.

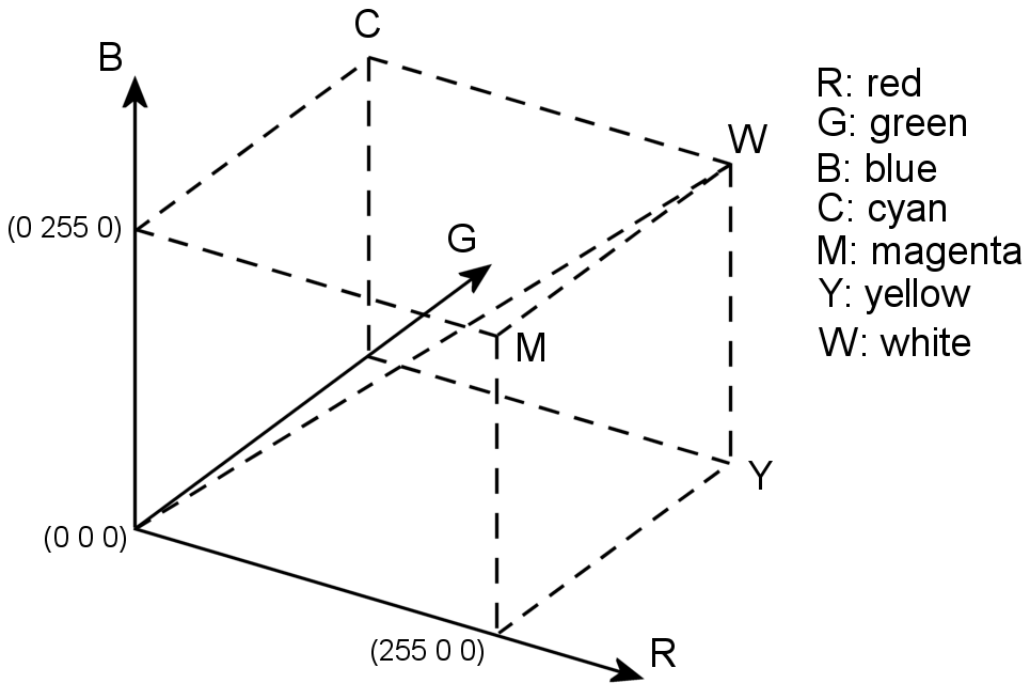


Figure 2.3: RGB color space representation

We can simply define any color by an ordered triplet $[red, green, blue]$ which represents its coordinates in the RGB cube. Black has zero intensities in red, green or blue, so it has the coordinates $[0, 0, 0]$. At the opposite corner of the color cube, the white $[255, 255, 255]$ has maximum intensities of each color. Full intensity red, having zero green or blue components, also is positioned at a corner of the cube at location $[255, 0, 0]$. Yellow, which is combination of red and green, is positioned at $[255, 255, 0]$. Cyan and magenta, which are combinations of green, blue

and red, blue, respectively, are at $[0, 255, 255]$ and $[255, 0, 255]$. Finally, note that a middle gray is at the exact center of the cube at location $[128, 128, 128]$.

2.4.2 YCbCr

RGB method is not the most efficient way to represent digital color pictures. R, G and B components are equally important and so must be stored with the same resolution. Moreover, to modify a color pixel intensity we must re-scale each single component. For these reasons, many video standards use luma (luminance Y) and two color difference (U, V or Cr, Cb) signals.

The RGB color system can be rotated to form a different color space such as YUV or YCrCb. The YCbCr color space was developed as part of ITU-R BT.601 [14] during the development of a world-wide digital component video standard. YCbCr is a scaled and offset version of the YUV color space. Y is defined to have a nominal 8 bit range of $16 \div 235$, while Cb and Cr are defined to have a nominal range of $16 \div 240$. The transformation between RGB and YCrCb is done according to linear equations 2.1 and 2.2.

$$\begin{aligned} Y &= 0.299R + 0.587G + 0.114B \\ Cr &= 0.564(B - Y) \\ Cb &= 0.713(R - Y) \end{aligned} \tag{2.1}$$

$$\begin{aligned} R &= Y + 1.402Cr \\ G &= Y - 0.344Cb - 0.714Cr \\ B &= Y + 1.772Cb \end{aligned} \tag{2.2}$$

The HVS is more sensitive to brightness than to color difference components, so it is possible to represent the chroma with a lower spatial resolution then luma, achieving a first compression step. There are several YCbCr sampling formats, such as 4:4:4, 4:2:2, 4:1:1 and 4:2:0, each one defining a different chrominance sub-sampling method.

2.4.3 YCbCr Sampling Formats

The notation $N : M : K$ defines the relative sampling rate within the three components $Y : Cb : Cr$ in the *horizontal* direction (a part the particular case of $K = 0$). Each sample is typically 8 bits (consumer applications) or 10 bits (professional video applications) per component. To display a YCbCr video sequence, first a conversion to 4:4:4 YCbCr format is necessary, using interpolation to generate the missing Cb and Cr samples.

- **4:4:4 Format** - The positioning of YCbCr samples for the 4:4:4 format are illustrated in Figure 2.4. Each pixel is formed by three samples: Y, a Cb, and a Cr, which leads to a 24 bits (30 bits for professional video) representation.

Figure 2.4: 4:4:4 YCbCr sampling format

- **4:2:2 Format** - Figure 2.5 shows the YCbCr samples positioning for the 4:2:2 format. For every two horizontal Y samples, there is one Cb and Cr sample, so each pixel requires 16 bits (or 20 bits for pro-video).

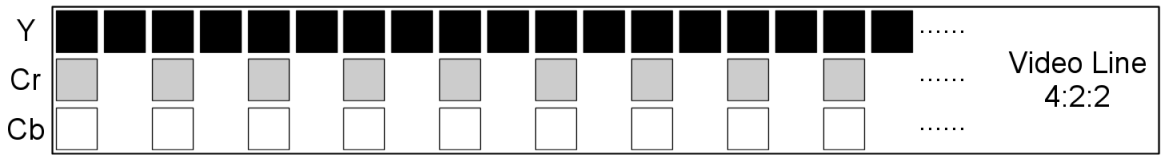


Figure 2.5: 4:2:2 YCbCr sampling format

- **4:1:1 Format** - Figure 2.6 illustrates the positioning of YCbCr samples for the 4:1:1 format (also known as YUV12), used in some consumer compression video applications. For every four horizontal Y samples, there is one Cb and Cr value. Each component is typically 8 bits, therefore each pixel requires 12 bits.

Figure 2.6: 4:1:1 YCbCr sampling format

- **4:2:0 Format** It is commonly used for video compression because rather than the horizontal-only 2:1 reduction of Cb and Cr used by 4:2:2, 4:2:0 YCbCr implements a 2:1 reduction of Cb and Cr in both the vertical and horizontal directions. This format (sometimes referred as YV12) requires 12 bits per pixel, as in 4:1:1. A common example of 4:2:0 sampling scheme is reported in figure 2.7.

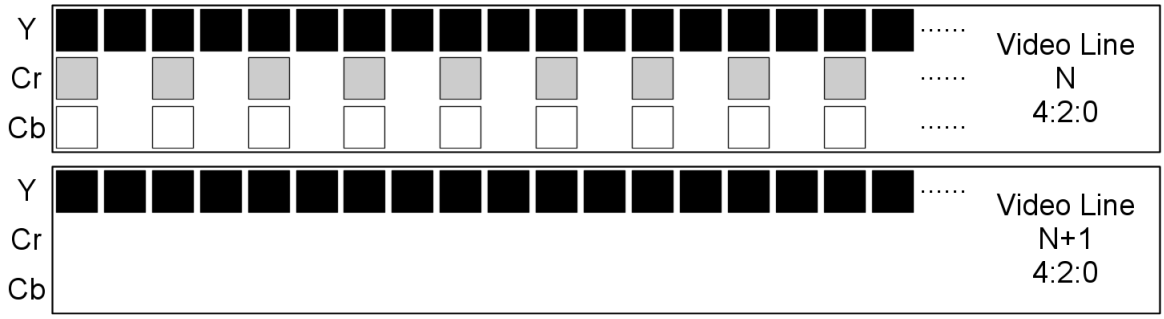


Figure 2.7: 4:2:0 YCbCr sampling format

2.5 VIDEO FORMATS

Most of video compression standards support a wide variety of video frame formats. The first step is to capture the video using one of the *intermediate formats* then compression and transmission follow. The Common Interface Format (CIF) is the starting point for a common set of formats widely used in video coding applications (figure 2.8). The choice of the frame resolution depends on the particular application and available storage or transmission capacity:

- **4CIF** - 4*times* Common Interface Format. It is a medium resolution format suitable for Standard Definition TeleVision (SDTV) and DVD-Video applications. The active frame size is 704×576 pixels.
- **CIF** - Common Interface Format or Common Image Format. The Common Interface Format was developed to support video conference. It has an active resolution of 352×288.
- **QCIF** - Quarter Common Interface Format. This video format was developed to allow the implementation of cheaper video phones. The QCIF format has a resolution of 176×144 active pixels.
- **SQCIF** - Sub Quarter Common Interface Format. As QCIF, this format is appropriate for mobile telephone multimedia applications where the display resolution and the encoding bit-rate are limited. The SQCIF format support a frame size of 128×96 active pixels.

ITU-R Recommendation BT.601-5 [14] defines a standard sampling rate and frame format for both PAL and NTSC television production video systems. The luminance component of the

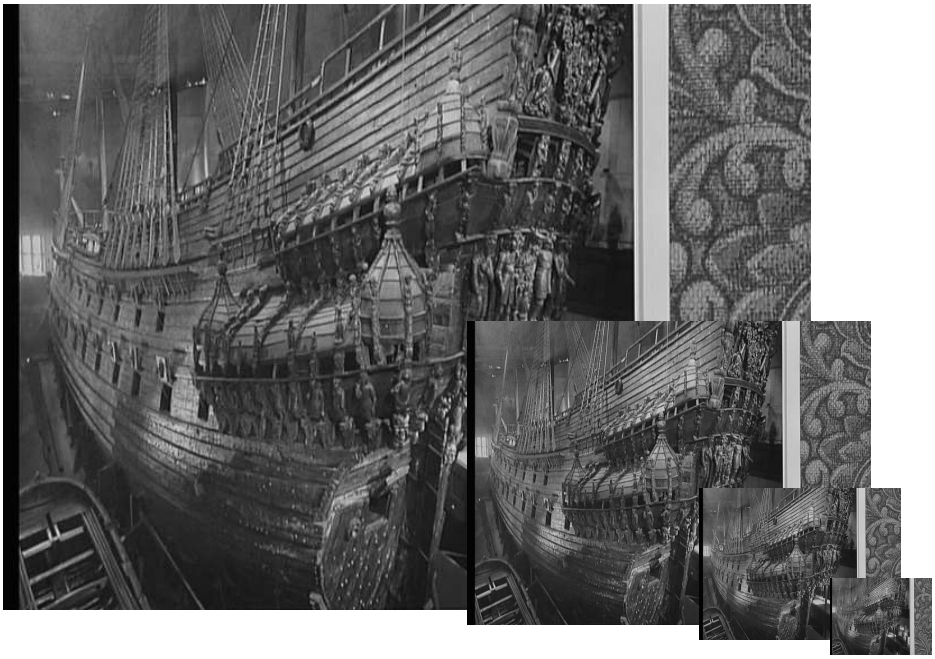


Figure 2.8: Luminance video frame at different sampling resolutions

video signal is sampled at 13.5 MHz and the chrominance at 6.75 MHz (refer to figure 2.9) to produce a 4:2:2 YCbCr component signal. Not all the samples correspond to the active video region. We have to take into account the horizontal and vertical blanking intervals, which are placed outside the edges of the display area.

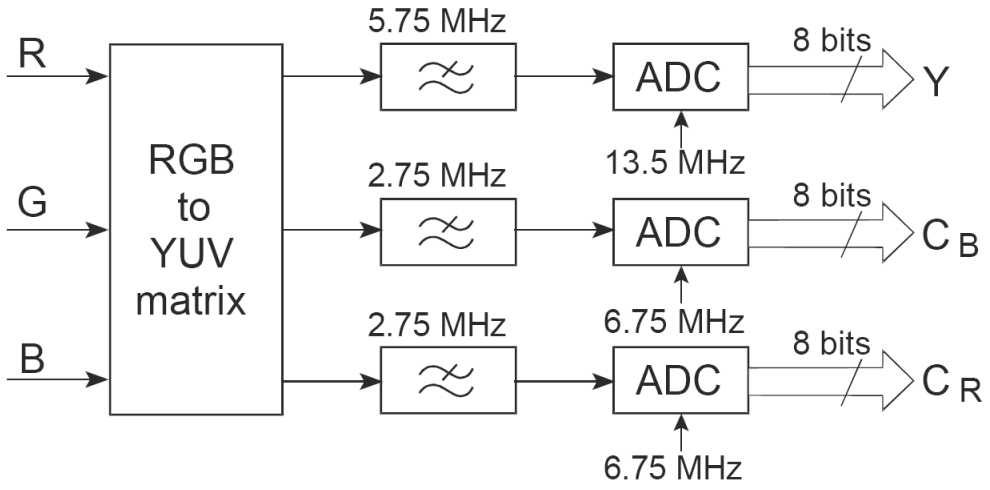


Figure 2.9: Digital video sampling scheme

The digital video parameters are listed in table 2.2, within which the number of samples per active line is $720+360+360 = 1440$ that is a constant for both PAL and NTSC systems. Another constant is the number of scanned lines per second: $576 \times 25 = 480 \times 30$. This means that the

lower PAL system frame rate (25 fps) is compensated for by a higher spatial resolution (576 active lpf) compared to NTSC system (480 active lpf), so that the total active video bit-rate is the same in each case (216 Mbps).

	NTSC (30 fps)	PAL (25 fps)
Field per second	60	50
Lines per frame	525	625
Y samples per line	858	864
Cr, Cb samples per line	429	432
Bits per sample	8	8
Total bit-rate (Mbps)	216	216
Active lines per frame	480	576
Y active samples per line	720	720
Cr, Cb active samples per line	360	360

Table 2.2: ITU-R BT.601-5 Parameters

2.6 QUALITY ASSESSMENT

Digital video systems exhibit fundamentally different artifacts than analog video systems. Lossy compression methods such as MPEG introduce distortions that highly depend on scene content and encoding rate. Traditional signal quality measurements are inadequate for the evaluation of the visibility of these artifacts. The assessment of perceived video quality is not a trivial task and although several methods have been developed, no any of them is able to achieve itself a sufficient video quality evaluation.

Methods for video quality assessment are divided into two categories: objective and subjective. Objective methods aim to mathematically estimate the impairment introduced to video during compression while the subjective counterparts aid in the compilation and statistical analysis of sample ratings generated by humans.

2.6.1 Subjective Quality Measurement

This type of evaluation is simply based on human beings. Several tests procedures for subjective quality evaluation are defined in ITU-R Recommendation BT.500-11 [15].

A commonly used procedure from the standard is the Double Stimulus Impairment Scale (DSIS) method. The subject is presented with pairs of sequences as displayed in the figure 2.10.

The reference sample is played first, followed by the impaired one. After each pair, the subject is given 8 seconds to vote with the following scale:

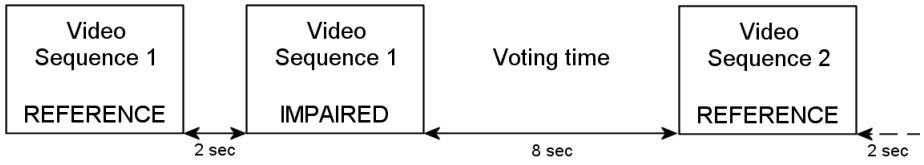


Figure 2.10: Diagram of the Double Stimulus Impairment Scale test

1. Very annoying
2. Annoying
3. Slightly annoying
4. Perceptible, but not annoying
5. Imperceptible

Another method that has been widely applied to evaluate high-quality TV sequences, is the Double Stimulus Continuous Quality Scale (DSCQS). This test is most suitable when there is no large quality variance between original and impaired sequences.

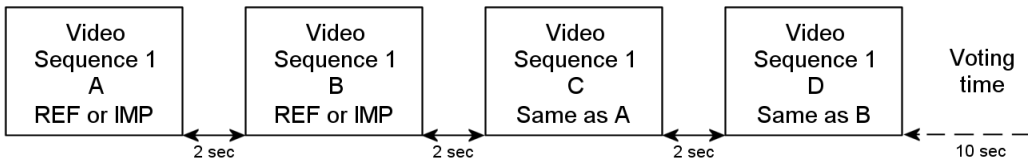


Figure 2.11: Diagram of the Double Stimulus Continuous Quality Scale test

As shown in figure 2.11 the sequences are played in two identical pairs. The order of the first pair is pseudo-random and is preserved in the second one. At the end of the two pairs, the subject is given 10 seconds to vote. This time the score is marked on a vertical scale which provides a continuous rating system in order to avoid quantization errors. After the test, the score is normalized in the range of $0 \div 100$ and the difference between the reference and impaired sequence is calculated. Finally, statistics such as standard deviation are achieved from the collected data set.

Tests such DSIS and DSCQS are considered to be valid measures of subjective video quality, however these methods suffer from practical problems:

- **Low reproducibility** - The results can change depending on the assessor and the video sequence under test. This variation can be limited by repeating the test with several sequences and several assessors.

- **Accessor useability** - After some tests, a *non-expert* assessor becomes familiar with the nature of video compression distortions and may give a biased score.

So a large pool of assessors is required to carry on this kind of tests which become extremely expensive and time-consuming.

2.6.2 Objective Quality Measurement

Over the years, many computational methods have appeared which are said to objectively compare video quality. The objectivity of these methods arise to the fact that human interaction is avoided. The original video sequence and the impaired one (in this case the encoded/decoded video) are feed to an algorithm which calculates the distortion (figure 2.12).

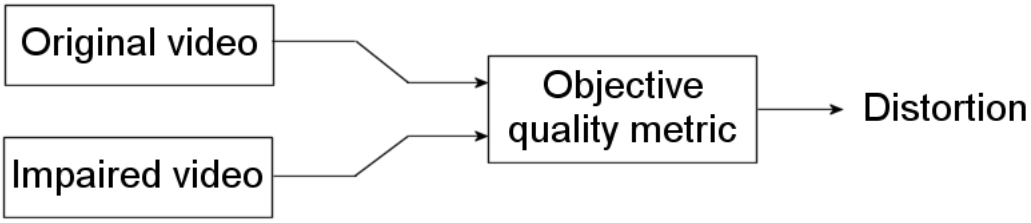


Figure 2.12: General scheme for an objective quality measurement method

This model was initially applied to still images and later extended to video sequences by simply applying the image quality metric to every frame of the video sequence.

The simplest method that compiles to this model is the Root Mean Square Error (RMSE) which calculates the *distance* between two images. RMSE can be applied to digital video by averaging the results over each frame. For an $M \times N$ original image $f(m,n)$ and impaired image $f^{'}(m,n)$, RMSE can be calculated using equation 2.3.

$$RMSE = \sqrt{\frac{1}{M \times N} \sum_{M=0}^{M-1} \sum_{N=0}^{N-1} [f'(m,n) - f(m,n)]^2} \tag{2.3}$$

Signal to Noise Ratio (SNR) is the simple mathematical evolution of RMSE and it is defined by equation 2.4.

$$SNR = \frac{\frac{1}{M \times N} \sum_{M=0}^{M-1} \sum_{N=0}^{N-1} [f^{'}(m,n)]^2}{RMSE^2} \tag{2.4}$$

The Peak Signal to Noise Ratio (PSNR) (equation 2.5) is a further RMSE based objective method. The value $(2^n - 1)$ is the highest possible value in the image, where n is the number of bit per

sample. PSNR can be calculated easily and quickly and is therefore widely used quality measure.

$$PSNR_{dB} = 20 \log_{10} \frac{(2^n - 1)}{RMSE} \tag{2.5}$$

However, the way that HVS perceives color and motion plays a key role in defining video quality. Calculating the difference between two images as data samples is both rather inefficient and unrealistic. In other words, the distortion calculated by an objective quality metric like RMSE might not match with the subjective perception of a human being. For example, despite a bad RMSE, the subjective quality may be not affected if the impairments occur in zones rich of spatial details which achieve a masking effect.

Moreover, RMSE and all the other methods described earlier do not take in account two important elements: Natural motion (frame rate) and variant sensitivity of the human eye to contrast and spatial/temporal details.

2.7 VIDEO COMPRESSION

Compression is a reversible process that converts data to a format that requires less space. In the case of video, it is the process of compacting or condensing a digital video sequence into a smaller amount of bits, that can be stored or transmitted in a more efficient way.

If O is the size of the original or *raw* data and C is the size of the data in compressed form, we can define $R = C/O$, known as the compression ratio.

Despite lossy compression does not allow the reconstruction of an exact replica of the original data set, a good approximation can be generated, achieving higher compression ratios than lossless techniques. Generally the accuracy of the approximation decreases as the compression ratio increases.

A device (software or hardware) that compresses data is referred to as an *encoder* or *coder*, whereas a device that decompresses data is known as a *decoder*. Furthermore, a device that acts as both coder and decoder is known as *codec*.

The success of data compression largely depends on the data itself and some data types are inherently more compressible than others. Some elements within the data are more common than others and most compression algorithms exploit this property, known as *redundancy*. The greater the redundancy within the data, the more successful the compression of the data is likely to be.

A Video compression algorithm (2.13) is based on three key techniques: Spatial compression, temporal compression and entropy coding:

- **Spatial Compression** relies upon removing redundancies within areas which own to the same frame (intra-frame).

- **Temporal compression** exploits similarity between successive frame (inter-frame), performing motion estimation and compensation [16].

- **Entropy coding** groups a set of lossless compression methods developed to increase data-packing efficiency for transmission or storage purposes.

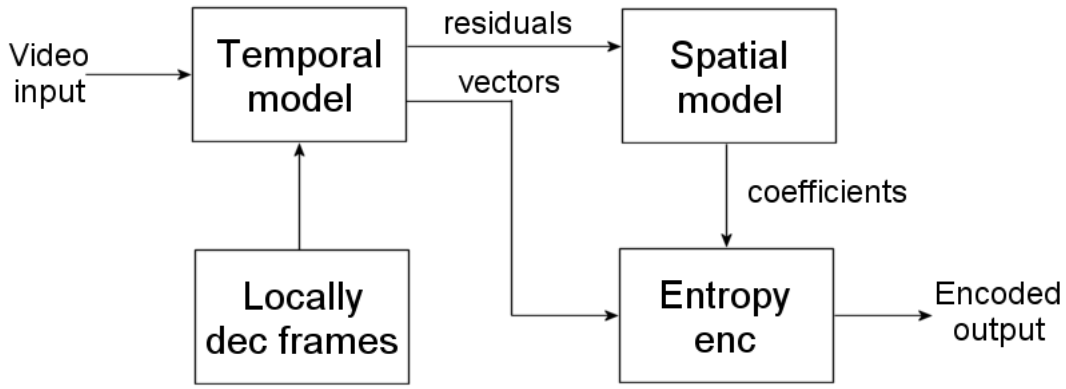


Figure 2.13: Video encoder general block diagram

2.8 SPATIAL REDUNDANCY

A digitized natural video image consists of a grid of sample values. Figure 2.14(b) shows the similarity between a natural video image (Figure 2.14(a)) and a spatially-shifted copy of itself ($2D$ auto-correlation function). The pyramid-shape graph means that similarity is maximum corresponding to a zero shift and it drops off in the case of any other position of the spatially-shifted copy. The gradual decreasing slope indicates that image samples are locally highly correlated. This means that in natural digital image, neighboring samples on a scanning line are normally similar, leading to a redundancy within the image, known as spatial redundancy.

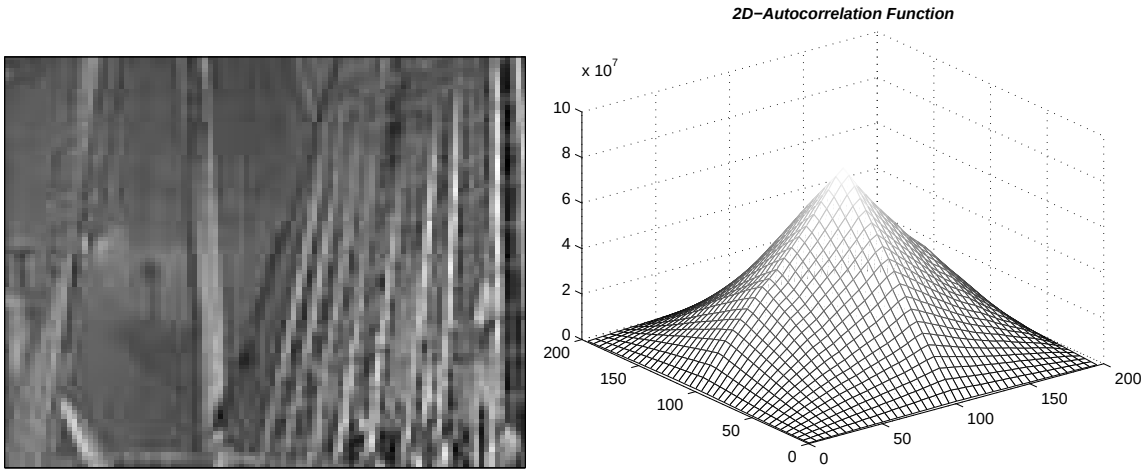
Intra-frame compression techniques (mostly applied to still images) exploit a redundancy reduction within the image and can be applied to individual frames of a video sequence. Practical intra-frame methods typically use four main techniques:

- **Prediction** - De-correlates data using previously encoded samples.

- **Transformation** - De-correlates and compacts the data using a transformed domain.

- **Quantization** - Reduces the precision of the transformed data.

- **Reordering** - Arranges the data to group together significant values.



(a) (b)

Figure 2.14: $2D$ autocorrelation function of picture sample (a)

2.8.1 Predictive Coding

Spatial prediction is sometimes described as *Differential Pulse Code Modulation* (DPCM) [17, 18], a term borrowed from a method of differentially encoding PCM samples in telecommunication systems. As for motion compensation, DPCM generates a prediction and subtracts this prediction to the current frame to form a residual. If the prediction is successful, the energy in the residual is lower than in the original frame and information can be represented with fewer bits. DPCM is a serial algorithm which exploits a pixel by pixel prediction that is often performed using a square window as in figure 2.15.

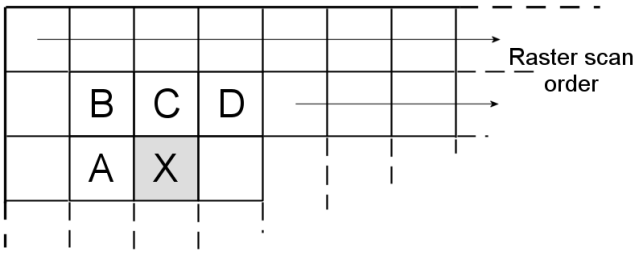


Figure 2.15: Spatial prediction pixel window

The encoder forms a prediction X for the current pixel P based on some linear combinations of previously-coded pixels: A, B, C and D (neighboring pixels in present and previous row) then the residual $R = P - X$ is calculated and sent to the decoder. The same prediction X is formed on decoder side and the current pixel is simply obtained as $P = R + X$.

If the encoding process is lossy then the decoded pixels $(A^{'}, B^{'}, C^{'}, D^{'})$ may not be identical to the original (A, B, C, D) and so the process could lead to a cumulative mismatch or *drift*.

To avoid this problem, the encoder must itself decode the residuals and reconstruct each pixel, computing the prediction on the same pixel values that will be available at decoder side.

The efficiency of this method relies on the prediction accuracy. However it is usually not possible to choose a predictor that works well in all conditions and better performance may be obtained by an adaptive prediction scheme, where different predictors are selected depending on the local statistics of the image (flat, vertical, horizontal textures). The disadvantage is that the encoder may have to send additional information to indicate the choice of predictor to the decoder and so there is a trade-off between efficient prediction and the extra bits required.

2.8.2 Transform Coding

The transform stage of a full image or a residual-frame is a reversible process that converts the data into another domain (the transformed domain) where the correlation is lower and compression is easier. We can distinguish between two categories of transform:

- **Block based** - A whole frame or residual-frame is divided into many squared $N \times N$ blocks each of one is converted into a similar block of coefficients in the transform domain. Block transforms have low memory requirements and are well-suited to be used with block-based motion compensation residuals, although it usually suffers of *blockiness*, an artefact at block edges. Examples of block-based transforms are the Karhunen-Loeve Transform (KLT), Singular Value Decomposition (SVD) and the well known Discrete Cosine Transform (DCT).

- **Image based** - Image-based transform operates at full image or frame level (or a large section of the image often known as *slice*). Image transforms such as the Discrete Wavelet Transform (DWT) have been shown to out-perform block-based transform for still image compression but they have higher memory requirements and do not *fit* well with block-based motion compensation [19].

DCT

The most popular transform used in image coding is the Discrete Cosine Transform (DCT, the sampled version of the cosine transform [5, 4]), extensively used in two-dimensional form in many MPEG standards. DCT and inverse DCT equations are shown in 2.6 2.7, where X_{ij} is a matrix of video samples and Y_{xy} the related matrix of coefficients. Because transformation of large images can be prohibitively complex, it is common to decompose the image into smaller square blocks

X_{ij}^k and code each one separately.

$$Y_{xy} = C_x C_y \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} X_{ij} \cos \frac{(2j+1)y\pi}{2N} \cos \frac{(2i+1)x\pi}{2N} \tag{2.6}$$

$$X_{ij} = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} C_x C_y Y_{xy} \cos \frac{(2j+1)y\pi}{2N} \cos \frac{(2i+1)x\pi}{2N} \tag{2.7}$$

$$C_k = \begin{cases} \sqrt{\frac{1}{N}} & (k=0) \\ \sqrt{\frac{2}{N}} & (k>0) \end{cases} \tag{2.8}$$

MPEG exploits a decomposition into 8×8 pixel blocks, which can be fully represented by a weighted sum of 64 DCT basis functions (Figure 2.16), where the weights are just the corresponding DCT coefficients (Y_{xy}) [5]. Since the transform requires multiplication by fractions (Equation 2.6), an higher resolution is needed, resulting in coefficients that have longer word-length than the pixel samples. Typically, each 8-bit samples results in a 11-bit coefficient. Although this process does not in itself result in compression, the DCT coefficients, when read in an appropriate order, tend to be good candidates for compression using run length encoding or predictive coding.

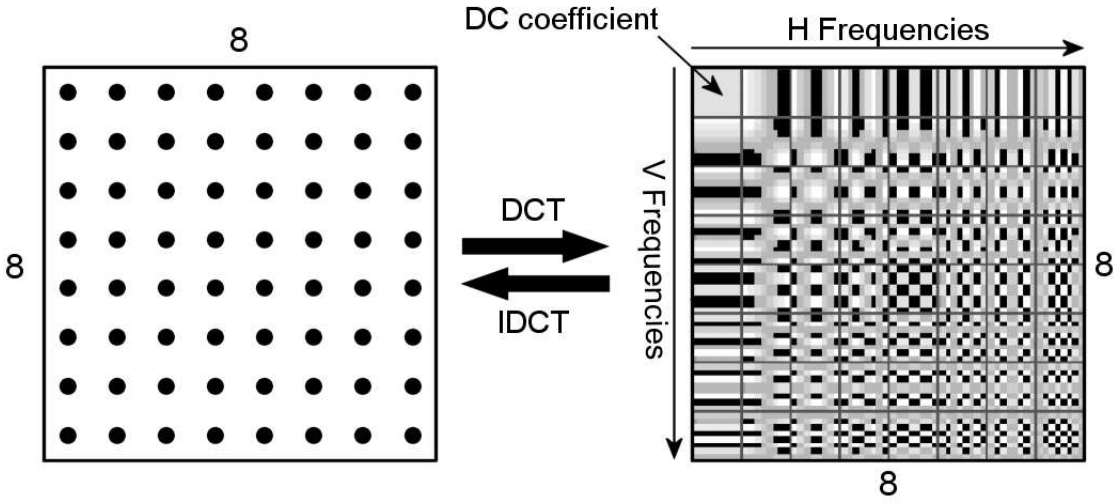


Figure 2.16: 8×8 DCT basis patterns

Figure 2.16 shows the results of an inverse DCT of each of the individual coefficients of an 8×8 transformed block. In the case of the Y signal, the top-left coefficient is the average brightness or DC component of the whole block. Moving across the top row and down the left column, horizontal and vertical spatial frequencies respectively increase. In real pictures, different vertical and horizontal spatial frequencies may occur simultaneously. Using the coefficients in the DCT block, all possible combinations can be achieved.

Thus combining the 64 coefficients of the 2-D DCT will result in the original 8×8 pixel block. In the case of color pictures, each Y, Cb and Cr component is organized into separated 8×8

blocks which are individually transformed.

In much real program material, many of the coefficients will have zero or near-zero values and, therefore, will not be transmitted. This fact results in significant compression that is virtually lossless. If a higher compression factor is needed, then a quantization step is required. The most useful property of DCT coded blocks is that the coefficients can be coarsely quantized without seriously affecting the quality of the image that results from an inverse DCT. However, this quantization will reduce accuracy of the coefficients and will introduce losses into the process. With care, the losses can be introduced in a way that is least visible to the viewer.

DWT

All transforms suffer from uncertainty because the more accurately the frequency domain is known, the less accurately the time domain is known (and vice versa). In most transforms such as Discrete Fourier Transform (DFT) [20] and Discrete Cosine Transform (DCT), the block length is fixed, so the time and frequency resolution is fixed. The frequency coefficients represent evenly spaced values on a linear scale. Unfortunately, because human senses are logarithmic, the even scale of the DFT and DCT gives inadequate frequency resolution at one end and excess resolution at the other.

The wavelet transform [9, 5] is not affected by this problem because its frequency resolution is a fixed fraction of an octave and therefore has a logarithmic characteristic. This is achieved by changing the block length as a function of frequency. As frequency goes down, the block becomes longer. Thus, a characteristic of the wavelet transform is that the basis functions all contain the same number of cycles, and these cycles are simply scaled along the time axis to search for different frequencies. Figure 2.17 contrasts the fixed block size of the DFT/DCT with the variable size of the wavelet.

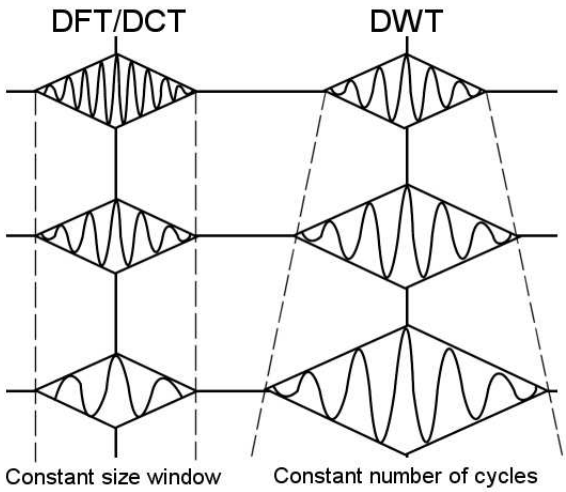


Figure 2.17: DFT, DCT and DWT frequency resolution

Wavelets are especially useful for audio coding because they automatically adapt to the conflicting requirements of the accurate location of transients in time and the accurate assessment of pitch in steady tones. For video coding, wavelets have the advantage of producing resolution scalable signals with almost no extra effort.

In moving video, the advantages of wavelets are offset by the difficulty of assigning motion vectors to a variable size block, but for intra-coding applications, this is not an issue. Wavelet coding has shown particular benefits for very-low bit-rate applications, in which the artifacts generated by excessive quantization of wavelet coefficients generally appear as *smearing*, which are much less objectionable than the *blockiness* that results from excessive quantization of DCT coefficients.

2.8.3 Quantization

Quantization is the process of mapping a signal with a rage of values X to a quantized signal with a reduced range of values X_q. In this way the quantizer signal is represented with fewer bits than the original. Quantization is not a reversible process, so the original signal can not be recovered from the quantized one without error (quantization error). A *scalar quantizer* maps one sample to one quantized output value and a *vector quantizer* maps a group of input samples (a *vector*) to a group of quantized values [21]. Equation 2.9 shows the process of rounding a fractional number to the nearest integer, which is a classical example of scalar quantization. Q is the quantization step and $\triangle Q$ the quantization error.

$$\begin{aligned} X_q &= round\left(\frac{X}{Q}\right) \\ X &= X_q \cdot Q + \triangle Q \end{aligned} \tag{2.9}$$

To quantize a 8×8 block of DCT coefficients, MPEG standards make use of weighting tables. The quantization step for each DCT coefficient is specified in the corresponding position of a *quantizer matrix* as expressed in equation 2.10, where $i, j = 1$ to 8.

$$\begin{aligned} X_q[i,j] &= round\left(\frac{X[i,j]}{Q[i,j]}\right) \\ X^{'}[i,j] &= X_q[i,j] \cdot Q[i,j] + \triangle Q[i,j] \end{aligned} \tag{2.10}$$

Inverse quantization (rescaling) is used at decoder side to reconstruct the original value. The difference between actual value $X[i,j]$ and reconstructed value $X^{'}[i,j]$ is still the quantization error $\triangle Q[i,j]$. It is possible to carefully design $Q[i,j]$ in order to better preserve visual quality.

The human perception of noise in pictures is not uniform but is a function of the spatial frequency. More noise can be tolerated at high spatial frequencies, as it is effectively masked by fine detail in the picture, whereas in plain areas noise is highly visible.

As the DCT splits the signal into different frequencies, it becomes possible to control the spectrum of the noise. Effectively, low-frequency coefficients are rendered more accurately than high-frequency coefficients by a process of weighting.

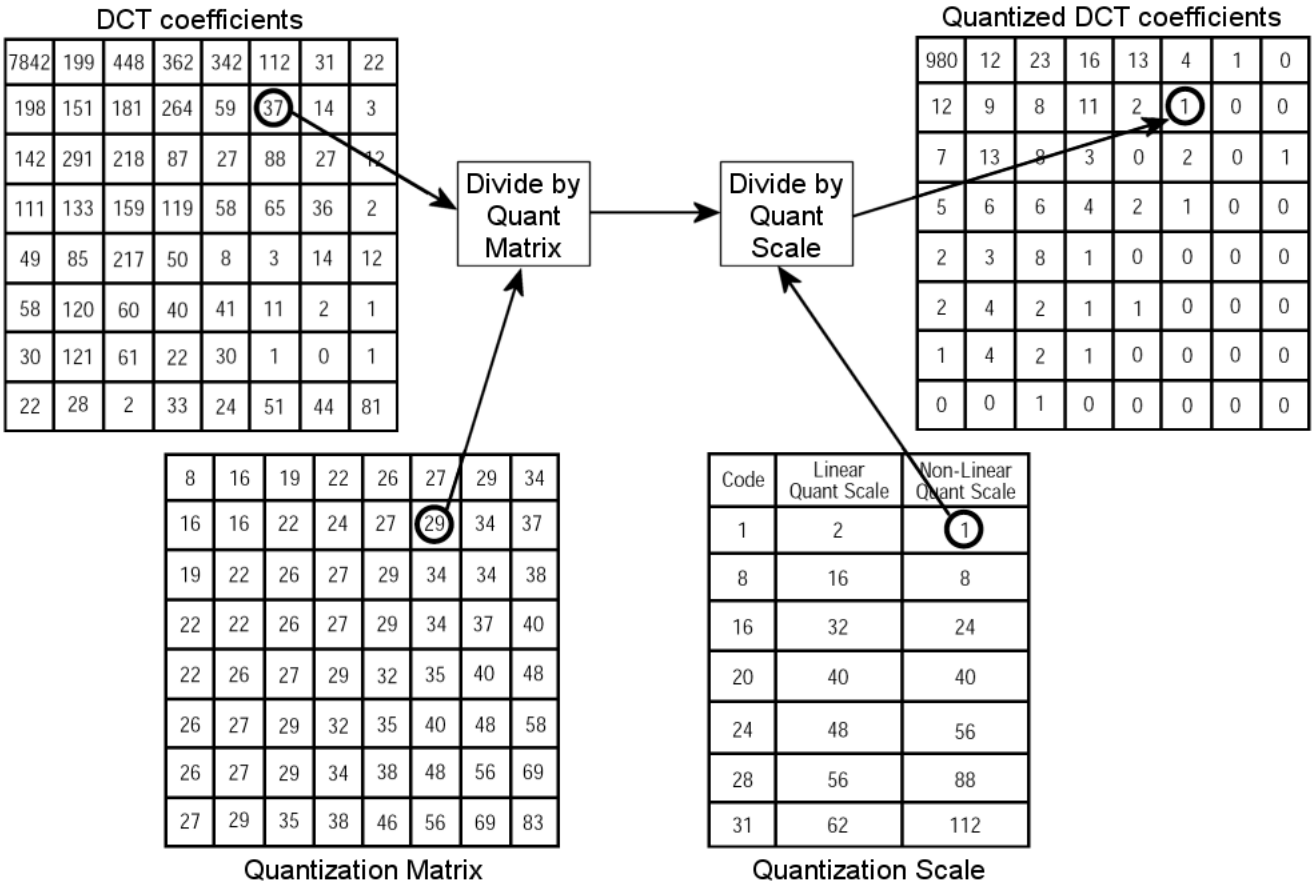


Figure 2.18: 8×8 DCT coefficients block quantization

Figure 2.18 shows the MPEG weighting process. The coefficients from the DCT are divided by constants that are a function of two-dimensional frequency. Low-frequency coefficients will be divided by small numbers, high-frequency coefficients will be divided by large numbers and the result is truncated to the nearest integer.

This truncation introduces the quantization error that increases according to the division factor. As result, coefficients representing low spatial frequencies are quantized with relatively small steps and suffer little increased noise. Coefficients representing higher spatial frequencies are quantized with large steps and suffer more noise. However, fewer steps means that fewer bits are needed and compression is achieved.

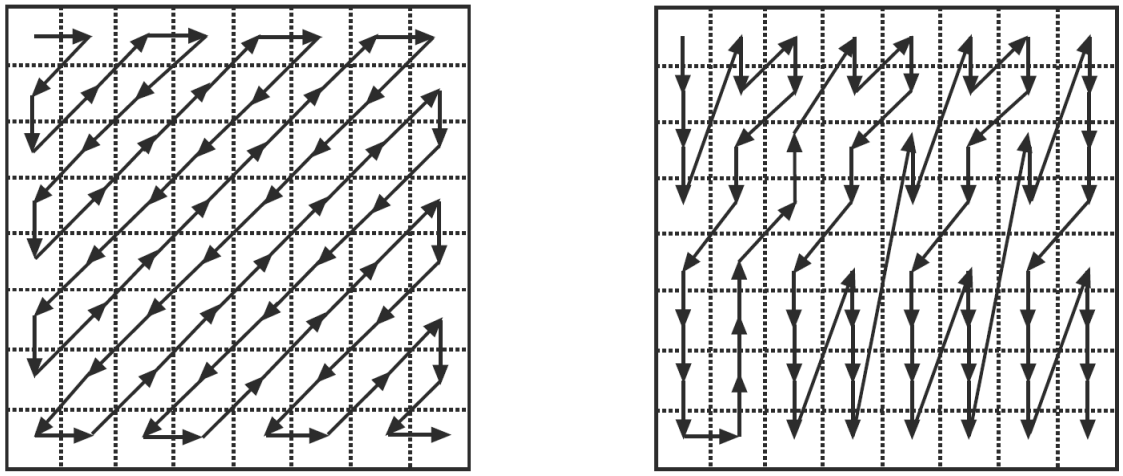
At decoder side, weighted coefficients are multiplied by inverse weighting factors to recover

their original magnitude. At high frequencies the multiplication factors will be larger, so the quantizing noise will be greater. Following inverse weighting, the coefficients will have their original DCT output values, plus a quantization error, which will be greater at high frequency than at low frequency.

The degree of compression obtained and, in turn, the output bit rate, is a function of the severity of the quantizing process. Different bit rates will require different weighting tables. Since the weighting table in use can be transmitted to the decoder, MPEG supports both standard and user-defined tables.

2.8.4 Reordering

The most significant DCT coefficients are generally found in or near the top-left corner of the matrix. After weighting, low-value coefficients might be truncated to zero. More efficient transmission can be obtained if the non-zero coefficients are sent first, followed by a code indicating that the remainder are all zero. Scanning is a technique that increases the probability of achieving this result, because it sends coefficients in descending order of magnitude probability. There are two main cases of scanning patterns, depending on wether the video sequence is progressive or interlaced.



(a) (b)

Figure 2.19: Zigzag scan order: (a) Classic, (b) Alternate

- In a non-interlaced system, the probability of a coefficient having a high value is highest in the top-left corner and lowest in the bottom-right corner. A 45 degree diagonal zigzag scan (figure 2.19(a)) is the best sequence to use here.
- In an interlaced picture, an 8×8 DCT block from one field extends over a 16×8 pixel screen area, so that for a given picture detail, vertical frequencies will appear to be twice as

great as horizontal frequencies. Thus, the ideal scan for an interlaced picture will be on a diagonal that is twice as steep. Figure 2.19(b) shows that a given vertical spatial frequency is scanned before the corresponding horizontal one.

Run-Level Coding

Run-Level Coding (RLC) is an entropy coding technique which is employed in MPEG standards. Higher-frequency DCT coefficients are often quantized to zero and so a reordered block will usually contain one or more clusters of non-zero coefficients near the start, followed by strings of zero coefficients (refer to figure 2.18).

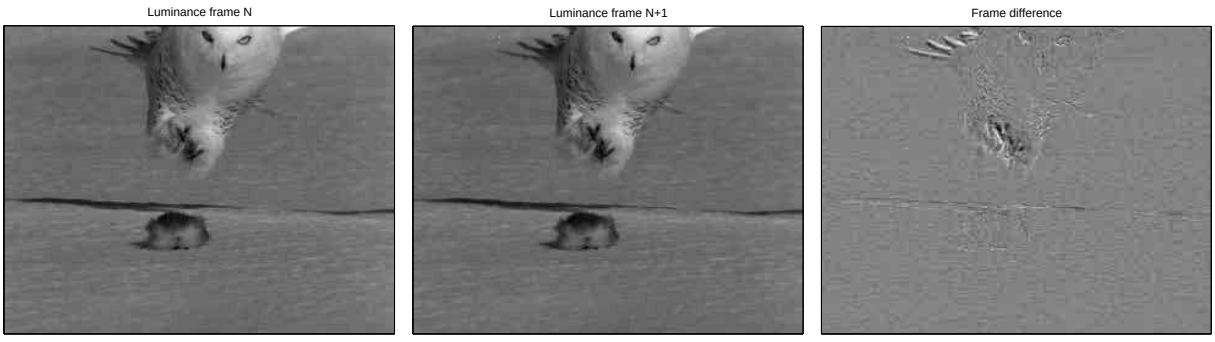
The large number of zero values are encoded by representing the array as a series of $(run, level)$ pairs where run represents the number of zero preceding a non-zero coefficient (repeated occurrence of zero) and $level$ indicates the magnitude of the non-zero coefficient. Finally, RLC transmits the number of zeros rather than each individual zero sample.

Input array:	$19, 0, 0, 0, -7, 5, 9, 0, 0, 4, 0, 0, 0, 0, -3,$ ..... $0, 0, 0, 0, 0$
Run-level:	$(0, 19)$ $(3, -7)$ $(0, 5)$ $(0, 9)$ $(2, 4)$ $(4, -3)$ ..... $(12, 1)$ EOB

A special case is required to indicate the final non-zero coefficient in a block. In two-dimensional RLC each run-level pair is encoded as in the example and a separate symbol EOB (End Of Block) indicates the end of the non-zero values.

2.9 TEMPORAL REDUNDANCY

There are several inter-frame compression techniques of various degrees of complexity, most of which rely on the general assumption that video frames which are close in time are also high spatially correlated (Figure 2.20). Therefore, when encoding a video frame, it is efficient to make as much use as possible of the information stored in previously encoded frames.



(a) (b) (c)

Figure 2.20: Example of frame difference: (c)=(b)-(a)

2.9.1 Difference Coding

Difference coding, or *conditional replenishment* method, simply considers the difference between the current frame and a previous reference frame, encoding the residual difference. When two frames are very similar, their difference will be simpler to encode than each single frame. Therefore, the reference is used as temporal estimation of the current frame and only pixels that have changed are updated.

An information overhead is associated with indicating which pixels are to be updated and if their number is large compression efficiency may be affected. At the cost of introducing some loss, two modifications can alleviate this problem somewhat:

- The intensity of many pixels will change only slightly and when coding is allowed to be lossy, only pixels that change significantly need be updated. Thus, not every changed pixel will be updated.

- Difference coding does not need to operate only at the pixel level, *block* level processing is possible. In the case of 4:2:0 sampling mode, four luminance blocks and two chrominance blocks form a *macroblock*(figure 2.21), which corresponds to a 16×16-pixel region of a frame.

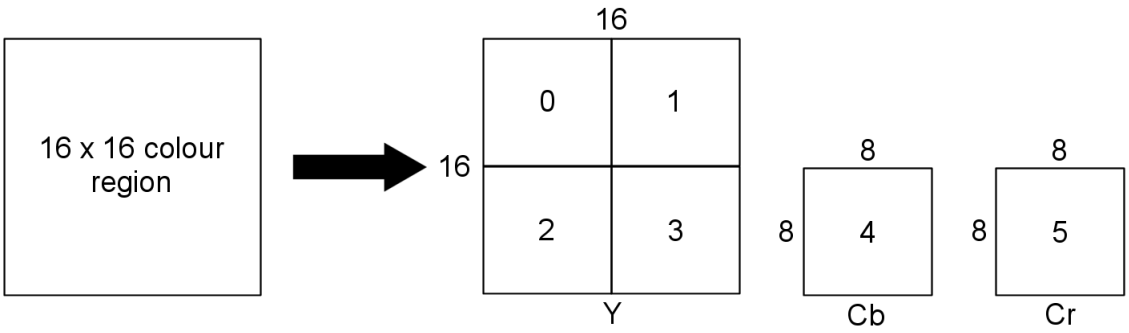


Figure 2.21: Macroblock (4:2:0)

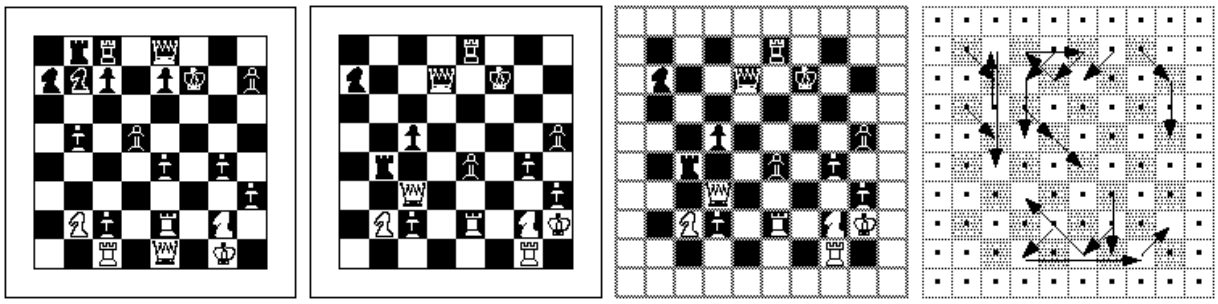
If the frames are divided into non-overlapping blocks and each block is compared with its counterpart in the previous frame, then only blocks that change significantly need to be updated. For example, if only those blocks of figure 2.20(b) which contain the bird, its shadow and the prey are updated, the resulting image might be an acceptable substitute for the original.

Updating whole blocks of pixels at once reduces the overhead required to specify where updates take place. The 320×240 pixels in the frame of figure 2.20(a) can be split into 1200 blocks of 8×8 pixels. Significantly fewer bits are required to address one of 1200 blocks than one of 76800 individual pixels. However, some pixels will be updated unnecessarily and discontinuities might be visible especially if large blocks are used. The suitable block size is a critical parameter which must achieve a trade-off between image quality and compression efficiency.

The obvious problem with difference coding prediction (even if block based) is that only objects that remain stationary within the image can be effectively coded. If there is a lot of motion or indeed if the camera itself is moving (even a very slow pan), then very few pixels will remain unchanged and a lot of energy remains in the residual frame (light and dark areas in figure 2.20(c)). This means that there is still a significant amount of information to compress after temporal prediction and much of the residual energy is due to object moving. Therefore, a better prediction may be formed by *compensating* for motion between the two frames.

2.9.2 Block Based Motion Estimation and Compensation

Considering two consecutive frames divided into uniform non-overlapping blocks, as shown in figures 2.22(a)(b), most of the motion that blocks undergo between frames is a translational motion. This assumption forms the based of the Motion Estimation (ME), which attempts to find for each block in the current frame, the best matching block in the previous frame, also known as target block.



(a) (b) (c) (d)

Figure 2.22: Motion vectors estimation

The motion of a block is represented by a Motion Vector (MV) that has two components, the first indicating horizontal displacement, and the second indicating vertical displacement. If the target block and matching block are found at the same location in their respective frames then the motion vector that describes their difference is known as a *zero vector*. The illustration in figure 2.22(d) shows the motion vectors that describe where blocks in the current frame can be found in past frame.

A search window is usually defined and bounds the area within which the encoder can perform the search for the best matching block (figure 2.23). Different criteria could be used to measure the similarity between two blocks. A common approach to find the best matching block is to evaluate the Sum of Absolute Differences (SAD) at every pixel location within the specified search window. This method is called full search or exhaustive search, and is usually computationally expensive but on the other hand yields good matching results.

A more computationally efficient approach is to restrict the search to only few points in the

search area, where there is a high likelihood of finding a good match. These search points are defined by predicted MVs which are calculated on previously encoded blocks.

Once that the each target block has been found, the recovered MVs are used to generate a motion prediction frame by rigidly translating the blocks in the reference one. The difference between the prediction and the actual frame is computed to produce the motion compensated residual to be encoded. This technique is called Motion Compensation (MC)

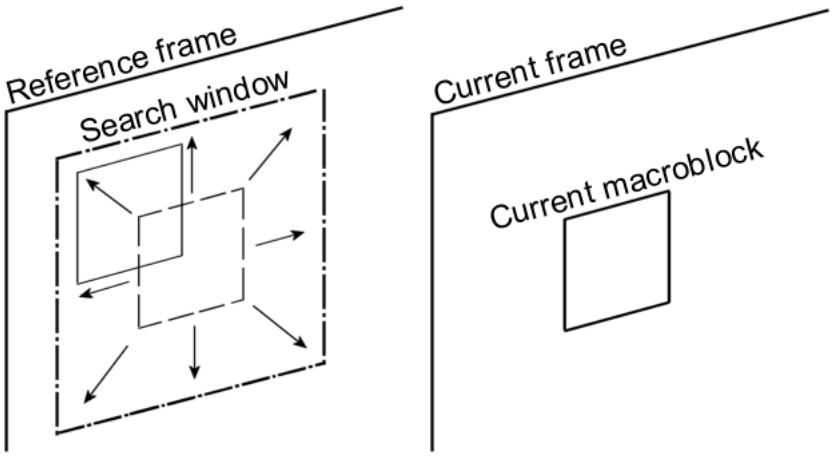


Figure 2.23: Motion estimation search window

Since fewer bits are required to code both residual blocks and MVs than actual blocks, a better compression is achieved. During decoding, the MVs are used to reconstruct the prediction frame (from the reference frame already decoded) which is added to the decoded residual to recover the actual frame.

Up to 40% of the bits transmitted by a MPEG encoder, might be employed for MVs data [22]. Fortunately, the high correlation between motion vectors and their non-uniform distribution makes them suitable for further lossless compression.

2.9.3 Bidirectional Motion Compensation

There are many variations on the basic ME and MC process. The reference frame may be a previous frame (in temporal order), a future frame or a combination of predictions from two or more previously encoded frames.

When an object moves, it conceals and reveals the background at its leading and trailing edge respectively. The revealed background requires new data to be transmitted no information can be obtained from a previous picture. A similar problem occurs if the camera pans; new areas come into view and nothing is known about them.

Bidirectional motion compensation minimizes this problem by using matching blocks from

both a past and future frame to code the current frame. If a background is being revealed, it will be present in a latter picture, and the information can be moved backwards in time to predict part of an earlier picture. Figure 2.24 shows the concept of bidirectional coding. On an individual macroblock basis, a bidirectionally-coded picture can obtain motion-compensated data from an earlier or later picture, or even use an average of both.

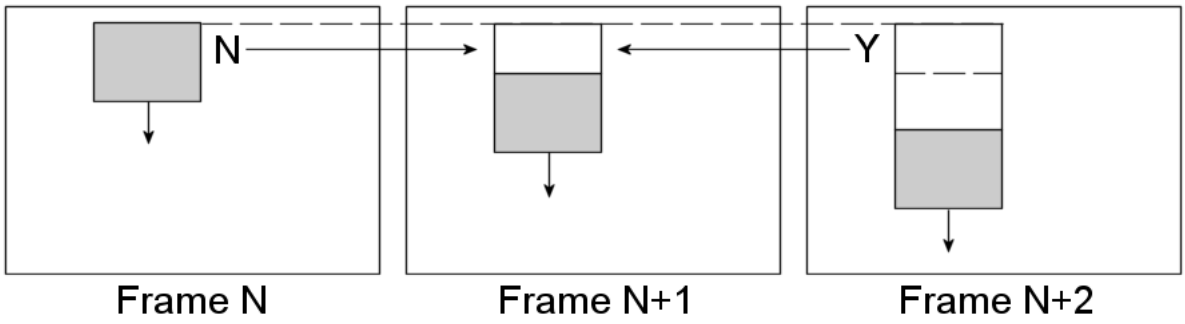


Figure 2.24: Bidirectional coding concept

Bidirectional coding is much more powerful than compression based on single past frame only, reducing the amount of residual data by improving the efficiency of possible prediction, which allows more blocks to be replaced by MVs. However, bidirectional MC requires that frames would be encoded and transmitted in a different order from which they will be displayed.

Whether there is a significant change between the reference and current frame (for example a scene change), it may be more efficient to avoid MC. Generally, an encoder may chose between *intra* and *inter* mode (encoding without or with MCP) according to the convenience for each single block. In the case of MPEG-1 and MPEG-2, the ME is performed on luminance component at macroblock (16×16 pixels) level and furthermore extended at the color component by hypothesis of correlation.

Moving objects in a natural video scene rarely follow a square 16×16-pixel boundaries and so a variable block size for ME and MC can certainly achieve better performances.

2.9.4 Variable Block Size Motion Compensation

The ME technique allows to reduce the residual frame energy by compensating each 16×16 macroblock. This energy can be further reduced by compensating each 8×8 blocks instead, and further more using 4×4 blocks. In general, smaller MC block sizes can produces better results. However, a smaller block size leads to higher complexity (more search operations) and an increase in the number of MVs to transmit. The benefit of reduced residual energy may be outweighed by the overhead of bits necessary to encode the MVs.

In order to exploit the advantages coming from the reduced block size, a trade-off between

block sizes and image content has to be found. Large block sizes can be chosen in flat homogeneous zones of a frame, while smaller block sizes can be selected around areas of high detail and complex motion. This technique is the Adaptive Motion Compensation Block Size (AMCBS), which has been introduced in H.264 codec [5].

2.9.5 Sub-pixel Motion Compensation

The motion of objects in the real world is continuous and does not necessarily match the sampling grid points after digitization of the scene. In some cases a better motion compensated prediction may be formed by predicting from interpolated sample positions in the reference frame [5].

Usually sub-pixel ME adopts a multi-resolution search approach, which achieves a significant saving in the number of operations over the full search method. The estimated integer-pixel displacement is obtained first through the classical ME method. Then the images are interpolated (in the area of interest only) and a finer search at sub-pixel resolution around this integer position is performed on these interpolated data. In the case of half-pixel resolution, we have to choose the best within 8 different displacements around the integer position.

In general, *finer* interpolation provides better MC (lower residual energy) at the expense of an increased complexity. In fact, sub-pixel ME requires more bits to code the MVs displacement fields, as their resolution is higher. Since a more accurate MC approach requires more bits to encode the MVs and fewer bits for the residual, whereas a less accurate MC requires fewer bits for the MVs and more bits for the residual, a trade-off between MC accuracy and compression efficiency have to be taken into account.

Sub-pixel MC is commonly used at *half-pixel* accuracy, as *quarter-pixel* is considered to be the limit of further incremental coding gain [4].

2.10 ENTROPY CODER

Entropy encoding is a reversible lossless process which converts a series of symbols associated to elements of the video sequence into a compressed bitstream suitable for transmission or storage [5, 4]. Input symbols may include quantized transform coefficients (previously run-level coded), MVs (a $[x, y]$ displacement vector for each motion compensated block with specified, integer or sub-pixel, resolution), markers (codes used to indicate a synchronization points in the sequence) headers (macroblock headers, picture headers, sequence headers, etc.) and supplementary information.

Once that a dictionary of symbols has been defined, then we can study the probability of occurrence of each particular symbol in real video sequences. In practice, some symbols will occur more often than others and this statistical information can be used to achieve further compression using Variable Length Coding (VLC). Frequently occurring symbols are converted

to short code words, and infrequent symbols are converted to long code words. However, many methods have been conceived in order to achieve entropy coding and not all of them make use of VLC.

- **Relative Encoding** - Certain symbols are highly correlated in local regions of the picture. For example the DC value of neighboring intra-coded blocks of pixels may be very similar. Neighboring MVs may have similar $[x, y]$ displacements and so on. Coding efficiency may be improved by predicting elements of the current block or macroblock from previously-encoded data and encoding the difference between the prediction and the actual value.

  For example, using Differential Pulse Code Modulation (DPCM), the stream 1 5 1 0 6 4 3 3 0 0 3 would be transmitted as $1 + 4 - 4 - 1 + 6 - 2 - 1 + 0 - 3 + 0 + 3$. DPCM is a predictive coding method (ref. to 2.8.1) and in the example, the encoder uses the previous value as prediction for the next one. The performances of a DPCM-based algorithm are strongly dependent on the efficiency of the achieved prediction.

- **Run-Level Coding** - Refer to section 2.8.4.

- **Huffman Coding** - Huffman coding is a classic compression technique that assigns variable length codes (VLC) to symbols, so that the most frequently occurring symbols have the shortest codes [23]. On decompression the symbols are reassigned their original fixed length codes.

  For example, in text compression applications, variable length codes are used in place of ASCII codes, and the most common characters, usually *space*, *e* and *t*, are assigned the shortest codes. In this way the total number of bits required to encode the data can be considerably less than what a fixed length representation achieves. Huffman coding is particularly effective where the data are dominated by a small number of symbols.

- **Arithmetic Coding** - Although Huffman coding is very efficient, it is only optimal when the symbol probabilities are integral powers of two. Arithmetic coding [24] does not have this restriction and is usually more efficient than the more popular Huffman technique. Although more efficient than Huffman coding, arithmetic coding is more complex.

- **Lempel-Ziv Coding** - Lempel-Ziv compressor [25] use a dictionary of symbol sequences. When an occurrence of the sequence is repeated it is replaced by a reference to its position in the dictionary. There are several variations of this coding technique, which are characterized by different dictionary management approaches. The most well known of these techniques is the Lempel-Ziv-Welch variation [26].

2.11 DPCM/DCT VIDEO CODEC MODEL

Most of video coding standards released since the early 1990s have been based on the same generic model of video CODEC which includes a motion estimation and compensation front end, a transform stage and an entropy encoder. The model is often described as a hybrid motion-compensated DPCM/DCT CODEC. Any CODEC that is compatible with H.261, H.263, MPEG-1, MPEG-2, MPEG-4 Visual and H.264 has to implement a similar set of basic coding and decoding functions.

Figure 2.25 shows the block diagram of a generic DPCM/DCT encoder and decoder that will be a useful reference for the discussion of specific video coding standards. There are two main data flow in the encoder, left to right (encoding) and right to left (reconstruction). This means that the encoder has at least twice the decoder complexity.

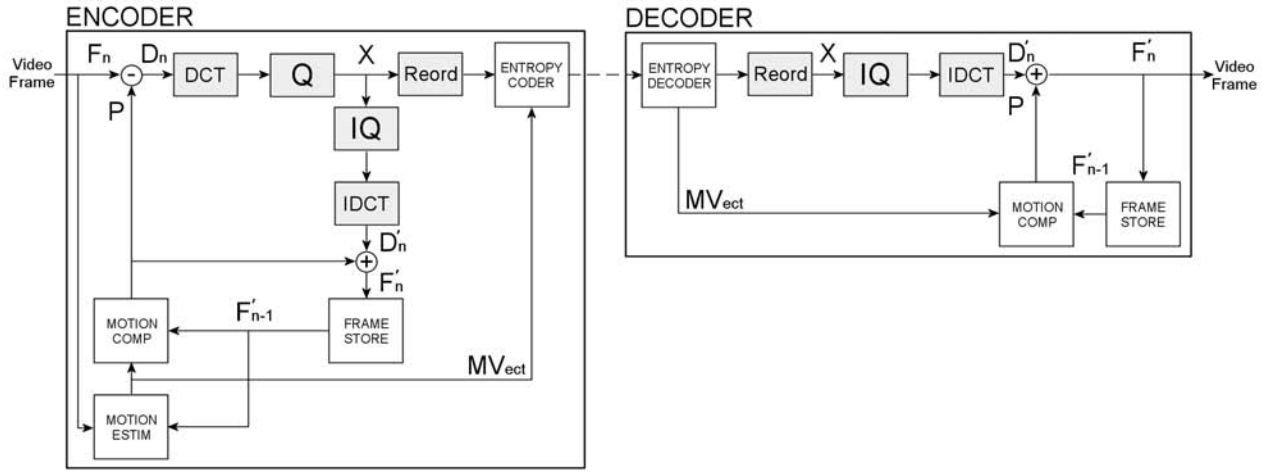


Figure 2.25: DPCM/DCT video encoder/decoder

- **Encoding** - The frame buffer contains the reference frame for temporal prediction: F'_{n-1}, a reconstructed copy of the previously-encoded frame. At the input there is the current video frame F_n, ready to be encoded in units of a macroblock (corresponding to 16×16 Y region and associated Cr, Cb samples). The motion estimator calculates MVs for each block in the current frame. P, a motion-compensated version of F'_{n-1} is generated and subtracted from the current frame F_n to create a residual error frame D_n. Each block of this difference frame is then DCT transformed. The DCT coefficients are then quantized, reordered, entropy coded and transmitted together with the MVs (entropy coded also).

- **Local decoding** - At the same time, to ensure that the encoder and decoder use identical reference frames for motion compensation, the quantized DCT coefficients X are scaled back by the quantization step size (see the rescaling block (IQ) in Figure 2.25) and inverse transformed (IDCT) to create the reconstructed residual D'_n and $F'_n = D'_n + P$, a local copy

of the reconstructed current frame. $F_n^{'}$ will be used as the prediction reference for the next frame.

- **Decoding** - The coded macroblocks are first entropy decoded, inverse-reordered and MVs are extracted. Then the DCT coefficients are scaled back by the quantization step size and put through inverse DCT transform. This gives back the reconstructed difference frame $D_n^{'}$. A motion-compensated reference frame P is created using the previous decoded frame $F_{n-1}^{'}$ and the MVs for the current frame $F_n^{'}$, that is simply reconstructed as $F_n^{'} = D_n^{'} + P$. This frame is displayed and is also stored in the decoder frame buffer to be used in the decoding of the next frame.

The encoding/decoding data flow described before does not cover all the cases. In the example, the motion compensated prediction P is generated using the previous reconstructed frame $F_{n-1}^{'}$. Using bidirectional MC (ref. to 2.9.3) a prediction B could be generated using two reference frames (past and future) and it is even possible not to use MC at all, exploiting intra-coding only. So a generic motion-compensated DPCM/DCT CODEC can be described as three CODECs, each one handling a different frame type:

- **I-frame** - It is encoded as a single image, with no reference to any past or future frames. The encoding scheme used is similar to JPEG compression. Each 8×8 block is encoded independently with the only exception of DC coefficient that is encoded relative to the DC coefficient of the previous block (DCPM coding). The block is first transformed from the spatial domain into a frequency domain using the DCT, then the data is quantized. Quantization is the only lossy part of the whole compression process. The resulting data is then run-length encoded in a zigzag ordering and entropy encoded to increase compression.

- **P-frame** - It is encoded relative to the past reference frame. A reference frame can be a P frame or I frame that has already been reconstructed. Each macroblock in a P frame can be encoded either as an I macroblock or as a P macroblock. An I macroblock is encoded just like a macroblock in an I frame. A P macroblock is encoded using MC as in example of figure 2.25. MVs may include half-pixel values, in which case pixels are interpolated. The residual term is encoded using the DCT, quantization, and run-length encoding. A P macroblock may also be skipped when it is related to a zero MV and an all-zero error term.

- **B-frame** - It can be encoded relative to the past or future reference frame, or both frames. The future reference is always the closest following I or P frame. The encoding for B frames is similar to P frames, except that MVs may refer to areas in the future reference frames. For B macroblocks that use both past and future reference frames, both of 16×16 corresponding areas are averaged.

In a coded video stream, the absolute image data (I frames) are interleaved with frames containing residual data (P frames and B frames). The starting I frame and all the P and B frames prior to the next I frame are called a Group Of Pictures (GOP). The larger the number of P and B frames in a GOP, the higher compression ratio is achieved. However, a long GOP can cause a long delay before the picture can recover from a transmission error. Moreover, a long GOP makes video authoring and editing problematic, as the video stream can only be manipulated at I frames.

2.12 TIME STAMPS

P pictures which have been used as reference frames by bidirectional coding will be decoded some time before the presentation, as they are needed in the B picture decoding process. To simplify B picture handling, the encoded sequence transmission order is different by the display order. For example, pictures that have to be presented in the order IBBP, will be transmitted as IPBB, according to the decoding order.

To assure a correct decoding process, MPEG exploits a simple mechanism based on time stamps, which are 33-bit samples taken from the System Time Counter (STC), driven by the 27 MHz reference encoder clock. Time stamps are periodically incorporated in each picture during encoding process, in order to communicate at decoder side when a picture have to be decoded and when presented. Two types of time stamps are therefore employed.

- **DTS** - The decoding time stamp indicates the time when an encoded picture should be decoded and stored in the frame buffer.
- **PTS** - The presentation time stamp indicates when a picture should be presented to the decoder output to be displayed.

B pictures are decoded and presented simultaneously so PTS only is needed. When an IPBB sequence is received, both I and P pictures are decoded before the first B picture. A decoder can only process one picture at a time; therefore the I picture is decoded first and then stored. While the P picture is being decoded, the I picture can be displayed. Finally, both I and P pictures have to be kept in the frame buffer till the related B pictures have been received and decoded.

However, the time stamp approach is efficient only if both encoder and decoder can access a common time reference. In other words, a synchronization method is needed.

To provide the synchronization between encoder and decoder, a further time stamp called Program Clock Reference (PCR) is used (Figure 2.26). The decoder generates its own STC based on its 27 MHz program clock. When the decoder receives a PCR_{enc} which is different from the local PCR_{dec}, then a drift compensation mechanism is activated.

PCR_{enc} time stamp is periodically sent to decoder side, where the difference $\Delta = PCR_{enc} - PCR_{dec}$ controls a feedback loop based on the VCXO shown in Figure 2.26, which generates the 27 MHz decoder reference clock. The feedback reaction makes $\Delta \approx 0$ varying the decoder reference

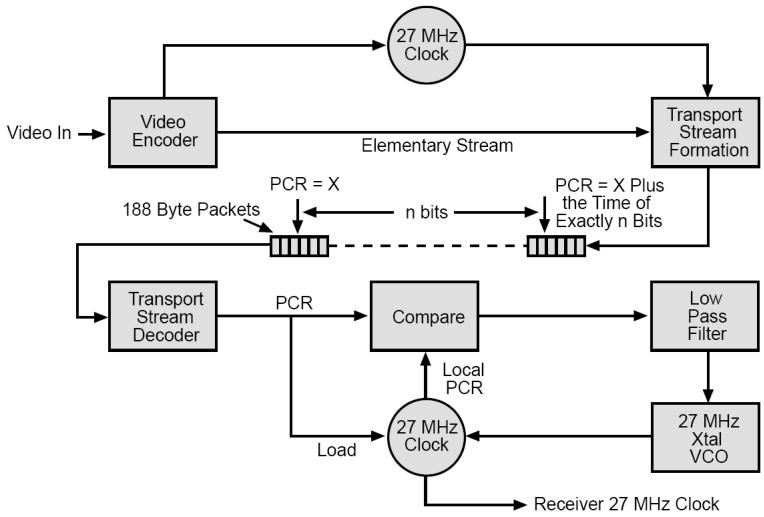


Figure 2.26: PCR based synchronization scheme between encoder and decoder

clock frequency in order to follow the encoder one. This delicate method allows both encoder and decoder to use the same synchronous time reference with the further result that video and audio streams can be synchronized together to make a program.

2.13 RATE CONTROLLED ENCODER

The DCT coefficient quantization, run-level and VLC coding produce a Variable Bit Rate (VBR) which depends on the complexity of the picture information and the amount and type of motion in the sequence. To achieve a Constant Bit Rate (CBR), which is required for transmission over a fixed bandwidth system, a buffer is employed to smooth out the bit rate variations. In order to achieve the requested CBR avoiding critical conditions of buffer overflow and underflow, a control feedback is applied to the coding process.

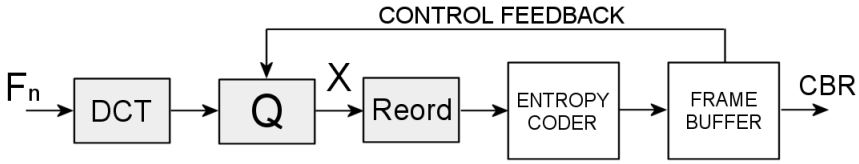


Figure 2.27: Rate controlled DPCM/DCT video encoder

The DCT quantization is often used to provide direct control of the buffer occupancy. As the buffer becomes full, the quantizer is made coarser to reduce the amount of bits used to encode each DCT coefficient. Viceversa, the DCT quantization is made finer as the buffer empties. Figure 2.27 shows a block diagram of a basic DCT spatial encoder with the buffer occupancy controlled

by the feedback on the DCT coefficients quantization block.

In principle, the feedback mechanism tunes the encoding quality in order to achieve the requested CBR. Therefore, reducing the output bit rate results in a lower quality of the encoded pictures. In a rate-constrained codec scheme, the encoding quality is a direct consequence of the output CBR and the buffer management method. For example, to transmit more TV channels into the same RF band, the output bit rate of an MPEG video encoder can be reduced to meet this possible requirement, at the price of a reduced video quality (e.g. by transmitting VHS-quality instead of broadcast quality).

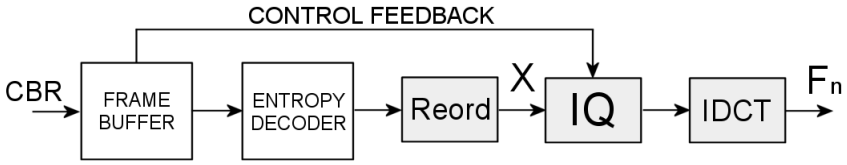


Figure 2.28: Rate controlled DPCM/DCT video decoder

At decoder side, a buffer of the same size is used to exactly reproduce the same VBR stream which was output at encoder side before buffering, which is required in order to correctly decode the video sequence. The encoding quality information are sent with the bitstream at decoder side and are used to achieve the same feedback control and therefore a correct decoding process.

If encoder and decoder are synchronized (which in MPEG is achieved with the time stamps mechanism), then it is possible to prove that the sum of both buffer occupancies is a constant B equal to the buffer size [27], as expressed in equation 2.11, where $B_e(t)$ is the encoder buffer fullness at time t, $B_d(t)$ is the decoder buffer fullness at time $t + D$ and D is a delay time needed to reach a stable state.

$$B_e(t) + B_d(t + D) = B \tag{2.11}$$

$$0 \leq B_e(t) \leq B \tag{2.12}$$

$$B \geq B_d(t + D) \geq 0 \tag{2.13}$$

In this case, if the encoder buffer is well controlled (no underflow or overflow conditions), the equation 2.12 is satisfied, and using the assumption 2.11, we find out equation 2.13 as direct consequence. Equation 2.13 simply means that the decoder buffer occupancy is always limited by the buffer size B and the empty level. Therefore, if the requested encoding bit rate does not cause critical situations of overflow ($B_e(t) > B$) or underflow ($B_e(t) < 0$) in the encoder buffer, a good behavior is expected at decoder side too.

2.14 CONCLUSIONS

The video digitizing and video coding concepts described in this chapter constitute the basis for understanding actual video standard coding algorithms. Motion compensated prediction, transform coding, quantization and entropy coding form the generic DPCM/DCT CODEC model that has been the main reference scheme of video compression for over 10 years.

This coding model is still the heart of MPEG-4 Visual and H.264, the most advanced coding algorithms standardized in the last years and not yet fully implemented on generic hardware platforms.

Chapter 3

State of The Art: MPEG-4 and H.264

3.1 INTRODUCTION

Over the past decade, there has been a significant interest in digital video applications. Consequently, many video coding standards have emerged that enable multiple video applications, including video telephony, video-conferencing, DVD and digital TV.

Some researches have converged in the standardization project of the Joint Video Team (JVT) formed by the joined forces of the ITU-T VCEG and ISO/IEC MPEG organizations. This project results in two international standards (MPEG-4 Video and H.264) for video coding that have the capability of approximately doubling the coding efficiency relative to the previous generation of video coding standards (MPEG-2) established just a few years ago.

The first part of this chapter gives a brief description of the video encoding evolution, since MPEG-1. Then, the main characteristics of MPEG-4 Visual [30] and H.264/MPEG-4 Part 10 [31] standards are described, giving an idea of the computational complexity and wide variety of supported applications.

3.2 ORGANIZATIONS AND STANDARDS

The standards provide the means needed to achieve interoperability between systems designed by different manufacturers, encouraging the growth of existing applications and sometimes the birth of new ones. The International Telecommunication Union Telecommunication standardization sector (ITU-T) is now one of two formal organizations that develop video coding standards, the other being ISO/IEC JTC1 (International Standardization Organization and the International Electrotechnical Commission, Joint Technical Committee number 1). The ITU-T video coding standards are called recommendations and they are denoted with H.26x (e.g. H.261, H.262, H.263 and H.26L), while MPEG-x is used for ISO/IEC standards (e.g. MPEG-1, MPEG-2 and MPEG-4).

3.2.1 ITU-T VCEG

ITU-T develops standards for telecommunication and to satisfy the then-emerging needs of the video telephony and conferencing applications. The H.261 standard, which was designed to operate at bit rates multiples of 64 kbps, is based on the hybrid block-based motion compensation and discrete cosine transform described in section 2.11 and forms the basis for all subsequent video coding standards. H.261 was initiated in 1984 and its final approval was given in 1990.

Motivated by the need for low bit rate (less than 64 kbps) video communication, the ITU-T began the work on the H.263 coding standard in 1993. Although its coding structure is based on that of H.261, H.263 supports new baseline coding methods as well as four optional modes that provide better picture quality at low bit rates with little additional complexity. Due mainly to its superior quality, H.263 became the predominant coding standard in video communication,

and it has been adopted in several network transport standards such as ITU-T H.324 (PSTN), H.320 (ISDN), H.310 (BISDN), H.323, and 3GPP. Approval for Version 1 of the H.263 standard was granted in 1995

Following the adoption of the first version of H.263, further enhancements were proposed, leading to Version 2 (H.263+) of the standard. H.263+ provides 12 new negotiable modes and additional features that improve video quality and increase robustness and functionality of the target video coding systems. Approval for Version 2 of the standard was granted in 1998. Version 3 of H.263 (H.263++) was introduced to add more features such as enhanced coding efficiency for low delay applications, enhanced error resilience for wireless video communications, and support for interlaced video sources. Version 3 of the standard was completed in 2001.

3.2.2 ISO MPEG

MPEG (Moving Pictures Expert Group) started in 1988 as a Working Group of the International Standards Organization (ISO) with the aim of defining standards for digital compression of video and audio signals. It took as its basis the ITU-T standard for video-conferencing and video-telephony H.261, with that of JPEG (Joint Photographic Experts Group) which was initially developed for compressing still images such as electronic photography.

It is important to note that the MPEG standards specify only the syntax and semantics of the bit-streams and the decoding process; they do not specify the encoding process. Much of the latter is left to the discretion of the coder designers and this gives scope for improvement as coding techniques are refined and new techniques developed.

Each MPEG standard is backward compatible with previous ones. For example MPEG-2 was engineered so that any MPEG-2 decoder will play back an MPEG-1 stream, ensuring a side-grade path for users who enter into MPEG with the lower priced MPEG-1 encoding hardware.

MPEG-1

The first goal of MPEG was to define a video coding algorithm for digital storage media; in particular, for the CD-ROM. The resulting standard was MPEG-1, published in 1994 [28]. MPEG-1 is restricted to non-interlaced video formats and it is primarily intended to support video coding of Standard Interchange Format (SIF, established as 352×240 pixels NTSC and 352×288 pixels PAL) at bit-rates up to about 1.5 Mbps. The target application is the compression of video and audio for CD playback, that forms the basis for Video CD (VCD) and MPEG-1 Layer-3 (MP3) formats.

MPEG-2

In 1990, MPEG began working on MPEG-2 [29], with the aim to extend MPEG-1 capabilities to the coding of interlaced pictures. The MPEG-2 standard, established in 1994, is designed to

support storage and broadcasting of *television-quality* video and audio at bit rates higher than 1.5 Mbps. MPEG-2 is not necessarily better than MPEG-1, since MPEG-2 streams at lower bit rates do not look as good as MPEG-1. But at its specified bit rates between 3-10 Mbps, MPEG-2 delivers true broadcast quality video at the full CCIR-601 [14] resolution (720×480 pixels NTSC and 720×576 pixels PAL).

MPEG-2 supports high definition formats (HDTV) at bit rates in the range of 15 to 30 Mbps, and has also received a lot of attention because it is the standard specified for DVD. The primary users of MPEG-2 are broadcast and cable companies who demand broadcast quality digital video and utilize satellite transponders and cable networks for delivery of cable television and direct broadcast satellite.

MPEG-3 was initially intended to cover HDTV, providing larger sampling dimensions and bit rates between $20 \div 40$ Mbps. However, it was discovered that MPEG-2 can be enhanced to cover the requirements of HDTV, so the MPEG-3 standard was abandoned.

MPEG-4

The MPEG-4 specification proposed an algorithm that achieves a better compression ratio than MPEG-2 and which also works well at very low bit rates. The MPEG-4 standard was initiated in 1995. Version 1 was released in 1998 (ISO/IEC 14496) and further tools and profiles were added in two amendments to the standard finalized in Version 2 in late 2001.

MPEG-1 and MPEG-2 provide interoperable ways of representing audiovisual content, commonly used on digital media and on the air. MPEG-4 extends this to many more application areas through features like: extended bit rate range, scalability, error resilience, seamless integration of different types of *objects* in the same scene, interfaces to digital rights management systems and powerful ways to build interactivity into content.

MPEG-4 is designed for use in broadcast, interactive and conversational environments, Such as web and television, not just the one after the other, facilitating integration of content coming from both channels in the same multimedia *scene*.

MPEG-7

The increasing availability of potentially interesting audio/video material makes its search more difficult. This challenging situation led to the need of a solution to the problem of quickly and efficiently searching for various types of multimedia material interesting to the user. MPEG-7 intends to answer to this need.

MPEG-7, formally called *Multimedia Content Description Interface*, has been formalized into a standard by September 2000 and gives a standardized description of various types of multimedia information. This description will be associated with the content itself, to allow fast and efficient searching for material that is of interest to the user.

MPEG-7 does not replace MPEG-1, MPEG-2 or MPEG-4. It is intended to provide complementary functionality to these other MPEG standards, representing information about the content and not the content itself. There are many applications and domains which will benefit from the MPEG-7 standard. A few examples are:

- Digital libraries (e.g. image catalogue, musical dictionary ...)
- Multimedia directory services (e.g. yellow pages ...)
- Broadcast media selection (e.g. radio channel, TV channel ...)
- Multimedia editing (e.g. personalized electronic news service, media authoring ...)

MPEG-21

Work on MPEG-21 *Multimedia Framework* standard started at the May-June 2000 meeting in Geneva. MPEG-21 seeks to describe a multimedia framework and sets out a vision for the future of an environment where delivery and use of all content types by different categories of users in multiple application domains will be possible.

A key assumption of MPEG-21 is that every human is potentially an element of a network involving billions of content providers, value adders, packagers, service providers, consumers, and re-sellers. Thus, besides client-server-based applications, peer-to-peer networking and the resulting flexibility of user roles have been an underlying part of MPEG-21 thinking since the early days of the standardization process.

The goal of MPEG-21 can be resumed to: defining the technology needed to support Users to exchange, access, consume, trade and otherwise manipulate Digital Items (e.g. a video collection, a music album, ...) in an efficient, transparent and interoperable way.

3.2.3 JVT

The ITU-T recommendations have been designed mostly for real-time video communication applications, such as video conferencing and video telephony. On the other hand, the MPEG standards have been designed mostly to address the needs of video storage (DVD), broadcast video (broadcast TV), and video streaming (e.g., video over the internet, video over DSL, video over wireless) applications.

For the most part, the two standardization committees have worked independently on the different standards. The only exception has been the H.262/MPEG-2 standard, which was developed jointly by the Joint Video Team (JVT) which consists of members of the two organizations. Recently, the ITU-T and the ISO/IEC JTC1 have agreed to join again their efforts in the development of the emerging H.26L standard, which was initiated by the ITU-T committee.

H.26L is being adopted by the JVT because it represents a departure in terms of performance from all existing video coding standards. Figure 3.1 summarizes the evolution of the ITU-T recommendations and the ISO/IEC MPEG standards.

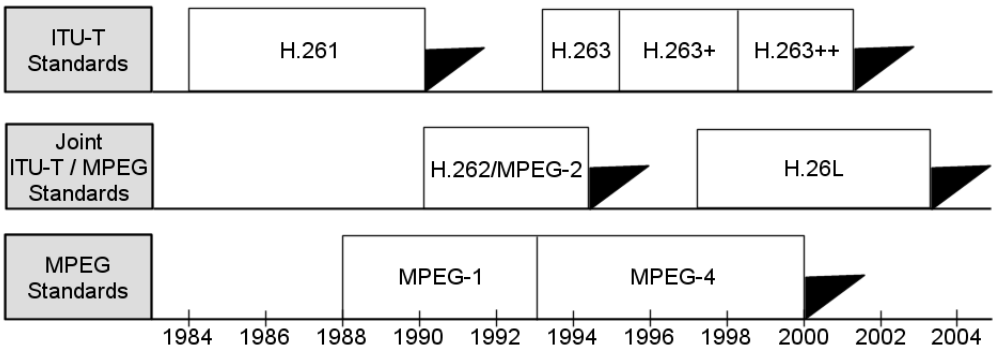


Figure 3.1: Progression of the ITU-T Recommendations and MPEG standards

Since 2001, this effort has been carried out cooperatively between VCEG and MPEG and has led to the development of a new standard, previously known as H.26L, entitled *Advanced Video Coding*. AVC is jointly published as ITU-T H.264 and ISO/IEC MPEG-4 Part 10. The main objective of H.264 standard is to provide a means to achieve substantially higher video quality compared to what could be achieved using any of the existing video coding standards.

3.3 MPEG-4 OVERVIEW

So far the industry has enthusiastically taken over MPEG-4 Video. MPEG-4 Video has been selected by several industries for a setting standard for next generation mobile communication and is being utilized to develop solutions for video on demand and related applications.

There are currently 16 Parts of MPEG-4 either published as International Standards or Technical Reports or still under development. The first 6 Parts of the standard correspond roughly to those of MPEG-2. The title of the first 5 is the same as MPEG-2, however, some significant differences of content exist. The full list of MPEG-4 parts is:

- **Part 1** - Systems: Scene description, multiplexing of audio, video and related information, synchronisation, buffer management, intellectual property management.
- **Part 2** - Visual: Coding of *natural* and *synthetic* Visual Objects (VO).
- **Part 3** - Audio: Coding of *natural* and *synthetic* audio objects.
- **Part 4** - Conformance Testing: Conformance conditions, test procedures, test bitstreams.
- **Part 5** - Reference SW: It is a complete publicly available software implementation of both encoder and decoder.

- **Part 6** - Delivery Multimedia Integration Framework: A session protocol for multimedia streaming.
- **Part 7** - Optimized Visual Reference SW: Optimized Software implementation of selected visual coding MPEG-4 tools.
- **Part 8** - Carriage of MPEG-4 over IP: Specifies the mechanism for carrying MPEG-4 coded data over Internet Protocol (IP) networks.
- **Part 9** - Reference Hardware Description: VHDL descriptions of MPEG-4 coding tools.
- **Part 10** - Advanced Video Coding: Efficient coding of *natural* video.
- **Part 11** - Scene Description and Application Engine.
- **Part 12** - ISO Base Media File Format.
- **Part 13** - Intellectual Property Management and Protection Extensions: Provides an extension to the current IPMP Framework that defines interoperability at the user level.
- **Part 14** - MP4 File Format.
- **Part 15** - AVC File Format.
- **Part 16** - Animation Framework eXtension (AFX): Animation framework will support rich 3D environment.

MPEG-4 is a standard that can be used in manifold application areas, each with their own agendas and practices concerning the business models of how technology is actually deployed. As again a number of patents are thought to be relevant for implementing the MPEG-4 standard and as MPEG is prevented by ISO rules to deal with patent issues, the MPEG-4 Industry Forum has been established as a not-for-profit organization external to MPEG with the goal of promoting the adoption of the standard and in particular creating the conditions for organizations dealing with patent licensing to be set up.

3.4 MPEG-4 VISUAL

The best way to understand MPEG-4's new paradigm is by comparing it to MPEG-2. In the MPEG-2 world, content is created from various resources such as moving video, graphics, text. After it is *composited* into a plane of pixels, these are encoded as if they all were moving video pixels. At the consumer side, decoding is a straightforward operation. MPEG-2 is a static presentation engine: if one broadcaster is retransmitting another broadcasters coverage of an event, the latter's logo cannot be removed, also for example, viewers may occasionally see the

word *live* on the screen when a broadcaster is showing third party live footage from earlier in the day. You can add graphic and textual elements to the final presentation, but you cannot delete them.

The MPEG-4 paradigm turns this upside down. It is dynamic, where MPEG-2 is static. Different objects can be encoded and transmitted separately to the decoder in their own elementary streams. The composition only takes places after decoding instead of before encoding. This actually applies for Visual Objects (VO) and audio alike, although the concept is a little easier to explain for visual elements.

This brings the interactivity of MPEG-4. All the objects can be encoded with their own optimal coding scheme: video is coded as video, text as text and graphics as graphics, instead of treating all the pixels as moving video, which they often really are not. MPEG-4 includes efficient coders for audio, speech, video and even synthetic content such as animated faces and bodies and all these coders are optimized for the appropriate data types.

3.4.1 Main Features

MPEG-4 Visual attempts to satisfy the requirements of a wide range of visual communication applications through a toolkit-based approach to coding of visual information. Some of the key features that distinguish MPEG-4 Visual from previous visual coding standards include:

- Efficient compression of progressive and interlaced *natural* video sequences. The core compression tools are based on the ITU-T H.263 standard and can out-perform MPEG-1 and MPEG-2 video compression. Optional additional tools further improve compression efficiency.

- Coding of individual *objects*. This means that the video information needs not be of rectangular shape as assumed in both MPEG-1 and MPEG-2 Video. The same applies for audio, which provides all tools to encode speech and audio at different rates and with different functionalities.

- Support for effective transmission over practical networks. Error resilience tools help a decoder to recover from transmission errors and maintain a successful video connection in an error-prone network environment and scalable coding tools can help to support flexible transmission at a range of coded bit rates.

- Coding of still *texture*. This means, for example, that still images can be coded and transmitted within the same framework as moving video sequences. Texture coding tools may also be useful in conjunction with animation-based rendering.

- Coding of animated visual objects such as 2D and 3D polygonal meshes, animated faces and animated human bodies.

- Coding for specialist application such as *studio* quality video. In this type of application, visual quality is perhaps more important than high compression.

3.4.2 Tools, Profiles and Levels

MPEG-4 consists of a large number of tools, not all of which are useful in any given application. In order to allow different market segments to select subsets of tools, MPEG-4 contains profiles, which are simply groups of coding tools. For example, the MPEG-4 Advanced Simple visual profile contains $\frac{1}{4}$ pixel motion compensation, bidirectional coding and global motion vectors, but shape coded video.

Profiles allow users to choose from a variety of toolsets supporting just the functionality they need. The MPEG-4 Visual profiles for coding *natural* video scenes are listed in Table 3.1 and these range from Simple Profile (coding of rectangular video frames) through profiles for arbitrary-shaped and scalable object coding to profiles for coding of studio-quality video. Table 3.2 lists the profiles for coding *synthetic* video (animated meshes or face/body models) and the hybrid profile that incorporates features from synthetic and natural video coding.

The concept of MPEG-2 Video Profiles has been extended to include the Visual, Audio and Systems parts of the standard, so that all the tools can be appropriately *subsetted* for a given application domain.

Profiles define a subset of coding tools and Levels define constraints on the parameters of the bitstream, so profiles exist at a number of levels, which provide a way to limit computational complexity, e.g. by specifying the resolution, the bit rate, the maximum number of objects in the scene, etc.

Table 3.3 lists the levels for the popular Simple-based profiles (Simple, Advanced Simple and Advanced Real Time Simple). The levels definitions place restrictions on the hardware complexity needed to encode/decode a given profile. For example, a multimedia terminal with limited processing capabilities and a small amount of memory may only support Simple Profile @ Level 0 bitstream decoding. Higher levels of Simple profile require a decoder capable to handle up to four Simple profile video objects (for example, up to four rectangular objects covering the QCIF or CIF display resolution).

3.5 H.264/AVC

H.264 provides tools for coding rectangular video sequences that are optimized for compression efficiency and seems to outperform the earlier MPEG-4 Visual codec, based on the older H.263 core. However the range of the available coding tools is more restricted than MPEG-4 Visual (due to narrowed focus of H.264) but there are still many possible choices of coding parameters and strategies. The main differences between H.264 and MPEG-4 Visual are summarized in table

MPEG-4 Visual profile	Main features
Simple	Low-complexity coding of rectangular video frames
Adv. Simple	Coding rectangular frames with improved efficiency and support for interlaced video
Adv. Real-Time Simple	Coding rectangular frames for real-time streaming
Core	Basic coding of arbitrary-shaped video objects
Main	Feature-rich coding of video objects
Adv. Coding Efficiency	Highly efficient coding of video objects
N-Bit	Coding of video objects with sample resolution other than 8 bits
Simple Scalable	Scalable coding of rectangular video frames
Fine Granular Scalability	Advanced scalable coding of rectangular video frames
Core Scalable	Scalable coding of video objects
Scalable Texture	Scalable coding of still texture
Adv. Scalable Texture	Scalable coding of still texture with improved efficiency and object-based features
Adv. Core	Combines features of Simple, Core and Advanced Scalable Texture profile
Simple Studio	Object-based coding of high quality video sequences
Core Studio	Object-based coding of high quality video with improved compression efficiency

Table 3.1: MPEG-4 Visual profiles for coding natural video

MPEG-4 Visual profile	Main features
Basic Animated Texture	2D mesh coding with still texture
Simple Face Animation	Animated human face models
Simple Face and Body Animation	Animated human face and body models
Hybrid	Combines features of Simple, Core, Basic Animated Texture and Simple Face Animation profiles

Table 3.2: MPEG-4 Visual profiles for coding synthetic or hybrid video

3.4.

H.264 is divided into two distinct layers: Network Abstraction Layer (NAL) and Video Coding Layer (VCL). NAL is responsible for packaging the coded data in an appropriate manner based on the characteristics of the network upon which the data will be used. On the other hand, VCL is responsible for generating an efficient representation of the video data. Hence, NAL takes care of the constraints of the underlying network and gives VCL a network independent interface.

Profile	Level	Typical resolution	Max. bit rate	Max. objects
Simple	L0	176×144	64 Kbps	1 S
(S)	L1	176×144	64 Kbps	4 S
	L2	352×288	128 Kbps	4 S
	L3	352×288	384 Kbps	4 S
Adv. Simple	L0	176×144	128 Kbps	1 AS or S
(AS)	L1	176×144	128 Kbps	4 AS or S
	L2	352×288	384 Kbps	4 AS or S
	L3	352×288	768 Kbps	4 AS or S
	L4	352×576	3 Mbps	4 AS or S
	L5	720×576	8 Mbps	4 AS or S
Adv. Real Time Simple	L1	176×144	64 Kbps	4 ARTS or S
(ARTS)	L2	352×288	128 Kbps	4 ARTS or S
	L3	352×288	384 Kbps	4 ARTS or S
	L4	352×288	2 Mbps	4 ARTS or S

Table 3.3: MPEG-4 Visual levels for Simple-based profiles

Comparison Tools	MPEG-4 Visual	H.264/AVC
Data types	Rect. video	Rect. video
	Arbitrary shaped VOs	
	Still texture and sprites	
	Synthetic or hybrid VOs	
	2D and 3D mesh objects	
Number of Profiles	19	3
Compression efficiency	Medium	High
Video streaming	Scalable coding	Switching slices
MC min. block size	8×8	4×4
MV accuracy	$\frac{1}{2}$ or $\frac{1}{4}$ pixel	$\frac{1}{4}$ pixel
Transform	8×8 DCT	4×4 integer DCT
Deblocking filter	No	Yes

Table 3.4: Comparison between H.264 and MPEG-4 Visual

3.5.1 Main Features

The VCL approach of H.264 is based on a hybrid block-based motion compensation transform-coding model 2.11, similar to that adopted in previous standards such as H.263 and MPEG-4 Visual. The main features of H.264 video coding are:

- Grouping of macroblocks into slices with flexible macroblock ordering.
- I/P/SP/SI/B slices.
- Variable block size motion prediction.
- Multiple reference frames.
- Quarter pixel accuracy motion prediction.
- Enhanced intra prediction modes.
- 4x4 Integer transform with no-mismatch.
- Advanced VLC/CABAC entropy coding techniques.
- In-loop de-blocking filter.

Motion Estimation and Compensation

From the motion estimation and compensation side, H.264 employs variable block size prediction, $\frac{1}{4}$ pixel accuracy, median and directional segmentation prediction and multiple reference frames. H.264 supports up to a maximum of 7 modes of variable block size motion prediction and a maximum of 16 motion vectors for each macroblock. This enables improved prediction thus providing higher granularity. The H.264 also supports the unrestricted motion prediction and the skip mode of operation.

Multiple Reference Frames

The H.264 standard offers the option of having multiple reference frames in inter-picture coding. Up to 6 different reference frames could be selected, resulting in better subjective video quality and more efficient coding of the video frame under consideration. Moreover, using multiple reference frames might help making the H.264 bitstream error resilient. However, from an implementation point of view, there would be additional processing delays and higher memory requirements at both the encoder and decoder.

Intra prediction

Intra prediction exploits the spatial redundancies between the adjacent blocks of a picture. So the pixel value of a block is first predicted from the adjacent blocks and the difference between the actual value and the predicted value is coded. H.264 supports 4 and 9 different intra prediction modes for macroblock and sub-blocks respectively.

Integer DCT

H.264 uses an integer DCT-like transform [32] which is an approximate form of the conventional DCT and its core part can be implemented by only adders and shifters. This eliminates the inherent transform mismatch between encoder and decoder which is otherwise introduced in conventional fixed point DCT. The block size of the transform is chosen to be 4×4 which helps reducing the blocking and ringing artifacts. H.264 also performs Hadamard transform on the DC luma and chroma component to increase the PSNR of the picture.

Quantization

In any compression, the major coding gain is attained in the quantization step. H.264 supports a total of 52 Quantization Parameters (QP) [32]. It uses non-linear quantization with an approximate increase of 12.5% in magnitude of step size from one QP to the other. The QP value of chroma is determined from that of luma. The wide ranges of quantizer step sizes make it possible for an encoder to control the trade-off between bit rate and quality in an accurate and flexible way.

De-blocking Filter

Strong motion isolation, enhanced intra prediction and arbitrary block sizes introduce visually distinguishable blocks. To solve this problem H.264 specifies the use of an adaptive de-blocking filter [33] that operates on the horizontal and vertical block edges within the prediction loop in order to remove artifacts caused by block prediction errors. The filtering is generally based on 4×4 block boundaries, in which two pixels on either side of the boundary may be updated using a 3-tap filter. Use of in-loop de-blocking filter reduces the blocking artifacts, hence improving the overall picture quality.

Switching P and B Slices

SP and SI slices [34] provide VCR-like functionality for bitstream switching, splicing, random access, fast-forward and error resilience/recovery. SP slices are designed to support switching between similar coded sequences (i.e., the same source sequence encoded at various bit rates) without the increased bit rates penalty of using I slices. A further type of switching slice, the SI slice, may be used to switch from one sequence to a completely different one, in which case it will not be efficient to use motion compensated prediction because there is no correlation between the two sequences.

Entropy Coding

In H.264, entropy coding can be performed using either a Context-based Adaptive Variable Length Codes (CAVLC) [35] or using Context-based Adaptive Binary Arithmetic Coding (CABAC) [36]. Although the use of a single Universal Variable Length Codes (UVLC) table is simpler, the major disadvantage is that a single table is usually derived using a static probability distribution model, which ignores the correlations between the encoded symbols. CAVLC to encode residual, zig-zag ordered 4×4 (and 2×2) blocks of transform coefficients, performing first a run-level coding to compactly represent strings of zeros and finally applying an adaptive VLC table. The choice of the table depends on the number of non-zero coefficients in neighboring blocks. Arithmetic coding method also needs a probability model at both encoder and decoder sides. However, to increase the coding efficiency, CABAC uses a dynamic probability model that is adapted to the changing statistics with a video frame, through a process called context modelling.

Error Resilience

For error prone networks, H.264 provides better error resilience as compared to previous standards. It has introduced the concept of Flexible Macroblock Ordering (FMO) and slice interleaving, which allows transmission of macroblock in non-raster scan order. Data partitioning is another error resilience mechanism included in the standard. In order to stop the propagation of error via inter prediction, H.264 provides additional error resilience features called redundant slice and spare macroblocks. These slices and macroblocks carry duplicate information which can be used if the corresponding previous slices or macroblocks were not correctly decoded.

3.5.2 Tools, Profiles and Levels

The H.264 standard defines three profiles: the Baseline, Main and the Extended Profile, each of them having 15 levels. Table 3.5 lists the main features of H.264 profiles, whilst table 3.6 shows the different levels.

Coding Tools	Baseline	Extended	Main
I,P slices	✓	✓	✓
CAVLC	✓	✓	✓
Error Resilience	✓	✓	
SP, SI slices		✓	
B slices		✓	✓
Interlaced		✓	✓
CABAC			✓

Table 3.5: H.264/AVC Profiles

The Baseline profile which has minimal features and least computational load is suited for low delay application. It also exploits a sufficient degree of error resilience to compensate for the data loss in error prone networks.

The Main profile primarily targets broadcast and other high end applications where computational load is not an issue. This profile provides highest compression as it uses features like bidirectional coding and CABAC. Main profile does not have error resilience feature as it expects the network to be error robust.

Level	Max. fps	Typical resolution	Max. bit rate	Max. ref. pictures
L1	15	176×144	64 Kbps	4
L1.1	7.5	352×288	192 Kbps	3
L1.2	15	352×288	384 Kbps	6
L1.3	30	352×288	768 Kbps	6
L2	30	352×288	2 Mbps	6
L2.1	30	352×480	4 Mbps	6
L2.2	12.5	720×480	4 Mbps	5
L3	30	720×480	10 Mbps	5
L3.1	30	1280×720	14 Mbps	5
L3.2	60	1280×720	20 Mbps	4
L4	30	1080×1920	20 Mbps	4
L4.1	30	1080×1920	50 Mbps	4
L4.1	60	1080×1920	50 Mbps	4
L5	72	$2K \times 1K$	135 Mbps	5
L5.1	30	$4K \times 2K$	240 Mbps	5

Table 3.6: H.264/AVC Levels

Applications which can afford computational power but also have to operate over highly unreliable networks can go for extended profile. This profile employs all the error resilience features present in H.264. Extended profile is suited for storage and streaming applications as it provides features like switching, splicing, random access and other VCR-like functionalities.

3.6 CONCLUSIONS

The MPEG-4 Visual standard supports the coding and representation of visual objects with efficient compression and unparalleled flexibility. The diverse set of coding tools described in the standard are capable of supporting a wide range of applications. Amongst developers and manufacturers, the most popular elements of MPEG-4 Visual to date have been the Simple and Advanced Simple Profile tools and there is a clear industry requirements for efficient coding of

rectangular video frames.

AVC/H.264 provides mechanisms for coding video that are optimized for compression efficiency and aim to meet the needs of practical multimedia communication applications. The range of available coding tools is more restricted than MPEG-4, however many studies in literature indicates that the newly developed H.264 standard has the ability of outperforms MPEG-4 Visual [5].

Since MPEG-1, the evolution of MPEG standard has been focused on the development of more efficient compression schemes to achieve better encoding quality at lower bit rates, not taking into account the processing latency. Although the increased performance and flexibility, MPEG-4 does not provide yet a low delay coding profile.

Chapter 4

Coding/Decoding Delay in MPEG

4.1 INTRODUCTION

Since 1994, when the MPEG-1 standard: *Coding of moving pictures and associated audio for digital storage media at up to 1.5Mbit/s video* [28] was published, MPEG has evolved in the direction of an even higher compression efficiency and flexibility, introducing new coding tools and supporting more video resolutions and applications. The last result of this evolution is MPEG-4 that, with its 19 profiles is nowadays the most complex, rich and advanced framework to answer the need of the multimedia community. Despite this increased complexity, MPEG is not yet able to satisfy a sub-class of applications which demand a very low coding/decoding latency.

Nowadays broadcast and studio-production applications request latencies lower than 40 ms at 4CIF frame resolution and medium-high quality. What MPEG standard defines as low delay is a total encoding and decoding latency of around 150 ms. Nevertheless, the standard does not specify neither a minimum threshold nor the related encoding quality. Therefore MPEG does not supply a valid low delay mode which could satisfy the emerging group of digital video applications.

The MPEG encoding/decoding latency optimization becomes more and more an implementation problem which involves many variables and difficult trade-off between coding quality and bandwidth overhead. An existing MPEG-2 hardware solution, which actually achieves the best known results, is described to understand its advantages and limits.

4.2 MAIN CAUSES OF DELAY IN MPEG

The most common field where a low latency is needed, is video conferencing. However, this application does not require a high resolution (usually CIF or QCIF) and the main constrain is a very low coding bit rate (< 384 Kbps), which limits the image quality. In studio production applications, where the video resolution is bigger (4CIF) and a higher encoding quality is required, the encoding bit rate increases up to $30 \div 40$ Mbps. Quality and bit rate are related variables, in the sense that given a particular set of coding tools, if we increase the bit rate, the quality gets better and viceversa 2.13. Using a more efficient coding algorithm, we can obtain higher quality at lower bit rates, but the coding/decoding latency will not necessarily decrease.

Evaluate the coding/decoding delay in a very complex algorithm is a non trivial problem. In a very intuitive way, we can say that latency and complexity are in contrast, because the more complex the algorithm, the more *elaboration time* it requires to finish its job. However, this assumption is not always true. The real trade-off between complexity and latency is the level of parallelization achievable in the algorithm implementation.

MPEG standards employ a common set of features in order to assure a basic backward compatibility. Since MPEG-1, the subsequent evolutions have exploited the hybrid DPCM/DCT video codec model described in section 2.11, which have been more and more enhanced without

modifying its basic scheme.

In the following sections we analyze a set of basic MPEG features taking into account its contribute to the processing latency. Most of these features are used by all MPEG standards; therefore, a class of basic coding tools can be defined in order to minimize the coding/decoding delay in MPEG.

4.2.1 Data Format

MPEG exploits the hierarchical data structure shown in figure 4.1. The very basic data element is a matrix of 8×8 samples called *block* which can be of three types: luminance (Y), red chrominance (Cr), or blue chrominance (Cb). A group of 4 blocks, corresponding to an area of 16×16 luminance samples, forms the base of a *macroblock*, which is also composed by a number of additional chrominance blocks depending on the sampling format (refer to section 2.4.3). One or more consecutive macroblocks form a *slice*.

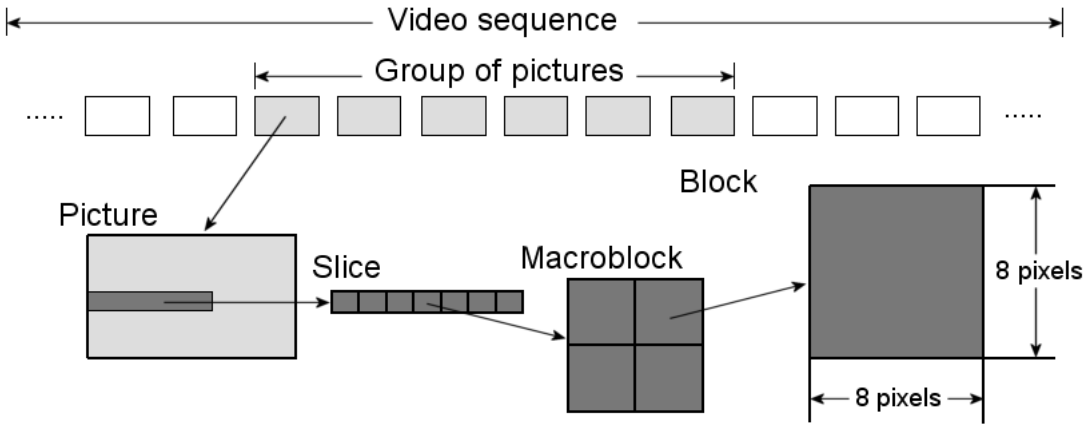


Figure 4.1: MPEG hierarchical data structure

As already mentioned in section 2.8.2, the use of blocks as elementary data suites well the hybrid DPCM/DCT motion compensated basic encoding MPEG scheme. However, from the latency point of view, the encoder cannot start the processing till it has received at least 16 lines of video, which gives a delay of $\simeq 1$ ms. Despite this minimum requirement, in many hardware implementations almost one input frame is stored before to start encoding, which leads to a more important delay of $33 \div 40$ ms depending whether PAL or NTSC system is employed.

4.2.2 Frame and Field Encoding

Many video sources generates interlaced video sequences (refer to section 2.3.3), where the top and bottom field are acquired and transmitted one after the other. Using a *frame-mode* encoding we have the advantage of a better spatial correlation between consecutive lines (especially in the case of poor movement), but before starting encoding we need to re-interlace fields (which

come alone) and store each frame in an input buffer as shown in figure 4.2. Moreover, if at the decoder output we want to generate the same interlaced sequence, we have to take into account an additional latency of half a frame time, which is necessary to store the two fields separately.

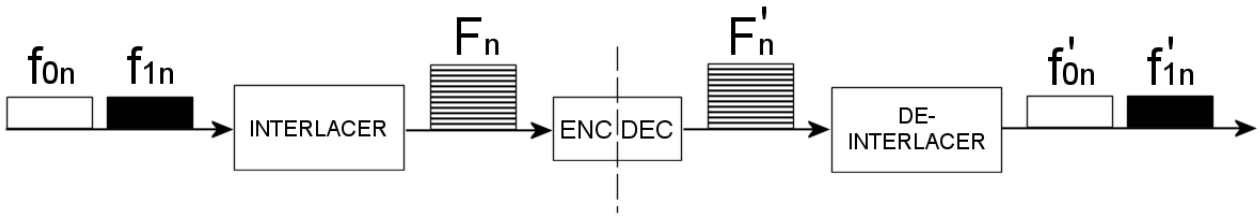


Figure 4.2: Frame encoding of interlaced sequences

On the other hand, the field-encoding mode (figure 4.3) avoids the input latency at encoder side and the output latency at decoder side at the price of a slight decreased vertical resolution. Furthermore, if the encoder exploits temporal redundancy using the MC and ME approach, it can be affected by a reduced efficiency due for example, to any object's pixels which are present in one field but not in the next (It is the case of horizontal lines). However, this effect is strongly dependent on the kind of video scene and in most cases is not very important.

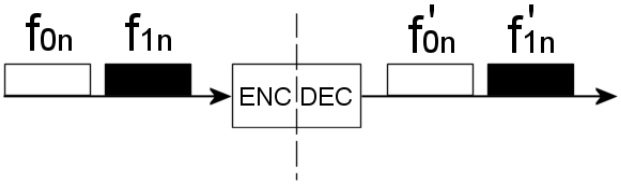


Figure 4.3: Field encoding of interlaced sequences

Using a frame-mode encoding process with interlaced sequences, we introduce a delay at encoder and decoder side that is globally greater than one frame time (40 ms and 33 ms for PAL and NTSC respectively).

The picture quality is generally the most appreciated parameter therefore, most of the hardware platforms do not care about latency and the field-coding mode is not widely supported.

4.2.3 GOP Size and Structure

A GOP (Group Of Pictures) is a series of one or more coded frames (I, P, and B pictures, refer to 2.11) intended to assist in random accessing and editing. A regular GOP can be described with two parameters:

- **N** - The GOP-length, is the number of pictures in the GOP.
- **M** - The GOP-structure, is the spacing of the P pictures.

The GOP illustrated in figure 4.4 can be described as N = 9 and M = 3. Generally the GOP structure repeats through the sequence, but N and M may be changed at any time during the encoding process. The smaller the GOP, the better the response to movement (since the I frames are closer) and editing capabilities, but the lower the compression. For example, sending I pictures only requires more than twice the bit rate of an IBBP GOP, however each I picture can be edited independently of each other.

The smallest GOP length is a single I frame, which corresponds to a GOP structure M = 1. There are no formal limits on the length of a GOP, but for transmission purposes a typical length of 12 or 15 pictures is employed.

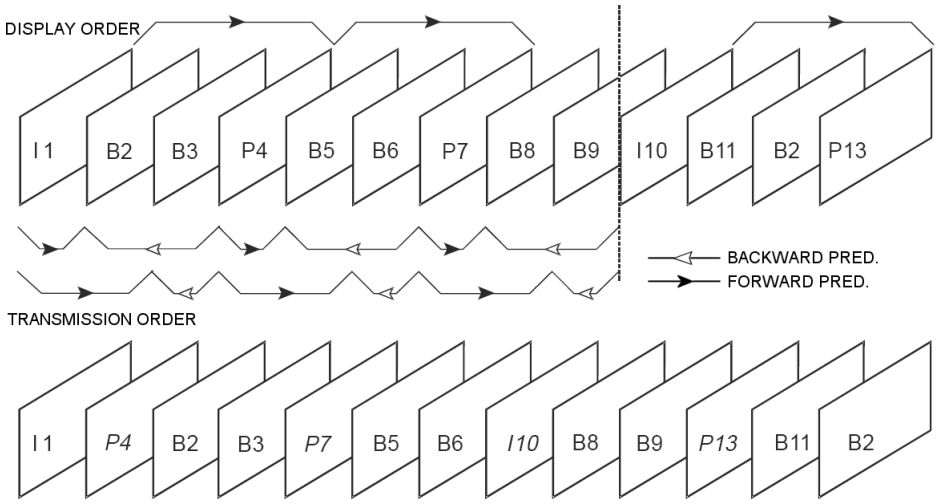


Figure 4.4: Transmission and display picture order

A GOP containing B pictures have the further disadvantage that transmission order and display order of the encoded pictures are not the same. In fact, while a P picture naturally follows the I or P picture from which it has been predicted, a B picture cannot be decoded until both of its references (I and P) have been received and decoded. Summarizing, in the coded bitstream, a GOP must start with an I frame and may be followed by any number of I, P, or B frames in any order. In display order, a GOP must start with an I or B frame and ends with an I or P frame. Figure 4.4 shows the same GOP sequence in display order at the top and in transmission order below. Note that, in transmission order B pictures always follow the reference pictures from which they have been predicted.

Despite the better efficiency and flexibility of bidirectional coding, which achieves up to 4 time less data if compared with a typical I pictures occupancy, the use of B pictures have to be restricted or avoided in a low delay coding scheme. Sending pictures out of display order requires additional memory at encoder side and a more complex processing scheme which introduces an important latency. In fact, a GOP-structure of $M \geq 1$ gives a latency contribution of $M - 1$ frames, which correspond to the buffering of the intermediate frames that occurs between the I

and P references which need to be coded and transmitted in advance.

4.2.4 Video Buffer Verifier

In Section 2.11 it was stated that the coded pictures resulting from the MPEG encoding of video are not of constant size. Their length depends on whether they represent an I, P or B picture and on the complexity of the visual scene. A video encoder therefore generates a variable amount of bits per picture and it outputs a VBR bitstream.

If the application requires a fixed-bandwidth transmission channel, then the VBR is generally smoothed to a CBR by a First-In First-Out (FIFO) memory known as the coder buffer (refer to section 2.13). On the decoder side a similar buffer is needed in order to reproduce the same VBR outputs by the encoder. This process involves two buffers and introduces a delay that is function of the chosen CBR and the buffer size.

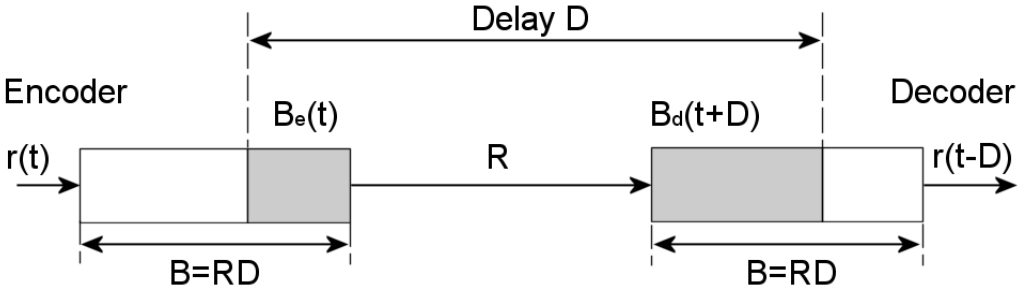


Figure 4.5: Encoder and decoder buffer delay

If the encoder and decoder buffer are well synchronized, it is possible to prove that the sum of their fullness is a constant B equal to the buffer size [27], as shown in figure 4.5 and stated in equation **??**.

$$B_e(t) + B_d(t + D) = B \quad (4.1)$$

$$B = RD \quad (4.2)$$

If R is the CBR rate then using the equation 4.2 we obtain that the global delay introduced by the rate conversion is $D = B/R$.

The MPEG standard does not specify the strategy to control the encoder buffer, but it gives a table of maximum sizes and encoding bit rates related to each profile and level. Therefore, it is up to the developer to find the best trade-off between delay and buffer size. This delay contribution may be very important at low-medium bit rates however, most implementations employ the maximum sizes suggested by the standard, which means a greater buffer delay contribution whatever the bit rate is.

The table 4.1 shows some buffer delay related to different profiles and levels in MPEG-2 standard. The minimum delay $\simeq$ 120 ms is always related to the maximum bit rate, which is

PROFILE AND LEVEL INDICATION	BIT RATE RANGE (Mbps)	MAX VBV BUFFER (bit)	DELAY RANGE (ms)
SP@ML	2 ÷ 15	1 835 008	122 ÷ 917
MP@LL	0.1 ÷ 4	475 136	119 ÷ 4751
MP@ML	2 ÷ 15	1 835 008	122 ÷ 917
MP@H-14	10 ÷ 60	7 340 032	122 ÷ 734
MP@HL	10 ÷ 80	9 781 248	122 ÷ 978
422P@ML	3 ÷ 50	9 646 080	193 ÷ 3215

Table 4.1: Buffer delay contribution in MPEG-2

often not exploited.

4.3 MPEG LOW DELAY SOLUTIONS

A low delay mode for real-time video communications, achieving a latency $\simeq$ 150 ms, is described in MPEG-2 standard [29]. This mode has been designed to cope with medium-low quality video applications at low resolution such as: visual telephony, video-conferencing and monitoring. Three main features have been exploited by MPEG:

- Low buffer occupancy
- No bidirectional coding.
- Intra slice coding.

A low buffer occupancy for both encoder and decoder is required for low delay. However, a constant latency is not automatically performed since the buffer size is kept constant and MPEG delegates to developers to specify a strategy to compensate the dependency on the working bit rate 4.2.4.

The total encoding and decoding delay is kept low by generating a bitstream which does not contain B pictures. Therefore, I and P pictures only are exploited by the coding process, which avoids the high frame re-ordering latency typical of bidirectional coding approach. Furthermore, to avoid I pictures, which require a large amount of bits, the intra-slice coding technique is employed.

As 150 ms are not enough to satisfy the requirements of studio production applications, which demands latencies less than 40 ms, the MPEG low delay mode does not provide a valid solution for our purpose. However, this method can be considered a good starting point which can be enhanced and integrated with an efficient VBV buffer control strategy for the case of real low occupancy.

4.3.1 VBV Buffer Size

The VBV buffer size is a crucial point to control the delay in applications which need a CBR coding (refer to section 4.2.4). Given a fixed buffer size, the delay is a function of the working bit rate R as stated in equation 4.3.

$$D = \frac{B}{R} \tag{4.3}$$
$$B = KR \tag{4.4}$$

If we want D to be a constant then B must be linearly dependent on the encoding bit rate R, as described in equation 4.4, where K is constant and it equals the requested buffer delay D. The standard table 4.1 can be referred to find out a bounding condition for K. Given the maximum buffer size VBV_{max}, we can state that:

$$B \leq VBV_{max} \tag{4.5}$$
$$K \leq \frac{VBV_{max}}{R} \tag{4.6}$$

Beyond the limitation of equation 4.6 we can make our choice of K. We know that the lower the buffer size the lower the delay, but on the other hand, a small buffer is more difficult to control and the risk of overflow and underflow increases with consequent loss of frames.

The ideal buffer should be able to compensate the input VBR peaks due to both spatial and temporal complexity variations in the video sequence. A first estimation of the minimum buffer size could be the difference between the size of the biggest and smallest encoded pictures which may occur in the video sequence. However, the information of the maximum possible variation is not enough, we also need to know how much time it takes. In general, three main variables can influence the buffer size:

- **Sequence** - The performances of a video coding algorithm are data dependent, so the results may change a lot using different input video sequences. For example, using the same average quality, a sequence made by the repetition of a still frame, as for example a bar code test, would require a smaller buffer than a more complicated sequence rich of movements, details and scene changing.

- **GOP** - The bit occupancy of an I picture is always bigger that a P picture, while B pictures get the lowest sizes. This means that a GOP made of I or P pictures only, can limit the difference between sizes of consecutive frames.

- **Control** - In a rate-controlled encoding scheme (2.13) an algorithm adjusts the encoding parameters to achieve a target bit rate. Therefore the controller efficiency can directly influence the required buffer size.

Finding the optimal solution for the minimum buffer size is a non trivial problem which involves many variables. An existing solution will be analyzed in the next sections, for the moment we can suppose that in a given encoding configuration the buffer keeps an average number $N \in \mathbb{Q}$ of coded frames. If P is the input frame rate (for example 25 Hz for PAL) then the average encoded frame size is R/P and B can be expressed as in equation 4.7 that combined with equation 4.4 gives us a simple expression for K.

$$B = N\left(\frac{R}{P}\right) \tag{4.7}$$

$$K = \frac{N}{P} \tag{4.8}$$

In the case of a very regular bitstream, where the average bits per frame is quite constant, the problem to find an optimal buffer size becomes to find the lowest number N which allows an efficient buffer control even in the case of a critical video sequence. In the case of field-mode encoding, we can halve N in all previous equations, getting half the delay than the frame-mode configuration.

$$B = \left(\frac{N}{2P}\right) R \tag{4.9}$$

4.3.2 I and P Only

In order to minimize the variations between consecutive pictures size, short GOP-length should be avoided. The simplest solution is an I-only scheme (GOP with $M = 1$). In this case the average bits per encoded-frame are equal to the average intra-coded picture size. As I-picture bit occupancy does not achieve high variance, this working condition agrees with the hypothesis of regular bitstream, needed by the buffer model of previous section. On the other hand, intra-coding is not very efficient itself, which means that a higher amount of bits is generated and higher bit rates are required to achieve a target quality.

Despite P pictures requires about half the data of an I picture at the same encoding quality, we cannot use a P pictures only scheme, because the forward prediction method in based on a reference frame which must be intra-coded at least for the first position in the GOP. Using a GOP based on both I and P pictures, we get the following sequence:

$$IPPP.....PPPP - IPPP.....PPPP - IPPP.....PPPP - IPPP..... \tag{4.10}$$

If the GOP-length is great enough then the average amount of bits per encoded-frame is equal to the average P picture size. However, each time that an I picture occurs, the buffer occupancy rapidly increases and the control feedback will decrease the coding quality to avoid the buffer overflow. This reaction would be visible in the decoded sequence as a periodic flash corresponding to a quick quality change.

To avoid this inconvenience a GOP-length=∞ can be employed. However, I pictures repetition represents an effective mechanism to stop the error propagation, which can otherwise really affect

the decoding process. Each P picture encodes the difference between two consecutive frames so, if an error occurs during the decoding process, a zone of the decoded picture would be affected and the error would spatially and temporally propagate over all the sequence. To solve this problem without using I pictures, an intra-slice encoding strategy can be adopted.

4.3.3 Slice Encoding

MPEG defines the slice as a series of consecutive macroblocks. Every slice shall contain at least one macroblock and shall not overlap. The position of slices may change from picture to picture but the first and last macroblock of a slice shall be in the same horizontal row of macroblocks. Slices shall occur in the bitstream in the order in which they are encountered, starting at the upper-left of the picture and proceeding by raster-scan order from left to right and top to bottom (illustrated in Figure 4.6 as alphabetical order).

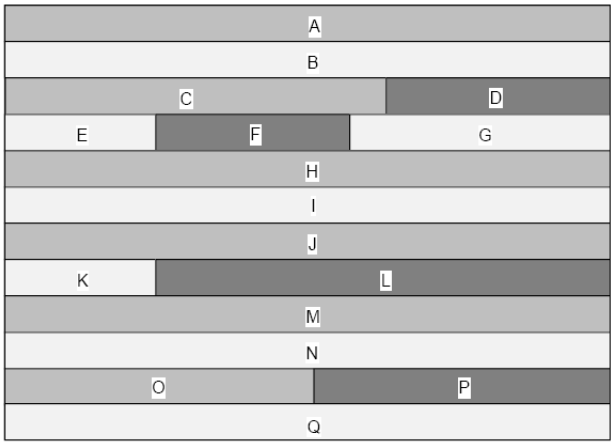


Figure 4.6: Example of restricted slice structure

The most basic method to achieve spatial localization of errors is to reduce the number of macroblocks in each slice. The increased frequency of re-synchronization points will limit the affected picture area in the event of a loss. It is effective in any transport or storage media, and in any profile since the slice structure is always present in MPEG coded video. The method results in a small loss of coding efficiency due to the increase of overhead information. From the perceived picture quality point of view, the performance of this method is generally dependent on the relative sizes between slice and picture. Therefore, the slice size should be decided by considering the picture size (in unit of macroblocks) and the trade-off between coding efficiency and visual degradation due to errors.

To avoid the disadvantages of intra pictures, some applications requiring low delay can choose to update the sequence by intra-coding only parts of a picture. This may provide the same kind of error resilience as intra pictures. For example, assuming that a constant number of slices per

picture from top to bottom are intra-coded, then the whole picture will be completely updated every n frames, where n is a constant which depends on geometrical parameters only.

If the intra-coded slice has small size compared to the whole picture, the buffer occupancy does not increase too much and the rate control can keep the quality of the intra-slice close to that of the surrounding non-intra macroblocks. In summary, the intra-slice method makes it possible to use P pictures in a GOP with length=∞, which represents the best trade-off between coding/decoding delay and average video quality.

4.3.4 VBV Buffer Rate Control

Rate control is not a part of the MPEG standard, but the standard group has issued non-normative guidance to aid in implementation [37, 38, 39]. A rate control algorithm dynamically adjusts encoder parameters to achieve a target bit rate R. It allocates a budget of bits to each group of pictures, individual picture and/or sub-picture unit in a video sequence and depending on the predicted next picture complexity, it calculates the quantization parameter QP to employ.

The quantization parameter QP determines the encoding quality and the real amount of bits needed to code a residual picture. QP is typically limited in changes to no more than ± 2 units between pictures, in order to guarantee stability and to minimize perceptible variations in quality. If the sequence's complexity varies quickly, as it is common at a scene change, QP cannot follow this variation at the same velocity, thus some pictures will be encoded at higher quality than the one allowed by the selected bit rate, which will result in a bit allocation overhead.

This data overhead will cause a VBV buffer occupancy increment which will be compensated later by the rate control algorithm itself. However, in a low delay application the VBV buffer size may be very small and this mechanism can generate a buffer overflow, with consequent errors and frame skipping at decoder side. So the lower delay we try to achieve, the higher the risk of overflow using a standard rate control method.

A way to overcome this problem is to take into account a bandwidth overhead. If R is the channel bit rate, we can set the encoding rate to R/α where $\alpha \geq 1$. The residual bandwidth $R(1 - 1/\alpha)$ will be used by stuffing packets during most of the time, but in the case of data overhead it will avoid the buffer overflow. In other words, we read the data from the buffer at a higher rate than expected, so the encoder buffer level is kept low for most of the time and if the picture complexity varies quickly we achieve a more efficient control.

Using this buffer control strategy, under normal working conditions, the encoder buffer is kept quite empty while the decoder buffer quite full, thus the VBV buffer delay is completely moved to decoder side. Therefore, the decoder implementation become a very important point in low delay applications.

4.3.5 NEL MPEG-2 Encoder

In late 2002, NTT Electronic Corporation (NEL) developed SC2010E module, which integrates an MPEG-2 422P@ML video encoder LSI, an MPEG-1 Layer 2 audio encoder DSP and a TS multiplexer. The SC2010E was the first flexible MPEG-2 hardware solution designed to achieve a low encoding delay at high image quality. Because of its compact size and low power consumption the SC2010E became the first candidate for SDTV mobile applications, and it has been integrated into LiveTools [11] equipment, a full mobile COFDM-based wireless system developed with EPFL [12, 40, 2] collaboration.

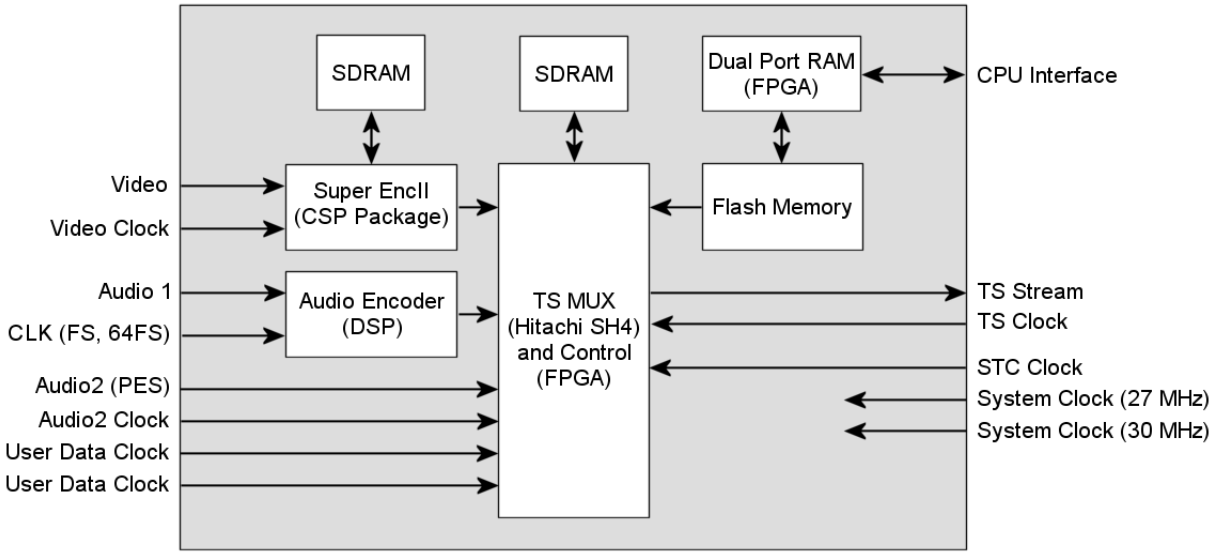


Figure 4.7: NEL encoder module generic scheme

Video Encoding

NEL encoder allows most of encoding parameters to be user-defined, for example the VBV buffer size, that is a key parameter of the implemented Ultra Low Delay (ULD) encoding mode. NEL ULD mode makes it possible to have a total 54 ms encoding/decoding delay, which is the minimum end to end delay actually measured on a MPEG-2 hardware platform at standard PAL resolution (46 ms in the case of NTSC). To achieve this performance NEL has designed the ULD mode employing all the low delay techniques already mentioned in the previous sections:

- **Field Encoding** - ULD uses field encoding to avoid input data reordering delays. So each picture is a field picture and the picture rate is doubled (50 Hz for PAL and 60 Hz for NTSC).
- **GOP-structure** - No B pictures are used (M=1). Moreover, the ULD has been implemented at SP@ML, so it makes use of only a basic set of MPEG coding tools.

- **GOP-length** - ULD is designed to employ a GOP like IPPPPPPP..... with infinity repetition of P pictures (GOP-length = ∞).

- **Intra-Slice Mode** - This technique guarantees an adequate error resilience without using I pictures so it is strongly recommended for the ULD mode.

- **VBV Rate Dependent Size** - NEL has implemented the VBV buffer size model described in equation 4.9, where the parameter $N > 1.4$ represents the number of encoded frames in the buffer and it is calculated depending on the requested delay D.

- **Bandwidth Overhead** - To avoid buffer overflow at encoder side, a bandwidth overhead is defined. The bandwidth multiplier α varies with the delay and the video source type (PAL or NTSC). The table 4.2 shows some values of α related to a supported low delay mode, with minimum delay of 84 ms for PAL and 70 ms for NTSC. We can state that the shorter the delay the higher is α, which means that to achieve a lower delay we have to increase the bandwidth overhead. In ULD mode an $\alpha \geq 3$ is recommended by NEL, wasting over 66 % of the entire bandwidth R.

DELAY TIME (Frames)	α MULTIPLIER NTSC	α MULTIPLIER PAL
2.1	2.80	2.60
2.2	2.40	1.90
2.3	2.00	1.70
2.4	2.00	1.40
2.5	1.70	1.30
2.6	1.50	1.20
2.7	1.40	1.15
2.8	1.30	1.15
2.9	1.30	1.15

Table 4.2: Suggested bandwidth multiplier for NEL low delay mode

The encoder and decoder buffer are usually synchronized using time stamps information sent with the bitstream (Refer to section 2.12). MPEG standard defines an internal System Time Clock (STC) based on the 27 MHz reference clock so the encoder generates its time stamps sampling the STC at certain points in time. A PTS stamp is generated each time the first byte of a picture inputs into the encoder, so PTS occurs at multiples of the picture time base and it is sent in the header of the related coded pictures.

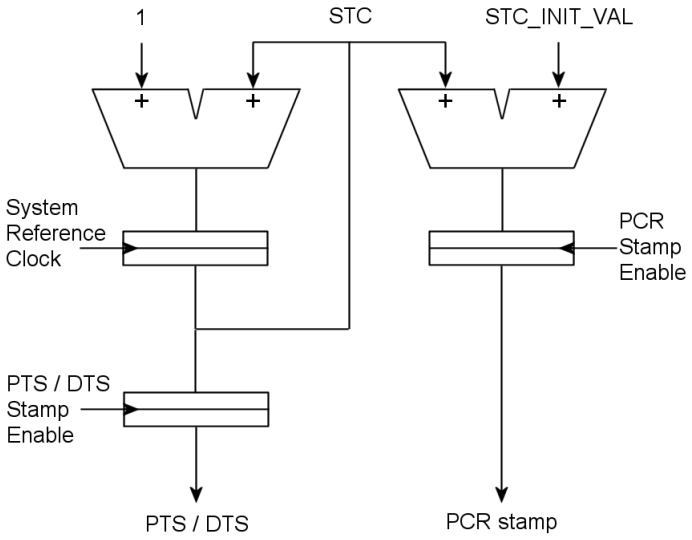


Figure 4.8: STC initial value in NEL implementation

In NEL implementation the generation of PCR stamps can be offset by the parameter STCInitVal which is added to the current STC as described in equation 4.11 and shown in figure 4.8.

$$PCR = STC + STCInitVal \tag{4.11}$$

This strategy does not affect the clock synchronization method but only the PCR counter initial value. If we use a STCInitVal < 0 the first PTS will be received by the decoder after a time $D = |STCInitVal|$ being the fist PTS $= 0$. So the STCInitVal can be used to control the overall encoding/decoding latency asking the decoder to output the first picture after a delay D. If the encoder/decoder configuration makes it possible to achieve a delay D then we can set:

$$STCInitVal = -D \tag{4.12}$$

This method, represented in figure 4.9, makes NEL design to be very flexible allowing the definition of sub-optimal coding strategies characterized by different trade-off within delay, quality and bandwidth overhead.

Table 4.3 shows the implemented modes on LiveTools system (based on NEL encoder) with their related delay. The coding quality varies with the target bit rate and the selected mode. To evaluate the video quality a typical five score levels scale has been used (Refer to section 2.6). To get a broadcast quality (corresponding to a score level = 3) using the MIN modes, we have to work close to 15 Mbps, within only 5 Mbps being used for effective video encoding. This is the expensive price to pay to achieve low delay in MPEG-2 NEL implementation.

LOW modes represent a better trade-off between delay and bandwidth, exploiting lower bit rates than MIN modes, because of the lower α multiplier (lower overhead), but achieving higher delays. LOW modes perform a broadcast quality at about 10 Mbps. The NORM modes are non

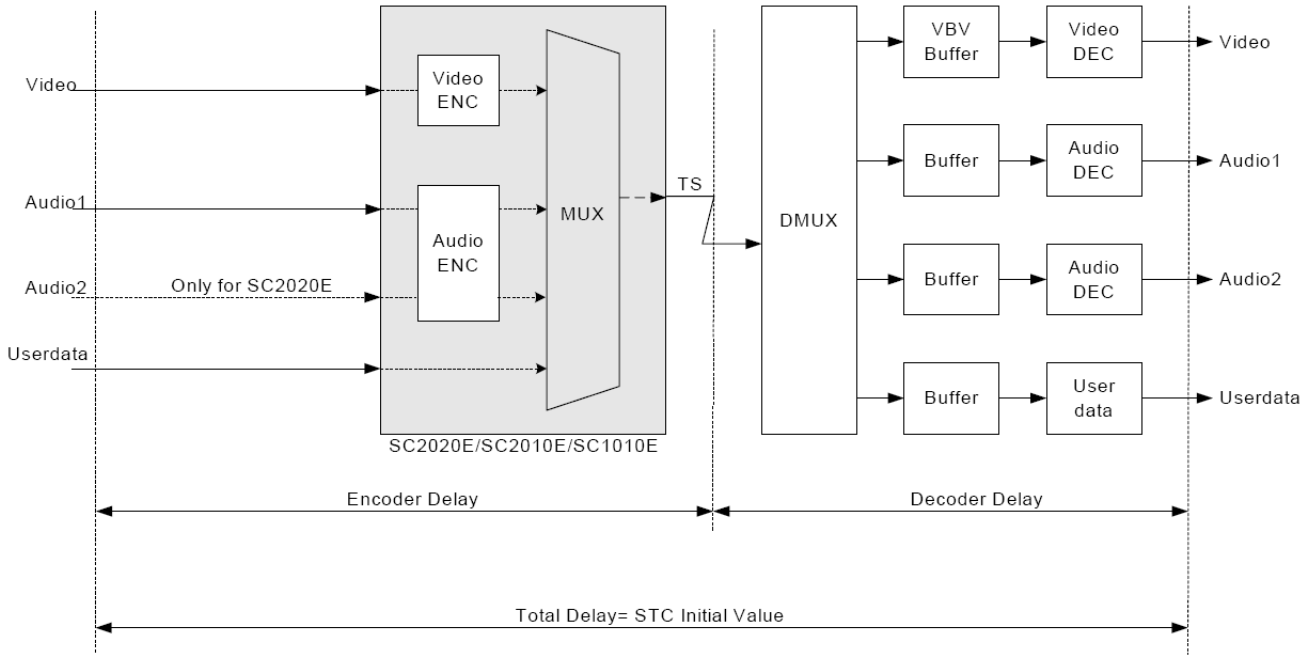


Figure 4.9: General encoding/decoding scheme with NEL module

MODE	GOP (N, M)	CODING UNIT	α MULT.	PROFILE&LEVEL	DELAY (ms)
420 MIN_{SLD}	N=∞ M=1	FIELD	3	SP@ML	54
420 MIN_{STD}	N=∞ M=1	FRAME	2.6	SP@ML	84
420 LOW	N=∞ M=1	FRAME	1.2	SP@ML	110
422 LOW	N=∞ M=1	FRAME	1.2	422P@ML	110
420 NORM	N=12 M=3	FRAME	1	MP@ML	270
422 NORM	N=12 M=3	FRAME	1	422P@ML	270

Table 4.3: LiveTools system coding modes and related delays

low delay, so the bandwidth is fully employed for video coding and we get a broadcast quality even at 5 Mbps.

Audio Encoding

NEL module integrates a MPEG-1 Layer 2 audio DSP encoder. Despite the 54 ms of ULD mode, the audio encoding/decoding process achieves a minimum delay of 100 ms. Thus, in order to correctly decode the audio stream, we have to shift the audio PTSs by 46 ms minimum. This limitation risks to vanish the good performances of ULD video processing. In consumer applications this result is still interesting, as 46 ms of delay between audio and video is hardly perceivable by common users, however it cannot be accepted in professional video applications.

4.3.6 IBM MPEG-2 Decoder

The LiveTools system integrates an IBM MPEG-2 digital audio/video decoder of CS24 family. This single-chip decoder is intended for professional audio/video editing and broadcast applications. The integrated video decoder is compliant with the ISO/IEC 13818-2 [29] standards for 422P@ML, MP@ML and SP@ML video decoding. In addition, the video decoder is capable of decoding both elementary streams and packetized elementary streams at rates of up to 50 Mbps. This capability enables high-bandwidth broadcast and studio-quality applications. The two audio decoders can be configured to handle both MPEG-1 and MPEG-2 Layers I & II audio in compliance with the ISO/IEC 13818-3 standard and furthermore, Dolby Digital audio. The integrated TS de-multiplexer accepts continuous input at a rate of up to 160 Mbps in parallel format or 60 Mb/sec in serial format. A generic block scheme the IBM decoder is shown in figure 4.10

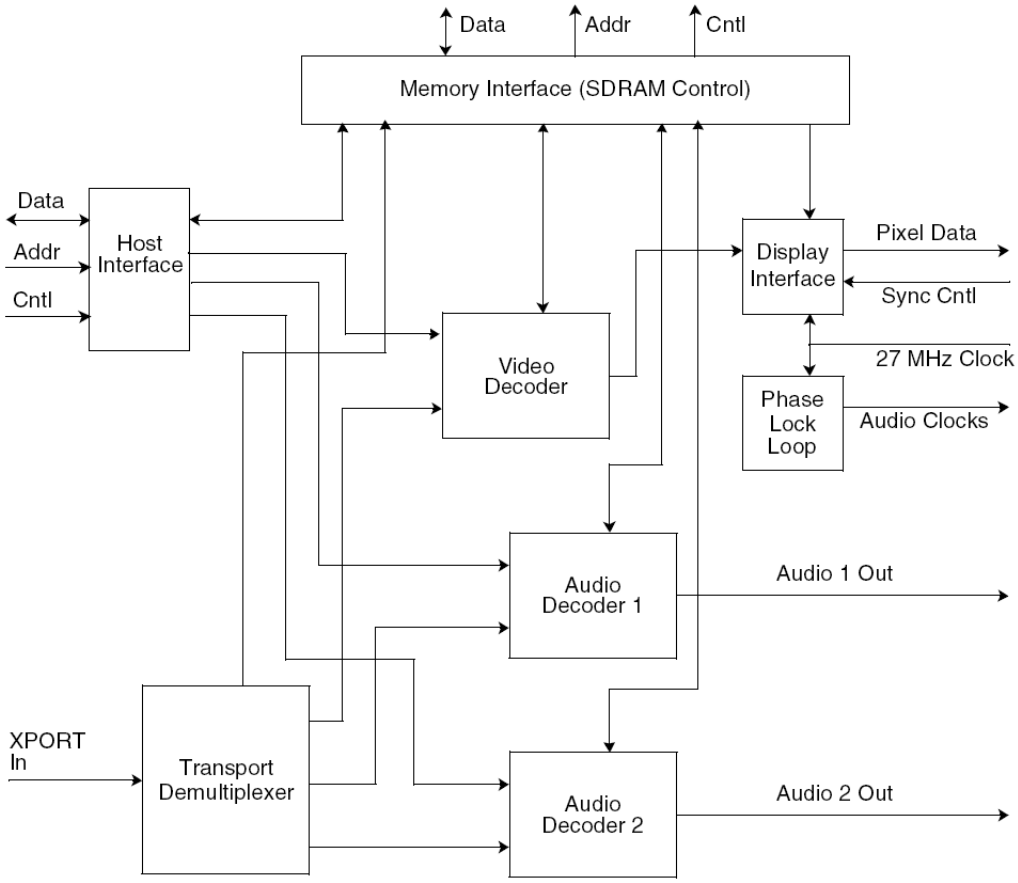


Figure 4.10: IBM decoder chip generic scheme

The IBM chip requires external memory to hold decompressed audio and video data during the decoding process. For this purpose a 16 Mb external SDRAM has been used, whose controller is shown at top of figure 4.10. In addition to storing the decompressed frames, the SDRAM also serves as a temporary buffer for the incoming compressed audio and video stream. The amount of storage required to hold the compressed data stream is a function of the data rate since higher

incoming rates require more buffering.

A frame buffer is necessary for MPEG regular decoding process. In GOP supporting B-frames, the SDRAM configuration will hold up to three decompressed frames of information. These three frames consist of two reference frames and one B-frame that is displayed as it is being decompressed. In the case where no B-frames are employed, the SDRAM configuration is required to hold two frames of data. One is the frame that is currently being decompressed and the other is the single reference frame.

The VBV buffer (implemented into the SDRAM) is different by the classic one analyzed in section 4.2.4. In principle IBM starts to decode data as soon as possible, therefore the delay associated to the VBV buffer appears much shorter than expected, but it is compensated by an equivalent delay at frame buffer level.

DTS are therefore ignored, the only important parameter is PTS, which tells IBM the exact time when it has to remove the data from the frame buffer and send it to the display interface. Much time is lost into the frame buffer, because IBM does not output any data before it has finished to decode the current picture. In the case of the 420 MIN_{SLD} mode at 54 ms (PAL): less than 4 ms are required by NEL encoder to generate the bitstream while the remaining 50 ms are required by the IBM VBV buffer, decoding process and frame buffer, which is a crucial point of IBM architecture. In fact, IBM works frame by frame and does not consider a field by field strategy that could save a considerable part of the global delay.

Video and audio synchronization is accomplished in the device by comparing the PTS with the STC. The result of this comparison determines whether any action is required for re-synchronization. When the difference between PTS and STC is within a fixed tolerance interval, the decoded picture is sent to the display interface. When the comparison is outside the tolerance, the current decoded picture can be repeated or the next B-frame can be skipped to recover synchronization. In the case that a B-frame is not available, synchronization is accomplished by skipping a P-frame.

The IBM chip has two main working modes: *Master* and *Slave*. When IBM is Master, it automatically generates its video synchronism signals, while in Slave mode it needs to be driven by external synchronism. If Master mode is set, IBM does not guarantee a constant decoding delay. Since the output video port becomes active, the display interface starts to output an empty video sequence whose synchronization words are generated by an inner Finite State Machine (FSM) which controls the global chip synchronization. When at power on the FSM is activated, the phase relation between the incoming PTSs and the inner synchronization is completely aleatory. As result we get a delay that we can model as in equation 4.13

$$D = D_{CODEC} + \Delta D \tag{4.13}$$

The term D_{CODEC} is a constant that depends on the actual PTS offset, while ΔD is an aleatory party which can varies between zero and one frame time (40 ms PAL, 33 ms NTSC) depending on the power on state. IBM does not give any control on the inner FSM, and no

way has been found to compensate the additional delay ΔD. The implemented solution to this problem propose a method to synchronize the IBM chip with the input bitstream using a Slave mode configuration.

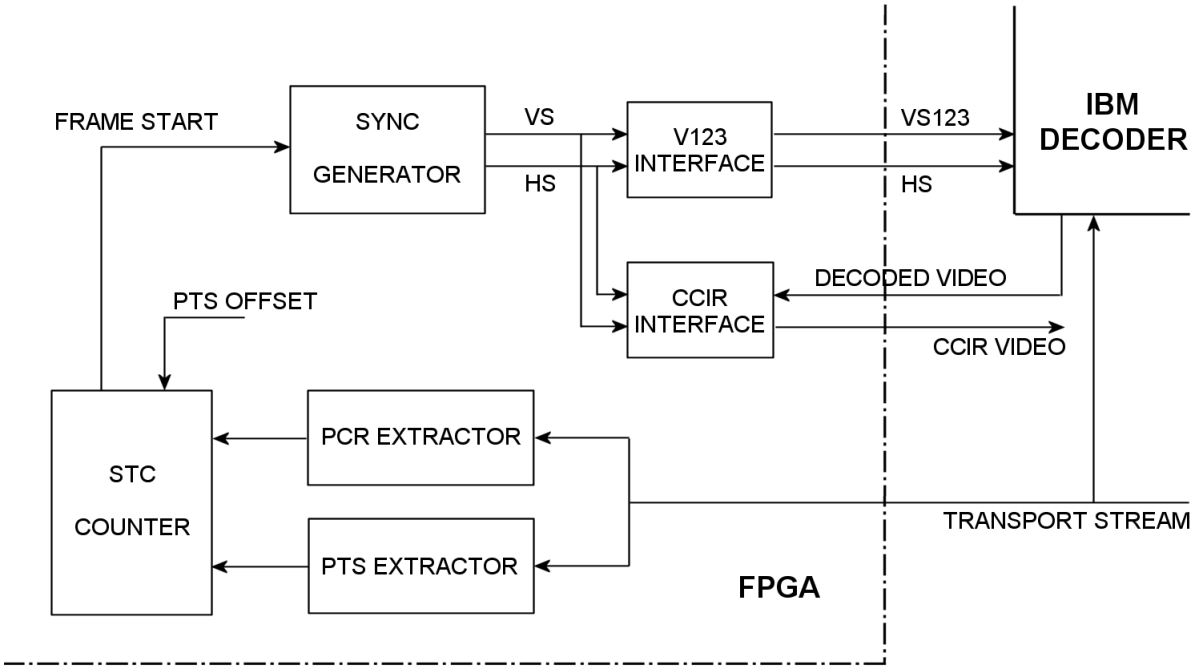


Figure 4.11: General scheme of DIG method

When the IBM works in Slave mode, both the control and synchronization signals have to be provided from outside, thus an auxiliary hardware device is required, which in LiveTools system has been implemented into a Xilinx xc2v500 FPGA. To retrieve timing information from the bitstream, we first need to extract the PCR time stamps and set up the STC counter which is the absolute time reference. PTSs can be recovered and compared with the STC. Each time a PTS matches the STC, we ask IBM to output a picture. Using a PTS offset, the timing can be adjusted in order to get $\Delta D = 0$. This compensation method, whose proposed name is DIG (Decoding Internal Genlock) stabilizes the delay and achieves the minimum allowed latency, therefore has been necessary in LiveTools low delay system implementation. A general scheme of DIG is shown in figure 4.11.

4.4 CONCLUSIONS

The analysis of MPEG low delay mode puts in evidence that only a sub-set of coding features can be employed to achieve low latency, with a consequent loss of compression efficiency. The buffer size and its control strategy become a critic point in the latency optimization. NEL low delay solution overcomes the buffering problems by employing over 66% of bandwidth overhead, that

is the expensive price to pay for a minimum latency of 54 ms at SD resolution and PAL format.

At decoder side, a delay stability problem has been solved in LiveTools system with DIG method. However, this technique is a complicate solution to a fundamental problem of synchronization on IBM chip that underline the limits of standard MPEG platforms to operate with critical delay constraints. Moreover, IBM decoder does not support a full field decoding mode and it requires to store a full frame before to start the decoded sequence output. Finally, IBM does not supply a real optimized low delay solution.

The actual implementation of NEL MPEG-1 Layer 2 audio DSP encoder achieves a minimum delay of 100 ms against the 54 ms of ULD video mode. Therefore, an audio low delay coding technique is required in order to support the minimum latency which have been achieved on video encoding side.

Chapter 5

HERMES Codec

5.1 INTRODUCTION

The state of the art of MPEG low delay coding has been analyzed in the previous chapter which puts in evidence limits and lacks of the best known solution. In this chapter, we introduce a non-standard algorithm which is the base of HERMES, a video encoder at very low delay for studio-production application. A prototype version of HERMES has been implemented on a FPGA platform and is still under development and optimization. HERMES intends to be an alternative solution to MPEG only for a restricted field of application where a coding/decoding delay shorter than one single frame at high quality is required.

5.2 CODEC CORE

HERMES codec algorithm is based on Differential Pulse-Code Modulation (DPCM) [41] which offers an attractive means of picture coding when low complexity is required. This is an important point in real time applications and encoding of high resolution pictures, as for instance broadband television. A basic DPCM system is shown in Fig 5.1

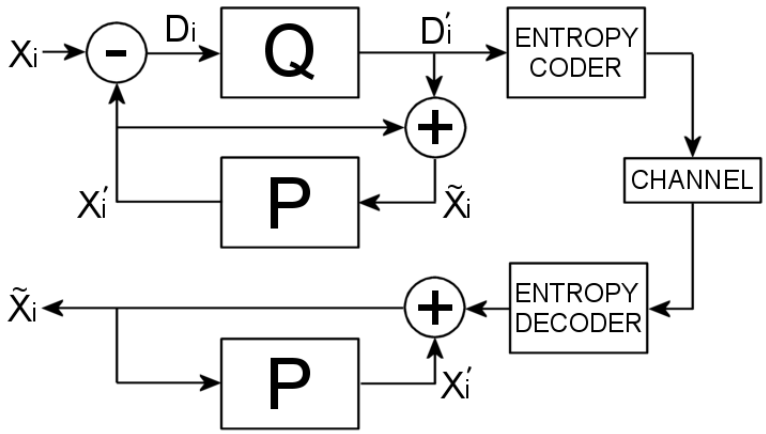


Figure 5.1: Block diagram of a DPCM codec

Instead of transmitting the actual sample X_i directly to the receiver as in PCM, the quantized difference $D_i^{'}$, between the actual sample X_i and a predicted value $X_i^{'}$ is transmitted. The received signal $\tilde{X}_i$ is the sum of the transmitted difference $D_i^{'}$ and the predicted signal $X_i^{'} = P(\tilde{X}_i)$. Therefore the equations which describe the encoding/decoding process can be expressed as follows:

$$D_i = X_i - X_i^{'} \tag{5.1}$$

$$D_i^{'} = Q(D_i) \tag{5.2}$$

$$\tilde{X}_i = D_i^{'} + X_i^{'} \tag{5.3}$$

$$X_i^{'} = P(\tilde{X}_i) \tag{5.4}$$

Each input sample X_i is a luminance or chrominance component of a 4:2:2 YCbCr sequence 2.4.3 according to ITU-R Recommendation BT.601-5 [14]. The generic DPCM encoding algorithm can be described in two steps:

- First, calculate the prediction error D_i 5.1 related to the actual pixel component (Y, U or V) under processing.
- Second, evaluate the quantized prediction error $D_i^{'}$ 5.2 which is processed in two ways:
  - Coded using variable length codes by the Huffman entropy coder (refer to section 2.10)to be sent over the transmission channel to DPCM decoder.
  - Added to the previous predicted pixel $X_i^{'}$ 5.3 to generate the actual locally decoded pixel needed by the prediction module to calculate next prediction 5.4.

An interesting property of this encoding scheme is that quantization error E_q and reconstruction error E_r are identical. To show that, we can simply combine equations 5.1 and 5.3 as follows:

$$E_q = D_i - D_i^{'} = (X_i - X_i^{'}) - (\tilde{X}_i - X_i^{'}) = X_i - \tilde{X}_i = E_r \tag{5.5}$$

Thus, in the absence of quantization distortions ($E_q = 0$) and transmission errors the reconstruction error is zero ($E_r = 0$), which means that the recovered signal $\tilde{X}_i$ is equal to the original one X_i.

The predictor $P(x)$ calculates the value $X_i^{'}$ in the same way on both encoding and decoding sides, from previous reconstituted samples $\tilde{X}_i$. If the prediction is accurate, the prediction error D_i will be small in most parts of the picture. As the entropy of the difference D_i is in general substantially smaller than the entropy of the original input signal X_i, the use of an entropy coder (i.e. a Huffman code [23]) on the quantized difference $D_i^{'}$ can further reduce the bit rate.

The quantization block $Q(x)$ introduces distortion, such as granular noise, slope overload and edge busyness, due to inadequate resolution of the quantization step. Better results are achieved by switching the quantization characteristic depending on estimations of the expected prediction error amplitude.

In DPCM algorithm we know that $E_q = E_r$ (refer to equation 5.5) so the finer the quantization stage, the lower the corresponding reconstruction error, with the same magnitude. A finer quantizer has more levels and it generates more bits. On the other hand, an efficient prediction generates low residuals to be quantized allowing to use coarser quantizers without increasing E_q. Therefore DPCM efficiency is mostly related to the prediction accuracy while the quality can be modulated using different quantization steps.

DPCM algorithm processes pixel by pixel achieving an intra-frame only redundancy reduction, therefore DPCM cannot exploit a high compression efficiency. However, the low complexity, low memory requirement and serial processing are very attractive properties which make DPCM a good candidate to develop a proprietary ultra low delay codec.

5.2.1 Activity Function

This function evaluates the luminance variations around the actual X pixel position using previously reconstructed pixels $\tilde{X}_i$ in order to be reproducible at decoder side too. The activity magnitude is an estimation of the local picture complexity which is employed to select the suitable quantization resolution. If A, B, C and D are the values of neighbors luminance components,

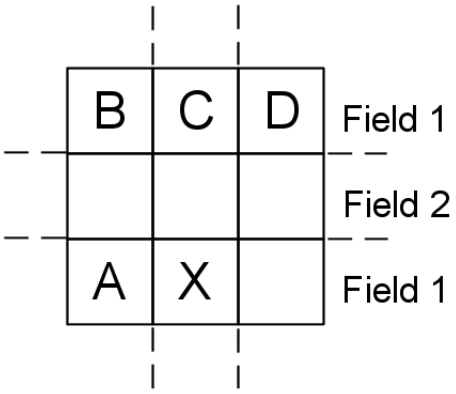


Figure 5.2: Neighbor pixels scheme

whose positions are shown in figure 5.2, then the activity related to pixel X is calculated as follows:

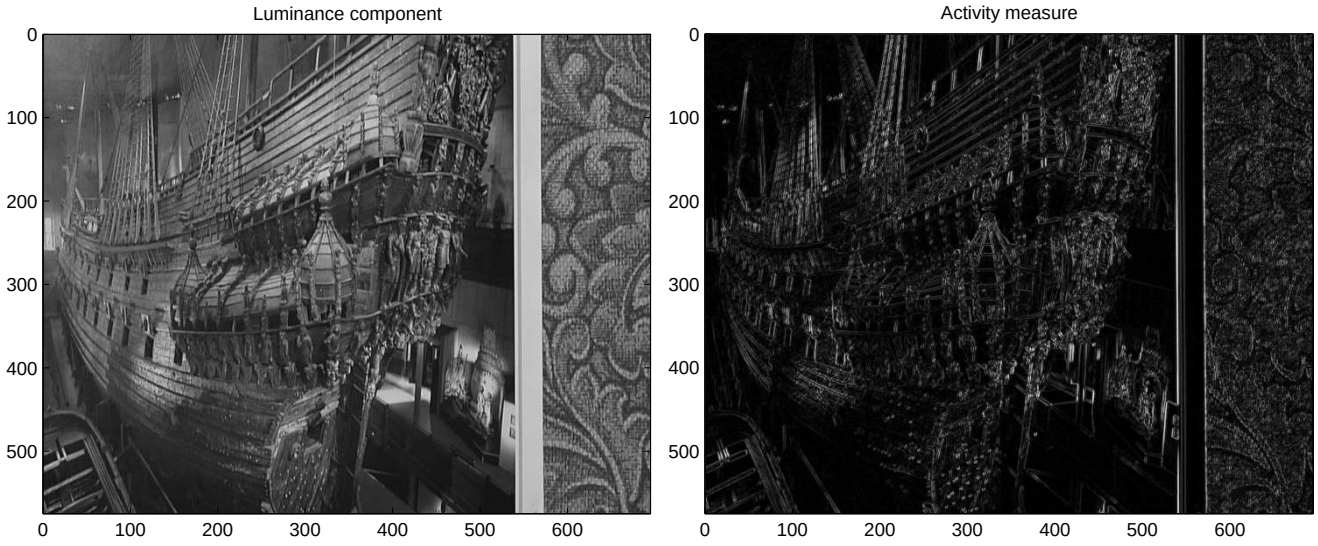
$$Act(X) = MAX\{|A-B|, |A-C|, |A-D|\} \tag{5.6}$$

To have $Act(X) \simeq 0$, it must be $A \simeq B \simeq C \simeq D$, which means that the selected luminance components are similar, thus we are processing a flat zone of the frame picture without any new information. In the case of $A \neq B$ or $A \neq C$ or $A \neq D$ then $Act(X) > 0$ and its magnitude estimates the local complexity around pixel X.

Figure 5.3(b) shows the activity function $Act(X)$ corresponding to the test picture 5.3(a). Dark zones in figure 5.3(b) corresponds to low values of $Act(X)$ and viceversa for light areas. We can remark that $Act(X)$ is nearly independent on the direction of the edge and is proportional to the slope of the edge. Further, $Act(X)$ decreases very fast with distance from the edge, which are good requirements to get a valid estimation of the local picture complexity.

5.2.2 Median Adaptive Predictor

As *adaptive prediction coding* we mean those methods in which several different predictors are operated simultaneously and one of them is selected as the actual predictor by a certain algorithm [42]. The optimum selection method is to keep the predictor which gives the minimum prediction error for each input sample. Unfortunately, this method is very impractical because it requires some overhead information to be transmitted to the decoder in order to identify the selected



(a) (b)

Figure 5.3: Example of activity measurement

predictor. However, we can employ a sub-optimal selection method which could be reproduced at decoder side without extra information requirement.

DPCM coders are often based on linear prediction methods however, better results can be achieved by using non-linear schemes as the Median Adaptive Prediction (MAP) [18, 43, 44]. The MAP is a method to achieve predictor selection in adaptive prediction coding context. The principle of the MAP is as follows: suppose that N predictors are used and that their prediction values are denoted in order of magnitude as

$$X_1^{'}, X_2^{'}, X_3^{'}, ..., X_{N-1}^{'}, X_N^{'} \tag{5.7}$$

The MAP selects $X_{N/2+1}^{'}$, that is the median of those N prediction values, as the actual prediction value. Since the adaptive selection of predictors in this method is made only through magnitude comparisons within values which can calculated at decoder side as well, no overhead information needs to be transmitted to the decoder. Therefore, it is possible to perform an adaptive control of predictors on a sample-by-sample basis.

The performances of MAP depends on the number and type of used sub-predictors. HERMES codec is based on a median type predictor [45] which is loaded by linear sub-predictors $X_N^{'}$, according to the generic scheme in figure 5.4.

If only pixels from the same line are used to calculate $X_i^{'}$, then the sub-predictor P_i is referred to as mono-dimensional (1-D). Otherwise, If pixels from the previous lines are included, P_i is a bi-dimensional (2-D) sub-predictor. Generally, 2-D predictions result in SNR improvements of around 3 dB as compared to 1-D predictions [4], however the subjective quality improvement is even more important than this number would suggest, which is mainly due to the elimination of

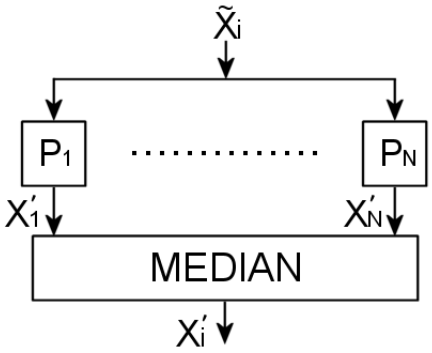


Figure 5.4: Generic MAP scheme

the jaggedness around non-horizontal edges. The only disadvantage of 2-D prediction is that it requires the buffering of previous lines.

The number of pixels employed in the prediction is called the order of the predictor. In general, a higher order predictor outperforms a lower order one, but studies performed on television images have demonstrated that there is only a marginal gain beyond a third-order predictor [43]. HERMES codec employs the following adaptive median predictor, which has shown good performances in several comparison tests [45]:

$$P(X) = MED\{A, C, L, R, A\} \tag{5.8}$$

where

$$L = \frac{A}{2} + \frac{(C+D)}{4} \tag{5.9}$$

$$R = A + C - B \tag{5.10}$$

where

- **A** is the previous pixel in the same line (refer to figure 5.2), the simpler 1-D horizontal predictor.
- **C** is the previous pixel in the same column (refer to figure 5.2), the simpler 1-D vertical predictor.
- **L** is the overall level predictor [41], which can be considered as a fundamental 2-D predictor structure in DPCM coding.
- **R** is a 2-D predictor called $-45°$ ramp.

5.2.3 Nonuniform Adaptive Quantizers

Quantizers are usually nonuniform in order to exploit the visual masking effect. Visual masking means that the human eye is less sensitive to distortions at high spatial frequencies than in plain

areas, thus video noise is effectively masked by fine details in the picture, whereas in plain areas it is highly visible. For this reason, coarser quantization can be achieved where large variations within neighbor samples occur.

The DPCM coder efficiency can be improved by using an adaptive quantization method. Depending on the local luminance activity 5.2.1 the most suitable quantizer can be selected from an available set. If the activity increases we can operate a coarser quantization without introducing visible distortions in the decoded picture. The adaptive quantization has been implemented using classes of three quantizers having the same number of quantization levels but different nonuniform quantization steps. Each quantizer in the same class is related to quite the same coding quality, which varies more within different classes.

The human visual system is less sensitive to chrominance resolution than to luminance resolution 2.4.2, thus the adaptive quantization method has not been adopted for chrominance processing too. After few tests, it became clear that a single 5-levels quantizer provides a suitable solution for both chrominance components [18].

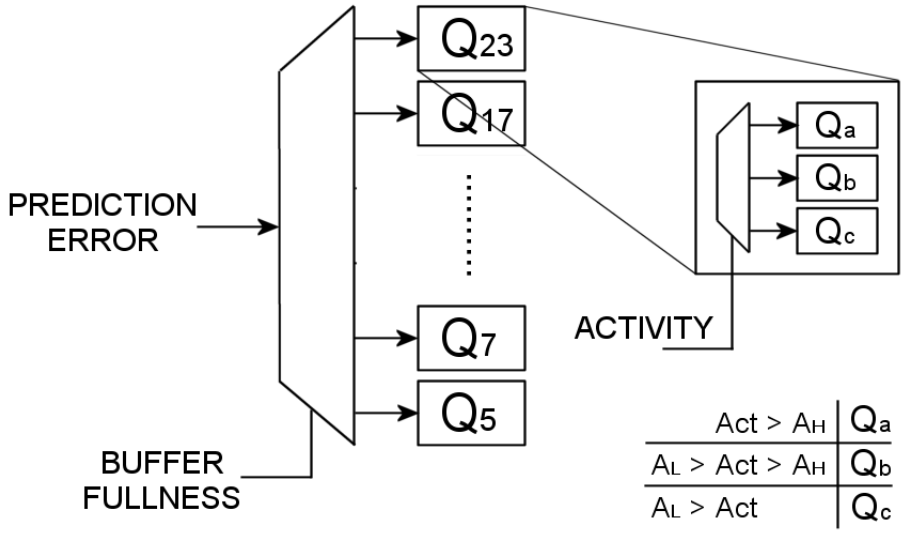


Figure 5.5: HERMES codec adaptive quantization scheme

In the HERMES codec, seven classes of quantization Q_N are used to achieve gradual control over luminance component As shown in figure 5.5, each class Q_N contains three N-levels nonuniform quantizers, which are adaptively selected using two thresholds A_L, A_H on the activity magnitude. In a rate-controlled encoding scheme, a class Q_N is selected using a feedback control from the output buffer (refer to section 2.13). In summary, HERMES provides a 21 different quantizers which can be used to adjust the encoding quality in a typical rate-controlled scheme.

The number of quantization levels determines the amount of compression possible for the DPCM coder. An optimum quantizer should have the smallest number of levels which gives no visual impairments. To investigates this limit, a set of quantizers with levels N=23, 17, 13, 11, 9,

7, 5 have been implemented. Each quantizer have been designed accordingly to a subjective design method, in which the quantization levels have been graphically determined by approximating the visibility of quantization error as a function of the prediction error [46, 47].

5.2.4 Conditional Coding

The DPCM coding algorithm is based on the transmission of the prediction error, thus the use of an efficient prediction method is critical to achieve maximum data compression. The better the prediction the lower the prediction error and consequently a smaller amount of bits are required to encode the residual information. Even a complex prediction would not always be able to give a completely reliable estimation of the signal, exploiting in every case a prediction error close to zero. However, if we could isolate the pixels related to a good predictions we could avoid transmitting their related prediction error (assumed to be very low), saving bits and improving the coding efficiency.

The conditional coding is a method that classifies the input pixel into two different groups: the pixel to be coded and transmitted and the ones that can be reconstructed only using the prediction as reliable estimation of their value. The estimation of the prediction reliability should be causal to avoid the need to send an information overhead. So we cannot base the conditional coding on the real prediction error magnitude, because at decoder side there is no way to evaluate it before its transmission. Thus, as in the case of best prediction selection 5.2.2, we have to use a method based on some approximation which therefore, can fail sometimes (resulting in an increase in the reconstruction error) but which is globally efficient.

Experiments on several images [17] show that there is a rather wide range of values for the activity function (Refer to section 5.2.1) for which the reliability of the prediction in very high. This occurs in regions of the picture where the variation rate of the luminance is low, and consequently the signal is easy to predict. Since we can tolerate some isolated coding errors in a very small percentage of pixels, in these low-frequencies regions, it is reasonable to omit the transmission of the prediction error of these pixels, resulting in a significant saving in bit rate.

HERMES codec implements the conditional coding strategy shown in figure 5.6, where a single reliability threshold A_R is introduced to make the decision. If the activity function $Act(X)$ related to the actual pixel X is $\leq A_R$ then we assume a null prediction error and we do not transmit it. To avoid further error propagation in the case of many consecutive non-transmitted pixels, we chose to always transmit the prediction error after a non-transmitted one, independently from the value of the activity function. As result, possible errors are dispersed, which reduces their visibility.

The reliability threshold introduces a further method to modulate the compression efficiency and therefore the encoding quality. In fact, if higher thresholds are selected, the probability to activate the conditional coding mechanism increases and viceversa, if the threshold is close to

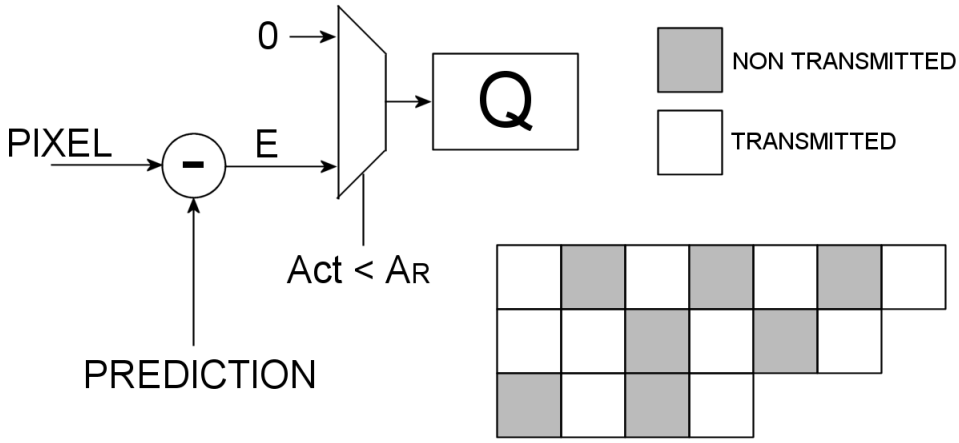


Figure 5.6: HERMES codec conditional coding scheme

0, no conditional coding will be achieved. Finally, the reliability threshold and the quantization class selection method, allow the definition of a complete quality scale which is required in a rate-controlled coding scheme.

5.2.5 Entropy Coding

To further improve the encoding efficiency, the quantized prediction error is then code-converted by using a VLC based entropy coder 2.10. The VLC set is a Huffman code [23] whose maximum word-length has been limited to take into account hardware complexity. Each code is related to a single quantizer and has been obtained using an algorithm of length-restricted codes generation [48, 49], based on modified probability distributions. The first step is to calculate the probability distributions related to the quantizer Q_N, which has been done running the HERMES software simulator with a reference natural video sequence as input. Once we have the probabilities $P_N(l_i)$, where l_i is one of the N quantization levels, we specify the maximum wanted code length and we use the already mentioned method to calculate the modified set of probabilities $P'_N(l_i)$ and the corresponding Huffman code. Despite the length restriction, the VLC codes are very close to the optimum ones. However, a loss of efficiency may occur during the coding of a sequence whose characteristics are very different from the reference one which has been employed to retrieve the probability distribution.

Figure 5.7 shows a general scheme of the luminance and chrominance encoding paths. HERMES encoder receives a 4:2:2 PAL or NTSC sequence compliant to ITU 601 standard [14]. Each video component is independently processed, so we have three main data streams: Y, U and V. Therefore at the output we still have three quantized prediction error streams $Q_N(E_Y)$, $Q_N(E_U)$ and $Q_N(E_V)$ at the same rates of the corresponding input data. If we multiplex together $Q_N(E_U)$ and $Q_N(E_V)$ we obtain a chrominance stream $Q_N(E_C)$ which has the same rate of $Q_N(E_Y)$.

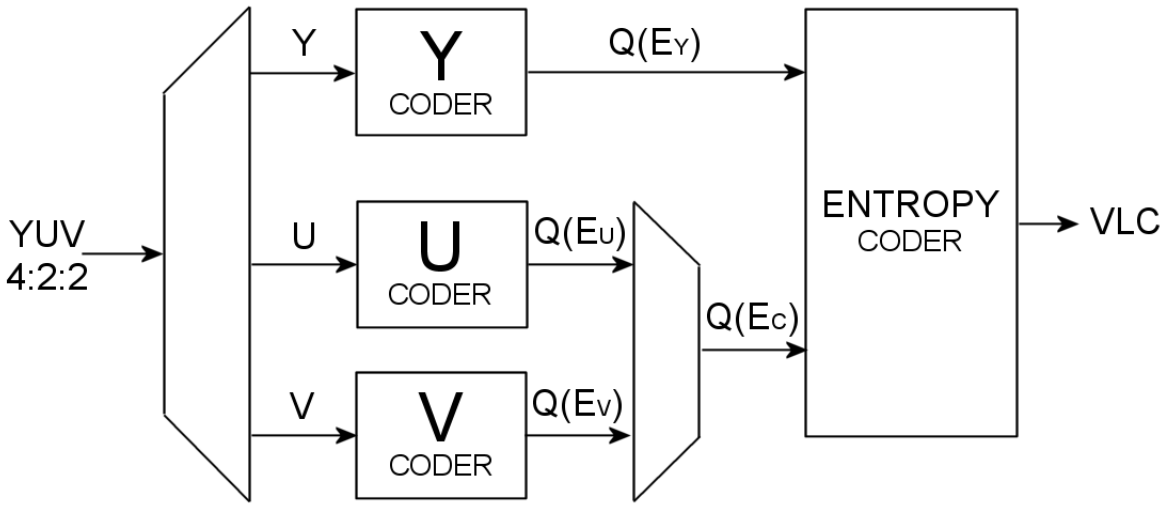


Figure 5.7: HERMES entropy coding scheme

Therefore, HERMES core outputs a couple ($Q_N(E_Y)$, $Q_N(E_C)$) every single cycle. Each quantized prediction error $Q_N(E)$ corresponds to a quantization level l_i that can be simply identified with its order number i, which is a positive integer $\in [0 \div N]$. Thus each couple ($Q_N(E_Y)$, $Q_N(E_C)$) becomes a combination of two positive integers (i, j). For that reason the entropy coding block considers each couple (i, j) as a symbol, where $j \in [0 \div 4]$ because the chrominance has a fixed 5-levels quantizer, while $i \in [0 \div N]$ with N=23, 17, 11, 13, 9, 7, 5 5.2.3.

In a natural video scene, not all the possible combinations between Y and C have the same probability. There is a set of colors which are not natural and do rarely occur. The same consideration is valid for Y values, top and bottom scale luminosity are less probable than middle scale ones. Thus, using a joined (Y, C) entropy coding, we can exploit the relationships between Y and C achieving an optimal solution. However, this method is strongly dependent on the reference video sequence that is used to generates the statistics. A better approach would be to use many reference sequences, each ones corresponding to a significative statistic model, in order to generate a set of VLC codes and achieve an adaptive Huffman coding selection. This method can further improve the entropy encoder performances and it will be object of a future optimization.
The actual HERMES entropy coder implements 7 of the 8 independent VLCs shown in table 5.1. The number of symbols required is simply: $N \cdot 5 + 1$ where an extra symbol is considered. This special symbol is the Escape Sequence Symbol (ESS) which is used as control word to terminate a coded sequence. It is important to notice that each VLC is related to a number of levels N which corresponds to three quantizers Q_N^a, Q_N^b and Q_N^c selected by the activity mechanism 5.2.3. This strategy can affect the entropy coder performances, however the VLCs related to Q_N^a, Q_N^b and Q_N^c are very close each others, and a little loss in efficiency is the price to pay to have a less complex and more robust hardware architecture. For the same reason the VLCs related to $N = 23, 17, 13$ have been constrained to a maximum code-length of 16 bits. The next table 5.2

Y QUANTIZER LEVELS (N)	C QUANTIZER LEVELS	SYMBOLS ($N \cdot 5 + 1$)	BIT per SYMBOL ε	L_{MAX} (BITS)	L_{MIN} (BITS)
23	5	116	5.3449	16	4
17	5	86	4.9247	16	3
13	5	66	4.5499	16	3
11	5	56	4.3378	15	3
9	5	46	4.0538	12	3
7	5	36	3.5825	13	3
5	5	26	3.1479	11	2
3	5	16	2.8257	9	2

Table 5.1: Number of symbols required by each Y quantizer

shows an example of VLC used for HERMES quantizers Q_5^a, Q_5^b and Q_5^c.

5.2.6 Chrominance Scrambling

According to the conditional coding method 5.2.4 in a flat zone of the picture under coding, only 50% of the pixels are coded and transmitted, which improves the codec performances. The decision to code or not to code a pixel X is based on the activity function magnitude: if $Act(X) \leq A_R$ then the area around pixel X is considered quite uniform, which means that the prediction error related to the pixels in that zone is very low, thus we do not encode X.

The activity function is calculated only on the Y components, which are not sub-sampled as the chrominance and are more significative to estimate the picture complexity. HERMES encoder outputs a couple ($Q_N(E_Y)$, $Q_N(E_C)$) which is jointly entropy coded 5.2.5. Therefore if we decide not to send the quantized residual $Q_N(E_Y) \equiv i$, then the corresponding chrominance component $Q_N(E_C) \equiv j$ is discarded as well. Thus at decoder side both luminance and chrominance components of X pixel, are reconstructed using their predictions only. This mechanism is active only one time every two pixels to avoid error propagation.

Unfortunately, the input chrominance sequence is like: $U_1V_2U_3V_4U_5V_6U_7$..... and the conditional coding method risks to selectively discard only one between U and V components within the picture zone where $Act(X) \leq A_R$. As shown in the example below, if all the odd couples are discarded, only the V component survives.

$$\begin{pmatrix} Y_1 \\ U_1 \end{pmatrix} \; Y_2 \; \begin{pmatrix} Y_3 \\ U_3 \end{pmatrix} \; Y_4 \; \begin{pmatrix} Y_5 \\ U_5 \end{pmatrix} \; Y_6 \; \begin{pmatrix} Y_7 \\ U_7 \end{pmatrix} \; Y_8 \; \begin{pmatrix} Y_9 \\ U_9 \end{pmatrix} \; Y_{10} \; \ldots\ldots\ldots$$
$$\begin{pmatrix} \\ V_2 \end{pmatrix} \; \begin{pmatrix} \\ V_4 \end{pmatrix} \; \begin{pmatrix} \\ V_6 \end{pmatrix} \; \begin{pmatrix} \\ V_8 \end{pmatrix} \; \begin{pmatrix} \\ V_{10} \end{pmatrix}$$

A simple solution to this problem is to modify the chrominance sequence input order like: $U_1U_3V_2V_4U_5U_7V_6$..... In this case the conditional coding cannot completely cancel the same

SYMBOL ID	(i, j) COUPLE	FREQUENCY	HUFFMAN CODE
1	ESS	100	00000000000
2	(0,0)	370677	00000000001
3	(0,4)	396648	0000000001
4	(4,0)	478850	0000000010
5	(4,4)	512400	0000000011
6	(1,0)	742453	000000010
7	(1,4)	794472	000000011
8	(2,0)	901808	000000100
9	(2,4)	964992	000000101
10	(3,0)	1339995	000000110
11	(3,4)	1433880	000000111
12	(0,1)	9795789	000001
13	(0,3)	10275072	000010
14	(4,1)	12654450	000011
15	(1,1)	13273600	00010
16	(3,1)	19620621	00011
17	(3,3)	20580608	00100
18	(4,3)	23831856	00101
19	(0,2)	24997888	0011
20	(1,3)	35411715	0100
21	(2,1)	37144320	0101
22	(2,3)	37367547	0110
23	(4,2)	48272350	0111
24	(1,2)	74845883	100
25	(3,2)	90910288	101
26	(2,2)	135083445	11

Table 5.2: Q_5 statistics and codes

chrominance component, as shown in the following example:

$$\begin{pmatrix} Y_1 \\ U_1 \end{pmatrix} \; Y_2 \; \begin{pmatrix} Y_3 \\ V_2 \end{pmatrix} \; Y_4 \; \begin{pmatrix} Y_5 \\ U_5 \end{pmatrix} \; Y_6 \; \begin{pmatrix} Y_7 \\ V_6 \end{pmatrix} \; Y_8 \; \begin{pmatrix} Y_9 \\ U_9 \end{pmatrix} \; Y_{10} \; \ldots\ldots\ldots$$

This operation of *scrambling* can be achieved by a simple operations of shifting and muxing. The scrambling block generates only one cycle shift and it is auto-reversible, which means that applying it to a previously scrambled sequence, the original order is restored. Therefore the scrambling block is used at encoder input to modify the chrominance sequence and at decoder output to restore it.

5.2.7 Slice Coding

DPCM systems are very sensitive to error propagation during decoding process, because each pixel is coded using the information carried by the previous ones. It is the case of the prediction function, which use the locally decoded pixels A, B, C and D to generate a prediction related to the actual pixel X, as shown in figure 5.8. Thus a punctual transmission error can be propagated over the whole sequence without stopping. To avoid error propagation it is necessary to break each video frame into smaller units called slices, as in figure 5.8. Each slice is made of a fixed number of lines, and it is completely independent on the other ones. Each slice can be decoded

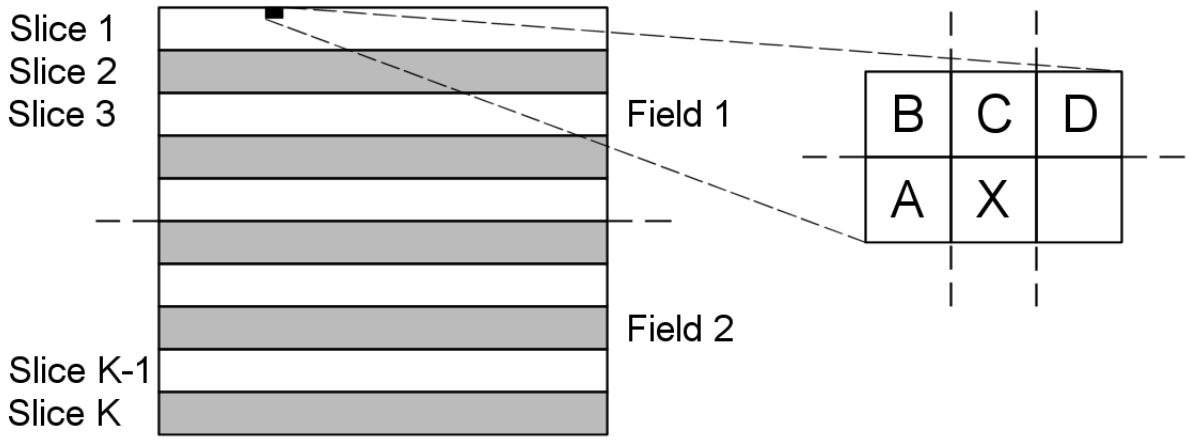


Figure 5.8: HERMES slice coding scheme

without requiring any information form previous or next one, so in the case of a decoding error, the propagation stops at the end of the current slice. At this purpose, each first line of a slice is not coded using the previous one as reference, but assuming $B = C = D = 128$, which means that a uniform middle scale line is considered. This slightly decreases the coding efficiency of each first line, because the only valid prediction left is the previous pixel A. A complete analysis on the artifact generated by this condition can be found in the *Coding Impairments* section 7.5.

In HERMES codec the slice becomes the basic coding structure after the pixel. Each slice is encoded as a different picture and using the same class Q_N of N-levels quantizers. In this way the coding quality is kept uniform inside each slice and it can vary only from slice to slice. The class of quantization N is an essential parameter needed to start the decoding process, therefore N is transmitted to decoder side as information overhead before the corresponding encoded slice data.

5.3 RATE LIMITATIONS

HERMES is a rate-controlled codec, therefore a memory buffer with a control feedback is used to smooth the VBR due to the variable picture complexity and to obtain the CBR requested by the transmission channel. This classic scheme is the same which have been introduced in section 2.13. However, we have to analyze the specific working conditions and system performances to achieve a suitable solution.

The buffer size is a key point (refer to section 4.2.4) to obtain a low coding/decoding delay. HERMES codec introduces a latency of few clock cycles, thus the whole coding/decoding delay D depends on the buffer size B and the requested constant bit rate R as described by the already mentioned equation 5.11.

$$D = \frac{B}{R} \tag{5.11}$$

In LiveTools [11] DVB-T basic system framework, the maximum utilizable bit rate is around 23 Mbps, which corresponds to a 64-QAM modulation at code rate 2/3 and guard interval 1/16. If we consider 2 Mbps of bandwidth overhead for both audio and system layer, we get a maximum video encoding rate of 21 Mbps. Therefore the first step is to investigate the best trade-off between quality and delay at video bit rates $R \leq 21$ Mbps.

To have a first idea of the problem we can estimate the minimum coding efficiency ε (expressed in bit per symbol), which is required to satisfy the rate constraint. HERMES encoder associates each input couple (Y, C) to a symbol (i, j) which is then entropy coded, so if the frame dimensions are L and H, there are $L \times H$ symbols per frame. Given a frame rate P and a conditional coding gain of ν (not transmitted symbols), the equation 5.12 describes our upper bounding condition.

$$R = \varepsilon(L \cdot H)(1-\nu)P \leq R_H \tag{5.12}$$

$$\varepsilon \leq \frac{R_H}{(1-\nu)(L \cdot H)P} \tag{5.13}$$

In the case of a standard PAL sequence (L=720, H=576 and P=25 Hz) and supposing that 30% of symbols are not transmitted (ν=0.3) we get $\varepsilon \leq 2.89$ bit per symbol. Looking at table 5.1 we discover that the coarsest 3-levels quantizer class only, can satisfy the requirement of $\varepsilon \leq 2.89$. Therefore $R_L \simeq 22$ Mbps is the theoretic HERMES bottom bit rate limit, which corresponds to

the lowest coding quality. Moreover, this condition does not match with the standard DVB-T upper bit rate limit of 23 Mbps, which is too close to R_L.

However, a more efficient version of the LiveTools DVB-T system is currently under development ant it will be able to support up to 60 Mbps using an enhanced OFDM modulation scheme [2]. This larger bandwidth makes it possible to integrate HERMES into LiveTools system in order to exploit the ULD features at rates $R > R_L$ corresponding to higher coding qualities. Therefore, it is very important to study HERMES codec limits in order to improve its performances and optimize its design.

5.4 BUFFER and DELAY

R/P bit is the average size of a coded frame, which is made by a number H/l of elementary coding structures called slices, where l is the fixed number of video lines per slice. Therefore, we can model the buffer size B as a multiple n of the encoded slice size, as described in equation 5.14.

$$B = n\left(\frac{R}{P}\right)\frac{l}{H} \tag{5.14}$$

$$D = \frac{B}{R} = \frac{n \cdot l}{P \cdot H} \tag{5.15}$$

As direct consequence of equations 5.11 and 5.14 we get the expression 5.15, which describes the delay D in terms of geometrical factors only. The number of line per slice l cannot be too small, as the more slice per frame we use, the more information overhead we have to introduce (each slice has its own fixed information header) to achieve a correct decoding process. Moreover, each slice introduces a coding inefficiency on its first line, thus using few lines per slices we decrease the codec performances. On the other hand, a high l can vanish our efforts to quickly break the error propagation.

Before making any estimation on n, we need some hypothesis on the encoded slice size. Using the medium-high N= 17 quantization class (ε=4.9247), and with l=8, we can estimate the amount of bit required to encode a slice. Using equation 5.12 and given that the number of slice per frame is H/l, we obtain the following expression for the encoded slice size S_{ENC}.

$$S_{ENC} = \varepsilon(l \cdot L)(1 - \nu) \tag{5.16}$$

In the worst case we can consider that the conditional coding is never active ($\nu = 0$) and we can increase S_{ENC} by 20% to take into account a certain entropy coding inefficiency. What we get is $S_H \simeq 35$ Kb. In the best case we have 50% of conditional coding ($\nu = 0.5$) and we can consider $\varepsilon = 3$, which is the shortest code-length for N= 17. Thus we get $S_L \simeq 9$ Kb. So the suitable buffer size B must be greater than 26 Kb, which corresponds to an average slice size calculated with ε=4.9247 and $\nu \simeq 0.1$.

The variation $\Delta S = S_H - S_L$ itself is not enough to determine the buffer size B, we need to know in which time interval ΔT it may occur. HERMES buffer control can change the encoding quality (quantization class N) only at the slice beginning, so a rapid variation $\Delta S/\Delta T$ could not be compensated and may cause overflow. A simple condition to be satisfied is that the maximum variation cannot overcome half of the buffer size, as described in equation 5.17, where $R\Delta T$ is the amount of bit output at constant rate R in the time interval ΔT.

$$\frac{B}{2} \geq \Delta S - R\Delta T \tag{5.17}$$

HERMES generates a quite high bit rate even in the best coding conditions, because at least 50% of the pixels are always coded and transmitted. Thus, VBR $\geq R_{MIN}$, where R_{MIN} is the bottom rate limit which can be estimated by equation 5.12. Therefore not all the CBRs are allowed and a simple but coarse condition is that $R > R_{MIN}$ as well the VBR. If we suppose that ΔT is not shorter than the slice time and we consider the worst case $R = R_{MIN}$, then in equation 5.17 we can use $R_{MIN}\Delta T = S_L$, which leads to the following bounding condition for B.

$$B \geq 2(S_H - 2S_L) \tag{5.18}$$

In the really worst case we can assume that no compression is achieved, thus $S_H \simeq 92$ Kb which corresponds to 8 lines of 1440 input data bytes. Using equation 5.18 we get $B \geq 148$ Kb, which corresponds to $n = 6$ if we consider an average slice size of 26 Kb.

Therefore we can state that a buffer size of $\simeq 150$ Kb is enough to smooth the HERMES variable data rate. Using equation 5.15 we get $D \simeq 4$ ms at medium-high coding quality (N= 17), which is a very good target compared to the minimum $D = 54$ ms actually achievable by the best MPEG-2 system implementation (4.3); despite the high bit rate of 37.5 Mbps resulting from equation 5.15.

5.4.1 Buffer Control Strategy

The feedback control system has to guarantee the requested output CBR avoiding conditions of buffer overflow/underflow. Every slice beginning, the control selects the suitable quantization class Q_N in order to modulate the amount of bit generated by HERMES core. Thus, the feedback is not continuous in time, as it acts every $T_k = k(\Delta T)$ where ΔT is the slice time and $k \in \mathbb{N}$.

As shown in figure 5.9, the control is based on two main variables: the actual *buffer fullness* B_F and the corresponding *arrival rhythm* $\Delta B_F/\Delta T$. There are two kinds of possible control schemes, which depend on the working conditions:

- **Soft** - The buffer fullness B_F is quite stable around half the buffer size $B/2$ (Gray zone of figure 5.10), the arrival rhythm module is bounded and its sign is not always positive. In this condition we expect not visible differences within the coding quality of consecutive slices.

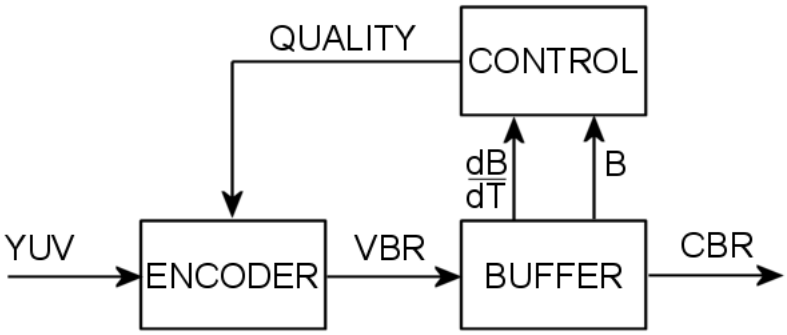


Figure 5.9: General buffer control feedback scheme

- **Hard** - The buffer fullness B_F is out a safe zone within $B/2 \pm aB$ and the arrival rhythm is still positive or negative, so we have to react quickly in order to avoid a buffer overflow/underflow condition. In this critic situation a rapid quality variation is performed and the sequence can be affected by quantization noise.

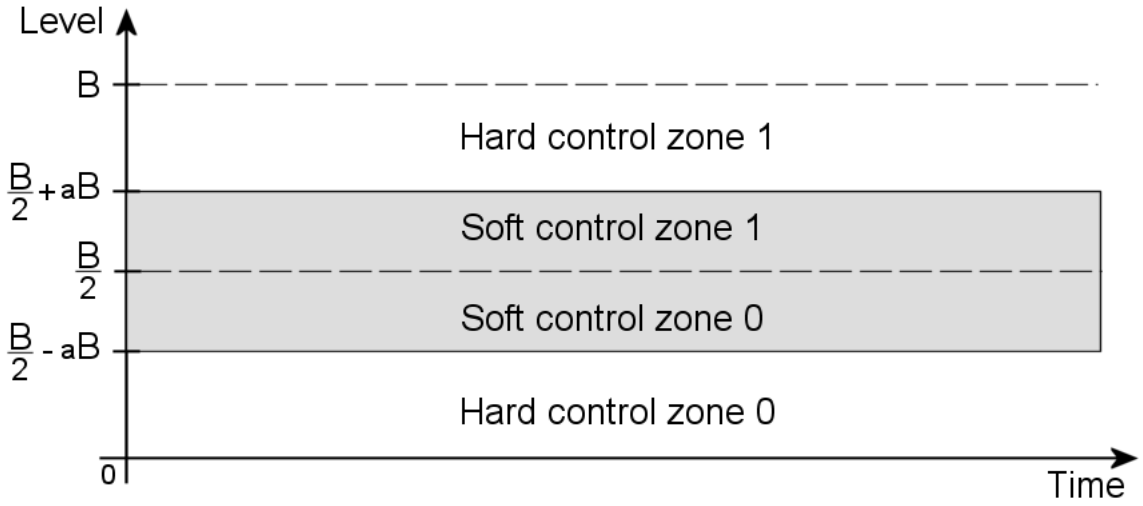


Figure 5.10: Buffer hard/soft control zones definition

If the system is well designed then the soft control works for most of the time and the encoding quality is gradually distributed in a clever way. The coded sequence have to be fluid, without any visible artifact due to slice structure or encoding quality variation within slices, both in time and space domain. The hard control should act for short times only, in high critical conditions due to a very stressing and unusual sequence.

The encoding quality allocation is a delicate problem. Each field is made by a number of consecutive slices, whose complexity changes. If the coding quality between consecutive slice in the same frame is varied too much then a disagreeable visual effect is introduced. The same

limitation occurs between slices of consecutive fields in corresponding positions. Therefore, there is a double space/time constraint on the slice quality allocation which can limit the control efficiency.

Once the requested channel rate R is fixed, we can calculate the reference slice occupancy S_{ref} which corresponds to the amount of bits requested during the slice time. S_{ref} is expressed in equation 5.19, where H and l are the number of lines per frame and per slice, while P is the frame rate.

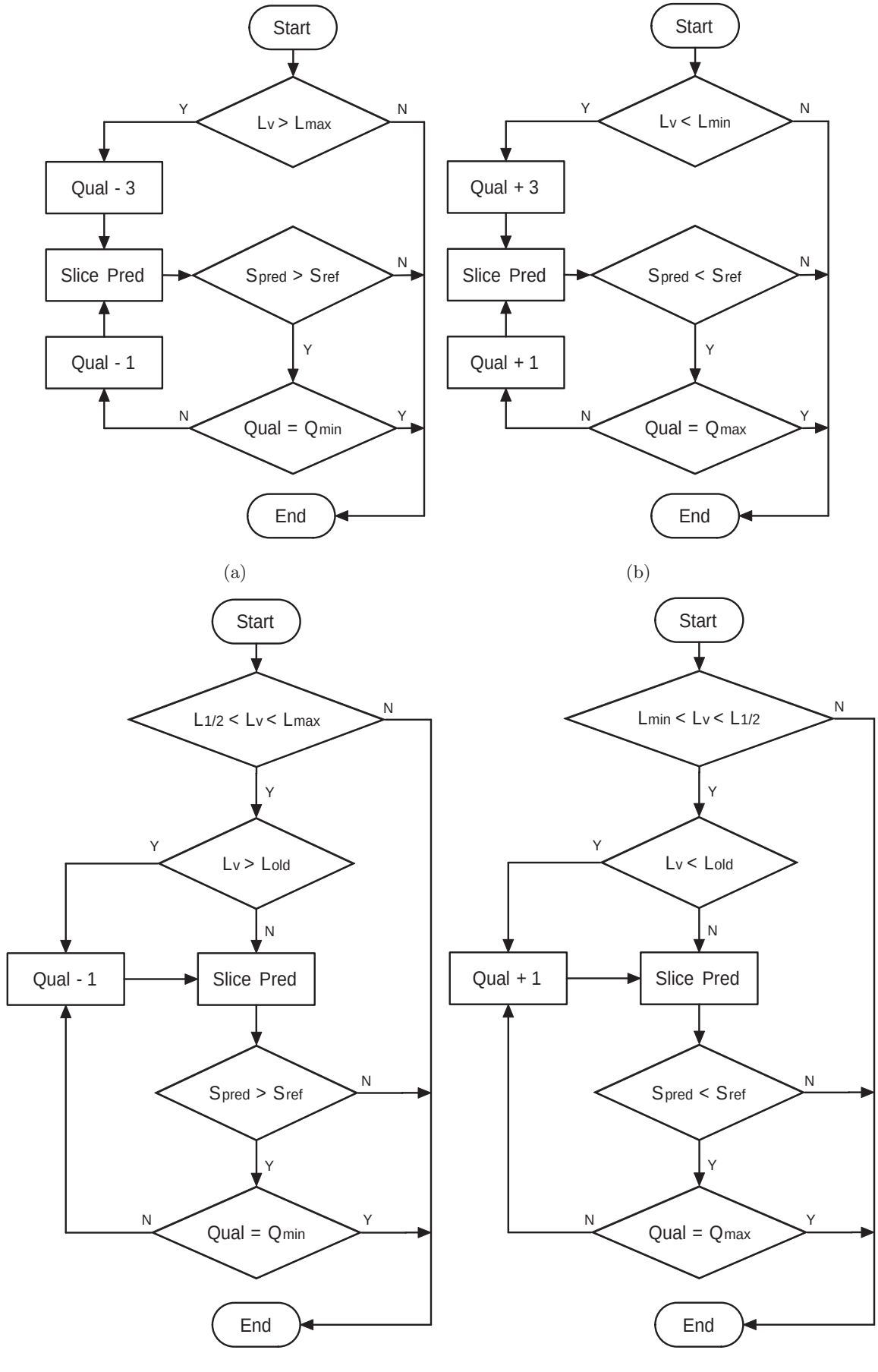


Figure 5.11: Buffer control strategy: (a)-(b) Hard control, (c)-(d) Soft control

$$S_{ref} = \frac{Rl}{HP} \tag{5.19}$$

If we force HERMES to generate encoded slices whose occupancy is close to S_{ref}, then we get a controlled constant level in the buffer.

The encoding quality is the only parameter which allows to control how many bits per slice are generated by the HERMES encoding process, thus there is a direct relation between quality and compression ratio. This relation can be easily investigated in order to characterize the system performance (refer to section 7.2) and to describe a simple HERMES encoder model which allows us to make predictions about next slice occupancy. So, equation 5.20 gives us a simple estimation of the amount of bits required to encode a slice at selected encoding quality q, where $r(q)$ is the related compression ratio and S_{raw} is the amount of *raw* bits per non-encoded slice.

$$S_{pred} = \frac{S_{raw}}{r(q)} \tag{5.20}$$

The buffer control strategy is described in figures 5.11(a)(b)(c)(d) by four flow charts, each one associated to a different control zone defined in figure 5.10. L_v is the buffer level occupancy while L_{max} and L_{min} correspond to the edges of the gray zone in figure 5.10.

In the case $L_v < L_{min}$ we are in the hard zone 0 and the control specified by the flow chart 5.11(b) is active. The first action is to increase the quality by a step of 3, in order to rapidly react to the lack of input bits. Then S_{pred} is calculated and compared with S_{ref}. The control keeps on increasing the encoding quality q until $S_{pred} > S_{ref}$, which means that L_v becomes greater.

When $L_v > L_{min}$ while we are still below the middle buffer level $L_{\frac{1}{2}}$, we are into soft zone 0 (figure 5.10) and the control flow chart 5.11(d) is active. First of all, the arrival rhythm sign is checked. If it is negative, the control preventively increases the quality by 1 to avoid L_v fluctuations around a constant level. Then S_{pred} is calculated and the control behaves exactly as described in the previous case, trying to increase L_v in order to reach $L_{\frac{1}{2}}$. As L_v overpasses $L_{\frac{1}{2}}$, the control described in figure 5.11(c) reacts in order to decrease L_v towards $L_{\frac{1}{2}}$.

Finally this kind of control increases the coding quality q until $L_v \simeq L_{\frac{1}{2}}$ and then it tunes q around an average coding quality $q^{'}$ which corresponds to S_{ref}, attempting to keep L_v close to $L_{\frac{1}{2}}$. The fluctuations $q - q^{'}$ are the direct consequence of the different amount of information carried by consecutive slices. Therefore more complex slices are coded at qualities lower that $q^{'}$, while less complex slices are coded at qualities higher than $q^{'}$.

To take into account the space/time constraint the previous employed encoding qualities q are stored and each first slice of a new field is encoded using a quality which takes into account the related one employed in the past corresponding slice.

In order evaluate this control strategy, the definition of both a quality scale and the corresponding compression ratio $r(q)$ is required. However, a complete analysis is carried out in section 7.2. In the next section a further control strategy is presented. The following method use a second

HERMES encoder which works in parallel, as fixed quality reference, in order to make predictions on quality allocation.

5.4.2 Constant Quality Reference

A method to distribute the slice quality in a clever way is based on the estimation of the picture complexity, which allows an accurate prediction of the incoming slice occupancy, making it possible to find the best trade-off between the selected quantization class N and the slice complexity (in terms of spatial details). At this scope we have to define a complexity function $\mathbb{C}(k)$ where $k \in \mathbb{N}$ is the slice position within the field. Since consecutive fields are high correlated, $\mathbb{C}(k)$ can be calculated for every slice in a field and used to determine the suitable coding quality for the corresponding slice in the next field.

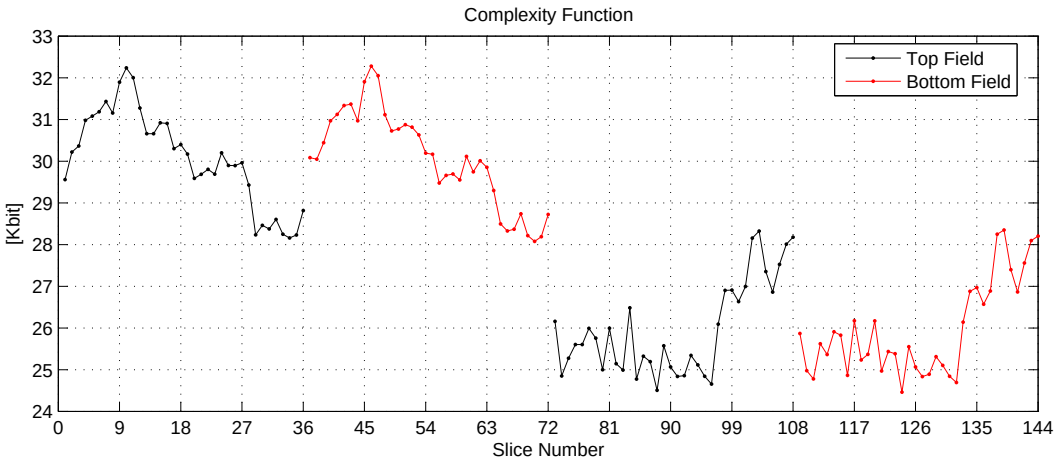


Figure 5.12: Not-normalized complexity function

If the requested output rate is R and the frame rate is P, then the average amount of bit per encoded field is simply given by equation 5.21, where T_f is the field time.

$$\bar{F}_{ENC} = R \cdot T_f = \frac{1}{2}\left(\frac{R}{P}\right) \tag{5.21}$$

To evaluate $\mathbb{C}(k)$ a reference encoder which works in parallel and at constant quality N can further store the amount of bit $S_H(k)$ needed to encode each slice k. Figure 5.12 shows an example of function $S_H(k)$, which is itself a complexity evaluation. Making a normalization we get equation 5.22, where the sum is done on the slices in the same field.

$$\mathbb{C}(k) = \frac{S_H(k)}{\sum_k S_H(k)} \tag{5.22}$$

Therefore $S_H(k)$ can be accumulated at every single slice end to calculate the total amount of bit requested to code each field at constant quality. Using these information the buffer occupancy at

time t_k can be predicted as described in equation 5.23.

$$B_N(t_k) = B_N(t_{k-1}) + \lambda S_H(k) - R\Delta T \tag{5.23}$$

where

$$\lambda = \frac{\bar{F}_{ENC}}{\sum_k S_H(k)} \tag{5.24}$$

The constant λ is the ratio between the average bit per field $\bar{F}_{ENC}$ and the real amount of bit needed to code the previous field at quality N. This prediction is possible only after the first field processing. If we sum both sides of equation 5.23 over the whole field, we get equation 5.25, where the sum of ΔT is simply the field time T_f. Using equation 5.24 and 5.21 we obtain that the final fullness $B_N(t_f)$ is equal to the initial $B_N(t_0) = B_0$ calculated on the previous field, as the term between braces in equation 5.26 is null.

$$B_N(t_f) = B_N(t_0) + (\lambda \sum_k S_H(k) - R \sum_k \Delta T) \tag{5.25}$$

$$B_N(t_f) = B_N(t_0) + (\bar{F}_{ENC} - R \cdot T_f) \tag{5.26}$$

$$B_N(t_f) = B_0 \tag{5.27}$$

Therefore, the predicted buffer fullness does not overflow/underflow but it is kept constant over each field to a value B_0 that we can impose to be $B/2$, half the buffer size. To perform the control we can use the difference ΔB between the predicted fullness and the real one.

$$\Delta B(t_k) = B_N(t_k) - B(t_k) \tag{5.28}$$

A simple control can be achieved using $N = 17$ as reference quality, defining three different zones related to a coefficient $\alpha < 1$ which can be tuned to calibrate the system.

$$\Delta B(t_k) > \alpha B \implies N = 23 \tag{5.29}$$

$$-\alpha B \leq \Delta B(t_k) \leq \alpha B \implies N = 17 \tag{5.30}$$

$$-\alpha B > \Delta B(t_k) \implies N = 13 \tag{5.31}$$

If the requested rate $R < \bar{R}_{13}$ where $\bar{R}_{13}$ is the average rate related to a coding quality $N = 13$, than the control cannot avoid the overflow. In a similar way the control fails when $R > \bar{R}_{23}$, and an underflow condition occurs. The maximum control efficiency expected for $R \simeq R_{17}$.

During normal operation, the real buffer occupancy $B(t_k)$ is forced to follow the predicted one and the control preserves the same average buffer fullness at the end of each field. What is more, any risk of fullness drifting is avoided because a new prediction is calculated for every field. The cost of the parallel reference encoder is not too high, because HERMES encoder does not requires many resources, thus the constant quality reference may be an alternative way to implement a soft control method.

In the case of a scene change, the prediction based on the previous field can be very inefficient, thus a hard control may be implemented to avoid critical conditions. For this purpose, the method described in previous section can be employed, for the hard zones only (Refer to figure 5.10 and flow charts 5.11(a)(b)).

5.4.3 Vertical Blanking Discontinuity

The ITU-R Recommendation BT.601-5 [14] takes into account horizontal and vertical blanking periods (Refer to figure 5.13), which correspond to the time intervals where the old analog synchronism took place. During blanking time there are no active video samples, therefore in digital video applications these intervals often carry additional information such as audio or teletext.

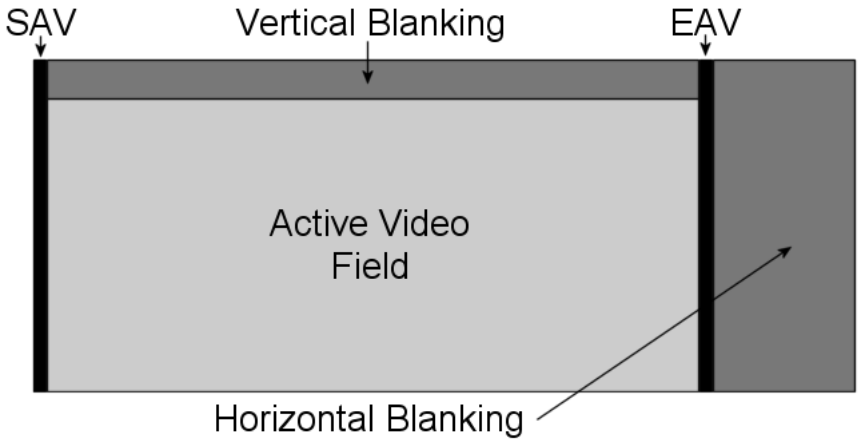


Figure 5.13: Video blanking intervals

From the buffer control point of view, the blanking periods generate a discontinuity of the input data rate, which suddenly falls to zero, while the output rate is still R. During blanking the buffer occupancy level decreases and the control feedback cannot react in any way because there is no input data at all. As a result the buffer is affected by regular fluctuations which reduce the control efficiency. The amplitude of these fluctuations depends on the requested data rate R and the blanking extension itself, which are listed in table 5.3.

	H BLANK (Words/Line)	V BLANK (Lines/Frame)	TOTAL WORDS PER FRAME	EQUIVALENT RATE (Mbps)
PAL	282	49	246810	58.2
NTSC	268	39	198860	55.7

Table 5.3: Horizontal and vertical blanking data space

Despite the horizontal blanking giving the main contribution to the equivalent rate of table 5.3, it has a short duration and it is distributed over all the entire frame (at each line end).

Therefore the horizontal blanking cannot seriously affect the buffer fullness. This is not the case of the vertical blanking which can take up to 24 consecutive lines between adjacent fields (PAL format). As one line takes 64 μs in both PAL and NTSC systems, the maximum possible buffer fluctuation is given by equation 5.32.

$$\Delta B = \frac{1.536R}{1000} \tag{5.32}$$

At 50 Mbps we get 76.8 Kb, which is a further limitation to the minimum buffer size.

The periodic buffer variation ΔB occurs every field time (20 ms and 16.5 ms for PAL and NTSC) have to be compensated by the control method, in order to avoid that vertical blanking may affect the coding quality. In fact, when the buffer level rapidly decrease of ΔB, the control could react by increasing the coding quality of next field in order to fill the buffer again. As result we would get a vertical coding quality variation which is not only related to the image complexity but also to vertical blanking mechanism.

A simple way to compensate this effect is to consider dynamic thresholds for both soft control zones and buffer half level, as shown in figure 5.14. The employed ramp function gets its maximum at every vertical blanking start, then it reaches its original level decreasing of the amount ΔB. In this way the vertical blanking interval is exploited in order to support a slightly higher average coding quality, at the cost of periodic fluctuations of the buffer level.

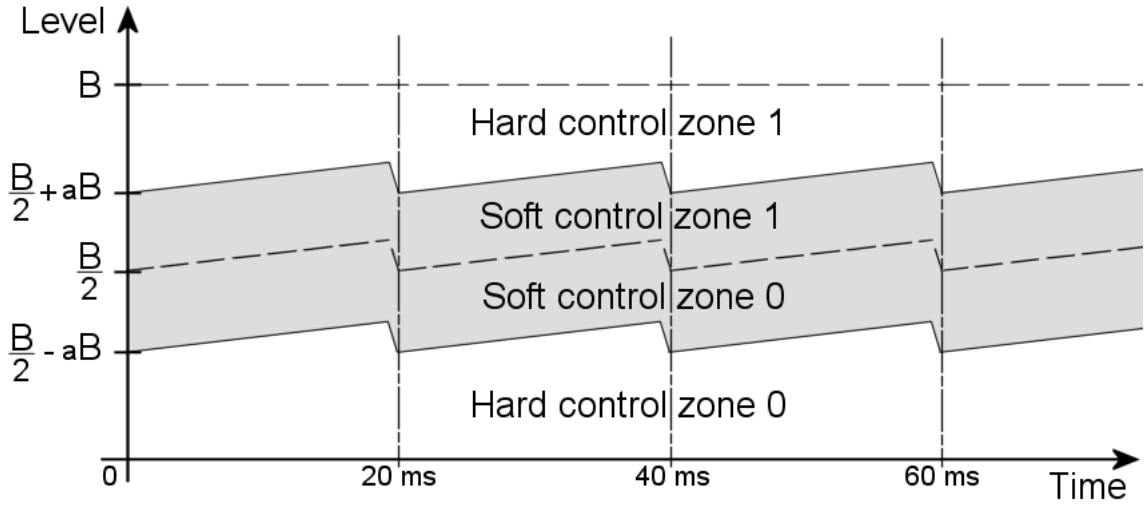


Figure 5.14: Buffer hard/soft control zones definition with blanking compensation

Finally both the previous buffer control methods (Refer to sections 5.4.2 and 5.4.1) have to be integrated with the vertical blanking compensation scheme of figure 5.14. In these conditions both methods can achieve a correct control if a suitable buffer size is employed.

5.5 SYSTEM LAYER

The MPEG system layer describes how elementary streams comprising one or more programmes are multiplexed together to form a single data stream suitable for digital transmission or storage. HERMES design adopts a MPEG-like system layer which has been conceived in order to carry timing and other information needed to de-multiplex the audio and video streams and to synchronize audio and video during playback.

For practical purposes, the continuous elementary streams carrying audio or video (compression layer) need to be broken into PES packets. These packets are identified by headers that contain overhead information. Audio and video PES packets are then multiplexed together to create a TS (Transport Stream) which is more suitable for transmission purpose.

The output of HERMES multiplexer is a contiguous TS of 8-bit-wide data bytes. The data rates must at least equal the total of the combined data rates of the contributing elementary streams which cannot overhead the total fixed channel data rate. The residual bandwidth is covered by stuffing TS packets.

5.5.1 Synchronization

HERMES provides a timing mechanism that ensures synchronization of audio and video. The implemented method is based on MPEG standard and two parameters have been taken into account: the programme clock reference (PCR) and the presentation time stamp (PTS, refer to 2.12). The time reference base runs at 90 KHz, which is a sub-multiple of the 27 MHz video clock. PCR and PTS values are expressed using 33 bits words, which allows to represent any clock cycle in a 24-hour period.

A PCR is a snapshot of the encoder system clock which is sent at decoder side using the system layer syntax. During decoding, PCRs values are used to adjust the system time clock (STC), which is a simple constantly running binary counter. As shown in figure 5.15 a feedback control based on an external VCXO is implemented at decoder side to achieve the clock synchronization. Finally, the PCR mechanism makes it possible to generate a copy of the encoder system clock at decoder side, which means that encoder and decoder can use a common time reference.

A PTS is a samples of the encoder system clock and it can be associated to both video and audio encoded data. The PTS represents the time at which the video picture has to be displayed or the starting playback time for the audio sequence. A constant PTS offset is added to the original PTS in order to delay the presentation, which is necessary to compensate the decoding time.

As shown in figure 5.15, we can receive an audio or video PTS. Every time that the STC matches the PTS, a START pulse is generated. Using this START signal we can directly enable the output of the corresponding audio frame or video slice.

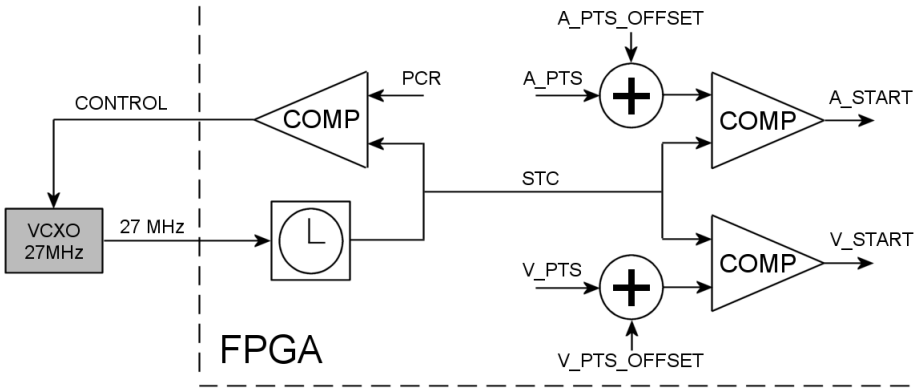


Figure 5.15: HERMES audio/video synchronization

5.5.2 Packetised Elementary Stream

The first packetization layer is the PES, in which the basic elementary stream (audio and video encoded data) is divided into packets of variable size up to few hundred kilobytes. Each packet is preceded by a PES packet header, which always begins with a START-CODE of 24 bits and a stream ID that identifies the contents of the packet as video or audio. The structure of a HERMES audio and video PES packet header is shown in figures 5.16 and 5.18.

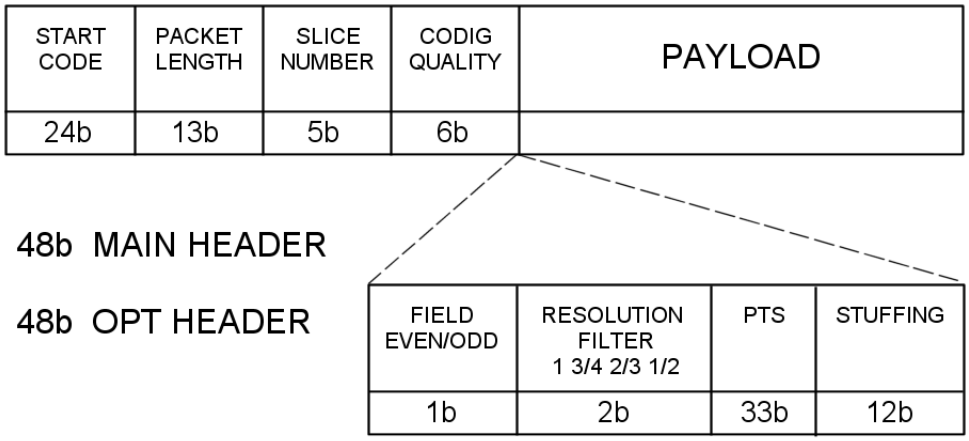


Figure 5.16: HERMES video elementary stream packet

Each video PES packet (Figure 5.16) carries one coded slice in the PAYLOAD section, whose length is specified by the PACKET LENGTH in units of 16 bit words. A maximum PAYLOAD of 128 Kb is allowed. The SLICE NUMBER is a continuity counter which gives us the display position of the coded slice. A maximum of 32 slices per field is allowed. However if a greater number of slices is required, it is still possible to increase the SLICE NUMBER field, decreasing the

PACKET LENGTH. Finally, the primary header ends with the CODING QUALITY parameter, which specifies the quantization class and the reliability threshold. If the PES packet carries the first slice of the field (SLICE NUMBER = 0) then the MAIN HEADER is followed by an OPTIONAL HEADER which contains the PTS and some additional information.

While the video PES packet length is variable, depending on the amount of information in the slice, the audio PES packet has a constant size because in HERMES design we directly pack the audio samples without encoding. Common implementations of MPEG layer 3 audio codec introduces a delay close to 80 ms which vanishes all the efforts to keep a low delay on video coding side. A MPEG layer 2 codec performs lower delay, but still not acceptable in our design. In the present implementation the raw audio samples are sent at decoder side without any processing, however a more efficient solution will be developed in future.

HERMES codec supports a single stereo channel sampled at a rate of 48 KHz. The PES audio packet payload structure described in figure 5.17 collects 384 samples of 24 bit each from both left and right channels for a total of 4 ms of audio. If we consider an audio PES packet as a frame then we get 10 audio frames each one video frame (In PAL format only).

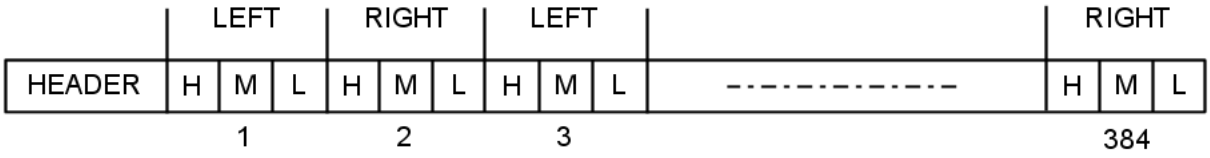


Figure 5.17: HERMES audio PES packet payload

The required audio bandwidth for a mono 48 KHz channel is 1.152 Mbps. Therefore, if R is the global transmission rate in unit of Mbps, then $200/R$ gives the percentage of bandwidth required by a stereo audio channel. For example, at $R = 40$ Mbps about 5% of the bandwidth is used to carry audio samples.

Figure 5.18 shows the header structure for a generic audio PES packet. The START CODE is followed by the FRAME NUMBER that is a continuity counter similar to SLICE NUMBER for video. Next 4 bit are used to specify the SAMPLE RATE which is 48 KHz by default. The audio PTS is carried by the OPTIONAL HEADER, which appears only when FRAME NUMBER = 0.

As we can notice from figures 5.16 and 5.18, both video and audio PES packets have a 48 bit MAIN HEADER and a 48 bit OPTIONAL HEADER. This has been conceived to simplify the packing/de-packing hardware, which works with 16 bit data busses.

5.5.3 Transport Stream

HERMES adopts the standard MPEG TS structure in order to exploit the same input/output interface of MPEG decoders/encoders, which is a requirements for the integration into MPEG-

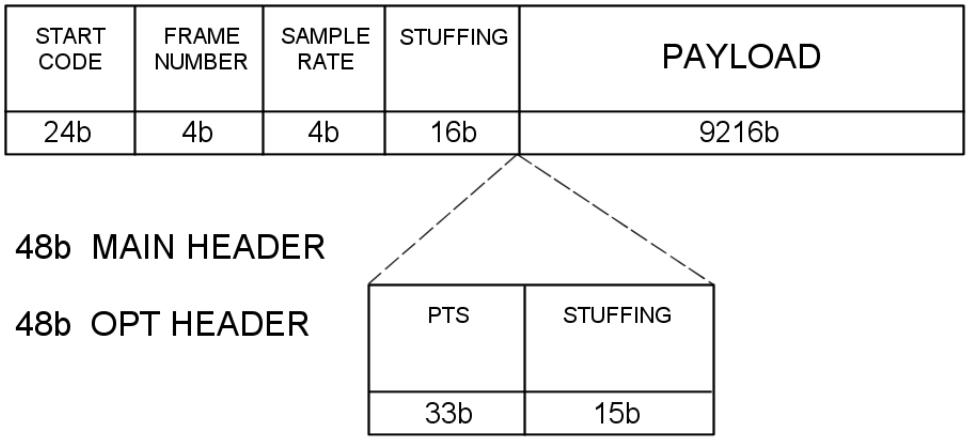


Figure 5.18: HERMES audio PES packet

based LiveTools DVB-T system. The TS consists of short fixed length transport packets which are always 188 bytes long. Figure 5.19 shows the generic structure of an MPEG TS packet, which comprises a 4 byte header followed by an adaptation field or a payload or both. The first byte of the packet header is the SYNC BYTE which get the constant value 47 (hex).

In a TS, each PES packet from the various elementary streams is divided among the payload parts of a number of transport packets according to the following constraints:

- The first byte of each PES packet must become the first byte of a transport packet payload. The START INDICATOR bit is set in the TS header if the first byte of the payload is also the first byte of a PES packet.

- Only data taken from one PES packet may be carried in any one transport packet. A PES packet is unlikely to fill the payloads of an integer number of transport packets, exactly. Thus the excess space is deliberately wasted by including an ADAPTATION FIELD of appropriate length for the final transport packet.

HERMES TS is designed to carry a single programme, which comprises one video and one audio packetised elementary stream. The 13 bit Packet Identifier (PID) field is used to distinguish within different types of transport packets as for instance: video, audio, stuffing and PCR packets.

The CONTINUITY COUNTER is incremented between successive transport packets belonging to the same elementary stream. This enables a decoder to detect transport packets loss and conceal the errors that can result from such an event.

5.6 CONCLUSIONS

The first part of this chapter has described HERMES: a DPCM-based scheme which achieves a low complex and ULD video compression algorithm, exploiting both conditional coding and slice

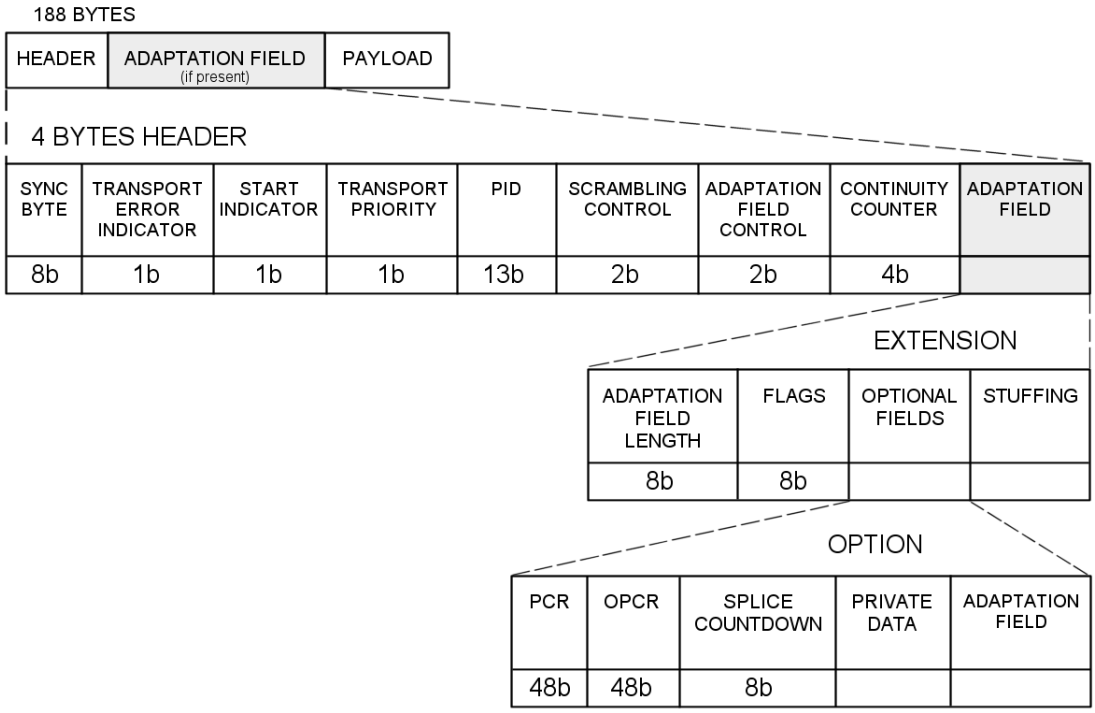


Figure 5.19: HERMES transport stream packet

coding techniques.

HERMES is an alternative to MPEG for applications which require a very short encoding/decoding latency and that have access to bit rates $R > 22$ Mbps for data transmission. Thanks to the last development of an enhanced OFDM modulation scheme [2], the integration of HERMES into LiveTools DVB-T system is achievable, which means a valid solution for wireless studio production applications.

The buffer size and its control is a key point to obtain a short coding/decoding delay. HERMES codec introduces a latency of few clock cycles, thus the whole coding/decoding delay D depends on the buffer size B and the requested constant bit rate R.

Some estimations about the required buffer size have been carried out, despite a strong dependency on the employed slice size. In the case of 8 lines per slice, a buffer of at least 150 Kb is enough to smooth HERMES output VBR to a constant rate close to 37.5 Mbps achieving a medium-high coding quality (N=17). A latency $D < 4$ ms is achievable in this working conditions.

Two different buffer control strategies have been proposed and analyzed taking into account a technique to compensate the blanking interval, which otherwise would introduce negative effects on the control strategy.

HERMES system layer have been defined according to a simplified MPEG model. A PCR/PTS based mechanism have been proposed in order to achieve audio/video synchrony. Finally the packetization of both elementary and transport streams have been discussed.

Chapter 6

FPGA Implementation

6.1 INTRODUCTION

In this chapter we describe the HERMES implementation on FPGA platform. We analyze the architecture of HERMES encoder and decoder, taking into account the main blocks and their working conditions. Each module described in this section has been developed, simulated and synthesized in VHDL using Xilinx ISE software tools. A prototype working version of HERMES codec has been implemented on a development Virtex II platform, which represents a first validation of the proposed architecture.

6.2 CLOCK REQUIREMENTS

The main video clock is 27 MHz, however the encoder design requires a further 108 MHz auxiliary clock in order to properly drive the quantization block. This high frequency clock is achieved using a Digital Clock Manager (DCM), which is a specific FPGA block that offers powerful clock management features, such as flexible clock multiplication and division.

The encoder input and decoder output need the 27 MHz video clock, while luminance and chrominance data are internally processed into two separated and parallel streams. Therefore a clock of 13.5 MHz is employed to drive most of the encoding/decoding logic. The advantage of using a halved clock is a more relaxed time constraint on combinatory paths, which is a critic point in HERMES codec. On the other hand we have a synchronization problem, because the phase relationship between the main video clock and the halved one must depend on the input CCIR data stream, otherwise we introduce a shift between the luminance and chrominance stream.

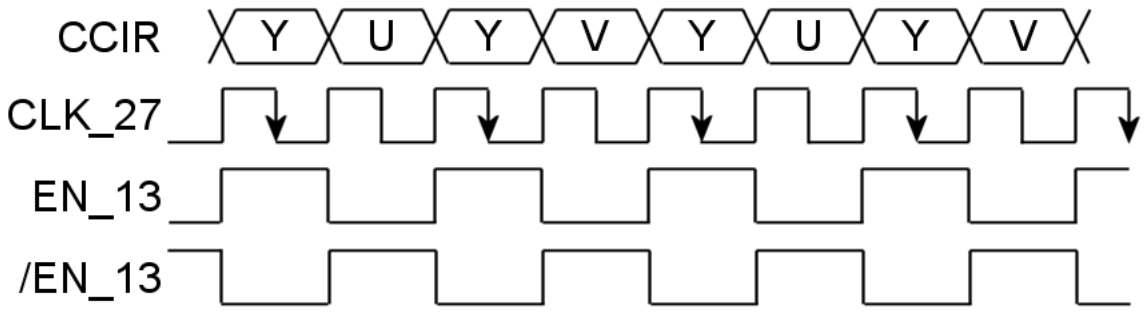


Figure 6.1: HERMES 13.5 MHz clock enable

Instead of using dedicated clock resources we can generate an enable signal $EN13$, as shown in figure 6.1. A generic signal as $EN13$ is more flexible than a clock signal and can be driven by any general purpose logic. In figure 6.1 $EN13$ selects the sequence of falling edges related to the 27 MHz video clock required to sample the luminance data. However, there are always two possibilities: $EN13$ and $/EN13$. In order to associate the right enable to the luminance and chrominance processing logic, the embedded synchronization words in the input 4:2:2 data stream

(SAV and EAV [50]) are used.

6.3 HERMES ENCODER

A general scheme of the developed HERMES video encoder core is shown in figure 6.2. Each block represented in the scheme corresponds to a VHDL module.

The input data stream is assumed to be 4:2:2 YUV format, ITU-R 656 compliant. The module DEMUX receives the raw video data and generates two parallel streams: chrominance C and luminance Y, together with their respective $EN13$ and $/EN13$ enable signals. A further important task of DEMUX is to read the SAV/EAV words embedded in the YUV stream, in order to generate some synchronization signals (SYNC) needed for the correct working of all the processing logic. The main SYNC information are related to the start/end of each active line and slice. Thus the slice structure is configured by DEMUX module, which requires some geometric parameters as: the number of samples per line, the number of lines per slice and the number of lines per field. All these parameters are charged into the VHDL code as combinatory constants which can be modified in a flexible way.

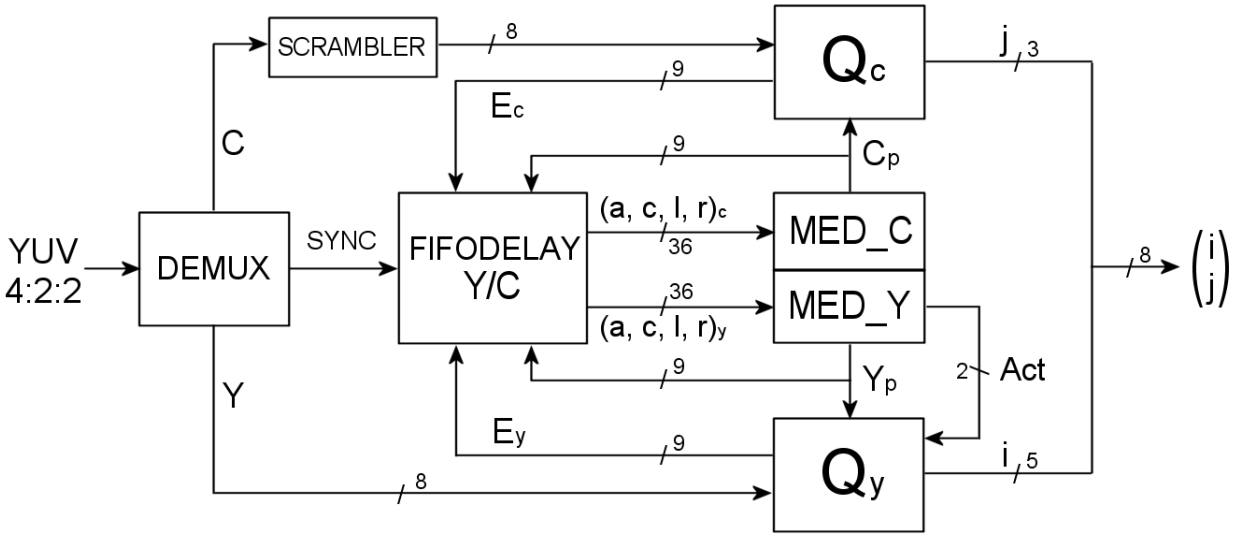


Figure 6.2: HERMES video encoder core

It is clear from figure 6.2 that the processing of Y and C components is completely separated and parallel. Each basic block has been implemented as two different versions, each one optimized for the processing of Y or C components respectively. The only exception is the SCRAMBLER block, which is required for the chrominance 5.2.6 only. Each corresponding Y/C module implements similar functions but with some differences due to the processing diversity between Y and C.

The quantization module Q_y/Q_c receives as inputs the Y/C sequence and the prediction Y_p/C_p

from the corresponding MED block. Q_y/Q_c generates as output the residual quantized difference E_y/E_c and the corresponding quantization level i/j, which are sent to the entropy coding module. Q_y also implements the conditional coding 5.2.4 method, therefore it receives the activity level already coded as a 2 bit word.

MED is a full combinatory logic module which performs the median selection within a set of values calculated by the FIFODELAY block, which consists of a single line FIFO buffer of locally decoded data. The main task of FIFODELAY is to calculate the required A, C, L and R linear predictors 5.2.2 and all possible intermediate values that can help MED to select the median.

HERMES core encoding latency is only two video clock periods corresponding to $\approx$ 72 ns. This result reflects the intrinsic serial processing of DPCM algorithm, which achieves the encoding on a pixel by pixel base. However, it also means that every 72 ns we must be able to calculate the requested linear predictors and the median within them. Therefore HERMES exhibits a critic combinatory loop that puts an upper limit to the maximum working frequency.

6.3.1 Line Buffer Module

The FIFODELAY module is not simply a line buffer, which allows to use the pixel of the previous video line, it is also a high pipelined preprocessing core that calculates the four linear predictors and an amount of intermediate values which helps the MED module to chose the median predictor. The basic structure of FIFODELAY Y module is described in figure 6.3. In the case of standard video resolution and 4:2:2 sampling format, we have 1440 samples per line, of which 720 Y and 720 C. In order to implement the shifting mask of neighbor pixels (refer to 5.2.7) we need the

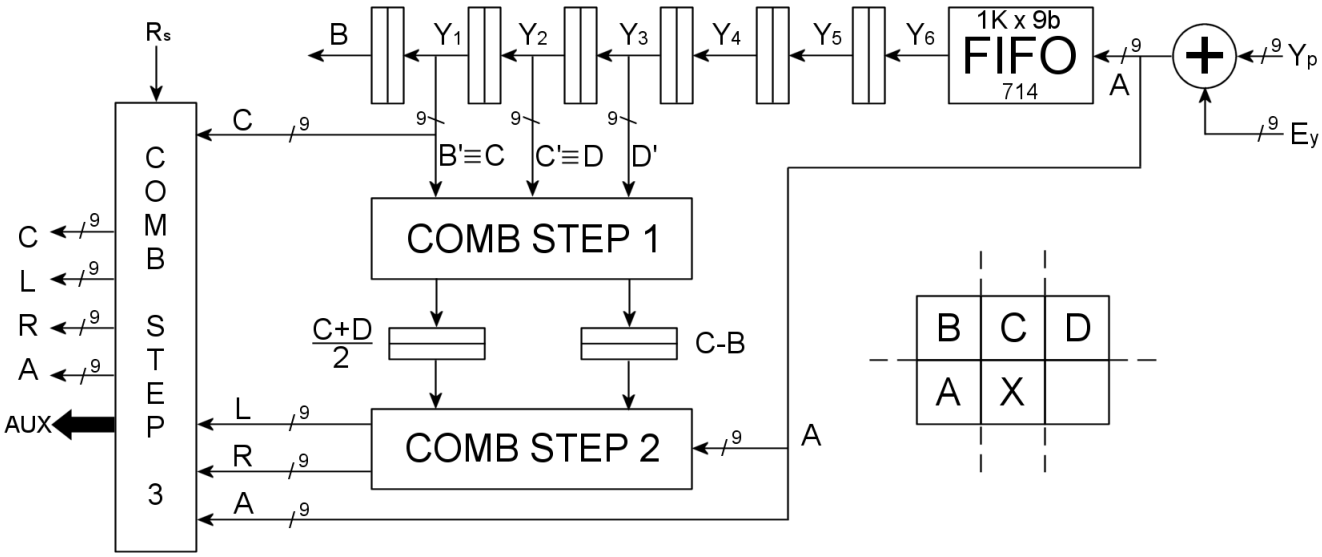


Figure 6.3: HERMES luminance FIFODELAY module

access to the last three consecutive pixel B, C and D on the previous line, and the previous

pixel A on the actual line. In the scheme of figure 6.3 we use a dual port memory bank $1K \times 9b$ configured as a synchronous FIFO with a depth of 714 locations. Then we achieve the line length of 720 positions using 6 shift registers. This structure is very flexible and can be adapted to a different line length by simply modifying the FIFO depth.

FIFODELAY module receives the predicted pixel value Y_p and the quantized prediction error E_y, then it reconstructs the previous coded pixel $A = Y_p + E_y$. Once we get A, we have to calculate the four linear predictors and select the median within them. This processing must be accomplished within the time of ≈ 72 ns, before that the input pixel will change. The delay generated by the combinatory logic does not meet the requirement, thus we have to find a way to break the long combinatory path into small ones.

Instead of waiting for the pixel values A, B, C and D and then starting the processing, we can pre-process all the intermediate values which depends only on B, C and D, because their past values B', C' and D' are already available in the shift register. As shown in figure 6.3, the STEP 1 calculates $(C+D)/2$ and $(C-B)$ one clock cycle before the corresponding A pixel would be ready. When A is input on next clock cycle, the linear predictors L and R can be directly calculated using the intermediate values sampled at STEP 2 input. The STEP 3 logic is used to impose the edge conditions on the predictors in order to guarantee the slice independency 5.2.7. As shown in figure 6.4, we can distinguish between two cases using a signal R_s (Row Start) to

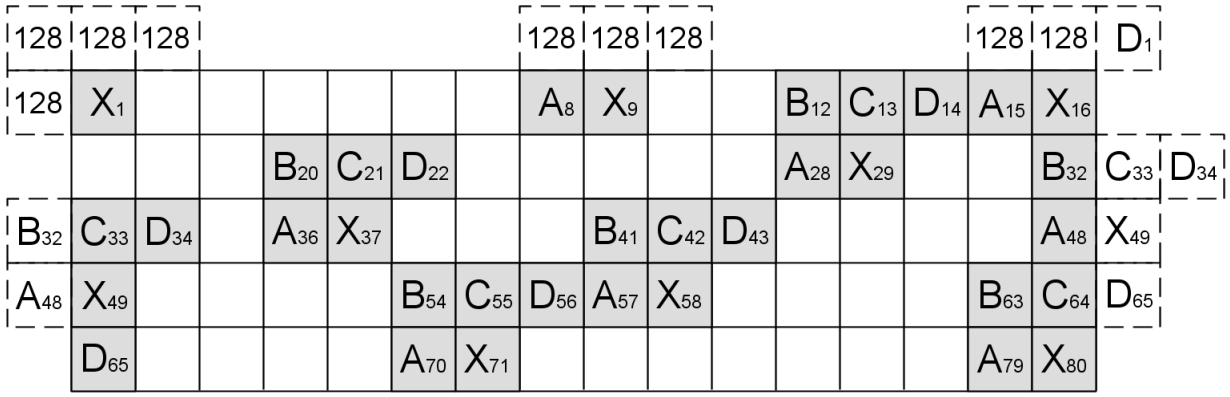


Figure 6.4: Slice coding edge conditions

identify the first line of a slice.

- If X is the fist data of the slice then $A = B = C = D = 128$, which means that $R = L = 128$.
- If X owns to the first line of the slice then $B = C = D = 128$, which means that $R = A$ and $L = A/2 + 64$.

We do not force any edge conditions for the intermediate lines in the slice because it is simpler to use the previous pixels, as if each line end data would be contiguous to the next line start data.

The scheme of the chrominance FIFODELAY module is shown in figure 6.5 and is similar to the luminance module of figure 6.3. The fundamental difference is that the C data stream is scrambled 5.2.6, so we have to select the right component at the right time. Furthermore, the C stream is formed by interleaved U and V components which are completely uncorrelated and have to be independently processed. In order to select the C' pixel, we use a simple multiplexer driven by the signal C_{sel}, which have to be generated according to the line synchronization.

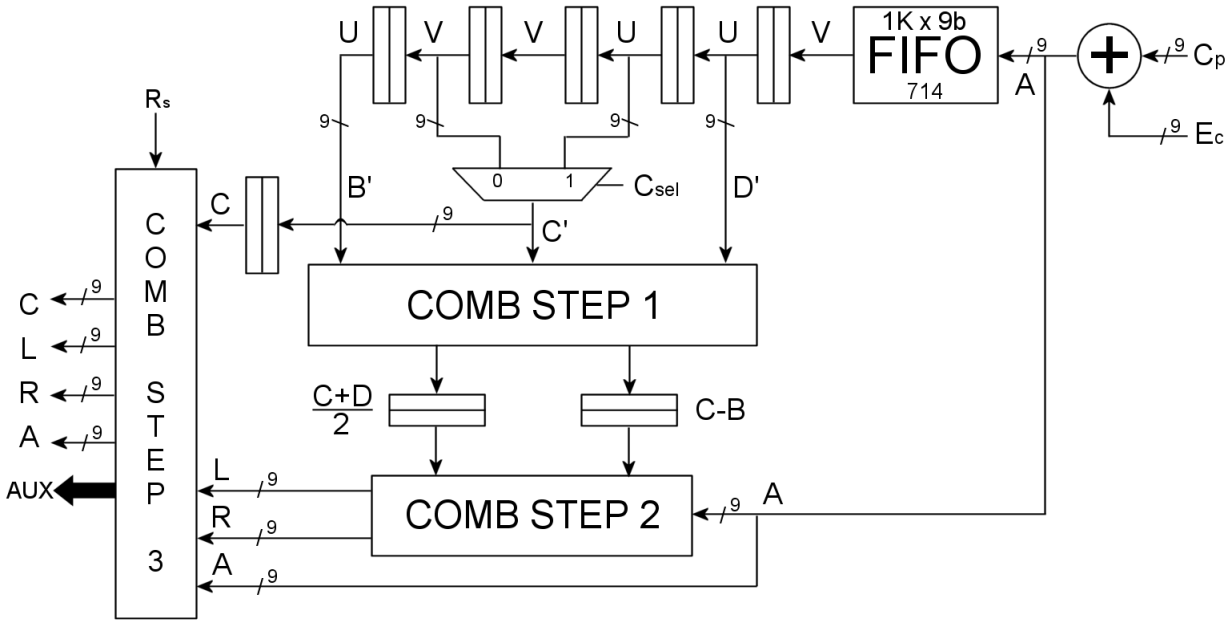


Figure 6.5: HERMES chrominance FIFODELAY module

The Y/C FIFODELAY modules have to work in parallel assuring the synchronization between the processing of the Y and C data streams. At this scope they have been joint together and the corresponding input FIFOs have been implemented into a single $1K \times 18$ memory bank. In this way each single data stored in the FIFO is a 18 bit word formed by a (Y, C) couple of locally decoded components.

The STEP 3 logic also receives intermediate values from both STEP 1 and 2 logic and use it to generate a set of auxiliary signals AUX, which are all the possible differences within the four linear predictors: $A-C$, $A-L$, $A-R$, $C-L$, $C-R$ and $L-R$. Therefore FIFODELAY module generates A, C, L, R and the AUX signals at the same time, which makes it possible to achieve a rapid selection of the median predictor in the combinatory MED module.

6.3.2 Luminance Quantization Module

The module Q_y performs the quantization of the residual error E between Y and the prediction Y_p. The quantization class N 5.2.3 has been renamed to Q_{sel} and coded as a 3 bit word. Each class gets three quantizers, which are selected by the activity Act. In the case of $Act = 00$ the

conditional coding is active and we simply force $E_y = 0$ in order to use Y_p as a reliable prediction of Y.

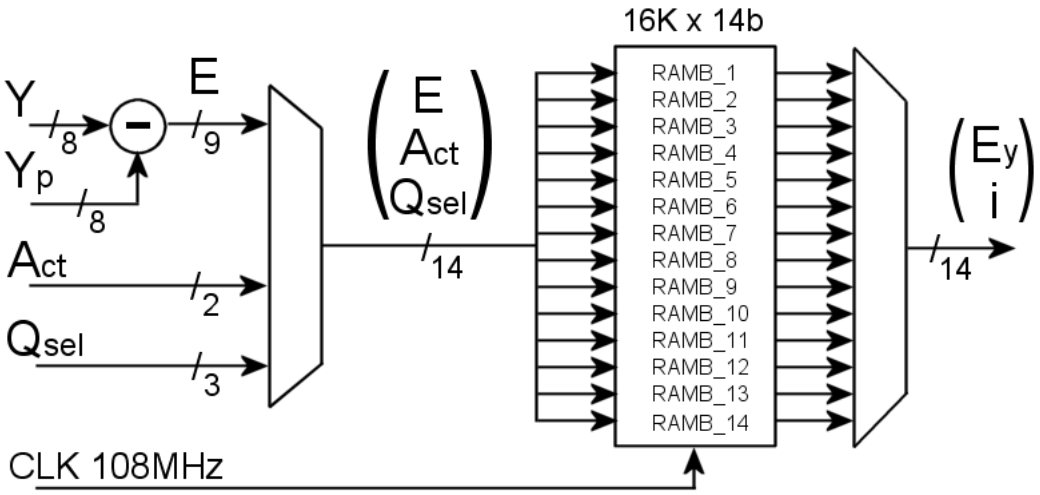


Figure 6.6: HERMES luminance nonuniform adaptive quantizers

The combinatory latency of the first Q_y logic implementation was too high and not constant. The latency changed depending on the chosen quantizer and even by the quantization level. Furthermore, this latency was added to the one required to calculate the median predictor and the final result was not acceptable.

A better solution is to implement Q_y using embedded FPGA memory banks, as shown in figure 6.6. The advantage of using a memory instead of a combinatory logic is that the latency is constant whatever the input is and we can use a high clock frequency (108 MHz) to drive the read port, to achieve a low readout delay. We have to implement a table having $9 + 2 + 3 = 14$ bit as input and $9 + 5 = 14$ bit as output. From the latency point of view, a good solution is to use 14 RAMB (RAM Banks) configured as $16K \times 1b$ and connected as expansion of the data bus.

The input 14 bit word is then connected to the address bus (ADD) and the reading enable flag is forced to stay always active ($WE = 0$) in order to perform a read operation at every rising clock edge. The data access time is < 3 ns, while at the frequency of 108 MHz the clock period is ≈ 9 ns. As shown in figure 6.7, the great advantage of this scheme is that the latency between a stable reading address and the corresponding output data is $T_a + 3 = 12$ ns maximum, where $T_a < 9$ ns.

Since the address bus is stable, we keep on reading at the same location and we get the same output. This method gives a very short latency, much better than the full combinatory case, however it may be expensive in terms of power consumption. Therefore, its use is limited to the quantization module, which represents a very critic case.

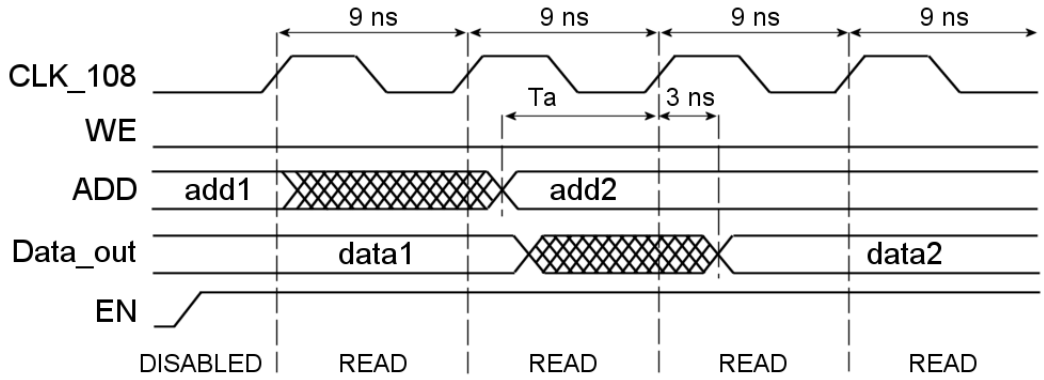


Figure 6.7: Reading temporization scheme

6.3.3 Entropy Encoder

The entropy encoder module has to replace each couple (i, j) with the corresponding variable length word of the selected Huffman code and outputs a VLC stream. We can divide the problem in two parts: first we generate a couple $(CODE, LEN)$ formed by a codeword $CODE$ and its related length LEN; $CODE$ is simply the variable length word corresponding to (i, j), expressed on $L_{MAX} = 16$ bit, where L_{MAX} is the maximum possible codeword length. The second step consists of merging the consecutive $CODEs$ taking into account their $LENs$, in order to generate a true VLC stream. The first step of Huffman coding is achieved by the module described in figure

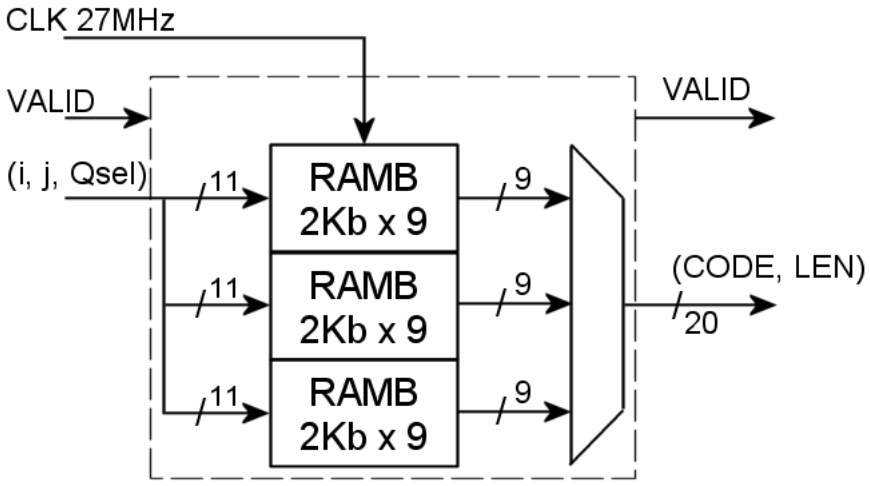


Figure 6.8: HERMES VLC encoder

6.8. We use three memory banks configured as $2K \times 9b$ to store the Huffman codes related to the quantization classes, for a total of seven VLCs and 432 codewords. On the address bus we input the 8 bit symbol (i, j) to be coded and the 3 bit word Q_{sel} to select the VLC to employ. The input $VALID$ signal is resampled and delayed using the 27 MHz reading clock in order to generate a

coherent $VALID$ signal for the output data. Despite the high flexibility and compactness, the design of figure 6.8 is not the best in terms of requirement. In order to save memory resources, a combinatory table can be easily employed.

The simple design of figure 6.8 outputs up to 27 bit of which only 20 are used because $L_{MAX} = 16$, therefore LEN can be represented on 4 bit. The limitation on L_{MAX} is required to define the entropy decoder architecture, which becomes more and more complex if L_{MAX} increases. $L_{MAX} = 16$ is a good choice because it does not affect too much the coding efficiency and it is a power of 2, which gives some advantages in performing data size conversions.

The second step is performed by the VLC SEQUENCER module (figure 6.10), which uses a circular $32 \times 1b$ merging buffer to create a 16 bit VLC stream. Each codeword $CODE$ is written into the buffer taking into account its corresponding LEN. When at least 16 bit of code have been written, the SEQUENCER outputs a word and empties the corresponding locations in the buffer. The VLC SEQUENCER is well integrated in the ES (Elementary Stream) generation, as shown in figure 6.10. Finally, another important task achieved by the VLC SEQUENCER is to calculate the elementary packet length, which is an output on 12 bit and is expressed in units of 16 bit words.

6.3.4 Audio Input Interface

The digital audio input processing is shown in figure 6.20. HERMES supports the 3 wires I2S serial interface for digital audio input/output, thus the fist step is the de-serialization in order to get the 24 bit PCM audio samples. This job is done by the I2S/PCM CONVerter module which also performs a clock domain adaptation between the serial input clock and 27 MHz video clock.

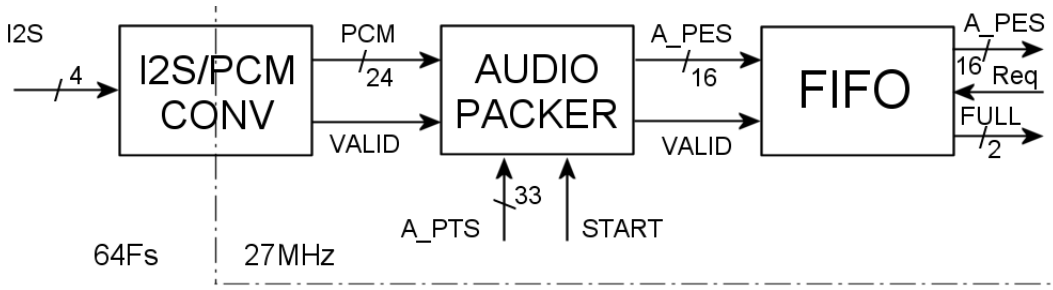


Figure 6.9: HERMES digital audio input

The AUDIO PACKER module has been introduced just after the converter I2S/PCM, because in the 27 MHz clock domain the average distance between consecutive PCM audio samples is ≈ 9 clock tics and the audio PES packet header takes a maximum of 96 bit $= 4 \times 24$ bit data words 5.5.2. Therefore we can directly output the actual PES packet header just after the arrival of last audio sample of previous PES packet. The packer module receives the audio PTS and a START

signal which is set when the first video frame is detected. In this way we can discard all PCM samples which come before the beginning of the first complete frame of the input video sequence. However, the fundamental audio/video synchrony is assured by the PTS mechanism, while the START pulse allows to begin the audio/video processing since a common time reference. Finally the audio PES packets are buffered into a small FIFO which interfaces the PES packer with the TS packer.

6.3.5 Packetization

The fist stage of packetization is the PES (refer to 5.5.2) generation. After the entropy coding, each slice becomes a sequence of variable length codewords, therefore any synchronization information is lost and without the packet structure it would be impossible to say where one slice ends and where the next one starts.

A video PES packet starts with a header which carries some useful information about the encoded slice, as for example the coding QUALITY, the SLICE NUMBER, the PTS and so on. This information is fundamental for the decoding process and the correct video synchrony recovery. A video PES packet ends with a special SLICE END CODE, which is used to reset the VLC decoder before the next slice comes.

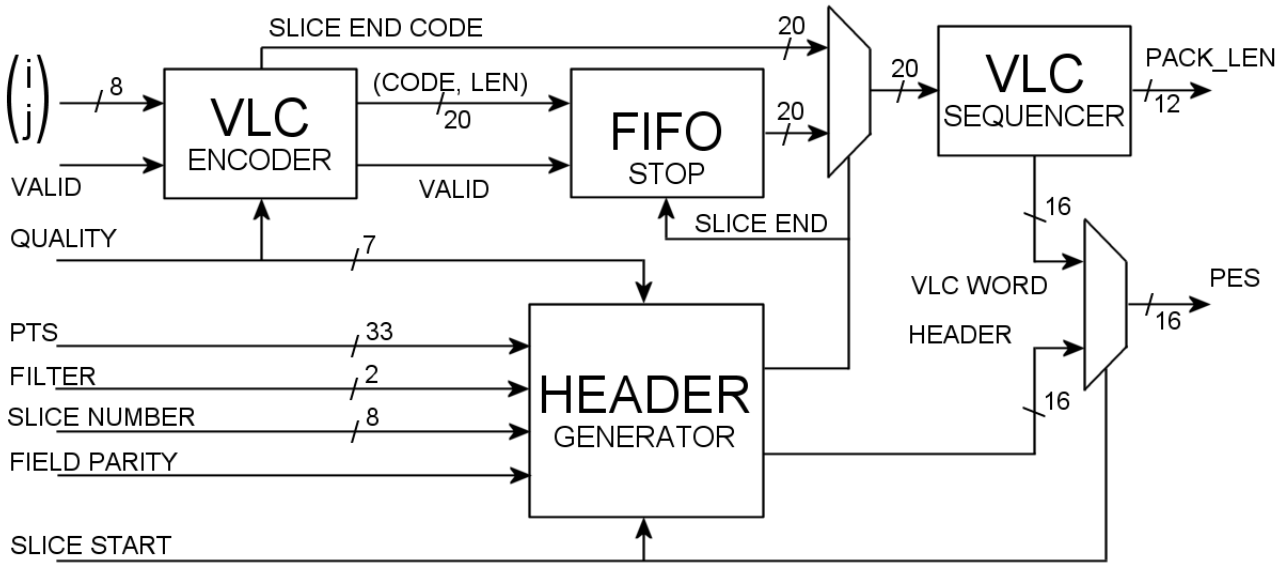


Figure 6.10: HERMES elementary stream packetization

In figure 6.10 the general scheme of the video PES packer is described. All the packet information is input into the HEADER GENERATOR, which also receives a SLICE START signal. When SLICE START is active the HEADER GENERATOR controls the insertion of the past SLICE END CODE and next slice packet HEADER. A small FIFO is implemented at VLC

ENCODER outputs in order to avoid data loss during the extra information multiplexing.

In fact, VLC coded data are continuously generated, even when a slice packet HEADER has to be inserted. In this case, the FIFO reading is disabled during the multiplexing. Afterward, a dedicated logic restarts the automatic FIFO reading.

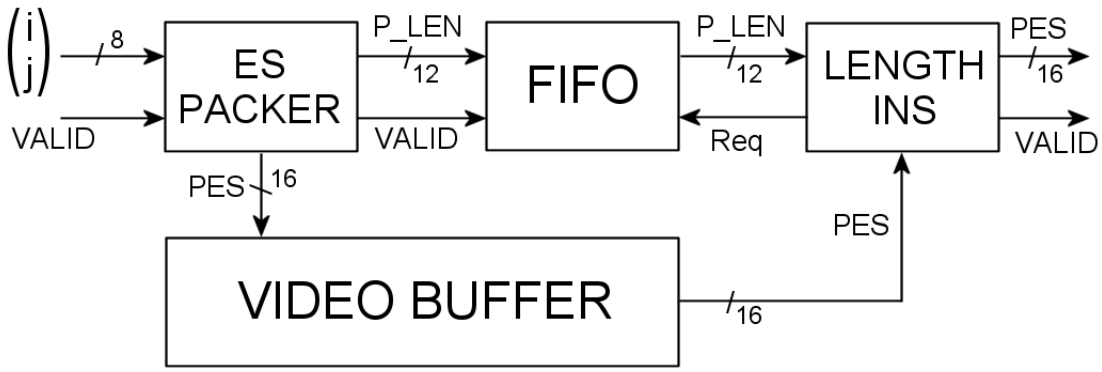


Figure 6.11: HERMES slice length insertion

The PES packet is finally output on a 16 bit bus and is stored into the output video buffer, as shown in figure 6.11. However, the packet length can be known only at the end of its related PES packet, after the slice has been fully processed by the VLC encoder and sequencer. If the video buffer stores more than one slice (which is the case described in section 5.4), we can send the packet length into a FIFO and perform the insertion later, when the PES is read out from the video buffer. The module LENGTH INS of figure 6.11 receives the PES stream and when a header start code is found, a data request is sent to the FIFO and LENGTH INS inserts the packet length word at the right position.

The module TS PACKER is both a data multiplexer and a TS (Transport Stream) packet generator. This module requests PES data from both video and audio input FIFOs, as shown in figure 6.12, then the corresponding TS packets are sent to the output FIFO, which achieves the clock domain adaptation between 27 MHz and the required channel rate. A further task accomplished by TS FIFO is the data conversion from 16 to 8 bit, which is requested by the output interface. TS PACKER also multiplexes PCR and stuffing packets in the TS in order to generate a continuous data stream without interruptions. The packet generation strategy is based on the audio/video FIFOs occupancies. A control logic continuously monitors the AFULL and VFULL signals coming from PES FIFOs and it decides the scheduling priorities in order to avoid overflow conditions. Every time AFULL $> 1/2$ or VFULL $> 1/2$, we schedule a read which is executed as soon as possible. Otherwise, if both audio and video FIFOs have a low occupancy then PCR or Stuffing packets are output. The PCR packets scheduling also takes in account a parameter that sets the minimum required PCR rate. The output TS FIFO occupancy is monitored as well, thus if TFULL $> 3/4$ we stop the write until TFULL $\leq 1/2$.

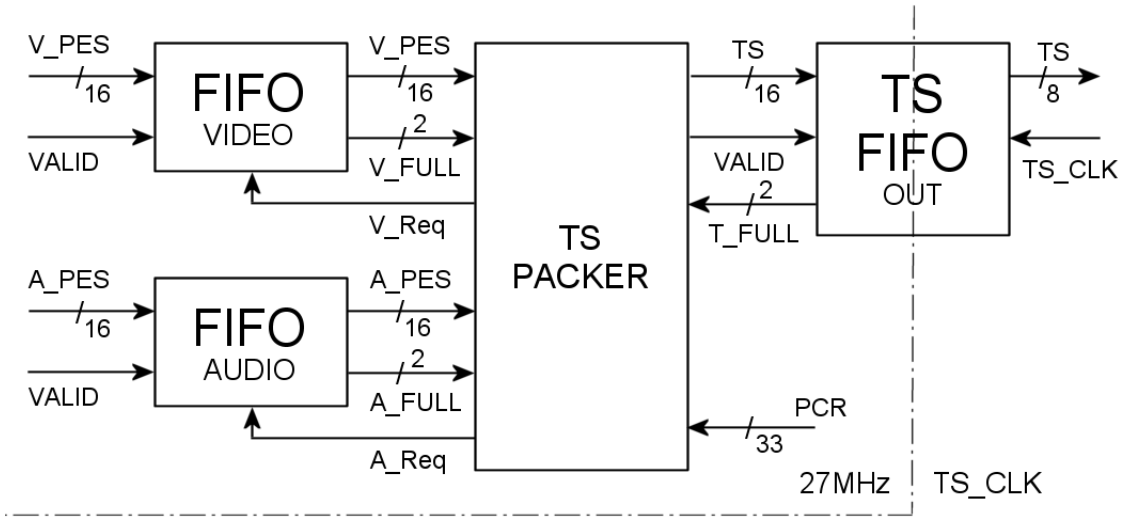


Figure 6.12: HERMES transport stream packetization

In order to correctly fragment each PES packet into many TS packets, we need to know exactly the amount of byte carried by the PES packet, which is the sum of the payload and the header size. In the case of audio PES packets, the payload is constant, while for video PES packets the payload size is given by PACK LEN parameter. The PES packets header size depends on FRAME NUM for audio and PACK LEN for video (refer to section 5.5.2). To retrieve these parameters before starting the read from FIFO, we can simply use the shift register logic shown in figure 6.13.

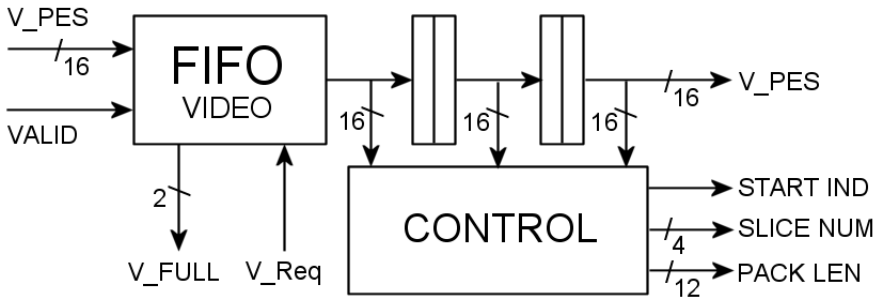


Figure 6.13: HERMES video shift control logic

The control logic monitors the PES data during the FIFO read. When a packet start code is found, the START INDicator signal gets high and the parameters PACK LEN and SLICE NUM are sampled. A similar logic has been implemented for the audio input FIFO as well.

$$CLK_{MIN} = \frac{V_b + A_b + PCR_b}{8} \tag{6.1}$$

The TS PACKER logic can work if the TS CLK $\geq CLK_{MIN}$ defined in equation 6.1, where

V_b, A_b and PCR_b are the video, audio and PCR bit rates respectively. The higher the difference between TS CLK and CLK_{MIN}, the more stuffing packets are used to fill the TS.

6.4 HERMES DECODER

The input of HERMES decoder is the TS which carries a single programme constituted by one audio and one video stream. Thus the first step is TS de-packing process in order to get the PES packets. After PES de-packing we perform the entropy decoding on video data and each slice is then sent to HERMES decoder which recovers the video luminance and chrominance samples. Instead of implementing an input VBV buffer (which stores encoded data) as in the classical design, we prefer to use a post-decoding frame buffer which can be used to implement a simple error correction strategy based on the repetition of last decoded frame. Therefore the input data are stored into a small FIFO just to perform the clock domain conversion (from TSCLK to 27 MHz) and are decoded as soon as possible. According to PTS information 5.5.1, the CCIR interface reads data from the frame buffer and finally, the YUV 4:2:2 decoded sequence is output.

The audio PES packets are separately processed and the output I2S interface is synchronized with the video stream. The audio samples are stored into a FIFO which carries out the same job as the frame buffer. However audio data represents only 1.38 % of video decoded data, so the audio frame buffer can be implemented into FPGA embedded memory.

6.4.1 De-Packetization

At decoder input we assume to receive a valid TS with its related TSCLK. The first module TS SYNC GENerator, shown in figure 6.14, performs a first analysis of input data stream, recovering the typical TS SYNC signal which identifies each TS packet beginning.

The SYNC BYTE value (refer to section 5.5.3) is not unique within a transport packet and can quite naturally occur in other fields, thus the simple identification of the SYNC BYTE is not a good strategy to recover TS SYNC. However, the fact that a sync byte will occur every 188 bytes within a transport stream enables TS SYNC GEN to lock onto this repetition and hence to identify the start of each incoming transport packet. In the case of a damaged input TS synchrony the SYNC ERRor signal is set and propagated to the inner blocks, while the TS VALid gets low until the next valid transport packet is found.

TS FIFO IN receives both TS and TS SYNC which are stored at same time. The main task of this asynchronous input FIFO is the clock domain adaptation between TSCLK and 27 MHz. The control of the FIFO is based on its fullness TSFULL. TS DEPACKER module begins to readout at TSFUL $> 3/4$ and it stops whenever TSFUL $< 1/4$. The FIFO overflow condition cannot occur because the reading rate is higher than the writing one.

TS DEPACKER module simply reads and de-packs the TS packets following the incoming

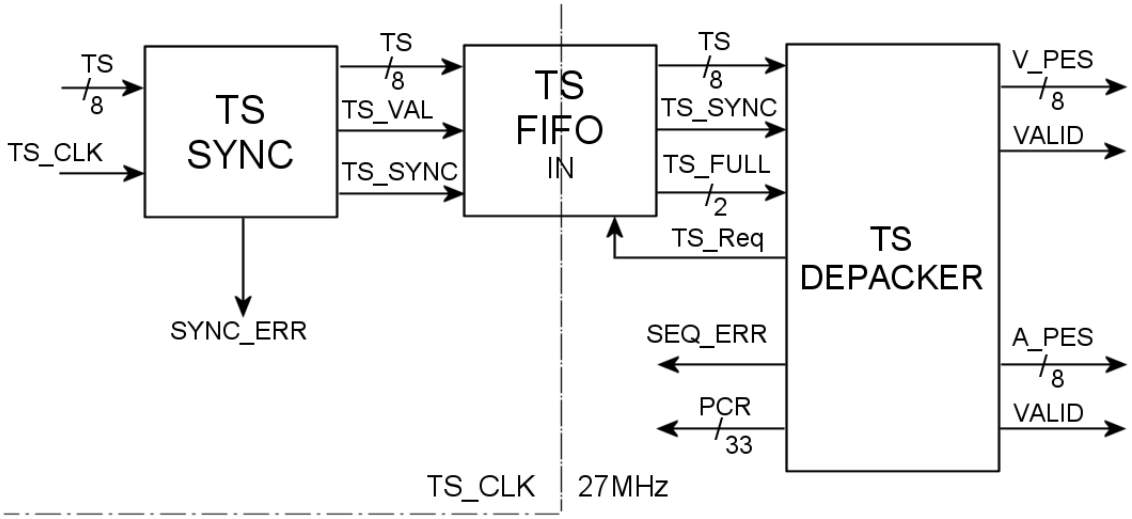


Figure 6.14: HERMES transport stream de-packetization

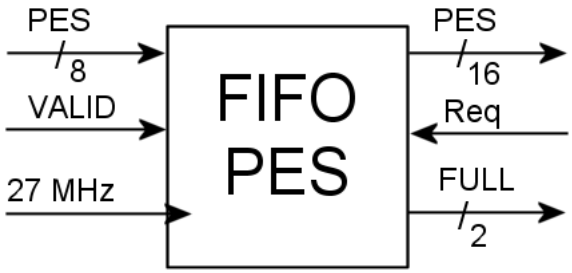


Figure 6.15: Audio/Video PES FIFOs

order fixed by the TS PACKER module at encoder side. The PCR is recovered from PCR packets while stuffing packets are discarded. In the case of anomalous TS packet structure, the SEQuence ERRor signal is set.

The de-packed audio/video data are then separated and stored into dedicated synchronous FIFOs which have the general structure described in figure 6.15. The PES FIFOs are used for temporary data buffering, while they also achieve a data width conversion from 8 to 16 bit, which is required by the following PES DEPACKER module.

Finally the VIDEO PES DEPACKER reads the video PES packets from the corresponding FIFO and sends the payload CODE words to the entropy decoder together with the needed Q_{SEL} parameter. Moreover the VIDEO PES DEPACKER recovers all the parameters carried by the PES header as: video PTS, SLICE NUMBER, RELY and so on.

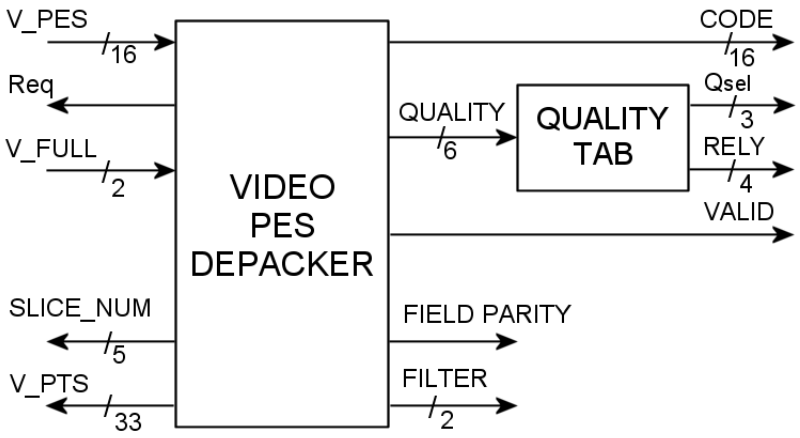


Figure 6.16: HERMES video PES de-packer

6.4.2 Entropy Decoder

The entropy decoder is one of the more complex modules in the decoder. A small input FIFO stores the CODE words coming from the VIDEO PES DEPACKER block. As shown in figure 6.17, the reading operations and the VLC DECODER control are performed by the FSM block. The STOP signal is generated in the case of FIFO EMPTY, otherwise a read is scheduled each time that LOAD is set by the VLC DECODER. The START signal is simply a local reset which is used to activate the VLC DECODER.

The input FIFO is small and has been implemented into a single FPGA memory bank. In fact the input data rate is low and the 27 MHz clock tics which take between two consecutive 16-bit PES words are given by the ratio $r = 27\text{MHz}/\text{TSCLK}$. If we consider a minimum code length $L_{MIN} = 2$ then in the worst case 8 clock tics are needed to fully decode a 16 bit word. The FIFO cannot overflow if each PES word is decoded before the next one comes, so if $r \geq 8$, which means a TS rate of 64 Mbps, that is higher than what we can ever expect.

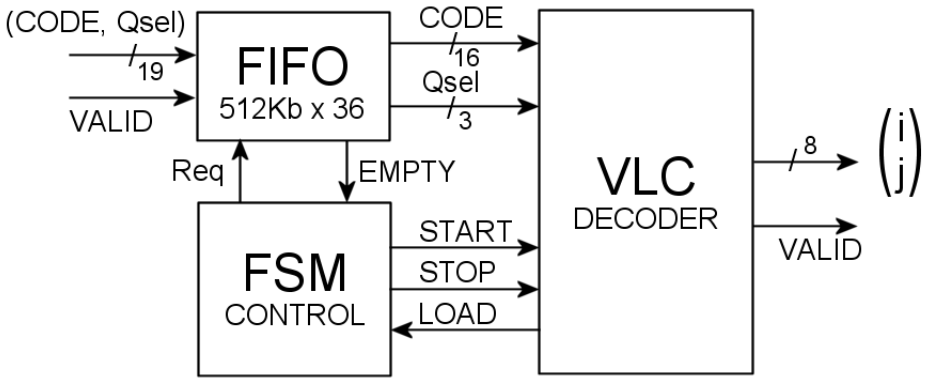


Figure 6.17: HERMES VLC controller

The main difficulty of Huffman VLC decoding is that we cannot know the length of next codeword in advance because it is variable, so the generic method performs an iterative search which can take time. The simpler VLC decoding approach is to serially read the VLC stream bit per bit while looking for a matching with one of the VLC codewords. When a codeword is identified then its length is known and it can be extracted and decoded. However this method would require a high working frequency to process the VLC stream correctly. The VLC DECODER block has been implemented using a parallel Huffman decoding algorithm based on the minimum redundancy prefix codes technique [51] which allows to decode a single codeword every clock tic.

The actual VLC DECODER solution can process 7 Huffman codes corresponding to each quantization class N=23, 17, 13, 11, 9, 7, 5 identified by the Q_{SEL} parameter. The maximum code-length $L_{MAX} \leq 16$ bits (refer to section 5.2.5), thus the input 32 bit register (H_{REG}, L_{REG}) shown in figure 6.18 within a dotted rectangle, can store at least 2 codewords. The BARREL SHIFTER selects within the 32 bits of input register a 16 bit window V which is shifted according to the SUM of LENgths of past decoded codewords. If LEN overpasses 15, a carry EN is generated and a new codeword is loaded into L_{REG} while the previous L_{REG} contents is transferred to H_{REG}.

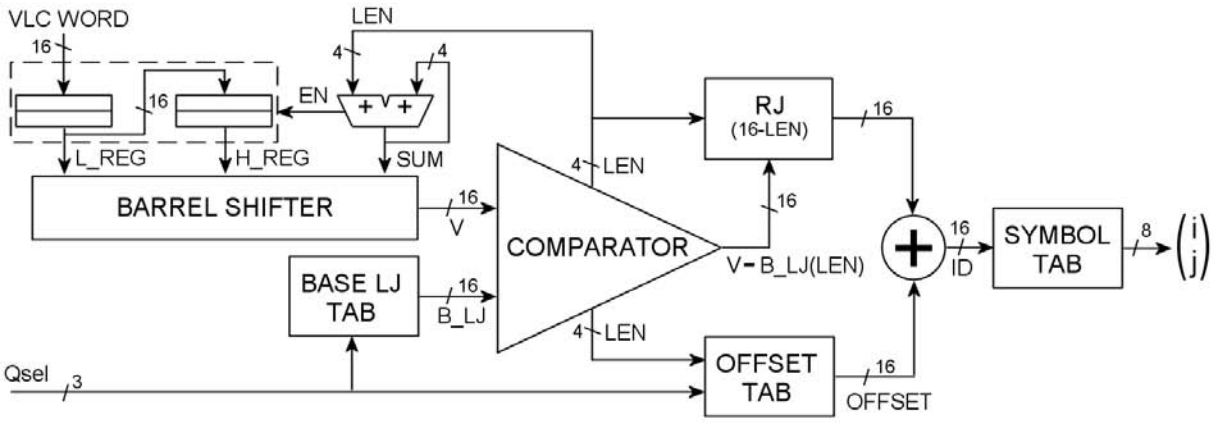


Figure 6.18: HERMES VLC decoder

The COMPARATOR identifies the actual codeword LEN by looking for the lowest LEN which satisfies the condition: $B_{LJ}(LEN) \leq V$, where $B_{LJ}(\cdot)$ is the left justified base table related to the VLC selected by Q_{SEL} [51]. As the possible values for LEN are restricted, the COMPARATOR has been designed to perform all possible comparisons in parallel in order to identify LEN within the shortest possible time.

Once LEN is known, the amount $V - B_{LJ}(LEN)$ is first left shifted by 16-LEN bit positions then right justified and finally added to the corresponding OFFSET(LEN). What we get is the symbol identifier ID which selects the decoded couple (i, j) in the symbol table. When ESS (End Sequence Symbol, refer to 5.2.5) is decoded then the SEQUENCE END signal is set and VLC

DECODER can start decoding the next sequence.

VLC DECODER block also checks the PES packet payload length according to the PACK LEN parameter 5.5.2. The actual VLC sequence length is simply calculated using a counter based on the EN carry signal, which generates a pulse each time that 16 bit of VLC sequence have been processed. If a mismatch occurs, the PAYLOAD ERRor signal is set.

6.4.3 HERMES Decoder Core

The HERMES video decoder architecture in figure 6.19 is composed of two sections: data input and decoding core.

The data input section is based on the input FIFO and the inverse quantization module Q^{-1} which acts as data controller and also generates some synchronization signals (as SLICE BEGIN), which are required by the decoding core (FIFODELAY module).

The decoder section is based on MED and FIFODELAY modules, which have been described in the previous sections. The 8 bit input data comes from the entropy decoder which uses a flag bit more to identify each slice beginning. The input FIFO in figure 6.19 avoids data loss when the decoder has to reconstruct a not transmitted pixel, so in the case of $A_{ct} = 00$.

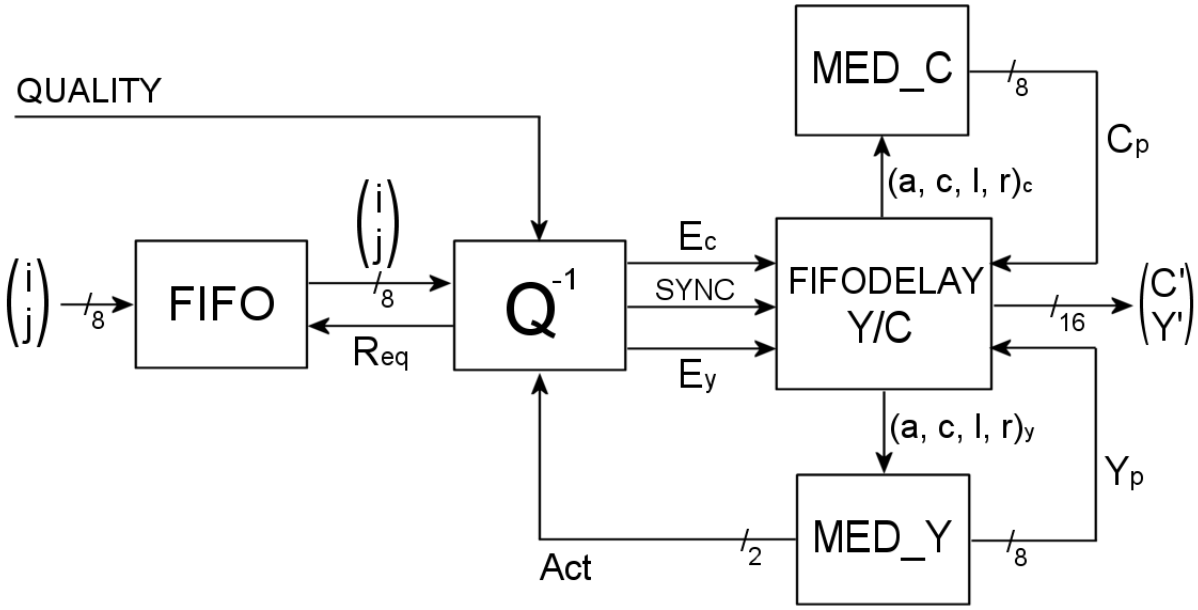


Figure 6.19: HERMES video decoder core

The inverse quantizer Q^{-1} requests (i, j) couples from the input FIFO and re-generates the corresponding error predictions (E_y, E_c) using both QUALITY and *Act* information. If the activity value is lower then the reliability threshold, Q^{-1} just stops reading from FIFO and it outputs $(0, 0)$. In this way HERMES simply decodes the actual pixel by using its prediction only. Finally the decoded C' is DE-SCRAMBLED 5.2.6 to recover the original chrominance sequence order.

6.4.4 Audio Output Interface

Figure 6.20 shows the AUDIO DEPACKER module, which reads the audio PES packets from the FIFO where they are stored by the TS DEPACKER (figure 6.14). The AUDIO DEPACKER recovers the information carried in the PES packet header, like FRAME NUMBER and audio PTS and performs a data size conversion from 16 to 24 bit in order to get the audio PCM samples.

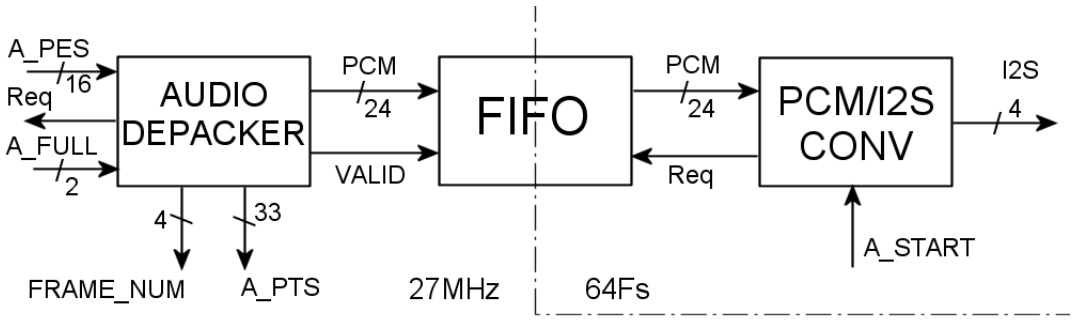


Figure 6.20: HERMES I2S input interface

A small asynchronous FIFO is used as clock domain adapter from 27 MHz to 64Fs. Finally the block PCM/I2S CONVerter reads 24 bit samples from the adaptation FIFO and generates the 3 wires I2S serial digital output. During the depacketization process, it is very important to distinguish between left and right audio samples. To add robustness to this design we store the left/right binary information together with the PCM sample using a simple flag bit.

6.4.5 Video Output

The decoded luminance and chrominance $(Y^{'}, C^{'})$ are then stored into an external memory which implements the VBV buffer. The video output CCIR 656 sequence is generated by the CCIR FORMatter block shown in figure 6.21. CCIR FORM is activated by the START signal coming from the synchronization block described in section 5.5.1. Once START= 1 the output formatter requests data to HERMES decoder which schedules read operations from the VBV buffer. CCIR FORM behaves as master, thus it requests data when it needs and it supposes that data are always available. On the other hand, HERMES decoder is the slave, so it has to provide data when requested.

FIFO FIELD is a sort shift register FIFO which can store a full video field. Therefore FIFO FIELD outputs the data corresponding to the actual pixel position in the last decoded field. In most cases the previous field is very similar to the actual one, thus if any error occurs HERMES decoder can set the DATA ERRor flag which selects FIFO FIELD as source. This recovery method achieves an efficient error concealing in the case of few TS packets loss which interests less than an entire field.

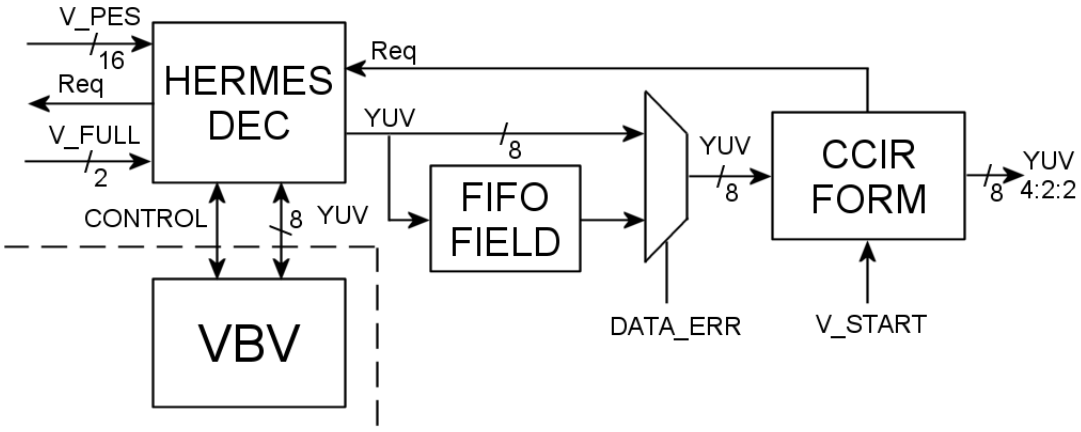


Figure 6.21: HERMES video output interface

6.5 IMPLEMENTATION

The previous system architecture has been first fully simulated at VHDL level and then implemented on the Xilinx multimedia board shown in figure 6.22. The test board is based on a Xilinx Viertex II FPGA and is equipped with a set of input/output audio/video standard interfaces. In order to test both the encoder and decoder design, a prototype version of HERMES codec has been implemented. This version does not take in account the video buffering stage and the input/output audio interface, therefore the encoder works in VBR mode, without rate control. Despite this simplification, the test design employs almost 80% of the proposed architecture, included the full HERMES encoder and decoder cores.Thus the VBR version still represents a good way to validate the real functionality of the global design.

The HERMES VBR implementation can be loaded with a standard DVD video source. As output we get the reconstructed sequence corresponding to a constant quality HERMES encoding. Using the buttons interface (figure 6.22) we can change the encoding quality selecting different quantization classes. An internal MIR can be used as optional source. The slice size can be configured in the VHDL code. This test has confirmed that the quantization class $N = 3$ produces a really poor coding quality, so it will be discarded in next implementations (Refer to section 7.2).

The implementation results are shown in figure 6.23 which confirms the low resources requirement for HERMES design. We can refer to table 6.1 to get a further comparison with the MPEG-4 SD FPGA IP core [52], still under development at Xilinx. The summary of table 6.1 is related to the basis video encoder only, without: audio encoder, system layer, buffer control and TS multiplexer.

Despite any similar information about AVC/H.264 on chip resource requirements are not yet available from Xilinx, a comparison with MPEG-4 in relative terms can be found in literature [53] .

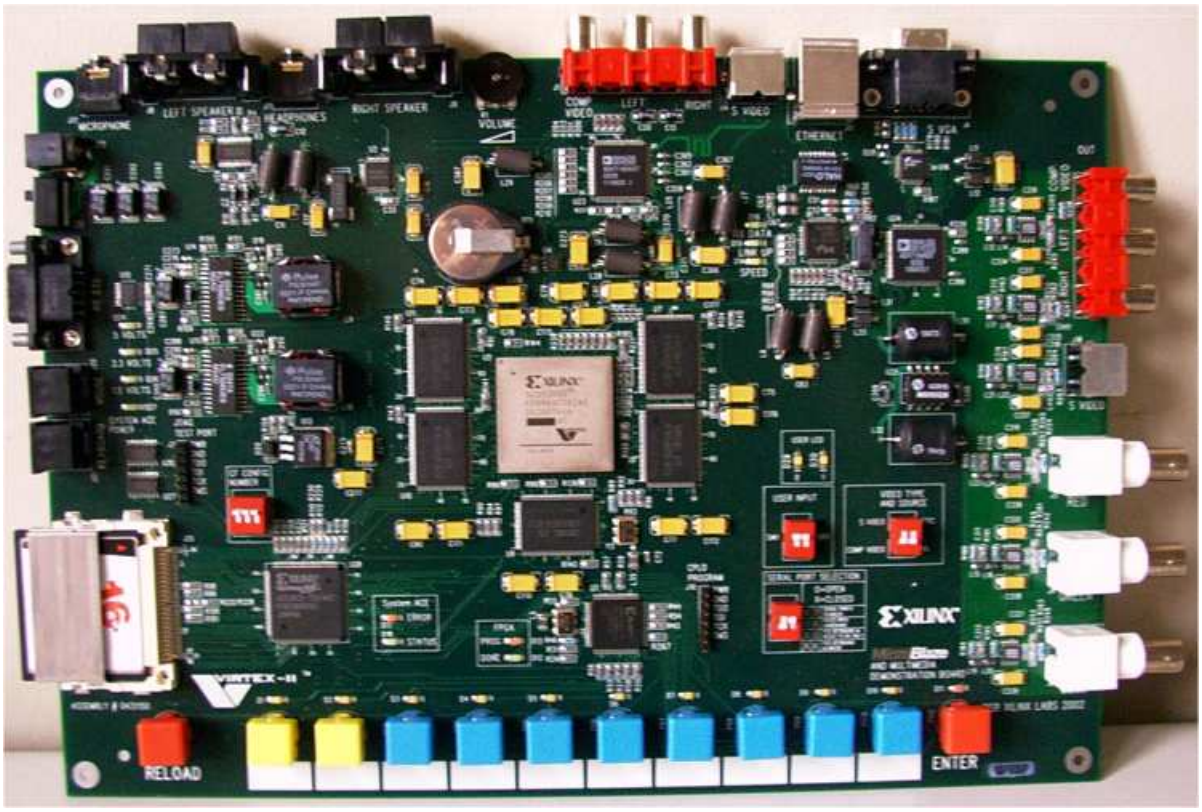


Figure 6.22: Multimedia development platform board

```
Design Information -- HERMES VBR ENCODER/DECODER
------------------

Target Device  : xc2v2000
Target Package : ff896
Target Speed   : -4
Mapper Version : virtex2

Design Summary
--------------

Logic Utilization:
  Number of Slice Flip Flops:         2,009 out of  21,504    9%
  Number of 4 input Luts:             4,694 out of  21,504   22%

Logic Distribution:
  Number of ocupied Slices:           4,225 out of  10,752   39%

  Number of Block RAMs:                  40 out of      56   71%
  Number of GCLKs:                        5 out of      16   31%
  Number of DCMs:                         2 out of       8   25%
```

Figure 6.23: HERMES VBR encoder/decoder FPGA occupation statistics

The encoder complexity increases with more than one order of magnitude between MPEG-4 Part 2 and AVC/H.264, with a factor 2 for the decoder. The AVC/H.264 encoder/decoder complexity

RESOLUTION	BRAMS	SLICES	MULTS
QCIF	16	8051	17
CIF	21	8330	17
4CIF	30	9000	17

Table 6.1: Xilinx MPEG-4 encoder IP core FPGA resources requirement

ratio is in the order of 10 for a basic configuration and grows up to 2 orders of magnitude for a complex one.

6.6 CONCLUSIONS

HERMES VBR architecture has been implemented on Xilinx FPGA platform, which has made it possible to verify the functionality of the most important VHDL blocks such as HERMES encoder/decoder cores. The design requires few on chip resources especially if compared to an MPEG architecture. To make real measurements of the coding/decoding delay we first need to implement a CBR HERMES design version, which includes the video buffering stage. However, the buffer control strategy described in section 5.4.1 has been fully tested using a C++ HERMES simulator and accurate predictions on global system latency are given in section 7.3.

Chapter 7

Simulation Results

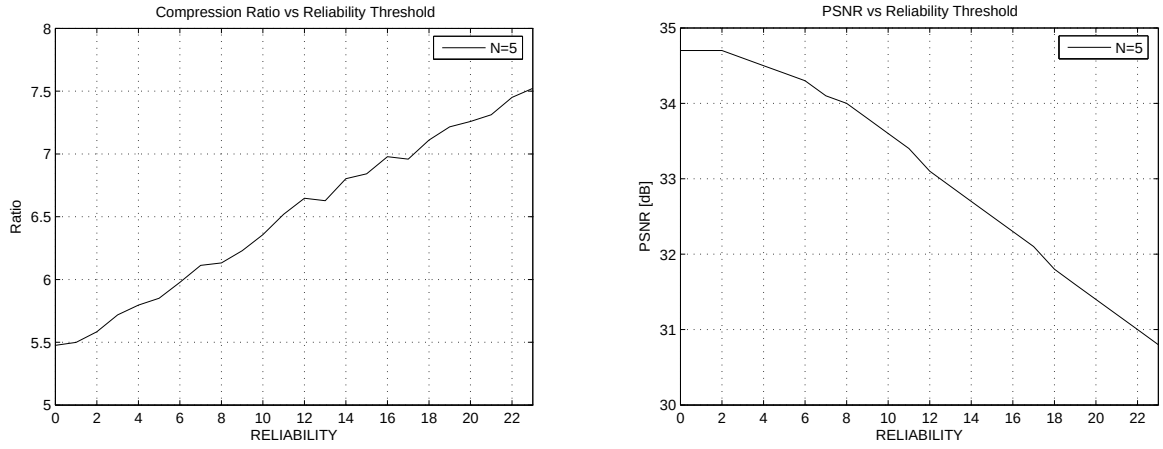
7.1 INTRODUCTION

This chapter analyzes HERMES algorithm performances using the C++ HERMES simulator which has been developed for this purpose. First a calibration method is introduced in order to define a coherent set of quality states which are required by the buffer control feedback. Applying the buffer control method described in section 5.4.1 we can trace the rate-distortion curve and make comparisons with MPEG. Finally we discuss about the visual impairments and further improvements.

7.2 QUALITY CALIBRATION

To achieve the required output CBR the encoding quality is tuned by the buffer control algorithm at each slice beginning. Therefore each slice is encoded at constant quality but different qualities can be used in different slices. To make this mechanism effective, it is fundamental that an upper quality level is associated to a greater amount of bits (lower compression factor) and viceversa. Thus we have to define the quality states according to a monotone relationship with the compression ratio.

A quality state is characterized by two parameters: the quantization class N=23, 17, 13, 11, 9, 7, 5, 3 (refer to section 5.2.3), and the reliability threshold $R_t \in \mathbb{N}$ (refer to section 5.2.4). If N is fixed and we increase R_t, we further decrease the number of bit to send, assuming as reliable a greater number of predictions. As shown in figure 7.1 we get higher compression ratios at the cost of lower objective coding qualities (PSNR).



(a) (b)

Figure 7.1: N=5 quantization class: (a) Compression ratio vs R_t, (b) Y-PSNR vs R_t

If we plot all the compression ratio vs R_t for each quantization class N, we get the curves of figure 7.2(a). As can be noticed, each curve has a quite linear shape and the chosen quantization

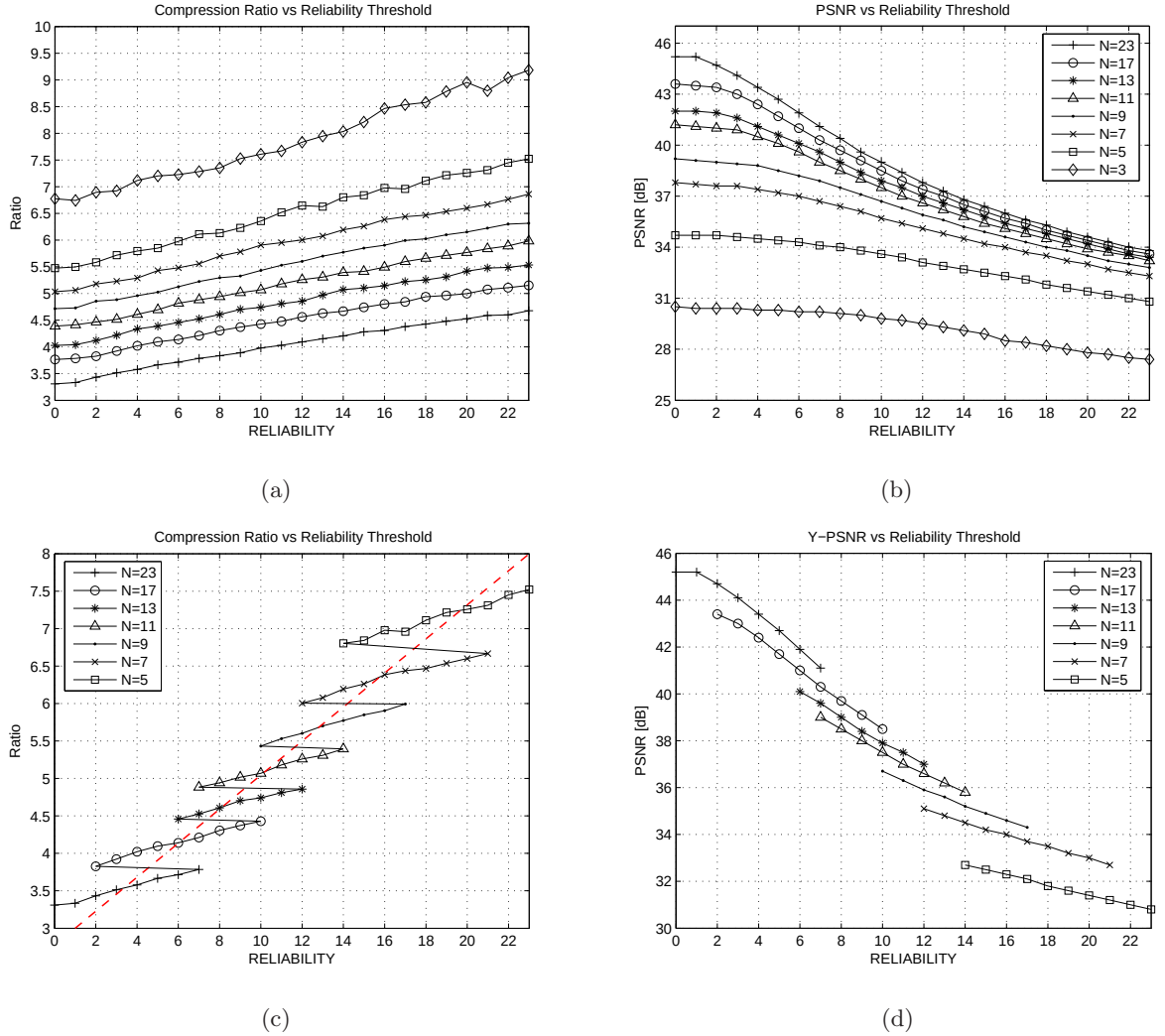
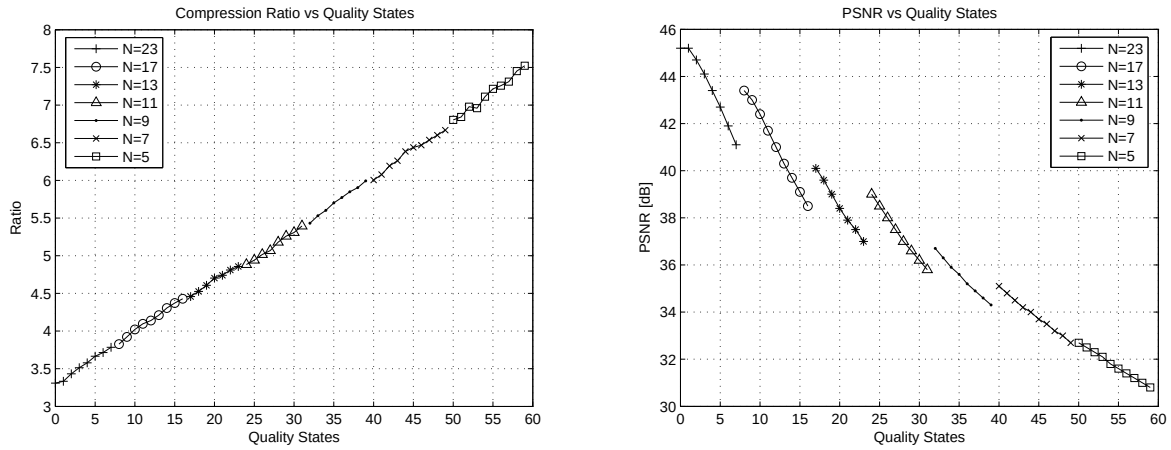


Figure 7.2: Quality states definition: (a)-(c) Compression ratio vs R_t, (b)-(d) Y-PSNR vs R_t

classes give a regular coverage of the zone between 3.3 and 7.5, apart the N=3 class which achieves higher compression ratio, but also very poor qualities (PSNR $<$ 30 dB, figure 7.2(b)). In order to assure a medium-high encoding quality at whichever bit rate, the N=3 quantization class has been abandoned in the final design. As shown in figure 7.2(b), the PSNR falls into a non-linear way when R_t increases. In order to limit the minimum PSNR we chose not to overcome $R_t = 23$, which corresponds to PSNR $\simeq$ 31 dB for N=5.

We now have to select a group of quality states from each class N=23, 17, 13, 11, 9, 7, 5 in order to uniformly cover the available compression ratio range. To do that we try to approximate the dotted linear shape in figure 7.2(c). The only criteria to jump between two classes is the monotony of compression ratio, thus if quality state $S_n \in Q_5$ and $S_{n+1} \in Q_7$ then must be $R_5(S_n) > R_7(S_{n+1})$, where $R_N(\cdot)$ is the compression ratio function related to class N. Figure 7.2(c) shows the results of this selection method, while in figure 7.2(d) the related PSNRs are shown.



(a) (b)

Figure 7.3: (a) Compression ratio vs quality scale, (b) Y-PSNR vs quality scale

State	Q_N	R_t	State	Q_N	R_t	State	Q_N	R_t	State	Q_N	R_t	State	Q_N	R_t
0	23	0	12	17	6	24	11	7	36	9	14	48	7	20
1	23	1	13	17	7	25	11	8	37	9	15	49	7	21
2	23	2	14	17	8	26	11	9	38	9	16	50	5	14
3	23	3	15	17	9	27	11	10	39	9	17	51	5	15
4	23	4	16	17	10	28	11	11	40	7	12	52	5	16
5	23	5	17	13	6	29	11	12	41	7	13	53	5	17
6	23	6	18	13	7	30	11	13	42	7	14	54	5	18
7	23	7	19	13	8	31	11	14	43	7	15	55	5	19
8	17	2	20	13	9	32	9	10	44	7	16	56	5	20
9	17	3	21	13	10	33	9	11	45	7	17	57	5	21
10	17	4	22	13	11	34	9	12	46	7	18	58	5	22
11	17	5	23	13	12	35	9	13	47	7	19	59	5	23

Table 7.1: HERMES quality states definition

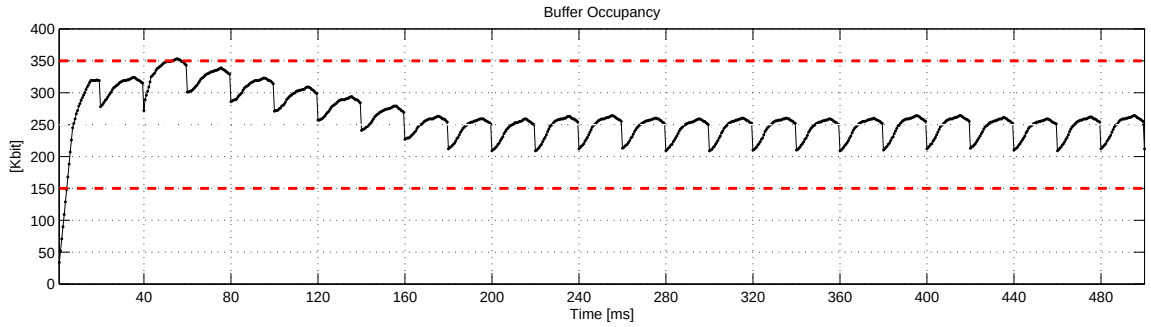
Finally we get a collection of 60 quality states which can be used to implement a rate control strategy in HERMES encoder (refer to section 2.13). As shown in figure 7.3(a), we get a perfect linear curve between compression ratio and quality states. However, the corresponding PSNR function of figure 7.3(b) is strongly non-linear and non-derivable, above all at higher coding qualities. The main problem of the PSNR function of figure 7.3(b) is that we may have variations up to 2 dB between adjacent states corresponding to different quantization classes. However the amplitude of these variations rapidly decreases at lower PSNRs, thus the resulting impairments are not quite visible.

To get a better PSNR curve we should eliminate some quality states across adjacent classes, which leads to the generation of gaps in the compression ratio curve of figure 7.3(a). These gaps can affect the stability of the rate control algorithm, thus we can consider this configuration as a good compromise from the stability point of view. To find a better trade off between stability and quality involves the complete redesign of the quantization classes and will be object of a future optimization.

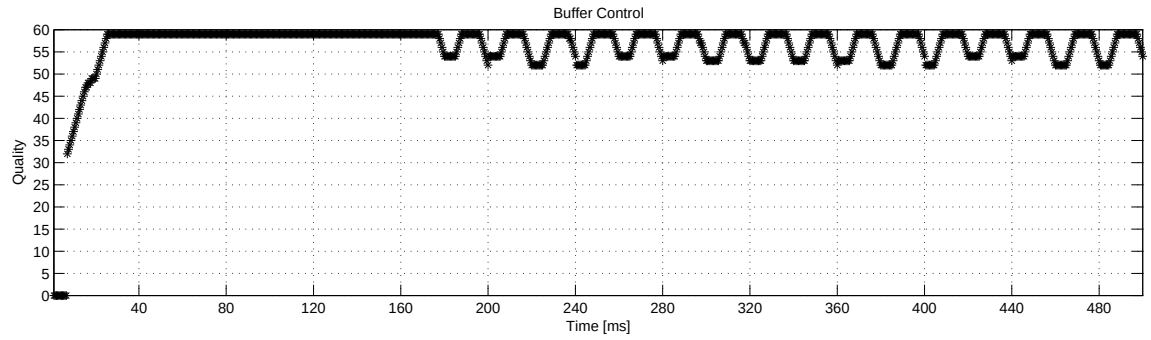
The defined quality states are listed in table 7.1. We have 60 states altogether, which go from 0 to 59, and the scale is reversed, in the sense that the state 0 corresponds to the best encoding quality, while state 59 corresponds to the worst one.

7.3 BUFFER CONTROL

The implemented rate control strategy is the one described in 5.4.1 which is based on both a *hard* and *soft* reaction depending on the *buffer fullness* B_F and its corresponding *arrival rhythm* $\Delta B_F/\Delta T$. This method is based on the quality states of table 7.1 and has been fully tested on the active range of encoding bit rates $20 \div 50$ Mbps.



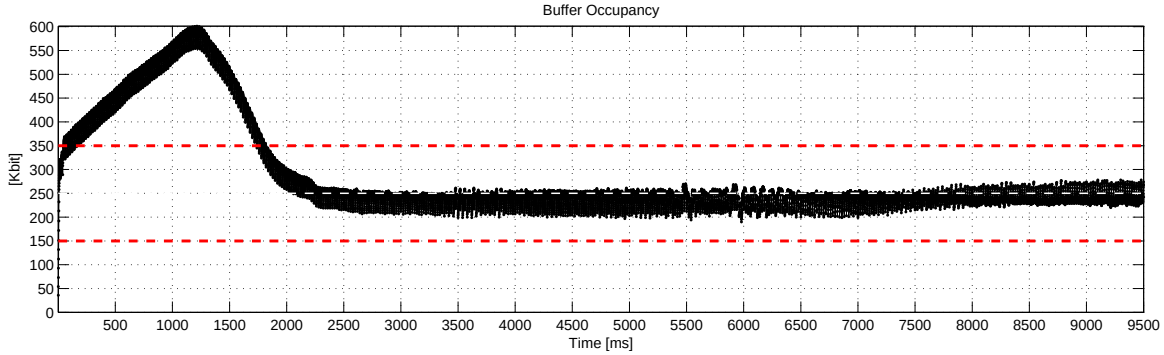
(a)



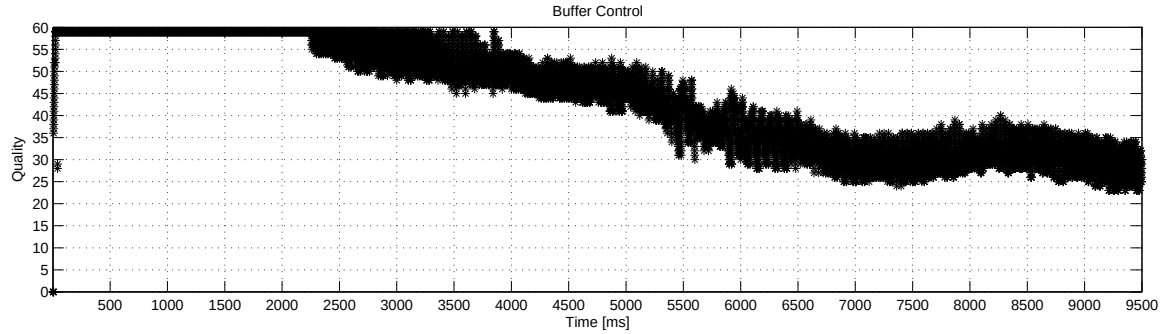
(b)

Figure 7.4: Occupancy level and encoding quality at 24 Mbps (a)-(b)

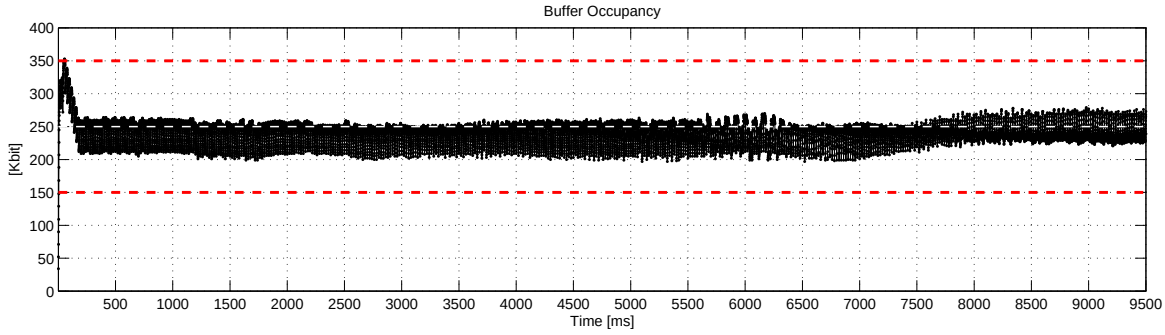
Figure 7.4(a) shows the evolution of B_F at encoder side, during the first 500 ms. The test sequence is a recent version of the well known *Mobile Calendar* in standard PAL 4:2:0 format. The following simulations have been carried out using the HERMES codec C++ software, configured to work in field-slice mode, with a constant slice size of 8 lines. In Figure 7.4(b) the buffer control acts at every slice end, deciding each time which quality to adopt to encode the incoming slice. Each point in figures 7.4(a)(b) has been collected at the end of a single slice coding. Every arc shape in figure 7.4(a) corresponds to a field formed of 36 slices. Two consecutive arcs take 40 ms and correspond to a frame. B_F rapid changes between adjacent fields is due to the vertical blanking interval, which has been compensated to avoid the negative effects over the encoding quality decision already introduced in section 5.4.3.



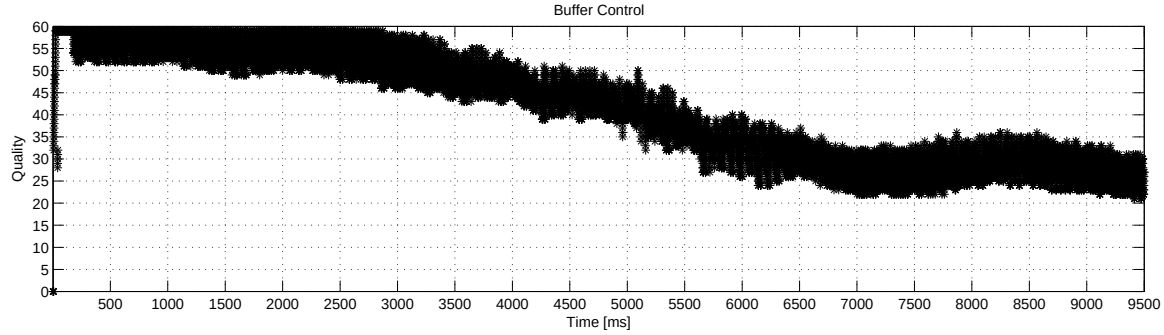
(a)



(b)



(c)



(d)

Figure 7.5: Occupancy level and encoding quality at 23 Mbps (a)-(b) and 24 Mbps (c)-(d)

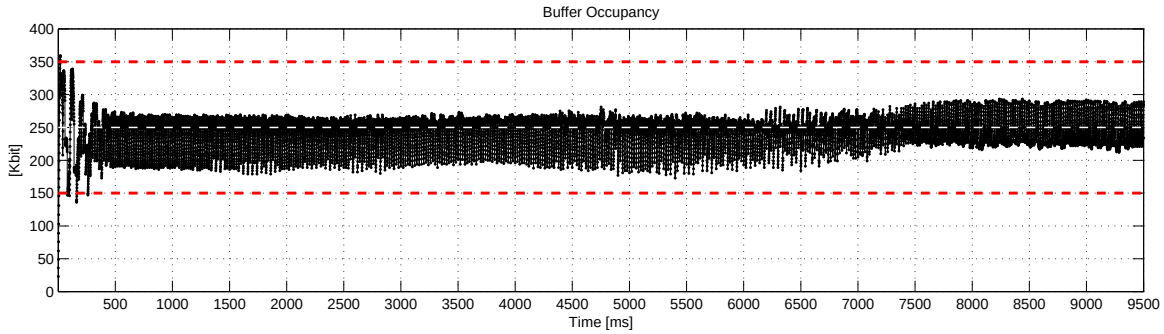
The dotted lines of figure 7.4(a) put in evidence two levels (150 Kbit and 350 Kbit) which divide the buffer depth into zones associated to different kinds of controls (refer to section 5.4.1). If $B_F < 150$ Kbit, as in figure 7.4(a) within first few ms, the *hard* control is activated and the best encoding quality (State 0 of table 7.1) is selected until $B_F = 150$ Kbit, then the *soft* control strategy is employed to constraint B_F between dotted lines and close to half buffer size (250 Kbit).

within the same field, the encoding quality distribution (Figure 7.4(b)) follows the incoming slices complexity function, which may show peaks and discontinuities Therefore rapid transitions are possible and have to be controlled. However, in natural images smoothed transitions are more common. Therefore we expect continuity between consecutive slices but for last and first slice of different fields. We also expect similar quality distributions within consecutive fields, which are supposed to be highly correlated.

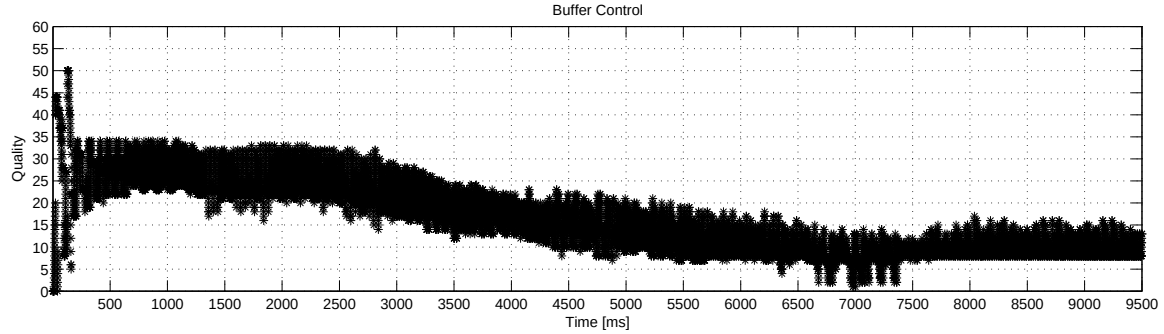
Figures 7.5(a)(b) show the buffer control evolution at 23 Mbps. In the first part of the simulation we can see that despite the control selects the lowest quality (State 59 of table 7.1), B_F keeps on increasing till the maximum occupancy of $\simeq$ 600 Kbit is reached. The first part of the input sequence carries more information, so it is more difficult to encode than the second part, what explain the better behavior of the control in figure 7.5(a) over the zone t $>$ 1200 ms. Therefore the rate 23 Mbps have to be considered as non-controllable, because even at the lowest quality, HERMES may produce a greater amount of encoded bits per second than the one read out from the buffer.

The first bit rate which correspond to a good control is 24 Mbps, whose control curves are shown in figure 7.5(c)(d). In this case B_F is under soft control during all the time. Thus, despite a low encoding quality is often employed, 24 Mbps is the lowest encoding rate limit for HERMES, related to the reference test sequence.

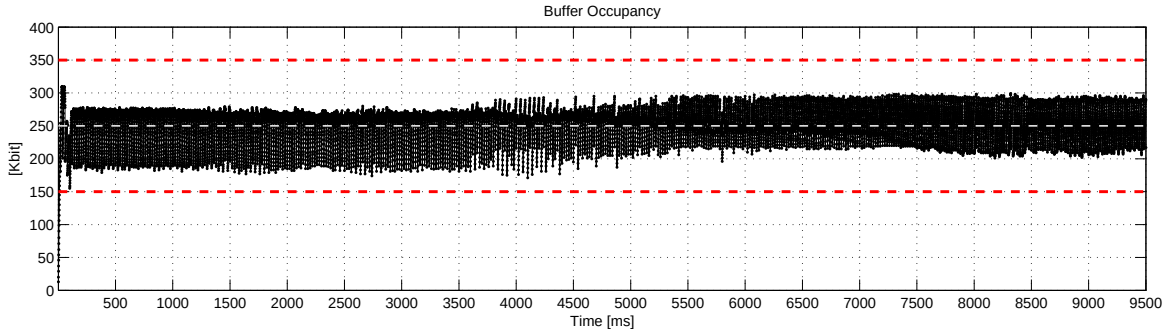
For encoding bit rates greater than 24 Mbps, the *soft* control is enough to constraint B_F between the dotted lines limits, while the employed encoding quality increases. Further examples are shown in figures 7.6(a)(c)(b)(d) which describe the buffer occupancy evolution (Figures 7.6(a)(c)) and the corresponding encoding quality (7.6(b)(d)) at rates of 35 and 45 Mbps.



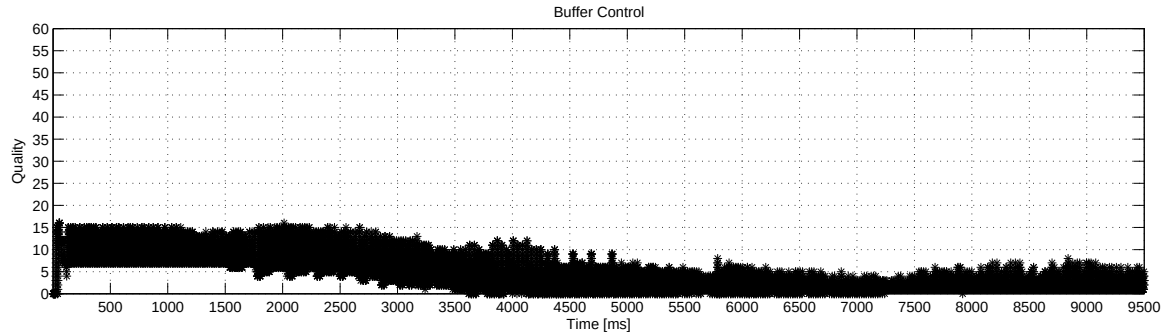
(a)



(b)

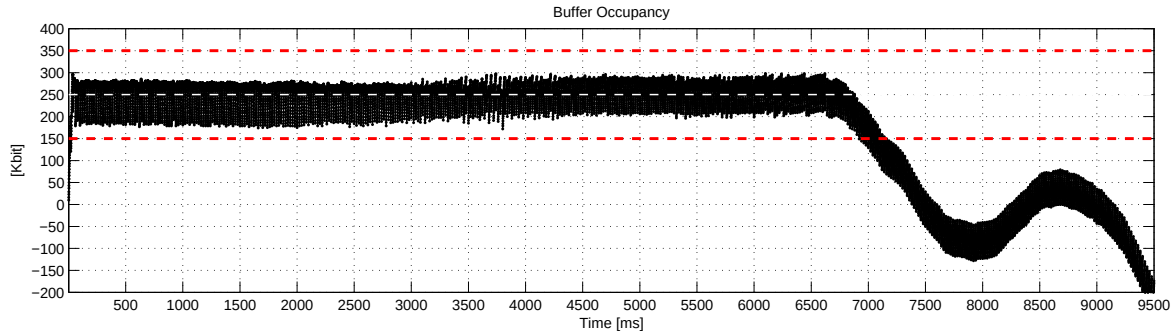


(c)

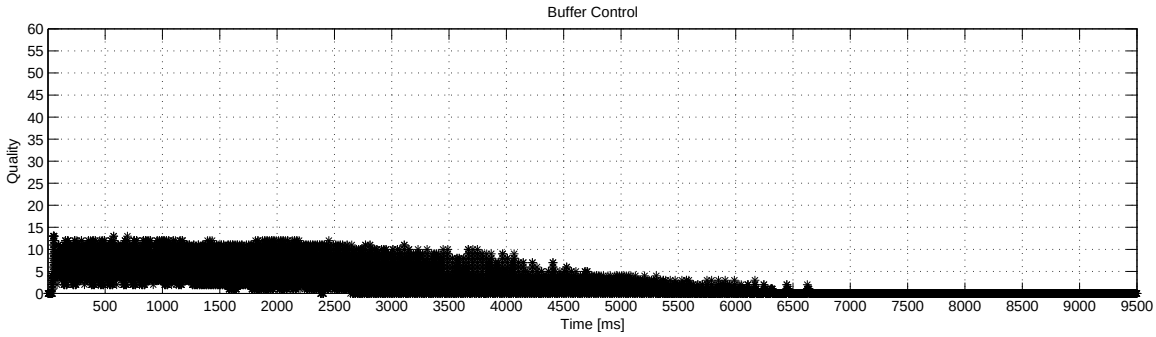


(d)

Figure 7.6: Occupancy level and encoding quality at 35 Mbps (a)-(b) and 45 Mbps (c)-(d)



(a)



(b)

Figure 7.7: Occupancy level and encoding quality at 48 Mbps (a)-(b)

At 48 Mbps, most of the sequence is encoded at maximum quality (Figure 7.7(b)), however if the amount of carried information decreases and the sequence becomes easier to encode then the control cannot avoid the underflow condition, as shown in figure 7.7(a). This means that even using the best encoding quality, at 48 Mbps HERMES does not produce enough bits to fill the buffer. This problem can be efficiently solved using stuffing techniques, which means sending empty packets in order to satisfy the channel request whenever the output buffer is quite empty. Therefore we can consider 48 Mbps as the top encoding rate limit which needs stuffing to assure the channel rate continuity.

Further tests have been carried out using the sequences listed in table 7.2, whose last column shows the minimum encoding bit rate required to achieve the buffer control. The sequence *Noise* has been generated using a pseudo random noise generator, in order to get something similar to the typical video pattern generated by a TV without input signal. *Noise* is not a natural video sequence and moreover, there is no spatial neither temporal correlation within close pixels. Therefore the over range limit of 75 Mbps can be explained.

The best case of 15 Mbps corresponds to *MIR* which is a simple color bars test sequence, thus without any movement and very high spatial and temporal correlation. On the other hand, many details and much movement are present in *Basketball*, a sport sequence which represents the worst case at 28 Mbps. Between *Basketball* and *MIR* we have different levels of coding complexity.

SEQUENCE NAME (YUV)	RESOLUTION	FRAMES	BOTTOM RATE LIMIT (Mbps)
Noise	720×576	150	75
Basketball	704×576	250	28
Mobcal	720×576	250	24
Harbour	704×576	300	23
Horses	720×576	110	23
Soccer	704×576	300	20
City	704×576	300	20
Crew	704×576	300	19
Ice	704×576	250	16
MIR	720×576	300	15

Table 7.2: Reference sequences used to test HERMES rate control algorithm

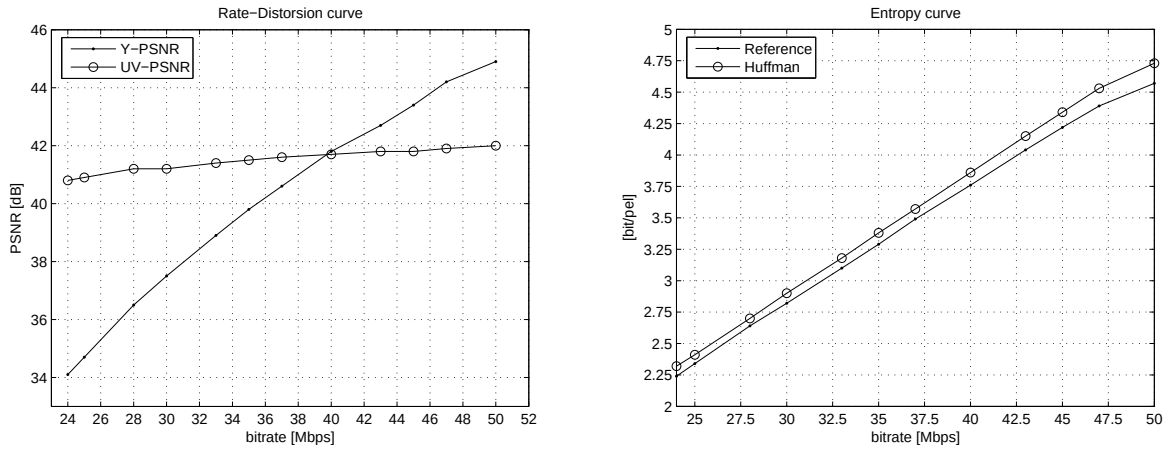
Finally we can state that: the buffer control strategy exposed in section 5.4.1 can efficiently constraint B_F into a *soft* control zone of amplitude 200 Kbit for encoding bit rates $R > 28$ Mbps. Therefore, even using a redundant buffer size $B = 400$ Kbit, we expect a delay $D < B/R \simeq 14$ ms. Moreover, the higher the bit rate R, the lower the delay D and the better the encoding quality.

7.4 HERMES PERFORMANCES

The figure 7.8(a) shows both luminance and chrominance rate-distortion curve, which plots the objective quality (PSNR) vs the encoding bit rate and it is a good method to represent the algorithm efficiency. As expected the Y-PSNR grows between 34 and 45 dB, which are the bottom and top limits found during the quality calibration in section 7.2.

Because the chrominance component is processed by a fixed 5-levels quantizer, we would expect a constant UV-PSNR, which instead slowly increases from 41 to 42 dB over the bit rate window. However, the reliability threshold is higher at lower bit rates, which means that a greater amount of couples (Y, C) are not encoded at all due to conditional coding (Refer to section 5.2.4). At decoder side both Y and C components will be reconstructed using their prediction only. This mechanism introduces an additional error which can explain the light UV-PSNR curve degradation at lower bit rates.

Figure 7.8(b) compares the average bit per pixel ((Y, C) couple) required by the implemented set of Huffman codes to the theoretical reference value based on the optimal set of Huffman codes calculated using real time statistics. The difference between the curves depends on two factors:



(a) (b)

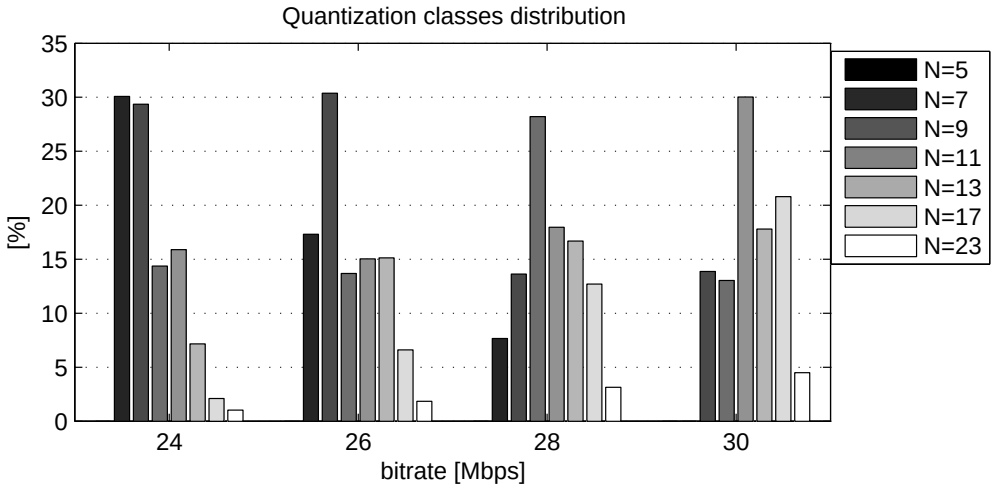
Figure 7.8: HERMES encoder: (a) Rate-Distortion Curve, (b) Entropy curve

- Given a quantization class N we use the same Huffman codes whatever sub-quantizer Q_N^a, Q_N^b or Q_N^c would be selected by the activity function (Refer to section 5.2.3).
- For $N = 23, 17, 13$ a set of length-limited Huffman codes with $L_{max} = 16$ bits [48, 49] have been adopted in order to reduce the hardware complexity.

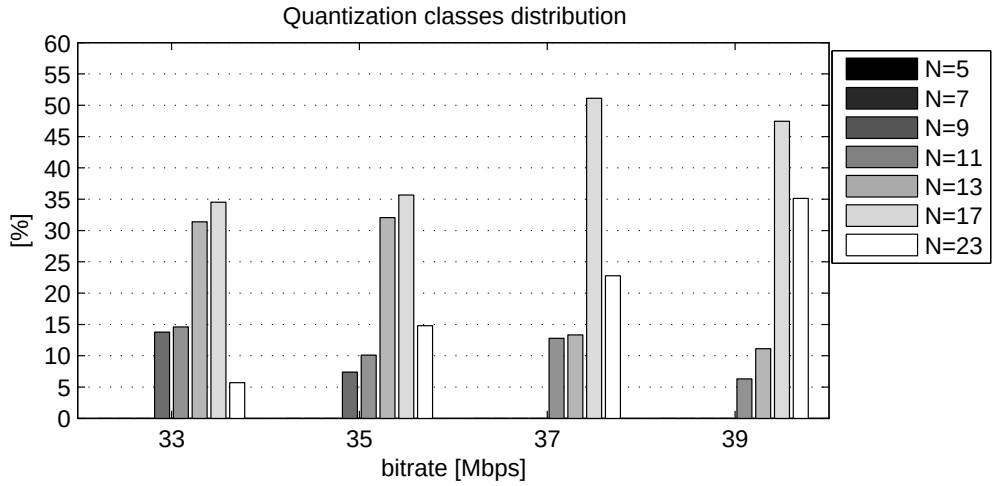
Looking at figure 7.8(b) we can state that the modified Huffman codes do not cause a sensitive loss of performances. The degradation is less than 0.12 bit/pel at lower bit rates and it increases to 0.2 bit/pel at 50 Mbps.

Figures 7.9(a)(b)(c) show the percentage of use for each quantization class at different bit rates. We can observe that the use of higher quantization classes increases with the bit rate. On the other hand the use of lower quantization classes decreases more and more until the complete abandon.

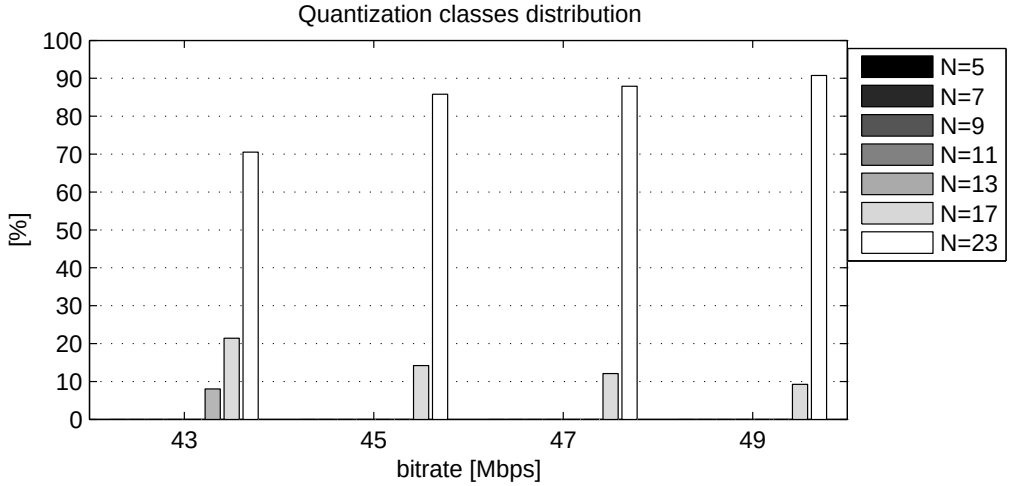
- Between 24 and 28 Mbps (Figure 7.9(a)) the classes N=5, 7, 9 are mainly used.
- From 30 Mbps $N = 5$ is no more employed and class $N = 11$ gets the higher percentage.
- At 33 Mbps $N = 13, 17$ are mainly used, while $N = 7$ has already been abandoned (Figure 7.9(b)).
- At 37 Mbps $N = 17$ has its maximum while $N = 9$ disappears.
- At bit rates higher than 43 Mbps (Figure 7.9(c)), $N = 23$ is used up to 90%.



(a)

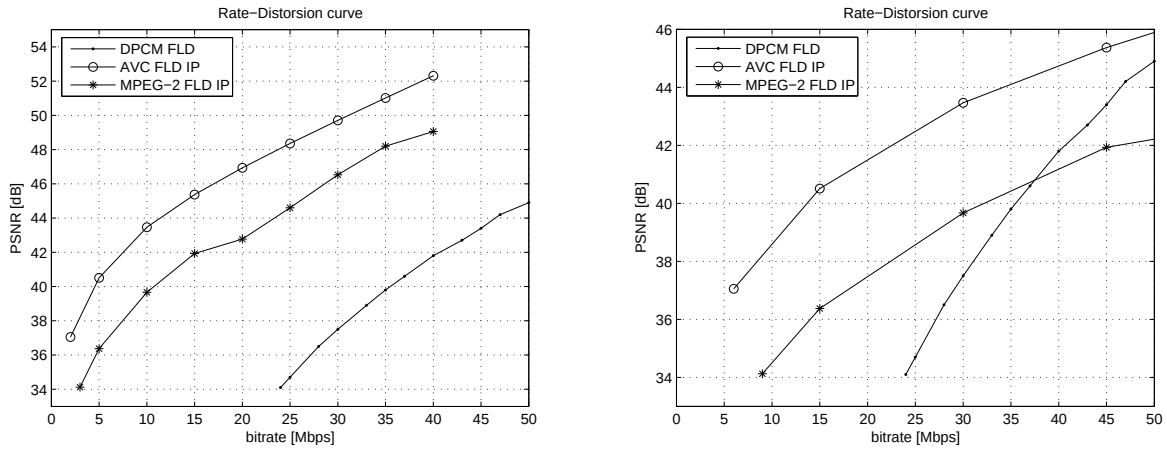


(b)



(c)

Figure 7.9: Quantization statistics: (a) Low bit rates, (b) Medium bit rates, (c) High bit rates



(a) (b)

Figure 7.10: MPEG vs HERMES Rate-Distortion Curve: (a) K=1, (b) K=3

Finally, we can compare HERMES encoder performances to MPEG. Figure 7.10(a) shows the rate-distortion curves of HERMES, MPEG-2 and AVC/H.264 respectively. The MPEG curves have been generated using dedicated video verification models which are: *MSYS Toolkit v1.1 (1994)* for MPEG-2 and the *JM Reference Software v8.2 (2001)* for AVC/H.264. To make a real comparison the MPEG software have been configured to work in ULD mode (refer to section 4.3):

- Field Encoding
- No B-pictures
- GOP-length = ∞
- Intra-Slice Mode

It is clear from figure 7.10(a) that MPEG outperforms HERMES from the compression efficiency point of view. The weakest point of HERMES is its low compression ratio flexibility, which covers the range $\simeq 3 \div 8$ only. However, a more fair comparison is shown in figure 7.10(b) where a bandwidth overhead factor $K = 3$ has been taken into account, according to the analysis presented in section 4.3. HERMES does not need a bandwidth overhead, because it exploits an intra-only coding, thus a rapid scene change does not affect HERMES buffer control as it does in MPEG systems. The result is that over 36 Mbps HERMES overpasses MPEG-2 encoding quality, using a much less complex hardware and above all achieving a coding delay $D < 12$ ms, which is 42 ms less than what has been performed with MPEG until now.

7.5 CODING IMPAIRMENTS

Section 5.2 shown that an interesting property of HERMES encoding scheme is the identity between quantization error E_q and reconstruction error E_r. However, the quantization noise E_q is not the direct cause of encoding artifacts in HERMES system. Actually it is the slice coding method (refer to section 5.2.7) which introduces annoying impairments on first slice lines.

Each first line of a slice is not processed using the previous line as reference, but a uniform line made of middle scale pixels is considered. This guarantees the slice coding independency condition, but it also affects the prediction efficiency. In the case that a edge shape would cause a rapid luminosity change across adjacent slices, then the pixels on the first slice line and close to that edge are roughly predicted, which means a great residual error that can lead to a great local quantization error.

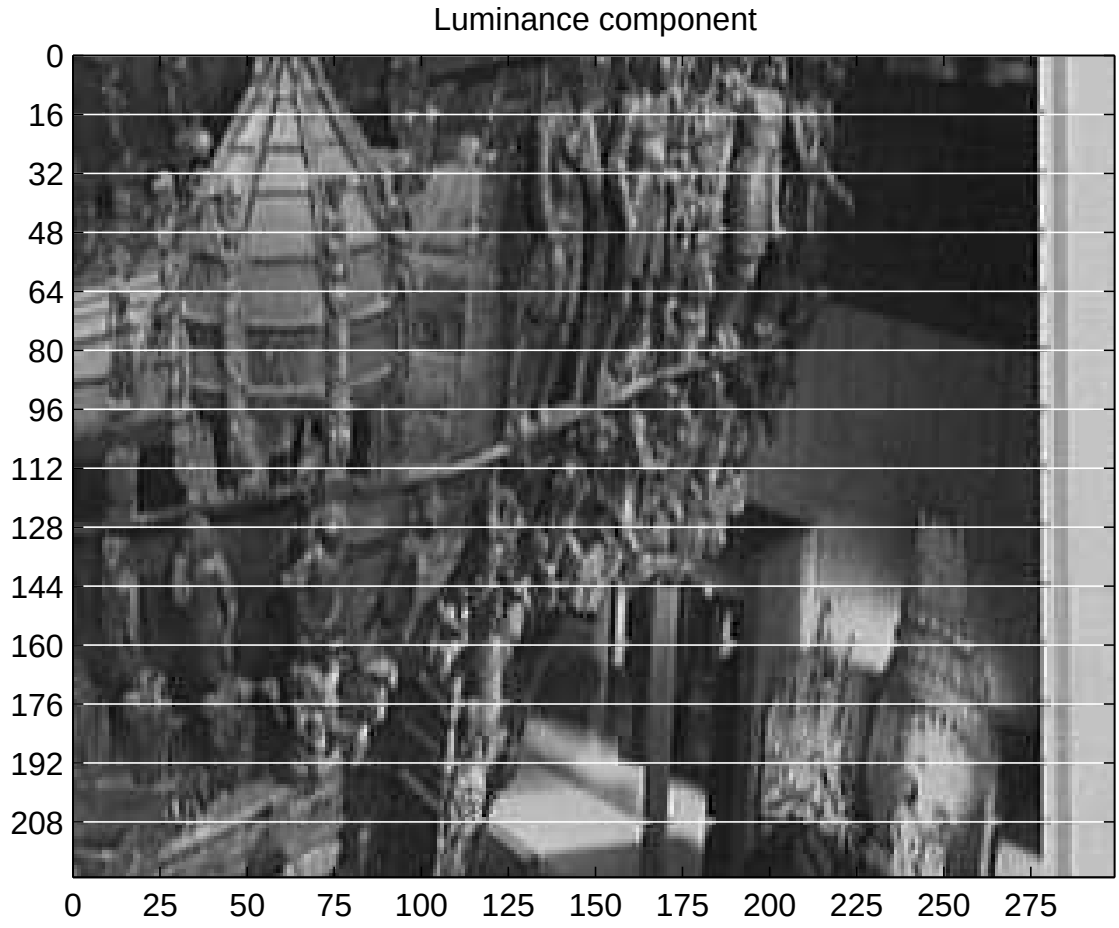


Figure 7.11: HERMES coding impairments

The problem is well represented in figure 7.11 which is the result of HERMES encoding/decoding process in field mode with slice size of 8 lines. Because of the line interlacing we have a periodicity of 16, thus all the first lines of field slices are located respectively at positions: 1-2, 17-18, 33-34,

etc., close to the horizontal white lines. The slice coding impairments appear as grey pixels which are clearly visible across the sharp luminosity edge close to column 275, along the white lines.

This artifact is further reduced when higher quantization classes are employed (higher encoding bit rates) and disabling the conditional coding option on each first line of a slice. However it does not completely disappear. In order to solve this problem of keeping the slice independency condition we can implement a slice overlapping method. One line overlap is enough to delete this kind of artifact. The price to pay is a slight decrease of coding efficiency, and some minor modifications on the processing point of view. A second possibility is the implementation of a *de-slicing* spatial filter at decoder side, which could be applied over the slice edges, along the vertical direction only.

7.6 CONCLUSIONS

A set of quality states have been defined in order to implement an efficient buffer control strategy which make it possible to simulate and characterize a rate controlled video codec based on HERMES core. The rate-distortion curve has been recovered and the system has been fully evaluated taking into account its limits and advantages.

The comparison with MPEG puts in evidence a reduced compression ratio flexibility. However HERMES does not require a bandwidth overhead to implement a ULD coding and it performs higher coding quality than MPEG-2 at bit rates $R > 36$ Mbps, achieving a global coding delay $D < 12$ ms with a redundant buffer size $B = 400$ Kbit. The slice-coding artifact have been described and a low cost solution has been proposed.

Chapter 8

Conclusions

8.1 INTRODUCTION

This chapter concludes the dissertation; it summarizes the main developments and results and indicates the direction of future work.

8.2 SUMMARY OF THE ACHIEVEMENTS

The demand for digital wireless links is more and more increasing in professional video domain, however some problems of compatibility are still open concerning applications which require to interface wireless and wired systems. When switching between cabled and wireless cameras, the residual coding/decoding latency makes it impossible to achieve the audio-video synchronization, leading to disagreeable effects. A simple solution to this problem is to use a very low delay compression algorithm which kept at minimum the processing latency.

Most of present wireless solution are based on MPEG-2 standard. However, MPEG does not represent a good solution from the latency point of view. The standard has evolved to satisfy the request of higher coding quality at lower bit rates, where the processing latency was not a critical constraint. This attitude also comes from the analysis of the most advanced MPEG standards. MPEG-4 and AVC/H.264 offer more efficient compression and unparalleled flexibility to meet the needs of practical multimedia communication applications. The diverse set of coding tools described in these standards address the wider range of applications ever supported. However, the algorithm complexity has reach high levels and compact implementations require big efforts from developers and manufacturers.

In order to implement a low latency system based on MPEG standard, a subset of simple coding tools can be identified. The study of an existing MPEG-2 low latency system makes it possible to understand the implementation issue coming from the VBV buffer control strategy, which puts in evidence a strong trade-off between latency and bandwidth overhead. MPEG also misses an audio low delay coding technique which is required in order to support an integrated audio/video codec. Further problems come from the available hardware platforms, which are often non optimized to achieve a low latency.

The introduction of MPEG-4 and AVC/H.264 seems not to give any contribution to the problem of latency, whose minimum has actually been identified around 54 ms for a SD resolution PAL video link, at the price of 66% of bandwidth overhead required to overcome buffering problems. To find better solutions, a non-standard algorithm has to be exploited.

Chapter 5 introduced HERMES, a non-standard algorithm which has been designed to perform a very low delay exploiting a light complexity. A prototype version of HERMES has been implemented on Xilinx FPGA platform, while the final design is still under development and optimization. HERMES intends to be an alternative solution to MPEG only for a restrict field of applications where a coding/decoding delay shorter than one single frame at high quality is

required.

HERMES implementation has made it possible to verify the functionality of the most important VHDL blocks such as HERMES encoder/decoder cores. The design requires few on chip resources especially if compared to an MPEG-4 architecture. The actual VBR HERMES implementation does not allow to make real measurements of the coding/decoding delay, however a buffer control strategy has been tested using a C++ simulator.

The comparison with MPEG puts in evidence a reduced compression ratio flexibility. However HERMES does not require a bandwidth overhead to achieve low latency coding and it shows higher coding quality than MPEG-2 at bit rates $R > 36$ Mbps, achieving a global coding delay $D < 16$ ms with a redundant buffer size $B = 400$ Kbit.

8.3 CONCLUSIONS AND FURTHER SCOPES

The optimization of the coding/decoding latency in MPEG is both a theoretical and implementation problem. The set of MPEG features which allows a low delay coding can be isolated and analyzed, but the real system performance depends on few key factors such as: field-encoding, intra-slice mode, infinite GOP and others, which are not actually widely supported by hardware developers. Furthermore, the set of MPEG low delay processing tools does not include bidirectional coding and impose strong constraints on the GOP, which limit the MPEG compression efficiency. To achieve a low delay coding in MPEG is possible at the price of higher encoding rates and a sub-utilization of the available coding tools supported by standard platforms.

On the other hand, the theoretical problem of the VBV buffer optimization for low delay purpose is still open. MPEG does not give any indications about the buffer control strategy, a task which has been completely delegated to the developer. Moreover, the best known implemented solution is based on a very inefficient method which waste a considerable part of the available bandwidth in the effort of keeping low the encoder buffer occupancy.

In this scenario, the last MPEG standards: MPEG-4 and H.264, which represent the state of the art in video compression, offer more and more complex features and higher compression efficiency, which in fact are not utilizable to achieve a really low delay coding scheme. Therefore, some applications as wireless studio production, prefer to employ non-standard platforms, which allow lower complexity and latency, despite the missing of inter-operability. That is why, low complex algorithms like the HERMES codec, acquire higher importance, since they propose a valid alternative to a MPEG-based low delay scheme.

HERMES codec has shown the ability to achieve latencies shorter than 20 ms with low resource requirements. A rate-controlled version of HERMES has been tested and compared with MPEG-2 and MPEG-4 codec. As similar low delay coding schemes, such as wavelets and DV codec, HERMES does not achieve high compression ratios. However, the coding quality is always over 34 dB and pictures are not affected by typical MPEG block-shape artifacts. The only artifact is

due to the slice-coding strategy and can be avoided employing some minor modifications to the processing scheme.

HERMES is an alternative to MPEG for applications which require a very short encoding/decoding latency having access to bit rates over 22 Mbps for data transmission. The development of an enhanced OFDM modulation scheme [2] makes it possible the integration of HERMES into LiveTools DVB-T system [11], providing a valid solution for wireless studio production applications.

HERMES coding scheme is not yet a final solution, but its encouraging performances justify further efforts in its optimization and development. A weak point of HERMES design is the entropy coding, which have to be enhanced in order to support an adaptive scheme capable to limit the actual strong dependency on the kind of sequence under test. An other important point is the audio, which is not compressed in the present design. Therefore, a low latency audio-coding method is required in order to save bandwidth. Finally, further investigations could improve the quantization classes, in order to better exploit the visual masking effect, achieving higher compression ratios.

Bibliography

[1] www.gigawave.co.uk/dcamdev.html

[2] R. Posega, *Enhanced OFDM System for Digital News Gathering*, Ph.D. Dissertation, EPFL, Lausanne, March 2005.

[3] Tim Ferguson *Digital Video Coding and Compression*, School of Computer Science and Software Engineering, Monash University, Australia.

[4] Kelth Jack, *Video Demystified. A Hand book for the Digital Engineer*, LLH Technology Publishing, Eagle Rock, VA, 3rd ed (2001).

[5] Iain E.G. Richardson, *H.264 and MPEG-4 Video Compression*, Video Coding for Next-generation Multimedia, The Robert Gordon University, Aberdeen, John Willey & Sons Ltd (2003).

[6] Jie Chen, Ut-Va Koc, K.J. Ray Liu, *Design of digital video coding systems*, Signal Processing and Communications Series, Marcel Dekker, AG (2002).

[7] V. Bhaskaran, K. Konstantinides, *Image and Video Compression Standards - Algorithms and Architectures - Second Edition*, Kluwer Academic Publishers, 1997.

[8] Pan Feng Nanyang, *Digital Television Terrestrial Broadcasting Primer*, Technological University, Singapore, TechOnLine Publications (October 2001).

[9] *Wavelet-based Color Image Compression: Exploiting the Contrast Sensitivity Function* Marcus J. Nadenau, Member, IEEE, Julien Reichel, Member, IEEE and Murat Kunt, Fellow, IEEE

[10] http://www.adamwilt.com/DVvsMJPEG.html

[11] www.livetools.tv

[12] www.epfl.ch

[13] www.xilinx.com

[14] ITU-R Recommendation BT.601-5, *Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios*

[15] ITU-R Recommendation BT.500-11, *Methodology for the subjective assessment of the quality of television pictures*, Question ITU-R 211/11, (1974 - 1978 - 1982 - 1986 - 1990 - 1992 - 1994 - 1995 - 1998 - 1998 - 2000 - 2002).

[16] Q. Wang, R.J. Clarke *Motion estimation and compensation for image sequence coding*, Signal Processing: Image Communication, Vol. 4, No. 2, pp 161 - 174, April 1992

[17] Gaetano Caronna, *Adaptive DPCM with Conditional Coding*, CSELT S.p.A., Signal Processing of HDTV, II, L. Chiarigione (ed.), Elsevier Science Publischers B.V. 1990.

[18] Tiina Jarske and Yrj Neuvo, *Adaptive DPCM with median type predictors*, Tampere University of Technology, Finland, IEEE Transactions on Consumer Electronics, Vol. 37, No. 3, August 1991.

[19] M. Ravasi, L. Tenze and M. Mattavelli A Scalable and programmable architecture for 2D DWT decoding, IEEE Trans. Circuits Systems Video Technology, Vol. 12, No. 8, August 2002.

[20] Kevin Brown, Katie Streit, Luke Hoban, Rob Gaddi, *2D Frequency Domain Filtering and the 2D DFT*, www.owlnet.rice.edu

[21] N. Nasrabadi and R. King, *Image coding using vector quantisation: a review* IEEE Trans. Commun, 36 (8), August 1988.

[22] Woo Young Choi, Rae-Hong Park, *Motion vector coding with conditional transmission Signal Processing*, Vol. 18, No. 3, November 1989.

[23] D. A. Huffman, *A method for the construction of minimum redundancy codes*, Proc. of the IRE, 40, p 1098-1101, (1952).

[24] I. Witten, R. Neal and J. Cleary, *Arithmetic coding for data compression*, communications of the ACM, 30 (6), june 1987.

[25] J. Ziv and A. Lempel, *A Universal Algorithm for Sequential compression*, IEEE Transaction on Information Theory, Vol. 23, No. 3, pp. 337-343, 1977.

[26] Welch, T. A., *A Technique for High Performance Data Compression*, IEEE Computer, vol. 17, no. 6, June 1984.

[27] P. A. A. Assuno and M. Ghanbari, *Buffer Analysis and Control in CBR Video Transcoding*, IEEE Transaction and systems for video technology, Vol. 10, No. 1, February 2000.

[28] ISO/IEC 11172-2, Information technology, *Coding of moving pictures and associated audio for digital storage media at up to 1.5Mbit/s video*, Geneve, Switzerland, 1994, MPEG1.

[29] ISO/IEC 13818-2, Information technology, *Generic coding of moving pictures and associated audio information: video*, Geneve, Switzerland, 1996, MPEG2.

[30] ISO/IEC 14496-2, Information technology, *Generic coding of audio-visual objects-Part 2: Visual*, Geneve, Switzerland, 1998, MPEG4.

[31] ISO/IEC 14496-10 and ITU-T Rec. H.264, *Advanced Video Coding*, 2003.

[32] H. S. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky, *Low-Complexity Transform and Quantization in H.264/AVC* IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, No. 7, July 2003.

[33] P. List, A. Joch, J. Lainema, G. Bjntegaard, and M. Karczewicz *Adaptive Deblocking Filter*, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, No. 7, July 2003.

[34] M. Karczewicz and R. Kurceren, *The SP- and SI-Frames Design for H.264/AVC*, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, No. 7, July 2003.

[35] G. Bjntegaard and K. Lillevold, *Context-Adaptive VLC Coding of Coefficients*, JVT document JVT-C028, Fairfax, May 2002.

[36] D. Marpe, H. Schwarz and T. Wiegand, *Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard*, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, No. 7, July 2003.

[37] MPEG 2 Test Model 5, Rev. 2, Section 10 *Rate Control and Quantization Optimization*, ISO/IEC/JTC1SC29WG11, April 1993.

[38] G. Sullivan, T. Wiegand and K.P. Lim, *Joint Model Reference Encoding Methods and Decoding Concealment Methods; Section 2.6: Rate Control*, JVT-I049, San Diego, September 2003.

[39] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini and G. Sullivan, *Rate-Constrained Coder Control and Comparison of Video Coding Standards*, IEEE Transactions on Circuits & Systems for Video Technology, Vol. 13, No. 7, July 2003.

[40] M. Lattuada, *Efficient Error Correction Solution for OFDM Based Digital Video*, Ph.D. Dissertation, EPFL, Lausanne, January 2005.

[41] P. Pirsch, *Design of a DPCM Codec for VLSI Realization in CMOS Technology*, Proc. IEEE, Vol 73, No 4, April 1985, pp. 592-598.

[42] K. A. Prabhu, *A Predictor Switching Scheme for DPCM Coding of Video Signals*, IEEE Transactions on Communications, Vol. 33, pp. 373-379, April 1985.

[43] H. Mukamy, S. Matsumoto, Y. Hatori, H. Yamamoto *15/30 Mbitls Universal Digital TV Codec Using a Median Adaptive Predictive Coding Method*, IEEE Transactions on Communications, Vol. 35, No. 6, June 1987.

[44] Chairat Rittirong, Yuttapong Rangsanseri, and Punya Thitimajshima, *Multispectral Image Compression using Median Predictive Coding and Wavelet Transform*, Department of Telecommunications Engineering, Faculty of Engineering King Mongkut's Institute of Technology Ladkraband, Bangkok

[45] J. Salo et al., *Improving TV Picture Quality with Linear-Median Type Operations*, IEEE Transactions on Consumer Electronics, Vol. 34, No. 3, August 1988, pp. 370-379.

[46] P. Pirsch, *Design of DPCM Quantizers for Video Signals Using Subjective Tests*, IEEE Transaction on Communications, Vol. 29, No. 7, July 1981, pp. 990-1000.

[47] R. C. Brainard et al., *Predictive Coding of Composite NTSC Color Television Signals*, SMPTE Journal, March 1982, pp. 251-258.

[48] M. Liddell A. Moffat, *Length-Restricted Coding Using Modified Probability Distributions*, Department of Computer Science and Software Engineering, Melbourne University, IEEE 2001.

[49] L. L. Larmore and D. S. Hirshberg, *A Fast Algorithm for Optimal Length-Limited Huffman Codes*, Journal of the ACM, Vol. 37, pp. 464-473, July 1990.

[50] ITU-R Recommendation BT.656-4, *Interfaces for digital component video signals in 525-line and 625-line television systems operating at the 4:2:2 level of Recommendation ITU-R BT.601 (Part A)*

[51] A. Moffat and A. Turpin, *On the Implementation of Minimum Redundancy Prefix Codes*, IEEE Transaction on Communications, Vol. 45, No. 10, October 1997.

[52] Xilinx Company *MPEG-4 Encoder V0.1*, Product Specification, June 9, 2005.

[53] S. Saponara, C. Blanch, K. Denolf, J. Bormans *The JVT Advanced Video Coding Standard: Complexity and Performances Analysys on a Tool-by-Tool Basis*, IMEC, Belgium, 2002.

Graziano VAROTTO CV

CONTACT INFORMATION

Ch. de Contigny 7
CH-1007 Lausanne

Private: (+41) 21 6011018
Mobile: (+41) 76 5635353
E-mail: graziano.varotto@epfl.ch

32 years old, born in Livorno (Italy)
Italian, B working permit for Switzerland
Single

EDUCATION

2003 PhD candidate at LTS3 laboratory. The research project is focused on the study and optimization of a low delay video coding algorithm, finalized to a VHDL implementation on Xilinx FPGA platform

2000 Military duty as Navy officer (14 months)

1999 Laurea di dottore in Ingegneria Elettronica at Universitá degli studi di Pisa with final mark 110/110. The thesis *Project of an FPGA-based associative memory for track finding in hadronic colliders* has been developed at INFN (National Institute of Nuclear Physic) in S.Piero a Grado (Pisa)

1992 Maturità scientifica at liceo *F. Cecioni* in Livorno. Final mark 56/60

PROFESSIONAL EXPERIENCE

2001 Research assistant at EPFL-ITS-LTS3 in the *Broadcast Project* framework. Collaboration with LiveTools Technology (www.livetools.tv) as R&D engineer in the domain of wireless broadcasting video applications

PUBLICATIONS

2000 *A pipeline of Associative memory boards for Track Finding* M.G. Bagliesi, A.Bardi, R.Carosi, M.Dell'Orso, P.Giannetti, G.Iannaccone, F.Morsani, M.Pietri, G.Varotto, Proc. IEEE Nuclear Science Symposium, Lion 15-20 Oct. 2000.

2000 *The Fast Tracker Processor for Hadronic Collider Triggers* M.G. Bagliesi, A.Bardi, R.Carosi, M.Dell'Orso, M.D'Onofrio, P.Giannetti, G.Iannaccone, F.Morsani, M.Pietri, G.Varotto, Proc. IEEE Nuclear Science Symposium, Lion 15-20 Oct. 2000.

2000 *Real Time Track Finding in CMS*, R.Carosi, G.Iannaccone, G.Varotto, CMS internal note 2000/023.

SKILLS

Video coding
Digital signal processing
Embedded system design

Design tools:	Synplify, Synopsys, Xilinx ISE, Protel
Simulation tools:	Modelsim, Matlab
Programming language:	VHDL, C/C++

LANGUAGES

French:	Fluent
English:	Fluent
Italian:	Mother tongue