

FPGA PROTOTYPING OF A WATERMARKING

ALGORITHM FOR MPEG-4

Wei Cai, B.E.

Thesis Prepared for the Degree of

MASTER OF SCIENCE

UNIVERSITY OF NORTH TEXAS

May 2007

APPROVED:

Elias Kougianos, Major Professor
Saraju P. Mohanty, Co-Major Professor
Shuping Wang, Committee Member
Dan Cline, Committee Member
Vijay Vaidyanathan, Program Coordinator
Albert B. Grubbs, Jr., Chair of the Department
of Engineering Technology
Oscar Garcia, Dean of the College of
Engineering
Sandra L. Terrell, Dean of the Robert B.
Toulouse School of Graduate Studies

Cai, Wei. FPGA Prototyping of a Watermarking Algorithm for MPEG-4. Master of Science (Engineering Technology), May 2007, 96 pp., 13 tables, 50 figures, references, 77 titles.

In the immediate future, multimedia product distribution through the Internet will become main stream. However, it can also have the side effect of unauthorized duplication and distribution of multimedia products. That effect could be a critical challenge to the legal ownership of copyright and intellectual property. Many schemes have been proposed to address these issues; one is digital watermarking which is appropriate for image and video copyright protection.

Videos distributed via the Internet must be processed by compression for low bit rate, due to bandwidth limitations. The most widely adapted video compression standard is MPEG-4. Discrete cosine transform (DCT) domain watermarking is a secure algorithm which could survive video compression procedures and, most importantly, attacks attempting to remove the watermark, with a visibly degraded video quality result after the watermark attacks. For a commercial broadcasting video system, real-time response is always required. For this reason, an FPGA hardware implementation is studied in this work.

This thesis deals with video compression, watermarking algorithms and their hardware implementation with FPGAs. A prototyping VLSI architecture will implement video compression and watermarking algorithms with the FPGA. The prototype is evaluated with video and watermarking quality metrics. Finally, it is seen that the video qualities of the watermarking at the uncompressed vs. the compressed domain are only 1dB of PSNR lower. However, the cost of compressed domain watermarking is the complexity of drift compensation for canceling the drifting effect.

Copyright 2007

by

Wei Cai

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my profound gratitude to my thesis advisors: Dr. Elias Kougianos (Major Professor), and Dr. Saraju P. Mohanty (Co-Major Professor) for sharing with me their wealth of knowledge, vision and insights in the areas of video compression, watermarking and VLSI architectures and for their support with all necessary resources to accomplish my research work. Without their kindly help, encouragement and guidance, it would have been impossible for me to complete this thesis.

I would also like to thank my committee member Dr. Shuping Wang from the Engineering Technology department for her assistance during my early academic studies and my colleagues at the VLSI Design and CAD Laboratory at the department of Computer Science and Engineering for their advice during this work. I also thank Dr. Nourredine Boubekri, Dr. Albert B. Grubbs, Dr. Robert G. Hayes, Dr. Michael R. Kozak, Dr. Vijay Vaidyanathan, Mrs. Sandy Warren, Mr. Bobby Grimes, Mr. Mark Zimmerer and the staff of the Department of Engineering Technology for their friendly help and support.

Special thanks to Cheryl, Ben and my parents.

TABLE OF CONTENTS

	Page
ACKNOWLEDGEMENTS	iii
LIST OF TABLES	viii
LIST OF FIGURES	ix
Chapters	
1. INTRODUCTION	1
1.1 Motivation	1
1.1.1 MPEG-4 Video Compression Standards	1
1.1.1.1 MPEG's History	2
1.1.1.2 MPEG-4 Visual Overview	3
1.1.1.3 MPEG-4 Video Objects	3
1.1.1.4 Intra, Predicted and Bidirectional Predicted VOPs	4
1.1.2 Watermarking	7
1.1.3 FPGA (Field Programmable Gate Array) Implementation	8
1.2 Problem Statements	9
2. RELATED WORKS AND LITERATURE	10
2.1 Video Compression Algorithms	10
2.1.1 Color Spaces Conversion and Sampling Rate	10
2.1.1.1 RGB Color Space	10
2.1.1.2 YCbCr Color Space	11
2.1.1.3 Sampling Rate	12

2.1.1.4 Macroblock	13
2.1.2 Motion Estimation and Motion Compensation	13
2.1.2.1 Motion Estimation	14
2.1.2.1.1 Four Motion Vectors per Macroblock	18
2.1.2.1.2 Sub-pixel Motion Estimate	18
2.1.2.2 Block Based Motion Compensation	20
2.1.3 Discrete Cosine Transform (DCT)	21
2.1.3.1 Fourier Transform	21
2.1.3.2 Discrete Cosine Transform (DCT)	22
2.1.4 Quantization of DCT Coefficients	29
2.1.4.1 Scalar Quantization	29
2.1.5 Zigzag Scanning of DCT Coefficients	31
2.1.5.1 Discrete Cosine Transform Coefficients Matrix	32
2.1.5.2 Compression at DCT/Frequency Domain	33
2.1.6 Entropy Coding	35
2.1.6.1 Variable Length Coding (VLC)	35
2.1.6.2 Huffman Coding	35
2.2 Video Watermarking	38
2.2.1 Watermarking at Spatial Domain	38
2.2.2 Watermarking at DCT Domain	39
2.2.3 Visible and Invisible Watermarking	41
2.2.4 Drift Compensation of Visible Watermarking in Compressed Domain	43
2.3 FPGA (Field Programmable Gate Array) Implementation	43

3. VIDEO COMPRESSION AND WATERMARKING ALGORITHMS.....	45
3.1 Video Compression Algorithms	45
3.1.1 Color Space Conversion and Sample Rate Algorithm	45
3.1.2 Motion Estimation Algorithm	45
3.1.3 Fast Discrete Cosine Transform (FDCT) Algorithm	47
3.1.4 Quantization Algorithm.....	48
3.1.5 Zigzag Scanning Algorithm	50
3.1.6 Entropy Coding Algorithm	51
3.1.7 MPEG Video Compression Algorithm	53
3.2 Watermark Embedding Algorithms.....	53
3.2.1 Watermarking Algorithm in Uncompressed Domain	54
3.2.2 Watermarking with Drift Compensation Algorithm in Compressed Domain	55
4. SYSTEM ARCHITECTURE	58
4.1 Architecture of MPEG Watermarking in Uncompressed Domain	58
4.2 Architecture of MPEG Watermarking in Compressed Domain	62
5. PROTOTYPE DEVELOPMENT AND EXPERIMENTS	67
5.1 System Level Modeling with MATLAB/Simulink™	67
5.1.1 System Level Modeling Methodology	67
5.1.2 Modeling Watermarking in Uncompressed Domain	68
5.1.3 Modeling Watermarking in Compressed Domain	70
5.2 System Level Modeling with VHDL and FPGA Performances.....	74
5.2.1 Controller Performance	75
5.2.2 2-D DCT Performance	76

5.2.3 Motion Estimation	77
5.2.4 Quantization Performance	78
5.2.5 Zigzag Scanning Performance.....	78
5.3 Discussions	79
5.3.1 The Video Quality of Video Compression and Watermarking	79
5.3.2 Physical and Timing Analyzing.	81
6. CONCLUSIONS AND FUTURE WORK.....	83
6.1 Conclusions.....	83
6.2 Future Work.....	83
6.2.1 MPEG-4 Video Compression	84
6.2.2 Watermarking.....	84
6.2.3 Hardware Implementation.....	85
REFERENCES.....	86

LIST OF TABLES

	Page
1.1 Prediction equations for I, B and P frames	6
2.1 Sub-pixel motion estimate in MPEGs.....	20
2.2 DC and AC coefficients	27
2.3 Huffman code example	36
3.1 Loeffler's fast 8 element 1-D Inverse DCT algorithm.....	47
3.2 DCT adders and multipliers in total	47
3.3 MPEG video compression algorithm flow.....	53
3.4 MPEG watermarking algorithm flow in uncompressed domain	55
3.5 MPEG watermarking algorithm flow in compressed domain.	56
3.6 Comparison of first 20 coefficients of simulation and MATLAB™ dct2.....	77
3.7 Compilation and Time report of 128X128 Y frame processing in 100Mhz clock..	78
3.8 Video quality metrics of video compression and watermarking..	80
3.9 Physical and Timing results for 128X128 YCbCr frames at 400Mhz..	81

LIST OF FIGURES

	Page
1.1 Video object and video object planes.....	4
1.2 Synthesize VOs into a scene.. ..	4
1.3 I, B and P frames, forward, backward and interpolated predictions.. ..	6
2.1 4:2:0 macroblock.....	13
2.2 The redundancy among movie frames.....	14
2.3 Searching regions overlap.....	16
2.4 Three searching regions for motion estimate.. ..	17
2.5 Even and odd signals.....	23
2.6 Constructing signals with cosine and sine transforms.....	23
2.7 Calculating an 8x8 2-D DCT with 1x8 1-D DCT.. ..	26
2.8 DCT and DST frequency domain coefficients.. ..	27
2.9 Comparing the matrixes before and after DCT.....	28
2.10 Zigzag scanning of DCT coefficient matrix.....	32
2.11 Compress image at DCT domain.. ..	33
2.12 Huffman tree for Huffman coding.	37
2.13 Embedding a watermark at mid frequency.....	39
2.14 Watermark embedding at 16X16 DCT coefficients matrix.....	41
2.15 Watermarking at uncompressed and compressed domain.	42
3.1 Motion estimate data path block diagram.....	46
3.2 Motion estimate flow chart.....	46
3.3 2-D DCT component data path block structure.. ..	47

3.4	2-D DCT algorithm flow chart.....	48
3.5	Quantization component data path block diagram..	49
3.6	Quantization algorithm flow chart.	49
3.7	Zigzag scanning component data path block diagram..	50
3.8	Zigzag scanning algorithm flow chart.	51
3.9	Entropy coding (Huffman) component data path block diagram.....	52
3.10	Entropy coding (Huffman) algorithm flow chart..	52
3.11	Watermark embedding algorithm flow chart.	54
3.12	Watermarking in uncompressed domain data path and flow chart.....	54
3.13	Watermarking in compressed domain and drift compensation.....	56
4.1	Block level view of MPEG video compression and visible watermark embedding module.	58
4.2	System architecture of MPEG video compression and watermarking in compressed domain.	59
4.3	System data path of watermarking in uncompressed domain (data bus width is 12- bits).	61
4.4	System address and signals of watermarking in uncompressed domain..	61
4.5	Block level view of MPEG video compression and visible watermark embedding module on compressed domain..	62
4.6	System architecture of MPEG video compression and watermarking in compressed domain.....	63
4.7	System address and signals in compressed domain..	66

5.1 Simulink™ system block set diagram for MPEG watermarking in uncompressed domain.	68
5.2 Watermarking in uncompressed domain results (resolution 240X320).	70
5.3 Simulink™ system block set diagram for MPEG watermarking in compressed domain..	72
5.4 Watermarking in compressed domain results (resolution 240X320)..	73
5.5 Side effect of drift compensation of blur moving object..	74
5.6 Controller's FSM states diagram..	75
5.7 Controller simulation. S0 and S1 for 297us in clock 50Mhz	76
5.8 2D DCT simulation. Total processing time: 1281ns in clock 100Mhz.....	76
5.9 Motion estimate simulation. Total processing time: 51112.7ns in 100Mhz.....	77
5.10 Quantization Simulation..	78
5.11 Zigzag scanning simulation. Total processing time: 1281ns in clock 100Mhz.....	78

CHAPTER 1

INTRODUCTION

1.1 Motivation

Presently, most multimedia products like audio, video, images and text are transacted in physical media, at physical stores. However, as broadband Internet becomes available to commercial and private users, those multimedia resources can be openly accessed by the masses, and could be distributed much more quickly and widely. From this trend, one can predict that as more and more songs, movies and images will be exchanged in the Internet, the download multimedia sales will finally surpass the traditional sales channels in the near future. This trend could benefit the multimedia product owners, but also could challenge their ownership because most multimedia resources are distributed in unsecured formats. Furthermore, this situation is further degraded by the fact that duplicating and distributing digital multimedia products is almost cost-free and instantaneous. To legal authorities, arbitrating the ownership of multimedia products is not easy, unless a mechanism can guarantee the genuine integrity of copyright. Multimedia watermarking is a secure solution for copyright declaration and intellectual property protection. This thesis will study one type of multimedia, namely video and its watermarking algorithm techniques.

1.1.1 MPEG-4 Video Compression Standards

MPEG (moving picture experts group) is a branch of the ISO (International Standardization Organization) for video/audio compression and related techniques in commercial and industrial applications. The famous MPEG-1, MPEG-2, and MPEG-4

video compression standards are the results of the group's work. New standards like MPEG-9 and MPEG-21 are still under development. MPEG-1 mainly is applied for video distribution with laser video compact disks (VCD); MPEG-2 is for high-definition television (HDTV) broadcasting and high quality movie/video distribution in digital video disks (DVD). MPEG-4 is the mainstream exchangeable video format in the Internet because MPEG-4 has higher and flexible compression rate, lower bit rate, and higher efficiency. Microsoft©, Real© and Apple© support MPEG-4 standard and already have embedded MPEG-4 decoders into some of their products. Other companies or organizations also provide MPEG-4 encoders/decoders or Codecs, and there are even free products such as Xvid™ [1]. The main techniques involved in MPEG standards are color space conversion and sampling rate, block-based motion estimation and motion compensation, discrete cosine transform (DCT), zigzag scanning and entropy coding compression. The data format of the MPEG stream is in hierarchical layers.

1.1.1.1 MPEG's History

In 1991, the first MPEG standard (MPEG-1) was finalized, which can compact audio and video high quality. The MPEG-1's audio specification, MPEG-1 Audio Layer 3 or popularly referred as MP3, is broadly accepted as a digital audio lossy compression format. Since it supports progressive frames only, MPEG-1 could not be utilized in television broadcasting. The next is MPEG-2 which was finalized in 1994. MPEG-2 supports interlaced field coding and scalability so that it is widely adapted in television video broadcasting. The movie industry accepts MPEG-2 for digital versatile disk (DVD) to distribute high quality movie/video products. To achieve greater compression rate and more flexible scalability, MPEG-4 was finalized in 1998. MPEG-4 can support very

low bit rates (<64kbps), and variable compression rates for different applications. It quickly becomes the main multimedia distribution format in the Internet [2].

1.1.1.2 MPEG-4 Visual Overview

The goal of this thesis is to implement video watermarking insertion, so only the video part or MPEG-4 Visual is researched. The MPEG-4 Visual consists of tools, visual objects, profiles and levels. The tools are coding functions to support specified video features. Objects, which are coded by the tools, are a video's elements, like rectangular frames, arbitrary shapes, still texture, animation models, etc. The profiles are a set of objects a MPEG-4 Codec will process. A brief description of MPEG-4 visual profiles is given as in [2]. Furthermore, a profile contains the levels which define the constraints of a bit stream's parameters. For example, profile advanced simple level 5 has typical resolution 720X576, maximum bit rate 8 Mbps, maximum objects 4 AS or simple [2] pp103. Generally, MPEG-4 refers to MPEG-4 visual profile advanced simple when discussing video exchanging format for Internet multimedia.

1.1.1.3 MPEG-4 Video Objects

The hierarchical structures and terms in MPEG-4 are different from those in MPEG-1 and MPEG-2. However, the MPEG-4 Visual part is extended from MPEG-2, so the different terms in some profiles could be the same as in the early MPEG standards. MPEG-4 manipulates a movie as a collection of video objects, not only the rectangle frames as previous standards. Each video object can be accessed individually, such as searching, browsing, cutting and pasting. For the video objects which exist within a certain span of time and at a particular point of time, an instance of a video object is defined as video object plane (VOP), which is shown in figures (1.1) and (1.2). The

older term “frame” is equivalent to VOP if the objects are rectangular frames. But “frame” could not properly describe the arbitrary-shape video objects in MPEG-4. However, for MPEG-4 visual advanced simple profile, the rectangle video object planes (VOPs) could be treated as frames, and in this thesis, the term “frame” and “rectangular video object planes (VOPs)” are interchangeable.

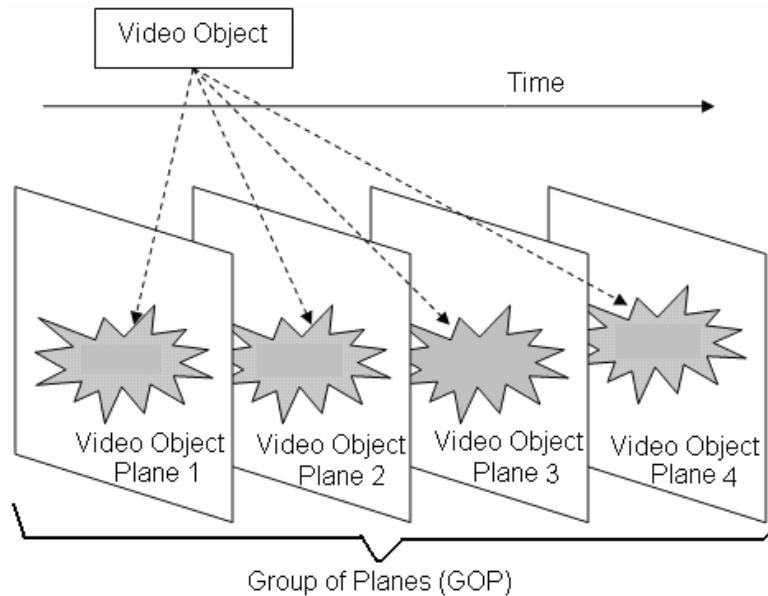
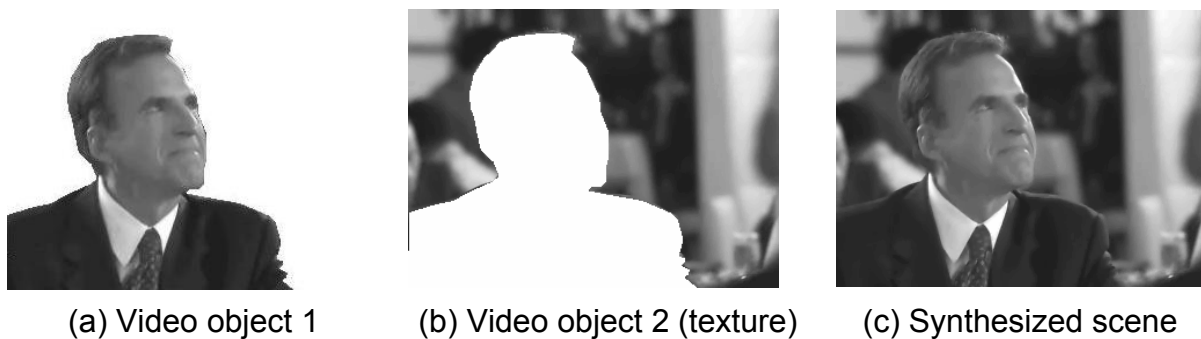


Figure (1.1) Video object and video object planes.



(a) Video object 1 (b) Video object 2 (texture) (c) Synthesized scene
Figure (1.2) Synthesis of video objects into a scene.

1.1.1.4 Intra, Predicted and Bidirectional Predicted VOPs

In early MPEG standards, these VOPs also are called intra frames, predicted and bidirectional predicted frames. In motion estimation and motion compensation, the ever

first base frame to predicate other frames is defined as Intra frame since there will be no temporal compression occurring, but only the compression within the frame itself. The intra frames are referred to as I frames in short. If one frame is reconstructed by predicting from other frames with motion estimation and motion compensation, it is called Intermedial frame or Inter frame in short. Furthermore, Inter frames can have two categories: Predicted or P frame and Bidirectional or B frame. P frames can be predicted from an I frame or another P frame while B frames need two frames, I and P or two P frames, to rebuild the frame. A group of VOPs (video object planes of MPEG-4) or GOPs (group of pictures in MPEG-1/2) will contain one I frame as base, some P frames and B frames interpolated between I frame and P frame or two P frames. For example, a GOP has one I frame, 7 P frames and 7 B frames interlaced between I and P frame or two P frames, for a total of 15 frames. Two such kinds of GOPs will be 30 frames for one second frame sequence of a standard NTSC video. A natural reduction of prediction is a previous frame as the base frame to predict other following. Actually, a later frame can also become a base frame to predicate previous frames. From this, three prediction directions are defined: forward prediction, backward prediction and interpolated prediction. Figure (1.3) demonstrates an example of I, B and P frames, forward, backward and interpolated prediction.

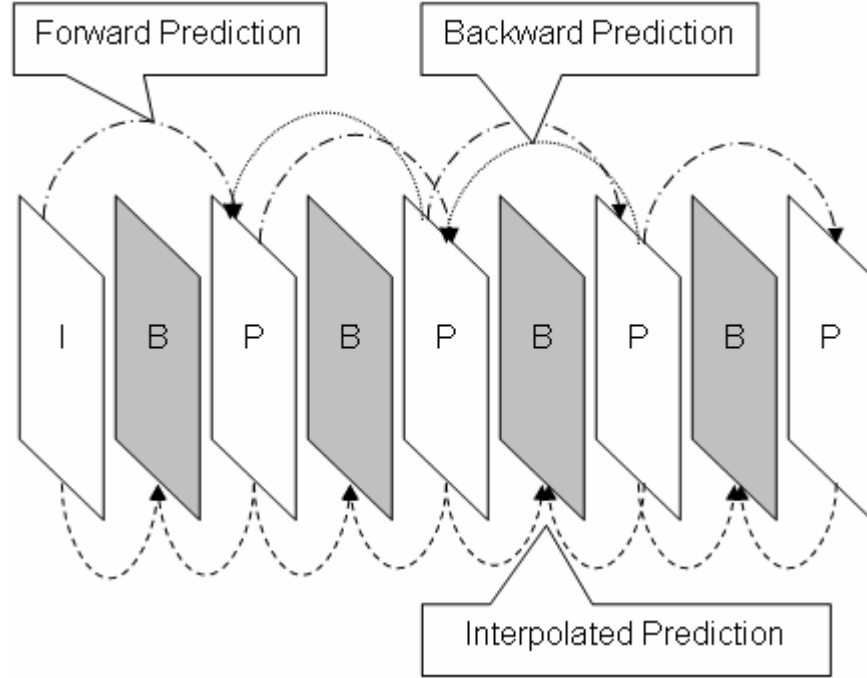


Figure (1.3) I, B and P frames, forward, backward and interpolated predictions.

From Figure (1.3), one can estimate that the coding sequence of a GOP is I-P-B-P-B-P-B..., which is not the video play back sequence I-B-P-B-P-B-P... because the P frame will always be coded before the B frame. Certainly, the B and P order in the decoding end must be re-sorted back: B followed by P. The B frame in Figure (1.3) will be predicted by interpolating a frame from two frames with a bilinear equation as [3]:

Table (1.1) Prediction equations for I, B and P frames.

Frame type	Prediction equation	Prediction error
Intra (I) frame	$\bar{F}_0(\bar{x})$	$F_1(\bar{x}) - \bar{F}_1(\bar{x})$
Forward prediction	$\bar{F}_1(\bar{x}) = \bar{F}_0(\bar{x} + mv_{01})$	
Backward prediction	$\bar{F}_1(\bar{x}) = \bar{F}_2(\bar{x} + mv_{21})$	
Interpolated prediction	$\bar{F}_1(\bar{x}) = \frac{\bar{F}_0(\bar{x} + mv_{01}) + \bar{F}_2(\bar{x} + mv_{21})}{2}$	

In the above table, \bar{F}_0 is Intra I frame; \bar{F}_1 is predicated frame; \bar{F}_2 is P frame; mv_{01} is motion vector related with forward prediction from \bar{F}_0 to \bar{F}_1 ; mv_{21} is motion vector

related with backward prediction from \bar{F}_2 to \bar{F}_1 ; F_1 is original frame, and Prediction Error is for motion compensation. B frames will greatly reduce temporal redundancy among frames and will not propagate errors like a P frame. The quantity of B frames interpolated between two frames could be adaptable according to compression ratio, and picture quality. But if there are too many interpolated B frames, much more time delay could occur [3].

1.1.2 Watermarking

Watermarking is embedding extra data, called a watermark, into a message while, at the receiving end, the embedded data can be detected or extracted by proper methods. Watermarking originally can be traced back to steganography, a technique to hide data into the host message without the knowledge of end users. The name of the watermark is taken from the special processing in paper, money bills and security bonds for security.

The categories of watermarking can be seen from different aspects: according to human perception, it can be visible or invisible; according to its strength, it can be robust or fragile; according to applications, it can be source based or destination based; according to document types, it can be text, image, audio or video; according to the working domain, it can be spatial or in the frequency domain [4]. Visible and invisible watermarking can both be used for copyright protection; additionally, invisible watermarking also is utilized in security applications such as covert communications. Robust watermarking can withstand attacks of the attempt to remove the watermark, unless the attacker is willing to accept downgraded image or video. On the contrary, fragile watermarking will be prone to corruption if any unauthorized attempts to modify

the original document occur. The strength of robust and fragile watermarking comes from digital cryptographic algorithms. A watermark can be embedded into text, images, audio or video as redundant data. The procedure of inserting a watermark into a host is called embedding, and the inverse procedure is called extraction or detection. Depending on embedding algorithms, some watermarks could not be extracted exactly as the embedded one; however, with proper detection algorithms, the existence of a certain watermark can be confirmed. This feature can also be desirable for copyright protection.

There are some schemes to attack video watermarking, described in [5], [6], [7], [8], and [9]. Due to the robust nature of DCT (discrete cosine transform) watermarking, visible watermarking in the DCT domain was chosen in this thesis to accomplish MPEG video copyright protection.

1.1.3 FPGA (Field Programmable Gate Array) Implementation

The implementation of watermarking could be in many platforms such as software, hardware, embedded controller, DSP, etc. For commercial applications like movie production, video recording, on-spot video surveillance, real-time response will be always required, so a software solution is not recommended due to its long time delay. Since the goal of this research is a high performance encoding & watermarking unit in an integrated circuit (IC) for commercial applications, and since FPGAs (field programmable gate arrays) have advantages in both fast processing speed and field programmability, it was determined that an FPGA is the best approach to build a fast prototyping module for verifying design concepts and performance. Two companies

Altera© [10] and Xilinx© [11] are chosen as the suppliers of prototype platforms to implement MPEG and watermark embedding algorithms.

1.2 Problem Statements

The MPEG-4 standard is not freely available so that the full text of the standard could not be obtained from the Internet. Library and published papers can be resources; and open source projects like Xvid™ become another source. Xvid™ only fulfils MPEG-4 visual profile advanced simple so that the module described in this work will not support more advanced algorithms.

First, the MPEG video compression module should be built with a high level architectural design tool like MATLAB/Simulink™ [12] to verify the video compression algorithms and their performance. In this step, the algorithms like discrete cosine transform (DCT), motion estimation/motion compensation, quantization, zigzag scanning and entropy coding will be implemented and tested in a high level language environment. Then a visible watermark algorithm will be embedded into a video. The solution for the problem of watermark drifting can be tested at this stage. After all the algorithms are verified, the working module will be converted into an appropriate intermediate description language for simulation and then programmed into an FPGA device for the prototype.

CHAPTER 2

RELATED WORKS AND LITERATURE

2.1 Video Compression Algorithms

Video compression algorithms in MPEG are color space conversion and sampling rate, discrete cosine transform (DCT) and inverse discrete cosine transform (IDCT), motion estimation and motion compensation, quantization, zigzag scanning and entropy coding.

2.1.1 Color Spaces Conversion and Sampling Rate

Human visual perception or human visual system (HVS) can only perceive a short range of wavelengths of light in the electromagnetic spectrum: from 400nm to 700nm, or commonly called seven-color rainbow: red, orange, yellow, green, blue, indigo and violet. Furthermore, the seven-color rainbow can be constructed from three-original colors. In the video and movie industries, two three-original colors are widely used: RGB and YCbCr. Each three-original color is called a color space.

2.1.1.1 RGB Color Space

RGB color space has three original colors: red, green, and blue. They are the colors human eyes can see, and their combination in different brightness will create all colors in the real world. In digital image or video, each pixel has three elements to represent the three colors in brightness. The three colors have the same importance in a final combination result; therefore, the three color's brightness must be stored in the same resolution.

2.1.1.2 YCbCr Color Space

YCbCr is another color space besides RGB. Y presents the luminance or brightness of a pixel; Cb and Cr are chrominance or color difference. RGB and YCbCr are interchangeable. According to the ITU-R recommendation BT.601, the exchange equations between RGB and YCbCr are [3]:

$$\left. \begin{aligned} R &= Y + 1.402Cr \\ G &= Y - 0.344Cb - 0.714Cr \\ B &= Y + 1.772Cb \end{aligned} \right\}, \quad (2.1)$$

$$\left. \begin{aligned} Y &= 0.299R + 0.587G + 0.114B \\ Cb &= 0.564(B - Y) \\ Cr &= 0.713(R - Y) \end{aligned} \right\}. \quad (2.2)$$

We can estimate that the Cb and Cr in Equation (2.2) could be negative values, but to the actual digital image or video pixels in 8-bits, the data range is 0~255 and only positives are permitted. So, the Equation (2.2) is adjusted as Equation (2.3):

$$\left. \begin{aligned} Y &= 0.299R + 0.587G + 0.114B \\ Cb &= 0.564(B - Y) + 128 \\ Cr &= 0.713(R - Y) + 128 \end{aligned} \right\}. \quad (2.3)$$

The RGB color space and YCbCr color space are interchangeable; however, they are not the same in applications. The RGB color space presents the final image for visual results, YCbCr color space is the intermediate data for image and video processing.

2.1.1.3 Sampling Rate

Because the human visual system is more sensitive to brightness than to difference in color, unlike RGB colors which are stored in the same resolution, Cb and Cr could be in lower resolution than Y although the alteration of the final image quality is still beyond human perception. That observation results in three types of image pixel sampling rates called 4:4:4, 4:2:2 and 4:2:0 sampling rates as shown in [2]. A 4:4:4 sampling rate means each pixel's Y, Cb, Cr will be sampled completely. Consider a 4 pixels group: the samples of them will contain 4 Y samples, 4 Cb samples and 4 Cr samples. That is where the term 4:4:4 sampling rate comes from. The 4:4:4 sampling rate preserves the complete fidelity of the image's luminance and chrominance components, so it is utilized in very high quality and resolution commercial, industrial, or military applications. 4:2:2 sampling rate will sample same the pixel group in 4 Y samples, but 2 Cb and 2 Cr samples. Compared with 4:4:4 sampling rate, the 4:2:2 sampling produces less samples and less data density. Even though some chrominance is ignored, the 4:2:2 sampling rate is still fair enough for high resolution commercial image or video reproduction. The more commonly one, 4:2:0 sampling in a 4 pixels group contains 4 Y samples, 1 Cb and 1 Cr sample. The 4:2:0 sampling rate is widely accepted in television broadcasting, video/movie industry. The MPEG standard adopts 4:2:0 sample rate for VCD, DVD, HDTV, etc.

The 4:2:0 sampling rate will reduce the sampling rate of color space into half that of the 4:4:4 sampling rate by neglecting 75% chrominance samples. However, because of the nature of human visual system, human eyes generally could not distinguish the difference. To different applications and algorithms, Y, Cb, and Cr could be combined individually to accomplish different effects.

2.1.1.4 Macroblock

In the MPEG standard, one macroblock is a 16x16, 8x8 or 4x4 sampling luminance or chrominance block, and it is the basic data element for processing. It is the object of the coding operations such as motion estimation, motion compensation, discrete cosine transform, quantization, and entropy coding. According to different MPEG algorithms, a macroblock's constituents could be of different pixel size. The most commonly accepted one is 16x16 block, called a pel. Regarding 4:2:0 sampling, with 4 Y block samples, 1 Cb and 1 Cr block sampling, if the macroblock is a 16x16 pixel block, it will contain four 8x8 sampling Y blocks, one 8x8 Cb and one 8x8 Cr blocks. One such 4:2:0 macroblock's details are shown in Figure (2.1):

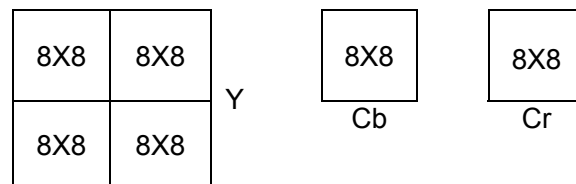


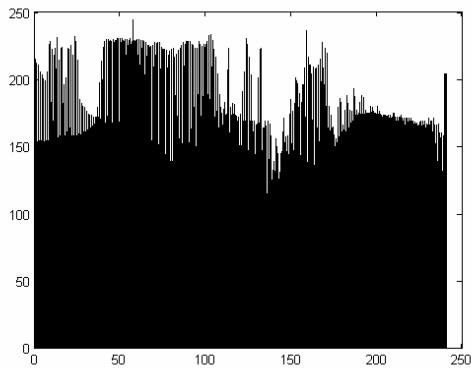
Figure (2.1) 4:2:0 macroblock.

2.1.2 Motion Estimation and Motion Compensation

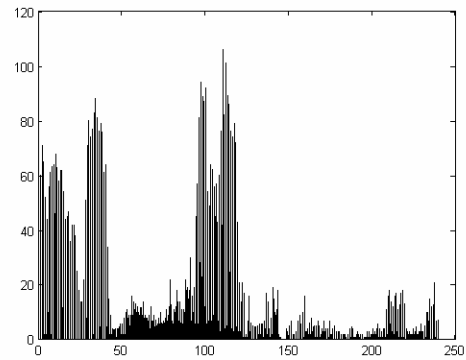
Video compression can be accomplished with color space sampling, DCT high frequency coefficient removing, quantization scaling, entropy lossless coding, and motion estimation with motion compensation in temporal domain. MPEG standard adopts spatial domain block based motion estimation and motion compensation. Actually, motion estimation and motion compensation also work in DCT domain because the position variables in spatial domain are exchangeable with the frequency variables in DCT domain. The goal of this work is an implementation of watermarking in MPEG video stream, so only the spatial domain motion estimation and motion compensation will be introduced.

2.1.2.1 Motion Estimation

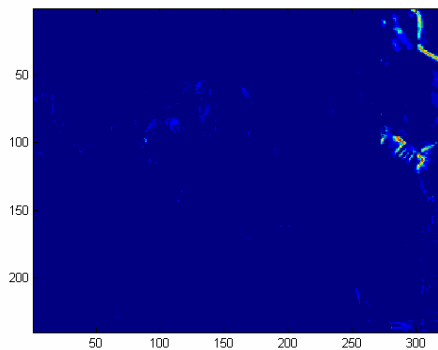
A video playing is actually a serial of frames that are been playing continuously. However, the human eye will view those continuous images as a movie because of human visual perception's persistence of vision. For example, to play a smooth and flicker-free movie, the television broadcasting standard NTSC requires a movie playing at a rate of 29.97 frames per second. If all those frames are transmitted without any compression, the communication bit rate will be very high and overflow most present communication carrier's bandwidth. For example, an uncompressed HDTV of resolution 1920X1080 in 30 frames per second will demand a bandwidth of 1.39 Gbps [2]. Figure (2.2) indicates the significant redundancy in two adjacent frames.



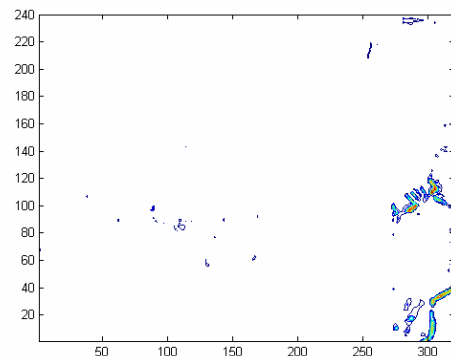
(a) Amplitude of a frame.



(b) Amplitude difference of two frames.



(c) Brightness difference of two frames.



(d) Brightness difference in contour.

Figure (2.2) The redundancy among movie frames.

The Figures of (2.2), (b) (c) and (d) indicates that the difference of two adjacent frames in a movie could be trivial. To remove the redundancy among frames, a based frame and the difference between two frames will be transmitted rather than two whole frames. That is the general concept of MPEG temporal compression model; however, MPEG compresses the frame differences further by motion estimation.

Before running motion estimation, an image is required to be split into smaller pixel groups called macroblocks as the element of the image rather than a single pixel for the compromise between efficiency and performance to analyze a video's temporal model. A macroblock commonly has the size of 16X16, 8X8 or 4X4 pixel. For two frames following each other in time, we can consider the difference between two frames is the macroblock position changing within a certain area of the frame. The variable to describe the position change is called motion vector. With the macroblock in the base frame and its two dimensional motion vector, the current frame can be predicted from the previous frame. The region in which the macroblock is sought for match could be a square, diamond, or arbitrary shape as in MPEG-4. For most applications, a square region is considered. For example, if the macroblock is 16X16 pixel size, the searching region will be 48X48 pixels block (some algorithms also use a diamond shaped region rather than a square one). The criterion of match for two blocks is the minimized difference between two blocks. For computation simplification, we apply sum of absolute difference (SAD) as the criterion for matching. Its equation is [3]:

$$SAD_N(x, y) = \begin{cases} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |c(i, j) - p(i, j)| - C & \text{for } (x, y) = (0, 0) \\ \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |c(i, j) - p(i + x, j + y)| & \text{otherwise} \end{cases} \quad (2.4)$$

Where, $c(i,j)$ are the pixels of current block, and $i, j = 0, 1, \dots, N-1$; $p(m,n)$ are the pixels of previous block for searching region, and $m, n = -R, -R+1, \dots, 0, 1, \dots, R+N-1$. From Equation (2.4), we can speculate that this SAD algorithm will search exhaust every where of the region in the step of one pixel, and the block with minimum SAD result will be taken as a match. To those applications whose searching speed is critical, some fast searching algorithms can be adapted, such as three-step search algorithm (TSS), four-step search algorithm (FSS), and cross search algorithm (CRS) [3]. In our design, exhaust SAD is used.

One problem will arise if the match block is located at the boundary across two adjacent regions; the match block could then not be found. The solution is to arrange two adjacent regions overlapping with each other with a width of macroblock minus 1. For example, for a 16X16 macroblock searching in a 48x48 region, the next region will have a 15X47 or 47X15 overlap with each other. If the match block is within the overlap and could not find the match, it can be matched in next region. This is shown in Figure (2.3).

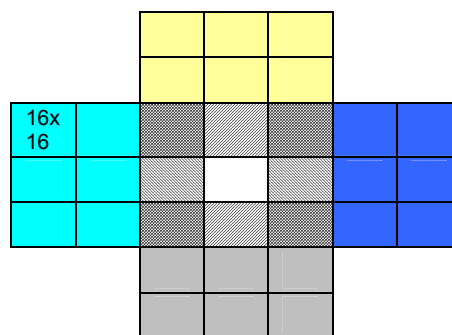


Figure (2.3) Searching regions overlap.

From this Figure (2.3), we observe that the region for 3X3 in macroblock sizes, only the center one, or the block's present position, does not overlap with other regions. That

guarantees that there could be no neglect in searching at the boundary area. One consideration of searching region block could be if the current block is at the image's boundary, what the searching region in the previous frame will be? Beside the above 3X3 region in macroblock size, there are another two cases needed to investigate: at a corner or at a boundary. For the corner macroblock, the search region in previous frame will be a 2X2 in macroblock size; for the boundary macroblock, it will be a 2X3 or 3X2 in macroblocks size. They are clearly shown in Figure (2.4):

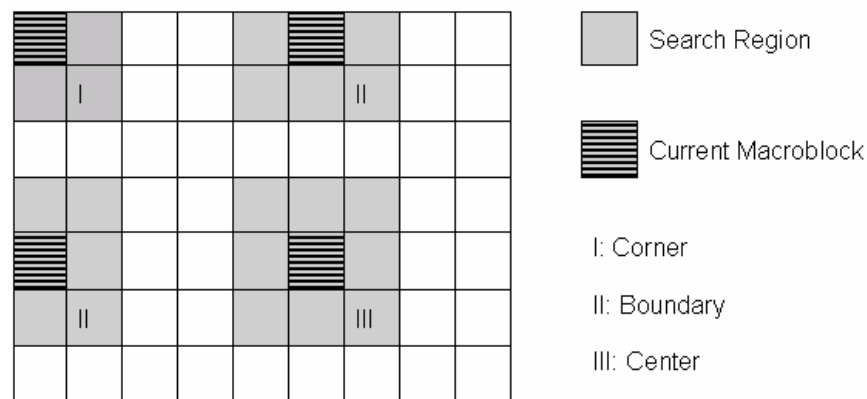


Figure (2.4) Three searching regions for motion estimate.

To make all searching region same size, four extra strips in 15X47 or 47X15 are added along image four boundaries. Another purpose of adding strips to solve the problem of matching a macroblock moving out of or into the boundary of a frame. In that case, the search could not have a match such that a disorder at the boundary will occur in the reconstructed frame. A simple solution is adding extra blank strips in width of 15 pixels along the outside of the image boundary. That ensures, even in the worst condition, at least 15 pixels will match in current macroblock and searching region. In the MPEG video compression, we only run motion estimation for Y macroblock. The Cb and Cr will directly use Y's motion vector. To the 4:2:0 sampling block, the motion vectors for Cb and Cr are half of Y's:

$$MV_{Cb} = MV_{Cr} = \frac{MV_Y}{2}. \quad (2.5)$$

For fine image quality and high resolution applications, some advanced motion estimation methods are introduced, such as four motion vectors per macroblock and sub-pixel motion estimate.

2.1.2.1.1 Four Motion Vectors per Macroblock

In 4:2:0 sampling rate video, the traditional motion estimation only generates one Motion Vector for the whole 16X16 pixel Y macroblock. From Equation (2.5), the motion vectors for the 8X8 Chrominance blocks can be calculated. An improved way for finer match is that the 16X16 pixel Y macroblock will be split further into four 8X8 blocks, and for each 8X8 block the motion estimation will be calculated individually to search for their own motion vectors [3] p92.

2.1.2.1.2 Sub-pixel Motion Estimate

The traditional motion estimation is also called integer-pixel motion estimation because it only produces the motion vectors in integer displacement values. But the real world is analog, and the image elements will be continuous, so the displacements of each pixel could not be necessarily integer values. In the case of presenting such images with integer displacement, an error could not be avoided. To accomplish further better match resolution, a motion estimation can be run on an interpolated bilinear frame, and the resulting motion vectors could be fractional displacements rather than integers. For example, if running half-pixel motion estimates, a motion vector could be (4.5, -1.5); if running quarter-pixel motion estimates, a motion vector could become (4.25, -1.75). To obtain sub-pixel motion estimates, the bilinear frames need to be

interpolated. For half-pixel motion estimate, it can be imagined that the original search region is zoomed in two times, and the blank pixel between two adjacent pixels will be interpolated two extra pixels according to bilinear algorithm. Similarly, the quarter-pixel motion estimate will zoom in the region in four times and four extra pixels will be inserted between two original pixels with bilinear algorithm computing results. The concept of sub-pixel motion estimate and interpolation is indicated in [3] p58. The bilinear equation of sub-pixel interpolation is:

$$\left\{ \begin{array}{l} A_1 = A_2 = A_4, \\ B_2 = \frac{A_1 + B_1}{2}, \quad C_2 = \frac{A_1 + C_1}{2}, \quad D_2 = \frac{A_1 + B_1 + C_1 + D_1}{4}, \\ B_4 = \frac{A_1 + B_1}{4}, \quad C_2 = \frac{A_1 + C_1}{4}, \quad D_2 = \frac{A_1 + B_1 + C_1 + D_1}{8}. \end{array} \right. \quad (2.6)$$

Understandably, if the sub-pixel motion estimate scheme is applied, both the compression and decompression ends need to interpolate extra frames. That will consume more computation resources and more time.

The sub-pixel motion estimation search approach could be implemented in two different ways. One is searching after inserting extra bilinear pixels; another is firstly running integer-pixel search, then interpolating bilinear pixels, later searching match. In the latter approach, so-called from gross to fine, one can avoid searching in bigger region so that it can reduce computation complexity and run much faster. For better resolution and smoother image, $1/8^{\text{th}}$, $1/16^{\text{th}}$ or further pixel motion estimation also could be considered. Actually, for balance of fine quality and fast processing time, most video compression standards accept half- and quart-pixel motion estimation. Table (2.1) gives the motion estimates commonly implemented in MPEG standard.

Table (2.1) Sub-pixel motion estimate in MPEGs.

<i>Standard</i>	<i>Integer ME</i>	<i>Half ME</i>	<i>Quarter ME</i>
MPEG-1	Yes	Yes	No
MPEG-2	Yes	Yes	Yes
MPEG-4	Yes	Yes	Yes

Motion estimation will reduce redundancy among frames in the temporal domain significantly. With motion estimation, only the base frame called Intra frame and the motion vectors are needed to be transmitted to predict the next frame. However, motion estimation will propagate and accumulate the errors created by prediction from the Intra frame and motion vectors. To compensate for the accumulated errors, motion compensation is introduced.

2.1.2.2 Block Based Motion Compensation

Even the best match in motion estimation could not guarantee two macroblocks are exactly same. The small difference between the prediction by motion estimation and the original image will keep on accumulating until the whole video image is smashed. A smart method called motion compensation is introduced [3] p63. As the block based motion estimation works in spatial domain, the block based motion compensation also corrects the prediction error in spatial domain. The motion compensation procedure can be describes as that, first the predicted frame will be built from the base frame and motion vectors from the motion estimate. Then the original frame related with predicted frame will be subtracted with the predicted frame, and the resulting difference is called residual frame. The macroblocks in residual frame are called prediction errors. The residual frame can compensate the error from the motion estimate. The motion compensation equation is defined as:

$$d(i, j) = c(i, j) - p(i + MV_x + MV_y), \quad i, j = 0, 1, \dots, N-1 \quad (2.7)$$

Where $d(i,j)$ is motion compensation, $c(i,j)$ is current frame to be predicated, p is predicated result, MV_x, MV_y are two dimensional motion vectors. Therefore the whole video compression in temporal model has only three elements: base frame, motion vector to predict the next frame produced by motion estimation, and the residual frame for motion compensation by subtracting between the original frame and predicted frame. They will be coded and transmitted to the receiver end. The decoding receiver will rebuild the movie with Equation (2.8) [3]:

$$\hat{c}(i,j) = \hat{d}(i,j) + p(i + MV_x + MV_y), \quad i, j = 0, 1, \dots, N-1 \quad (2.8)$$

2.1.3 Discrete Cosine Transform (DCT)

Discrete cosine transform is a mathematical tool to process signals like images or video. It will transform the signal variables from the spatial domain to the frequency domain or with its inverse transform, from the frequency domain back to the spatial domain without quality loss. The discrete cosine transform is the real part of the Fourier transform, and it can be quickly computed with hardware or software. For real-time video compression and watermarking processing, a fast discrete cosine transform will be implemented.

2.1.3.1 Fourier Transform

Before discussing the discrete cosine transform, the Fourier transform will be briefly introduced because the discrete cosine transform is derived from the Fourier transform. The Fourier theorem states that any signal can be constructed by summing a series of sines and cosines in increasing frequency. It is written as [13]:

$$F(u) = \int_{-\infty}^{+\infty} f(x)(\cos(2\pi ux) - i \sin(2\pi ux))dx \quad (2.9)$$

Here, $f(x)$ is the signal with time variable x , and $F(u)$ is the transformed result with frequency variable u . A very important feature of the Fourier transform is that an inverse function (Equation (2.10)) can transform the frequency domain expression back to the time domain expression [13].

$$f(x) = \int_{-\infty}^{+\infty} F(u)(\cos(2\pi ux) - i \sin(2\pi ux))du \quad (2.10)$$

Besides transforming the time domain back-and-forth to the frequency domain, the Fourier transform also can work on spatial domain from-and-to frequency domain. To process discrete signals like digitized images, sound, etc, which are discrete rather than continuous, the discrete Fourier transform and its reverse expressions are deduced as Equations (2.11) and (2.12) [13].

$$F(u) = \frac{1}{N} \sum_0^{N-1} f(x)(\cos(\frac{2\pi ux}{N}) - i \sin(\frac{2\pi ux}{N})) \quad (2.11)$$

$$f(x) = \sum_0^{N-1} F(u)(\cos(\frac{2\pi ux}{N}) - i \sin(\frac{2\pi ux}{N})) \quad (2.12)$$

Here $F(u)$ is discrete Fourier transform coefficient, $f(x)$ is input raw data, and N is the discrete frequency component for constructing the discrete Fourier transform.

2.1.3.2 Discrete Cosine Transform (DCT)

If an image is treated as a function of amplitude with the distance as variable, according to the Fourier theorem, that function can be built up with a series of cosines and sines in increasing frequency. When the function is with sine parts only, it is called Sine transform, and with cosine parts only, it is called cosine transform. All the Fourier transforms, the sine transform and the cosine transform have their specialized applications in image processing. However, in MPEG video compression and

watermarking, the cosine transform is the mostly commonly used one. To understand it, consider two signals, even and odd as in Figure (2.5):

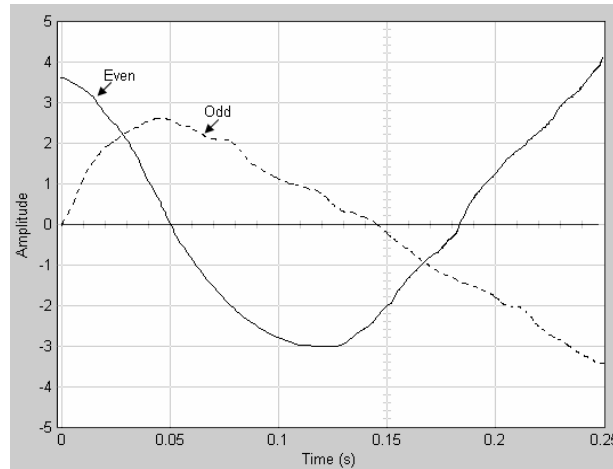


Figure (2.5) Even and odd signals.

The even signal has non-zero amplitude at time 0 or frequency 0 while the odd signal has zero amplitude. To construct the even or odd signal, either the cosine or sine transform function can be chosen, however, with cosine transform, the result for even signal requires less frequency range while with sine transform, the result for odd signal will have less frequency range. This can be indicated by Figure (2.6).

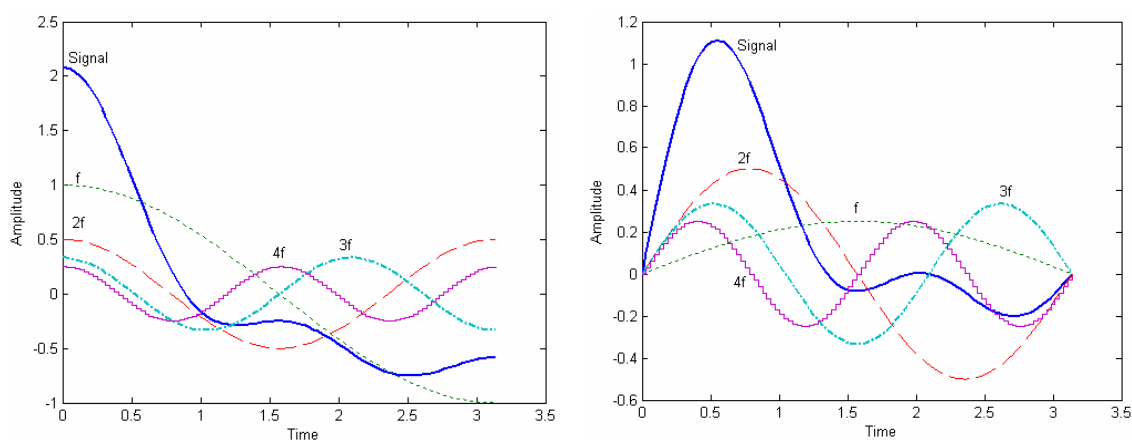


Figure (2.6) Constructing signals with cosine and sine transforms.

An image can be considered as an even signal because its average brightness or the brightness at frequency 0, generally, is of non-zero amplitude. Reasonably, building the image with Cosine transform could require less frequency parts than with the Sine one. A digital image, unlike one in a continuous mode in the real world, is in a discrete mode with the pixels as elements. Technically, the discrete cosine transform (DCT) is applied especially in the digital image processing. The reasons for applying discrete Cosine transform in digital image processing are, first, it can remove the correlation among image pixels in the spatial domain. Secondly, discrete cosine transform requires less computation complexity and resources. The one dimensional discrete cosine transform Equation (2.13) and its inverse transform Equation (2.14) are given by [14].

$$C(u) = \alpha(u) \sum_{x=0}^{N-1} f(x) \cos\left(\frac{\pi(2x+1)u}{2N}\right), \quad (2.13)$$

$$f(x) = \sum_{u=0}^{N-1} \alpha(u) C(u) \cos\left(\frac{\pi(2x+1)u}{2N}\right). \quad (2.14)$$

Here, $C(u)$ is discrete cosine transform coefficient, $f(x)$ is signal variable, N is element numbers, $u=0,1,2,\dots, N-1$.

For both Equations (2.13) and (2.14)

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & u = 0 \\ \sqrt{\frac{2}{N}} & u \neq 0 \end{cases} \quad for \quad (2.15)$$

The above one-dimensional discrete cosine transform algorithm will consume too much computation for a real time system. If computing an 8 element transform, it needs 56 adders and 72 multipliers. So, some fast algorithms are presented. Chen introduced a fast DCT algorithm in [15], and Leoffler presented an improved fast one dimensional DCT algorithm in [16]. Leoffler's fast algorithm of 8 elements DCT and inverse DCT [17]

was selected for this work. Because of the symmetrical feature of the Cosine Transform, the inverse discrete cosine transform can be directly obtained by reversing the direction of the discrete cosine transform.

The above one dimensional discrete cosine transform can only process one dimensional input data, however, images are two dimensional matrixes. Therefore, a two dimensional discrete cosine transform Equation (2.16) and its inverse transform Equation (2.17) are used for image processing [14].

$$C(u, v) = \alpha(u)\alpha(v) \sum_{y=0}^{N-1} \sum_{x=0}^{N-1} f(x, y) \cos \frac{(2x+1)u\pi}{2N} \cos \frac{(2y+1)v\pi}{2N}, \quad (2.16)$$

$$f(x, y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \alpha(u)\alpha(v) C(u, v) \cos \frac{(2x+1)u\pi}{2N} \cos \frac{(2y+1)v\pi}{2N}. \quad (2.17)$$

Here, $C(u, v)$ is the discrete cosine transform coefficient, $\alpha(u)$ and $\alpha(v)$ have been defined in (2.15), $f(x, y)$ is the input two dimensional matrix element, and N is input matrix row or column number.

For an 8x8 matrix with 8-bits for each coefficient, which is widely adapted as a unit data block in image processing, the data range of that discrete cosine transform coefficients can be estimated from Equation (2.16). Considering the worst condition, the value of one coefficient could be:

$$C_{\max}(u, v) = \alpha_{\max}(u)\alpha_{\max}(v) \sum_{y=0}^7 \sum_{x=0}^7 f_{\max}(x, y) = \frac{255 * 64}{8} = 2040, \quad (2.18)$$

$$C_{\min}(u, v) = -C_{\max}(u, v) = -2040.$$

From equations (2.16) and (2.17), we can estimate that the two-dimensional discrete cosine transform structure will still be complicated for hardware or software implementation in terms of resources. However, because the discrete cosine transform

is an orthogonal transform, the two-dimensional discrete cosine transform can be simply calculated by running the one-dimensional discrete cosine transform in rows and then the results are transformed again in columns as demonstrated in Figure (2.7) [14].

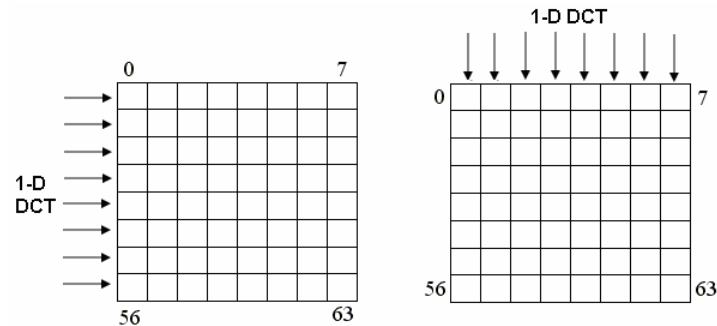


Figure (2.7). Calculating an 8x8 2-D DCT with 1x8 1-D DCT.

In the same manner, a two-dimensional inverse discrete cosine transform matrix can be obtained by executing the one-dimensional inverse discrete cosine transform two times.

The spatial correlation in an image cannot be compacted in the spatial domain because every pixel in the image is correlated with each other, and human visual perception can easily detect the position displacement at spatial domain. To remove the correlation among the pixels, discrete cosine transform can change the tightly correlated position variables at spatial domain into different discrete frequencies at frequency domain.

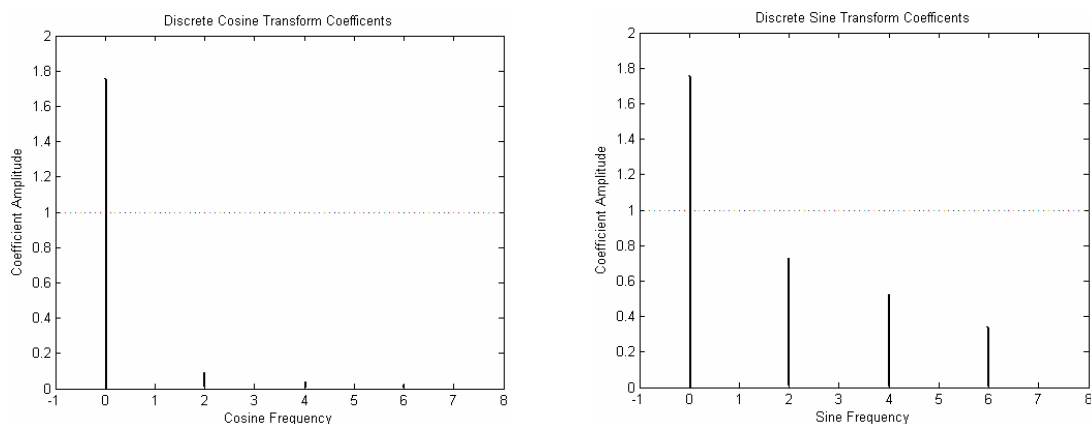


Figure (2.8): DCT and DST frequency domain coefficients.

From this figure, we can see that, for the same input signal, the coefficients generated by the discrete cosine transform will cluster in lower frequencies and their amplitudes decrease sharply while those by the discrete sine transform will spread among different frequencies and the amplitude change is not as sharp as the discrete cosine transform's. The meaning of each coefficient of the discrete cosine transform is: the first coefficient of the DCT is the DC part, and can be interpreted as the average value of the pixel matrix while all other remaining coefficients are the AC part. For example, to an 8x8 element pixel matrix, the DC coefficient is:

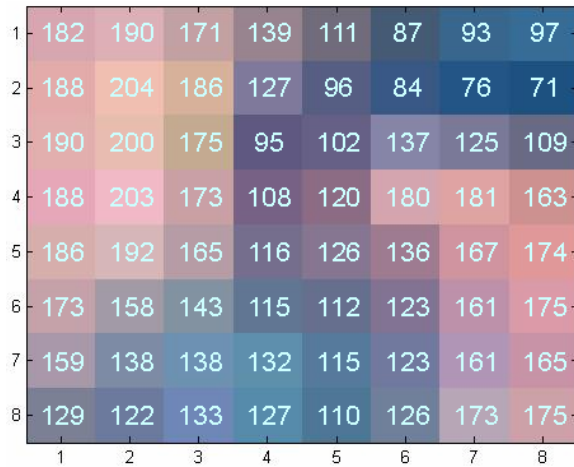
$$C(0,0) = \frac{1}{\sqrt{8}} \sum_{y=0}^7 \sum_{x=0}^7 f(x,y). \quad (2.19)$$

It is the mean of all pixel values of the 8x8 matrix in spatial domain. That DC coefficient indicates the average brightness of the matrix. Table (2.2) shows the locations of DC and AC coefficients in an 8x8 Discrete Cosine transform coefficient matrix.

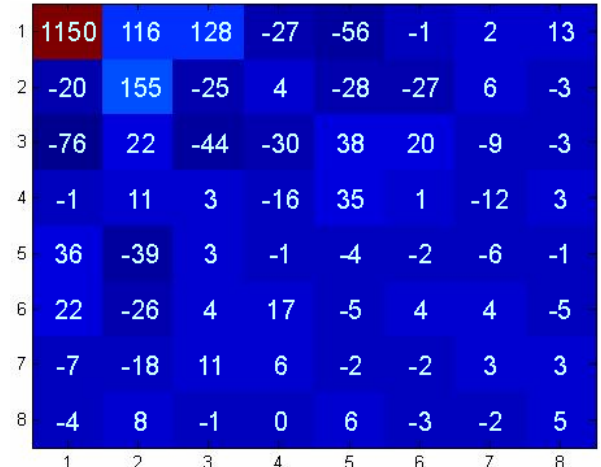
Table (2.2) DC and AC coefficients.

DC	AC	AC	AC	AC	AC	AC	AC
AC	AC	AC	AC	AC	AC	AC	AC
AC	AC	AC	AC	AC	AC	AC	AC
AC	AC	AC	AC	AC	AC	AC	AC
AC	AC	AC	AC	AC	AC	AC	AC
AC	AC	AC	AC	AC	AC	AC	AC
AC	AC	AC	AC	AC	AC	AC	AC
AC	AC	AC	AC	AC	AC	AC	AC

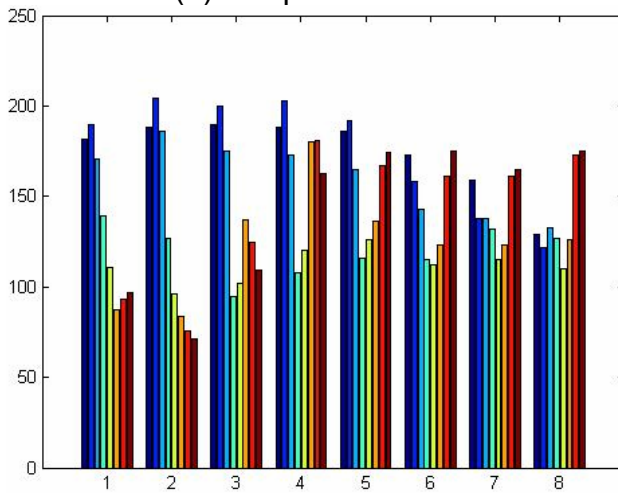
Furthermore, Figure (2.9) displays an 8x8 original pixel matrix and its discrete cosine transform coefficient matrix. The bar graphs of the coefficients clearly demonstrate that the energy at the frequency domain clusters at DC and lower frequencies.



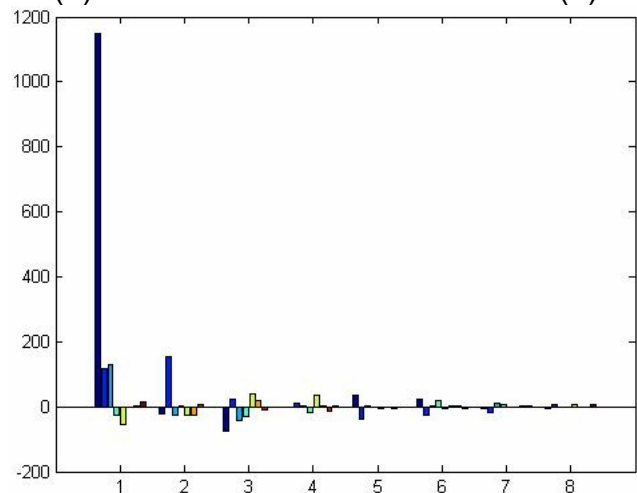
(a) 8x8 pixel matrix.



(b) 8x8 DCT coefficient matrix from (a).



(c) Amplitude bars in spatial domain.



(d) coefficient bars in frequency domain.

Figure (2.9) Comparing the matrixes before and after DCT.

To human visual perception, the high frequency parts in the frequency domain are not so sensitive as the lower ones. Even though we have difficulty in moderating an image or video's pixels in spatial domain without obviously disturbing its quality, in the frequency domain after discrete cosine transform, we can easily manipulate data at high frequency parts without degrading the image or video quality under human visual perception [14]. This feature is essentially useful for image compression and watermarking.

2.1.4 Quantization of DCT Coefficients

After discrete cosine transform, the correlation of pixels of an image in spatial domain already have been de-correlated into different discrete frequencies in frequency domain. Since human visual perception is more acute to the DC coefficient and low frequencies, a carefully designed scalar quantization approach can reduce data redundancy while keeping the fidelity of image.

2.1.4.1 Scalar Quantization

For most analog to digital conversion schemes, linear quantization is a simple and fast solution. Its drawbacks are:

- The resolution or dynamic range is the same for the entire data range, i.e., the greater digits have less derivative error while less digits have more error relatively.
- The bit rates for different values are same.
- Flickering noise exists around zero.

To solve these problems, some quantization schemes are introduced, such as A-law approach in telecommunications which adopts non-uniform quantization. But in the MPEG video compression standard, a uniform module called scalar quantization is adopted. The feature of the scalar quantization scheme is adaptive quantized step size according to discrete cosine transform coefficients of each macroblock [3].

The nonlinear scalar quantization introduces wider dead zone and greater step size at small input to achieve the following effects for MPEG video compression [3]:

- DC and low frequency coefficients will have fine step size while high frequencies will have more aggressive quantization step size.

- Wider dead zone around origin will block the small signal noise.
- The scalar quantization will generate more zeros which will benefit entropy coding.

For computation and hardware simplification, the scalar quantization step size can be chosen from pre-define tables as in [3]:

For the MPEG encoder and decoder, the quantization tables used are the default and being kept by both encoder and decoder. Intra frame will apply one quantization to determine the quantization step size while non-intra frame will use another table. By simple observation, we can find two tables approximately matches the 8x8 discrete cosine transform coefficient result, i.e., the DC and low frequency parts could have much finer step size, and high frequency parts will have grosser step size. To the non-intra frames or predicted and bi-directional frames, the residual results will have less bit rate and fixed quantization step size. Similarly, different quantization formulas are introduced as follows [3]. For intra frames:

$$Q(i, j) = RND\left(\frac{16 \times C(i, j) + \text{sign}(C(i, j)) \times qs \times qt(i, j)}{2 \times qs \times qt(i, j)}\right), \quad (2.20)$$

$$Q^{-1}(i, j) = \frac{2 \times qs \times qt(i, j) \times (Q(i, j) + 0.5 \times \text{sign}(Q(i, j)))}{16}, \quad (2.21)$$

$$\text{sign}(x) = \begin{cases} +1, & \text{for } x > 0, \\ 0, & \text{for } x = 0, \\ -1, & \text{for } x < 0. \end{cases} \quad (2.22)$$

Here, $Q(i, j)$: the quantization result, and $Q^{-1}(i, j)$: the inverse quantization results. RND : round function, $C(i, j)$: Discrete cosine transform coefficients, qs : quantization scale factor, the value is from 1 to 40. Smaller value will generate finer step size. $qt(i, j)$: the quantization table as Table (2.5) (a).

For the non-intra frames:

$$Q(i, j) = RND\left(\frac{16 \times C(i, j)}{2 \times qs \times qt(i, j)}\right), \quad (2.24)$$

$$Q^{-1}(i, j) = \frac{2 \times qs \times qt(i, j) \times Q(i, j)}{16}. \quad (2.25)$$

Here, $Q(i, j)$, $Q^{-1}(i, j)$, RND , $C(i, j)$, qs , $qt(i, j)$ are defined the same as the intra frame quantization equations above. $qt(i, j)$ here should be from Table (2.5) (b).

From the above scalar quantization and inverse quantization equations, we can conclude that the quantization algorithm will discard some details of discrete cosine transform coefficients to reduce bit rate, so it is not lossless computation. However, the effects of loss will trade off the benefits like extra “0” for the future compression in the entropy coding, better resolution for DC and low frequency coefficients, blocking small signal noise, and adaptability according to different resolutions for different applications.

2.1.5 Zigzag Scanning of DCT Coefficients

After the two dimensional Discrete Cosine transform, an 8x8 image pixel block will be transformed from the spatial domain to the frequency domain. As we have already seen, the coefficient matrix presents the coefficients of frequencies, and the low frequency parts will cluster in the upper left corner of the discrete cosine transform coefficient matrix. Zigzag scanning will rearrange the order of the matrix so that the coefficients are sorted by frequency in an ascendant order in a linear pattern.

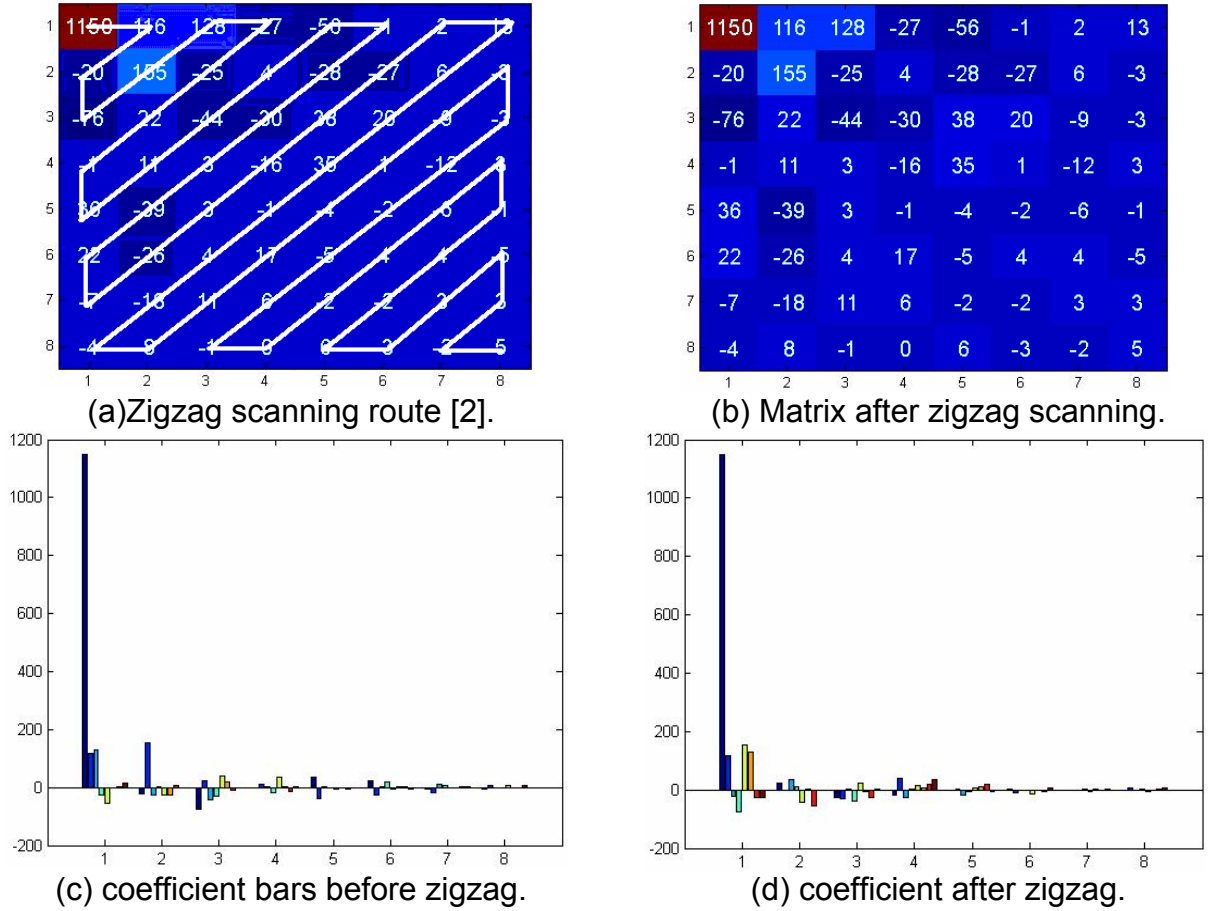


Figure (2.10) Zigzag scanning of DCT coefficient matrix.

2.1.5.1 Discrete Cosine Transform Coefficients Matrix

For ease of programming, the discrete cosine transform must be rearranged in a linear order with ascending frequencies. The processing is called zigzag scanning because the route to choose the coefficients in the matrix goes in a zigzag way as shown in Figure (2.17) (a).

From parts (c) and (d) of Figure (2.17), we observe that, after zigzag scanning, the coefficients are approximately sorted by their amplitudes. The purpose of zigzag scanning is to aggregate “0”s and other small values as much as possible to decrease bit rate. For the discrete cosine transform matrix from progressive video frames, the zigzag scanning route can go the zigzag way as in [3]. However, in MPEG2 and

MPEG4, both progressive and interlaced frames need to be supported; the blocks from interlaced frames or fields should be different from the progressive ones to achieve better aggregation of “0”s and smaller values. The alternative zigzag scanning route is described in [3] as well.

2.1.5.2 Compression at DCT/Frequency Domain

Three features make the compression work in the DCT domain rather than the spatial domain:

- 1) The discrete cosine transform will de-correlate the position in the spatial domain, and focus the energy into the DC coefficient while the AC coefficients will have small amplitudes in energy.
- 2) After the discrete cosine transform, the DC and AC coefficients of low frequencies will cluster at the upper left corner of the matrix and the AC coefficient of high frequencies will aggregate at the lower right corner. And,
- 3) Human visual perception is sensitive to DC and AC at low frequencies but not AC of high frequency energy, which relates to the fine details. There is no criterion on how many ACs will be chosen for a given quality of the reconstructed image. Reasonably, the better compression bit rate to accomplish, the worse the quality of reconstructed image. The trade off between compression and quality will compromise each other [14]. A group of compressed images in different compression rates by neglecting some AC coefficients are displayed in Figure (2.11):



(a) Rebuild with 1 of 64 coefficients.



(b) Rebuild with 4 of 64 coefficients.



(c) Rebuild with 16 of 64 coefficients.



(d) Uncompressed original image.

Figure (2.11) Compress image in DCT domain.

The method to process the original uncompressed image (d) is: the image is split into 8x8 pixel blocks. Then each 8x8 block will be transformed by DCT to create an 8x8 DCT coefficient matrix. After that, according to different compression rates, keep DC and some low frequency ACs. The remaining ACs will be replaced with "0". Finally, run the inverse DCT to rebuild a new image resembling the original one. By observing the images in Figure (2.12), we can conclude that even with 16 out of 64 DCT coefficients, the reconstructed image's quality is fairly acceptable for most video playback applications, but the compression rate achieved is 4:1.

2.1.6 Entropy Coding

After DCT and quantization compression, more compression still can be achieved. The code domain compression algorithms generally are called entropy coding, which includes Huffman coding, Arithmetic coding, etc. Unlike lossy compressions as in the color space, DCT and quantization procedures, the entropy coding compression is lossless. The basic idea of entropy coding is that the more frequently occurring symbols will be coded with short bits while uncommon ones with longer code bits such that the over all bit rate of the stream will be reduced.

2.1.6.1 Variable Length Coding (VLC)

From a general view, unlike fixed length coding such as ASCII, BCD, etc, entropy coding is a kind of variable length coding, which means that the code bits for different symbols are different. After the procedure of quantizing and truncating higher frequencies, most coefficient values of the DCT matrix will be zeros. With a proper variable length coding, most '0' coefficients are compacted. The variable length coding in MPEG standards is also called run level code: "run" refers to how many '0' precede a non-zero number; "level" presents the number's level or value. For example, a series of coefficients is: 7, 0, 5, 0, 0, 0, 4, 3, 0, -3..., they can be re-written into run level codes as: (0,7), (1,5), (3,4), (0,3), (1,-3).... The (R, L) pair like (1, 5) can be interpreted as one run '0' is preceding level '5'. Such run level coding will greatly compact bit length for a stream with many continuous '0' so that it will benefit DCT coefficient encoding.

2.1.6.2 Huffman Coding

To accomplish further compression in code domain after run level coding, entropy coding like Huffman coding is needed. Huffman originally contributed the idea of

building an optimizing symbol binary coding in his paper in 1952 [18]. According to a fundamental theorem of Shannon [19], the minimized binary code length of a symbol is:

$$L_s = \log_2 \left(\frac{1}{P_s} \right). \quad (2.26)$$

Where, L_s is the optimized length of a symbol; P_s is the probability of the symbol's occurrence in a message stream, and the entropy, the average number of bits for a total symbol is as:

$$Entropy = \sum_s P_s \log_2 \left(\frac{1}{P_s} \right). \quad (2.27)$$

Where, *Entropy* is lowest limit of the total average code length we can achieve in a compressed message. From the above equations, the optimized binary code lengths for the symbol probability of the power of 2 are easily obtained, but in most cases, the occurrence probability could not be always a power of 2; for example, 1/23, the code length generated by accepting approximate probability will not be the minimized one. Huffman's idea is using a Huffman tree to find out a unique code prefix. For example, a serial of symbols and their probabilities are as Table (2.3):

Table (2.3). Huffman code example.

<i>Symbol</i>	<i>Probability</i>	<i>Code</i>	<i>Bits(actual)</i>	<i>Bits(ideal)</i>
A	0.8	1	1	0.3219
B	0.02	0000	4	5.6439
C	0.03	0001	4	5.0589
D	0.07	001	3	3.8365
E	0.08	01	2	3.6439

The Huffman tree for searching code of the symbols in Table (2.3) is:

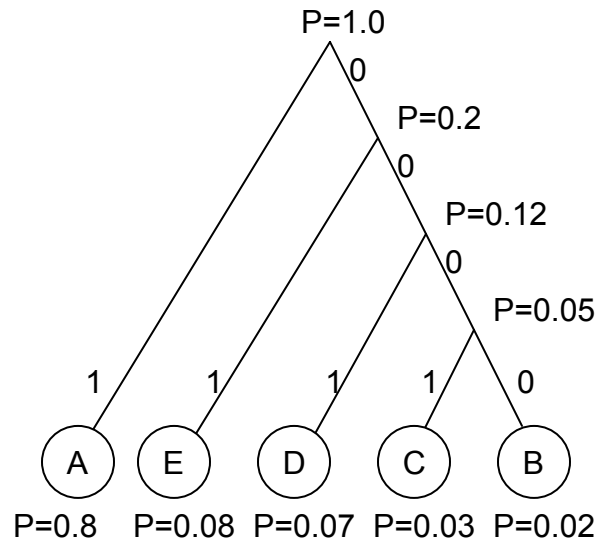


Figure (2.12) Huffman tree for Huffman coding.

Where, the tree's branches are arranged by their probabilities. The node's probability is the sum of two branches, and the two branches will be in code either '1' or '0'. If one branch is leaf, it is '1'; otherwise it is '0'; If both are leaves, the higher probability one is '1', and the lower one is '0'. By searching the tree from root to leaf, Huffman encoding can be achieved. Similarly, with the same Huffman tree, the Huffman codes are decoded to the corresponded symbols. However, for the compression purpose, to transmit all symbols' probabilities is not practical. In practice, a pre-calculated code table for generic image data is applied by both encoder and decoder. Besides no extra bits for probabilities table, Huffman tree searching is avoided so that improved processing speed results. A pre-calculated Huffman code table is [3], [14]. One observation from the above table is that it does not include all combinations of whole run-levels. It can be estimated that most run-level code combinations are in above table. For those are not in the table, they can be coded as: 6 bits ESCAPE, followed by 6 bits

run code, and then 8 or 16 bit level code [14]. For example, a run-level pair (45,113), its code is: 000001 101101 11100010.

With all above modules, a MPEG video compression module called hybrid DPCM/DCT is built as in [2].

2.2 Video Watermarking

The general idea of watermarking is embedding some extra data into a host message. The embedded information is a watermark, and the host message is a carrier. From the view of spread spectrum communication, the watermark is a message needed to be sent, and the carrier is a communication channel with noise. At a transmitting end, the embedding procedure will modulate the watermark message into a noise channel; on the other hand, the receiving end will extract the watermark message from the noise channel [20]. Watermarking applications could be in copyright protection, image authentication, data hiding, and covert communications [21]. In this thesis, only copyright protection will be discussed.

2.2.1 Watermarking at Spatial Domain

The first generation watermarking algorithms work in the spatial domain because it is less expensive and less demanding in computer complexity. One method is called LSB coding: the LSB bit of data byte will be modified for embedding watermark. LSB coding is brittle under attack by just masking the LSB of data bytes so that it is quickly replaced by other new techniques. Spread spectrum techniques can spread watermark in a wider spectrum against the attack which works on only one particular bandwidth [22]. Authors in [23], [24] implement spatial watermarking algorithms in low cost

hardware. However, spatial domain techniques could not easily achieve the needs of robust requirements.

2.2.2 Watermarking at DCT Domain

Similarly as DFT, DWT, the DCT watermarking is working at frequency domain. The examples of DFT (discrete Fourier transform) and DWT (discrete wavelet transform) watermarking are [25], [26]. But most commonly watermarking techniques are DCT domain based. We have seen that after changing working domain from spatial domain to DCT domain, the correlation of spatial pixels will be de-correlated into discrete frequency parts. The DC and low frequency coefficient of DCT matrix will determine most natural features of an image. After truncating higher frequency coefficients, the image fidelity will remain fair enough to human perception by applying inverse DCT. So, a natural approach is embedding a watermark DCT coefficients matrix into image DCT coefficients matrix in lower or middle frequencies area to achieve the robust watermark ([27], [28], [24], [30], [31], [32], [33]). Figure (2.13) shows the DCT watermarking inserting locations:

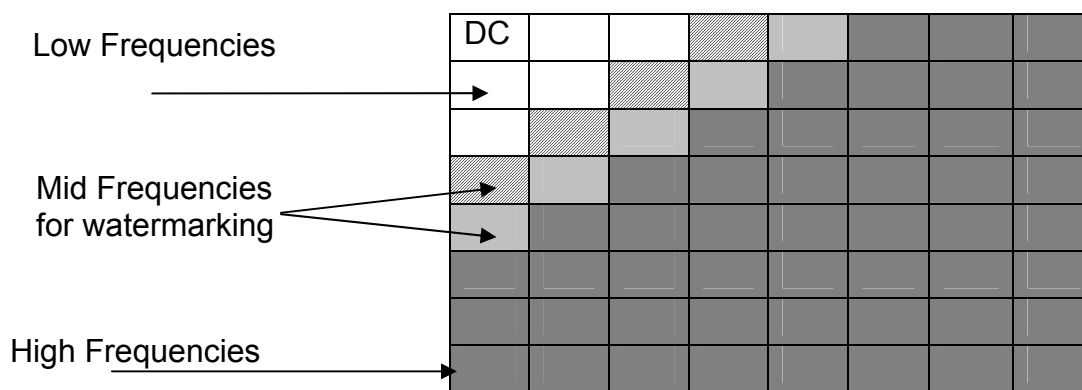


Figure (2.13) Embedding a watermark in mid frequency [34].

The robustness of DCT watermarking comes from the fact that if an attacker tries to remove watermarking at mid frequencies, he will risk degrading the fidelity of the image

because some perceptive details are at mid frequencies [35]. A watermark embedding equation is proposed in [28]:

$$C_w(i, j) = \alpha C(i, j) + \beta W(i, j). \quad (2.28)$$

Where, $C_w(i, j)$ is the DCT coefficient (i, j) after watermarking embedding; α and β are watermark strength factors which can determine whether the watermark is visible or invisible; $C(i, j)$ is original DCT coefficient before watermarking; $W(i, j)$ is watermark DCT coefficient.

Even though the above watermarking algorithms were originally applied to still images, considering that a video frame sequence could be treated as a series of still images, a reasonable approach is processing each frame of video as a still picture with above methods in spatial or frequency domain to embed watermark. The video watermarking in spatial domain is proposed as [36]; in DWT domain as [37], [38]; in DFT domain as [39]; in DCT domain as [20], [40], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50]; in Compressed domain (bit stream)[20], [44], [48], [49], [50], [51] or uncompressed domain (raw data) [20], [44]. The reason of DCT watermarking solution is more common in video is that MPEG video compression also requires DCT function. To reduce system complexity, watermarking in DCT domain is an understandable choice. If the watermarking subject is a compressed video stream, one option is directly watermarking the compressed video bit stream which is already in DCT domain rather than decoding the stream back to video frames for watermarking at spatial domain. However, if a visible watermark is embedded in compressed domain, the visible watermark will drift with moving object in the scene when the decoded frames are playing back such that drift compensation is needed [20], [44].

2.2.3 Visible and Invisible Watermarking

Invisible watermarking at DCT also can be implemented with watermarking Equation (2.25). By just adjusting the watermarking factors α and β , the watermark could become visible or invisible. The same equation is simply run to extract an invisible watermark [52], [53]. A safe way is inserting an invisible watermark in wherever of image DCT coefficients against attacker's removing attempts or use pseudo-random sequence to spread the invisible watermark DCT coefficients among frames' DCT matrix blocks. For this technique, both DC and AC could be subject of watermarking equation [41], [44], [45].

Visible watermarking in video is very commonly applied in video broadcasting. For the prototype working module, the image DCT 16X16 coefficients matrix will directly add with watermark image DCT 16X16 coefficients matrix as above Equation (2.14), and it is illustrated as:

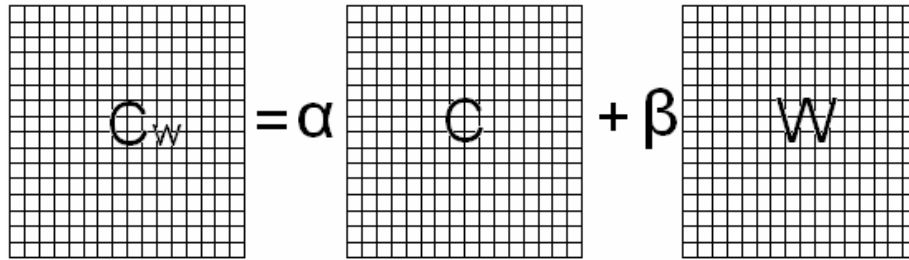
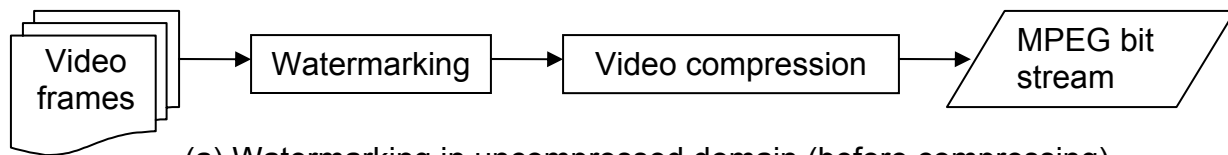
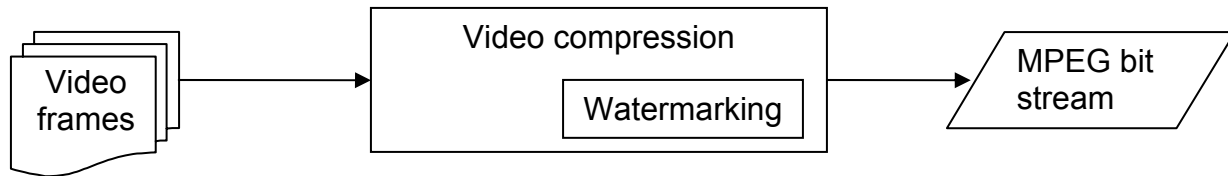

$$C_w = \alpha C + \beta W$$

Figure (2.14) Watermark embedding at 16X16 DCT coefficients matrix.

The reason to choose 16X16 block rather than 8X8 block is that in the watermarking embedding Y color frames, one pel is 16X16 pixels. Unlike mid frequency inserting, DC and all ACs of image DCT coefficients matrix will be modified by the watermark embedding because the watermarking will result in a noticeable and visible perception to human visual system. Two choices to insert watermark into frames of a video are watermarking before or after compression:



(a) Watermarking in uncompressed domain (before compressing).



(b) Watermarking in compressed domain (within compressing).

Figure (2.15) Watermarking in uncompressed and compressed domain.

The slight difference of the above two watermarking schemes is the location the watermark is inserted. The scheme Figure (b) of Figure (2.15) results in the watermark drifting with moving objects in a scene while scheme (a) does not have such issue. To understand scheme (b) and watermark drift, suppose a frame sequence of a video is three adjacent frames, the first frame is I frame, second frame is B frame and third frame is P frame in encoding (compressing) and decoding (decompressing) procedures. At compressing stage, after I frame runs DCT, I frame's DCT coefficient matrix is ready for mid-frequency watermark embedding with a watermark DCT coefficient matrix. If only I frame is watermarked, the same watermark also appears on predicted P and B frames because they are predicted from I frame as base frame with motion estimation and motion compensation. Since the motion vector indicates the macroblock's displacement between two frames, the watermark generally should keep still among frames. So after applying motion vector to rebuild the predicted frame, the watermark overlapping with moving object in the frames drifts back and forth with the moving objects. But why is it necessary to watermark compressed video stream rather than uncompressed one? The reason is most videos/movies could already have been in

compressed format. If decoding them to spatial domain and watermarking them, extra noise could be generated and PSNR (peak signal noise ratio) could be reduced.

2.2.4 Drift Compensation of Visible Watermarking in Compressed Domain

A solution called drift compensation is introduced by Hartung [20]. The scheme is: the un-watermarked inter frame (P or B) is subtracted from the watermarked inter frame (P or B) to extract the drifting watermark. The extracted drifting watermark to cancel the inter frame's drift and inter frames are embed the watermark again. The scheme's block diagram of drift compensation is given in [20]. Notice the drift compensation scheme works at spatial domain rather in DCT domain so the entropy decoding, inverse quantization and inverse DCT decodes quantized DCT coefficients to pixels. Assuncao and Ghanbari introduce a simplified scheme on DCT domain motion compensation to achieve drift compensation [54]. The simplified scheme requires the motion compensation be in DCT domain rather than in spatial domain so that at encoding procedure, two different types of motion compensations are generated, one from spatial domain motion compensation for MPEG decoding, and one from DCT domain motion compensation for drift compensation.

2.3 FPGA (Field Programmable Gate Array) Implementation

FPGA is a programmable logic device which allows a designer to change its internal logic gate connection with a hardware description language (HDL). The Most popular HDLs are VHDL, Verilog, System C, and AHDL. Because most engineers use MATLAB/Simulink™ to create mathematical modules for prototype design, Mathworks© [12], Altera© and Xilinx© also supply product modules to apply MATLAB™ codes or Simulink™ block sets directly to program the FPGA. The new generation Simulink™

can handle blocks less than 80 in number to program FPGA [12]. Altera©'s DSP Builder™ can link Mathworks©' MATLAB™ and Simulink™ to its Quartus II™ IDE tools, and then program FPGA [10]. Xilinx©'s similar product is DSP Generator™ [11]. Both Altera© and Xilinx© offer IP packages for speedy development. However, most developers are willing to use HDL languages to program FPGA to build their prototyping modules.

CHAPTER 3

VIDEO COMPRESSION AND WATERMARKING ALGORITHMS

The main algorithms for video compression are color space conversion and sampling rate, DCT and IDCT, quantization, zigzag scanning re-order, entropy coding. The watermarking algorithms are watermarking embedding and drift compensation if watermarking in compressed domain.

3.1 Video Compression Algorithms

Each algorithm of MPEG video compression is described; and the overall MPEG algorithm is integrated with individual modules.

3.1.1 Color Space Conversion and Sample Rate Algorithm

From color space conversion equations:

$$\left. \begin{aligned} Y &= 0.299R + 0.587G + 0.114B \\ Cb &= 0.564(B - Y) + 128 \\ Cr &= 0.713(R - Y) + 128 \end{aligned} \right\} \quad (3.1)$$

Where only total 7 adders and 5 multipliers are need, the VHDL codes are concurrent. The delay is from adders and multipliers so that it is not critical path. The sample rate is 4:2:0 so that every Y pixel is sampled while one of every 4 Cb and Cr is sampled.

3.1.2 Motion Estimation Algorithm

Motion estimation is in the critical path of video compression coding since most time delay occurs at this step. The SAD (sum of abstract difference) algorithm searches the 48X48 pixels square target region exhaustively to find out a matching 16X16 pixel macroblock. The output of this procedure is prediction error for motion compensation and motion vector. The data path block diagram is:

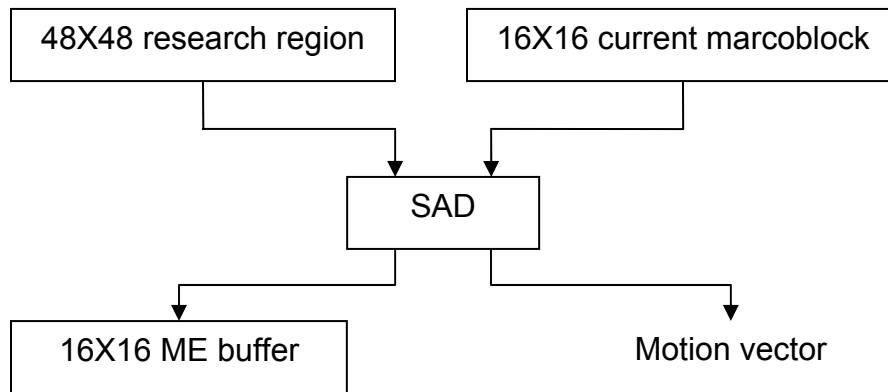


Figure (3.1) Motion estimate data path block diagram

The flow chart of motion estimate and motion compensation is:

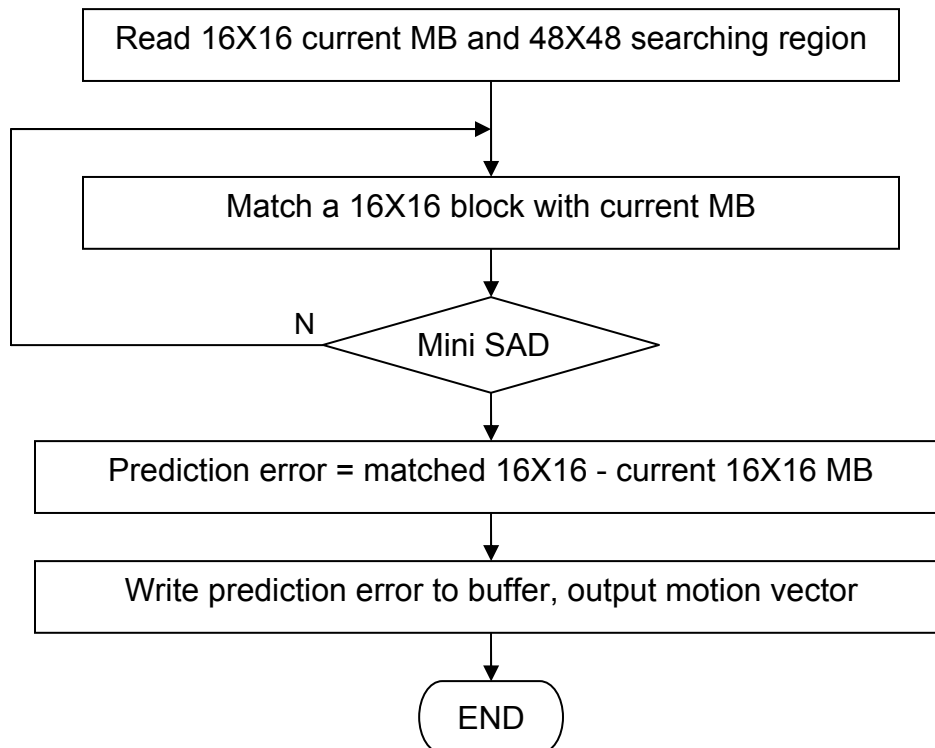


Figure (3.2) Motion estimate flow chart.

The VHDL program for motion estimate and motion compensation is sequential code, and causes much time delay.

3.1.3 Fast Discrete Cosine Transform (FDCT) Algorithm

The inverse fast DCT algorithm is from Loeffler [16]. For the ease of implementation, the algorithm is expressed as the table follows [55]:

Table (3.1) Loeffler's fast 8 elements 1-D inverse DCT algorithm.

Step 1	Step 2	Step 3	Step 4	Step 5
$b_0 = a_0 + a_4$	$c_0 = b_0$	$d_0 = c_0 + c_3$	$e_0 = d_0$	$f_0 = e_0 + e_7$
$b_1 = a_0 - a_4$	$c_1 = b_1$	$d_1 = c_1 + c_2$	$e_1 = d_1$	$f_1 = e_1 + e_6$
$b_2 = a_2 * m_1 - a_6 * m_2$	$c_2 = b_2$	$d_2 = c_1 - c_2$	$e_2 = d_2$	$f_2 = e_2 + e_5$
$b_3 = a_2 * m_2 + a_6 * m_1$	$c_3 = b_3$	$d_3 = c_0 - c_3$	$e_3 = d_3$	$f_3 = e_3 + e_4$
$b_4 = a_7 * m_3$	$c_4 = b_7 - b_4$	$d_4 = c_4 + c_6$	$e_4 = d_4 * m_4 - d_7 * m_5$	$f_4 = e_3 - e_4$
$b_5 = a_3$	$c_5 = b_5$	$d_5 = c_7 - c_5$	$e_7 = d_4 * m_5 + d_7 * m_4$	$f_5 = e_2 - e_5$
$b_6 = a_5$	$c_6 = b_6$	$d_6 = c_4 - c_6$	$e_5 = d_5 * m_6 - d_6 * m_7$	$f_6 = e_1 - e_6$
$b_7 = a_1 * m_3$	$c_7 = b_7 + b_4$	$d_7 = c_7 + c_5$	$e_6 = d_5 * m_7 + d_6 * m_7$	$f_7 = e_0 - e_7$
$m_1 = \sqrt{2} * \cos(6 * \pi / 16)$ $m_2 = \sqrt{2} * \sin(6 * \pi / 16)$ $m_4 = \sqrt{2} * \cos(3 * \pi / 16)$		$m_5 = \sqrt{2} * \sin(3 * \pi / 16)$ $m_6 = \sqrt{2} * \cos(\pi / 16)$ $m_7 = \sqrt{2} * \sin(\pi / 16)$		$m_3 = \sqrt{0.5}$

In the above table, $m_1 \sim m_7$ are pre-calculated constants, $a_0 \sim a_7$ are raw input values, and $f_0 \sim f_7$ are one dimensional IDCT transform results. The fast DCT algorithm reduces the number of adders and multipliers so that the execution of DCT is accelerated. The comparison of total number of adders and multipliers is:

Table (3.2) DCT adders and multipliers in total.

Type	Total adders	Total multipliers
Classic 1-D 1X8 DCT	56	72
Fast 1-D 1X8 DCT	26	14

The 2-dimensional DCT and IDCT algorithms can be achieved by running 1-dimensional algorithms in two times, one time in rows (horizontal) and one time in columns (vertical). The data path block diagram of 2-dimensional 8X8 DCT is as:

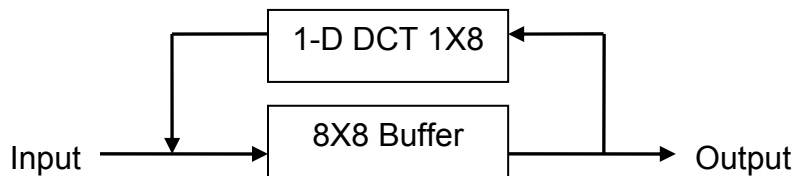


Figure (3.3) 2-D DCT component data path block structure.

The flow chart of 2-D 8X8 DCT is:

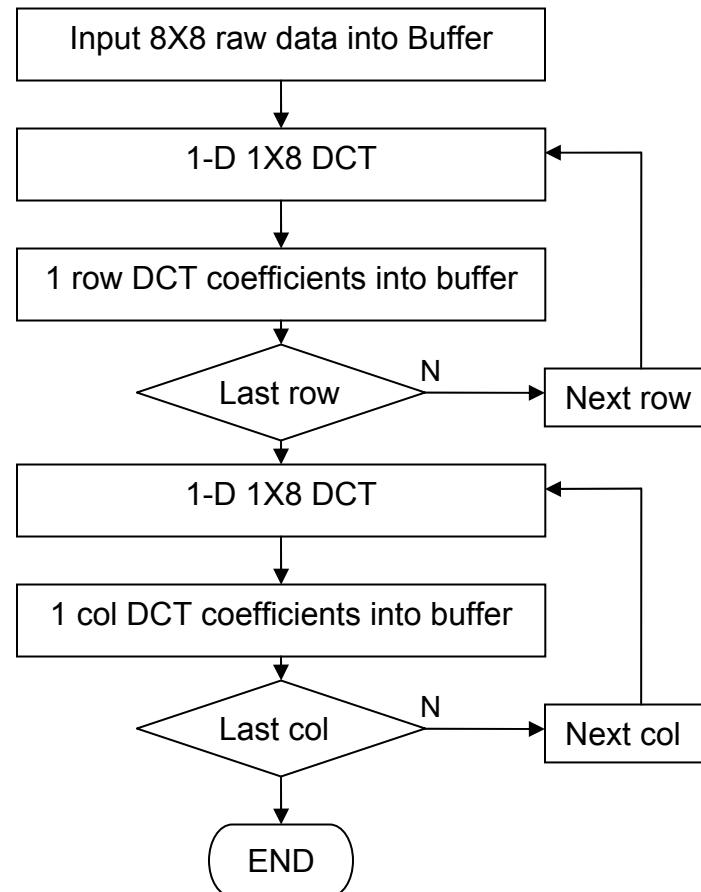


Figure (3.4) 2-D DCT algorithm flow chart.

The 2-D IDCT algorithm flow chart is similar as Figure (3.2). The only modification is replacing block “1-D 1X8 DCT” with “1-D 1X8 IDCT.”

Because of the 8X8 buffer, VHDL program for 2-D DCT/IDCT could not be concurrent, but must be sequential codes. That causes some time delay.

3.1.4 Quantization Algorithm

After DCT, the quantization algorithm quantizes the DCT coefficient matrix to generate more “0” coefficients at high frequency sections while keeping good resolution at lower frequency parts. For video compression, the quantization equation for Intra frames and non-intra frames and the quantization scale factor which determine

quantization resolution are different. The quantization equations are Equations (2.14), (2.15), (2.16) and (2.17); the quantization scale factor tables are from Table (2.4). The data path block diagram of the quantization procedure is:

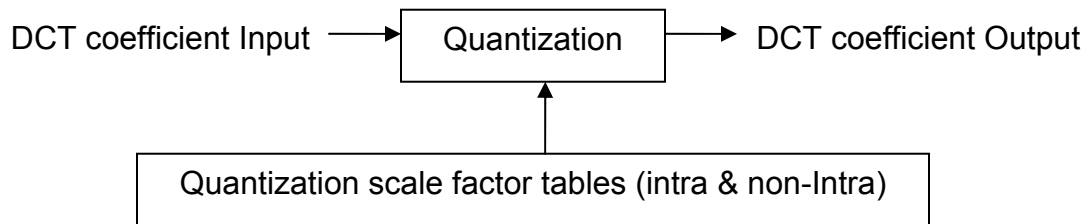


Figure (3.5) Quantization component data path block diagram

The flow chart of quantization for MPEG is:

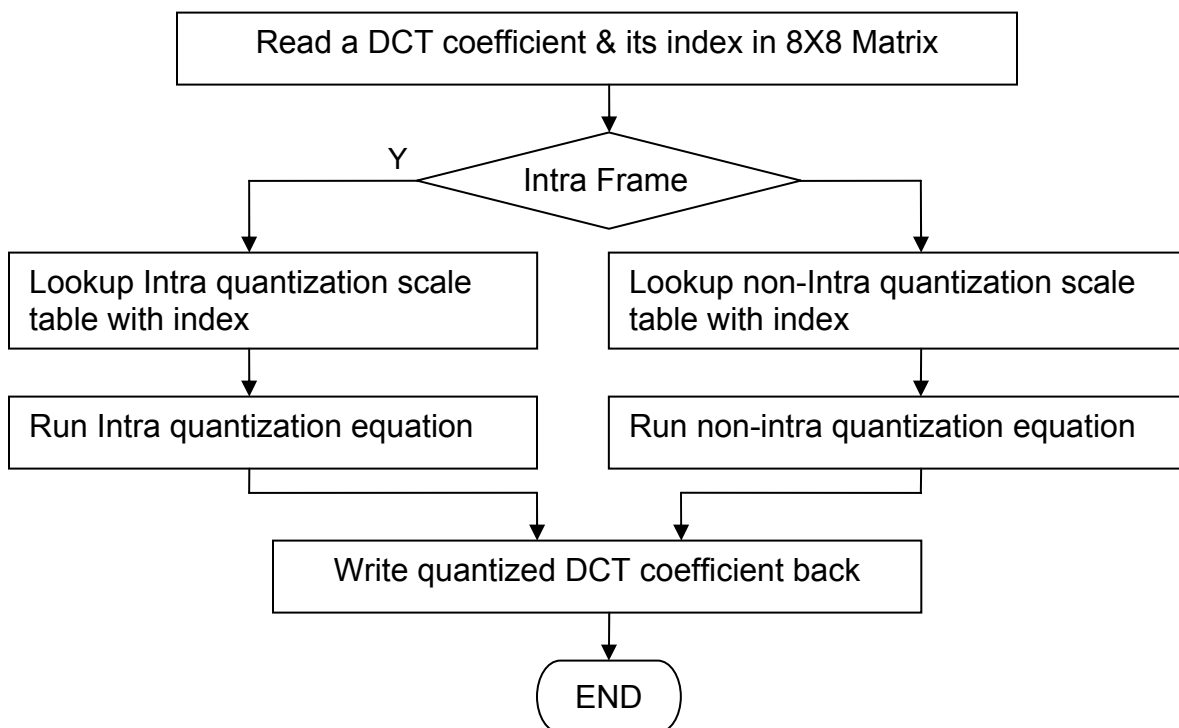


Figure (3.6) Quantization algorithm flow chart.

To VHDL programming, the quantization procedure could be concurrent codes to minimize time delay.

3.1.5 Zigzag Scanning Algorithm

Zigzag scanning re-orders the DCT coefficients of a matrix ascending in frequency. For progressive frames and interlacing fields, the zigzag scanning routes are provided as Table (2.5). The data path of zigzag scanning block diagram is:

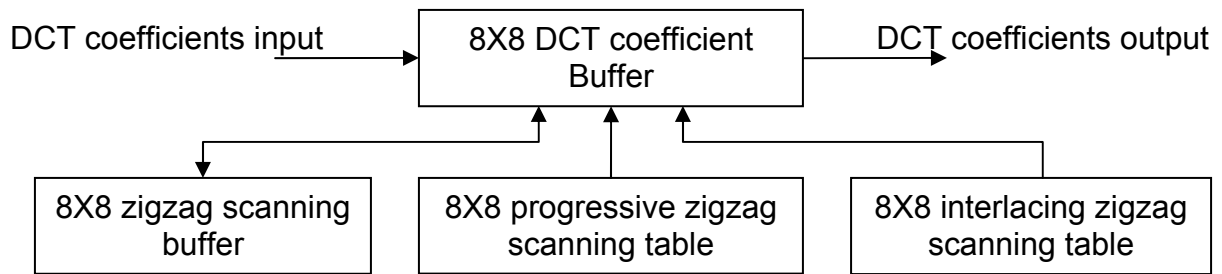


Figure (3.7) Zigzag scanning component data path block diagram.

The flow chart of zigzag scanning is:

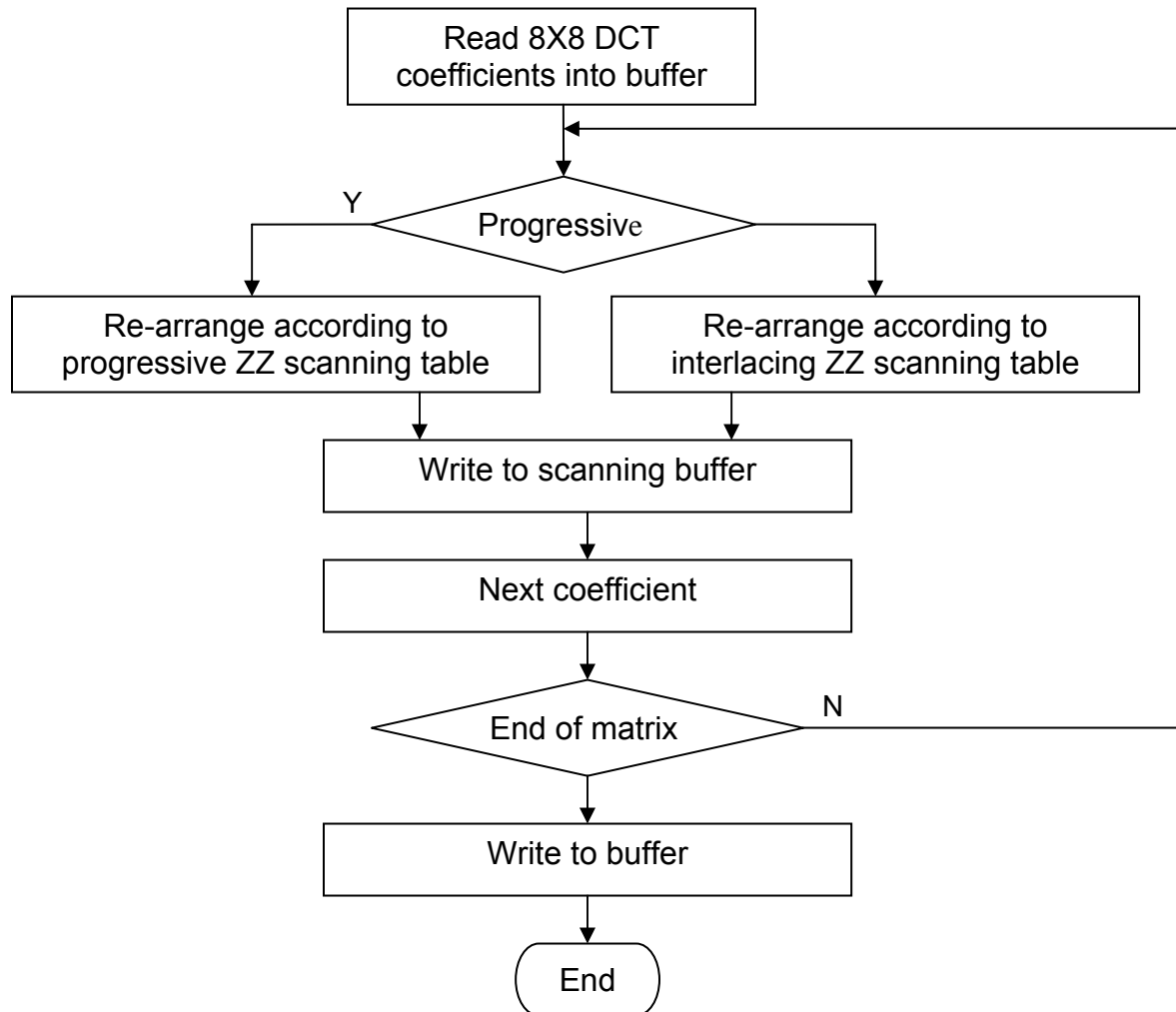


Figure (3.8) Zigzag scanning algorithm flow chart.

Because of two buffers in the structure, the VHDL program for zigzag scanning could not be concurrent codes. The sequential codes have smaller time delay.

3.1.6 Entropy Coding Algorithm

The entropy coding efficiency depends on the precision of calculation for computing each coefficient's occurring probability. However, calculating probabilities of all coefficients is impossible in real-time MPEG codec. One solution is utilizing pre-calculating Huffman code Table (2.8) for generic images. The entropy coding can apply

on the DC coefficients of different blocks, the AC coefficients within one block, and the motion vectors within one frame. Here the data path block diagram for the entropy coding of AC coefficients within one block is given:

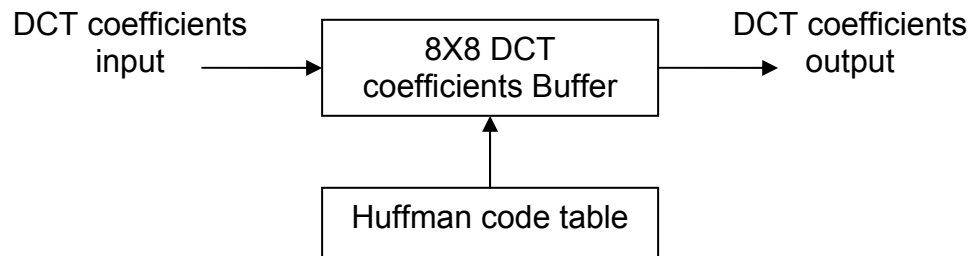


Figure (3.9) Entropy coding (Huffman) component data path block diagram.

The algorithm flow chart is:

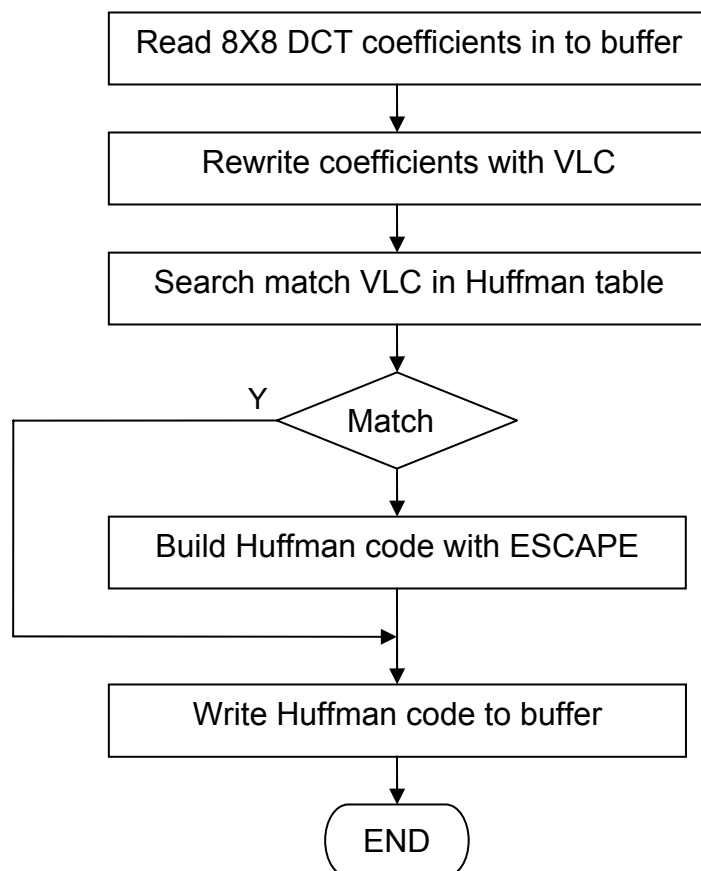


Figure (3.10) Entropy coding (Huffman) flow chart.

The entropy coding operated upon a block; the sequential code must be used.

3.1.7 MPEG Video Compression Algorithm

With above individual algorithm, the whole MPEG video compression algorithm as Figure (2.15) is described in procedure step flow as follows:

Table (3.3) MPEG video compression algorithm flow.

Input	Video RGB frames(NxM)
Output	MPEG stream
Step 1	RGB color frames converted to YCbCr frames
Step 2	YCbCr frames re-sampled according to 4:2:0 sampling rate
Step 3	YCbCr frames go to Buffer which hold a GOP (for example, 15 continuous adjacent frames).
Step 4	MPEG video compression starts. Y frame is split into 16x16 blocks, Cb and Cr are split into 8x8 blocks
Step 5	Only Y frames run motion estimate. Each 16x16 Y block rescale to 8x8 blocks. If the even first frame (I) of GOP, go Step 9; If P frame, go to Step 6; If B frame, go to Step 8.
Step 6	Y frame forward or backward motion estimate P frames with reference frames (I or P frames). The motion vectors (MV) and prediction errors of residual frame for motion compensation (MC) are found. If Y frame, go to Step 9;
Step 7	Find Cb, Cr motion vector and prediction error. Go to step 9
Step 8	Y frame interpolated motion estimate B frames with two P frames or I and P frames in bilinear algorithm. The motion vectors (MV) and prediction errors of residual frame for motion compensation (MC) are found.
Step 9	2-D DCT on blocks of frames from Step 5, 6, 7, 8.
Step 10	Quantize 2-D DCT coefficient matrix.
Step 11	Zigzag scan quantized 2-D DCT coefficient matrix.
Step 12	Entropy coding re-ordered 2-D DCT coefficient matrix and motion vector.
Step 13	Y, Cb and Cr frames to buffer
Step 14	Build structured MPEG stream from buffer

3.2 Watermark Embedding Algorithms

Two watermarking schemes are investigated: watermarking on uncompressed or in compressed domain. For the watermarking in compressed domain, drift compensation is required.

3.2.1 Watermarking Algorithm in Uncompressed Domain

Watermarking in uncompressed domain can be in spatial domain or frequency domain. Because of its robustness, the DCT domain watermark embedding algorithm is chosen. The watermark embedding flow chart is:

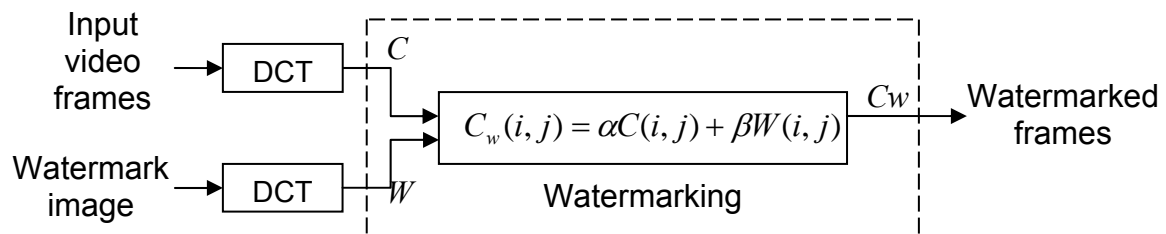


Figure (3.11) Watermark embedding component flow chart.

The data path and flow chart of DCT watermarking in uncompressed domain is:

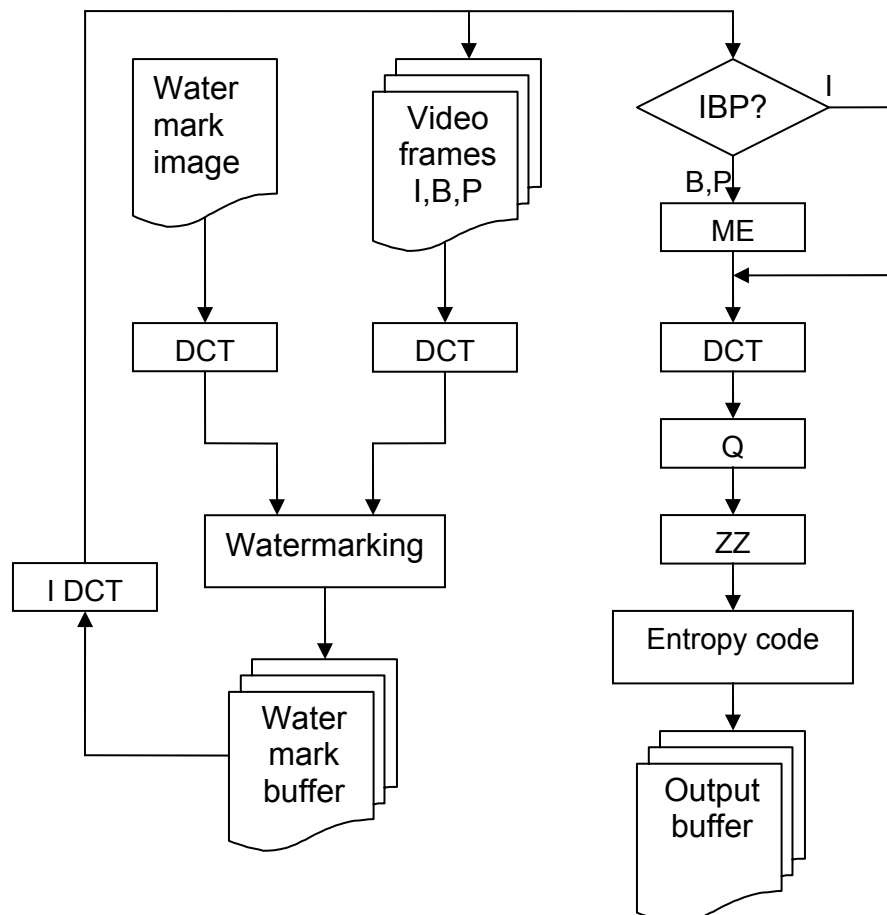


Figure (3.12) Watermarking in uncompressed domain data path and flow chart.

To clarify further above flow chart figure, a step flow to describe the watermarking in uncompressed domain algorithm is as the table follows:

Table (3.4) MPEG watermarking algorithm flow in uncompressed domain.

Input	Video RGB frames(NxM), watermark monochrome image(NxM)
Output	MPEG stream
Step 1	RGB color frames converted to YCbCr frames
Step 2	YCbCr frames re-sampled according to 4:2:0 sampling rate
Step 3	Split Y frame and watermark image into 8X8 blocks
Step 4	Each 8X8 block runs 2-D DCT to generate 8X8 DCT coefficient matrix
Step 5	Each 8X8 Y DCT matrix watermarked with a 8X8 watermark DCT matrix at same location as $C_w(i, j) = \alpha C(i, j) + \beta W(i, j)$ at DCT domain
Step 6	Each 8X8 watermarked matrix runs 2-D IDCT to transform back to Y color pixels
Step 7	Watermarked Y frame, non watermark Cb and Cr frames go to Buffer, which hold a GOP (for example, 15 continuous adjacent frames).
Step 8	MPEG video compression starts. Y frame is split into 16x16 blocks, Cb and Cr are split into 8x8 blocks
Step 9	Only Y frames run motion estimate. Each 16x16 Y block rescale to 8x8 blocks. If the even first frame (I) of GOP, go to Step 13; If P frame, go to Step 10; If B frame, go to Step 12.
Step 10	Y frame forward or backward motion estimate P frames with reference frames (I or P frames). The motion vectors (MV) and prediction errors of residual frame for motion compensation (MC) are found. If Y frame, go to Step 13;
Step 11	Find Cb, Cr Motion Vector and Prediction error. Go to step 13
Step 12	Y frame interpolated motion estimate B frames with two P frames or I and P frames in bilinear algorithm. The motion vectors (MV) and prediction errors of residual frame for motion compensation (MC) are found. If Y frame, go to Step 13; If Cb and Cr frames, go to Step 11.
Step 13	Run 2-D DCT on blocks of frames from Step 9, 10, 11, 12.
Step 14	Quantize 2-D DCT coefficient matrix.
Step 15	Zigzag scan quantized 2-D DCT coefficient matrix.
Step 16	Entropy coding re-ordered 2-D DCT coefficient matrix and motion vector.
Step 17	Build structured MPEG stream from buffer

3.2.2 Watermarking with Drift Compensation Algorithm in Compressed Domain

Watermarking on the compressed domain is also DCT watermarking, and drift compensation is essential, otherwise parts of the watermark drift with moving objects in the scene. The data path and flow chart of DCT watermarking in compressed domain are:

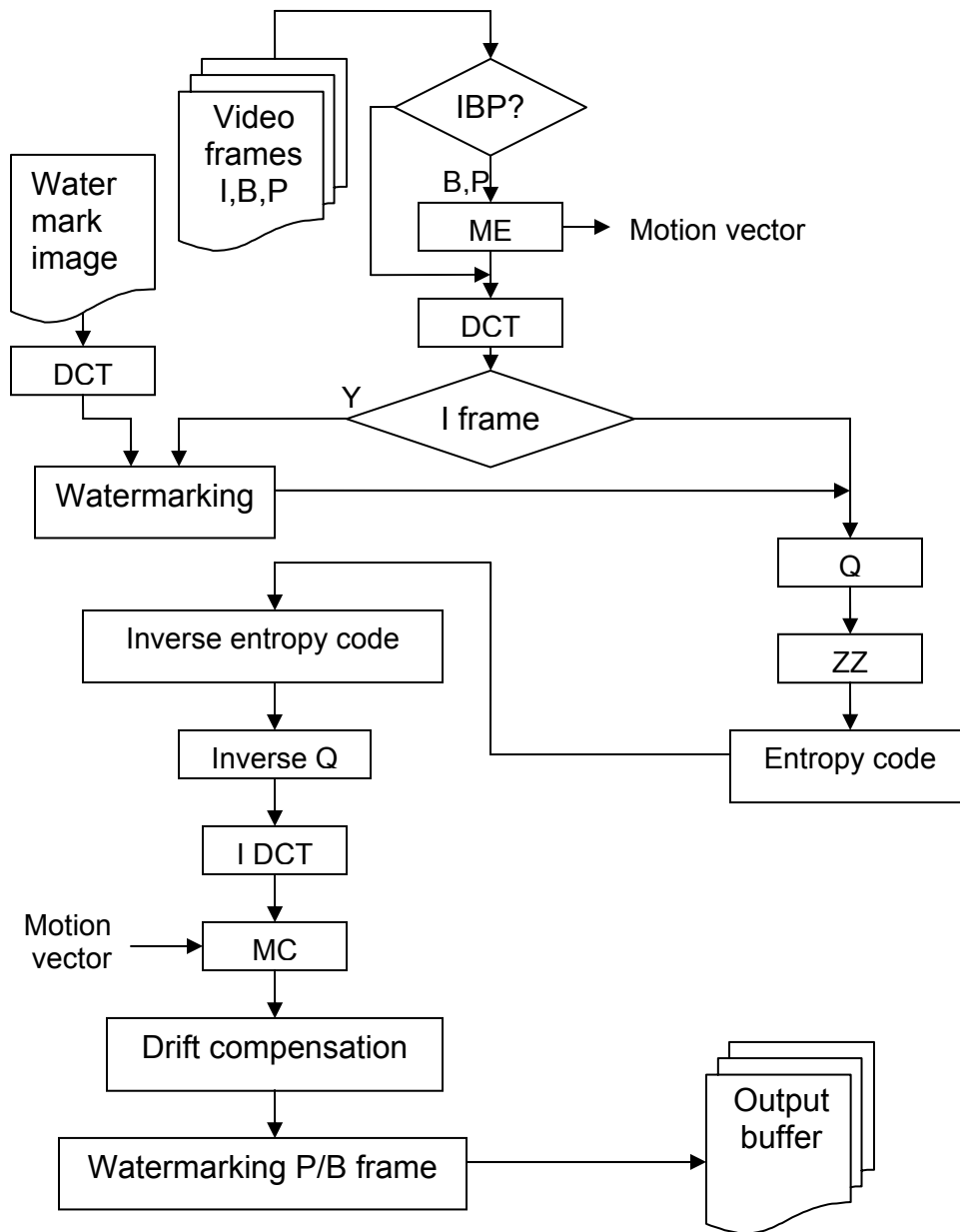


Figure (3.13) Watermarking in compressed domain and drift compensation.

Similarly, a flow step to clarify above figure to describe compressed domain watermarking and drift compensation is as Table (3.5):

Table (3.5) MPEG watermarking algorithm flow in compressed domain.

Input	Video RGB frames(NxM), watermark monochrome image(NxM)
Output	MPEG stream
Step 1	RGB color frames converted to YCbCr frames
Step 2	YCbCr frames re-sampled according to 4:2:0 sampling rate

Step 3	YCbCr frames go to buffer which hold a GOP (for example, 15 continuous adjacent frames).
Step 4	MPEG video compression starts. Y frame is split into 16x16 blocks, Cb and Cr are split into 8x8 blocks
Step 5	Only Y frames run motion estimate. Each 16x16 Y block rescale to 8x8 blocks. If the even first frame (I) of GOP, go Step 9; If P frame, go to Step 6; If B frame, go to Step 8.
Step 6	Y frame forward or backward motion estimate P frames with reference frames (I or P frames). The motion vectors (MV) and prediction errors of residual frame for motion compensation (MC) are found. If Y frame, go to Step 9;
Step 7	Find Cb, Cr motion vector and prediction error. Go to step 9
Step 8	Y frame interpolated motion estimate B frames with two P frames or I and P frames in bilinear algorithm. The motion vectors (MV) and prediction errors of residual frame for motion compensation (MC) are found.
Step 9	2-D DCT on blocks of frames from Step 9, 10, 11, 12.
Step 10	2-D DCT on the 1 st 8x8 block for each 16x16 blocks of watermark image
Step 11	Watermark Y of I, B, P frames with $C_w(i, j) = \alpha C(i, j) + \beta W(i, j)$ at DCT domain with blocks from Step9, 10
Step 12	Quantize 2-D DCT coefficient matrix.
Step 13	Zigzag scan quantized 2-D DCT coefficient matrix.
Step 14	Entropy coding re-ordered 2-D DCT coefficient matrix and motion vector.
Step 15	Cb and Cr frames to buffer
Step 16	Entropy decoding Y frame
Step 17	Inverse zigzag scanning
Step 18	Inverse quantization
Step 19	Inverse DCT
Step 20	If B, P frames, predicate frame with reference frame, motion vector and run motion compensation with predication error. Go to Step 25
Step 21	Original Y frame run video compression without watermarking as above without step 10, 11.
Step 22	Original Y frame run video compression as above except just watermarking I frame at Step 11.
Step 23	Decode MPEG stream from step 21, 22 respectively
Step 24	Extract drifting watermark by subtract decoded video frames between watermarked and un-watermarked frames from Step 23.
Step 25	Subtract IBP watermarked frames with drifting watermark frames
Step 26	MPEG compression Y frames again as Step 5,6,8,9,12,13,14
Step 27	Build structured MPEG stream from buffer

The procedure extracting the drift watermark of above in compressed domain could be simplified.

CHAPTER 4

SYSTEM ARCHITECTURE

The algorithms of visible watermarking in uncompressed domain and compressed domain are implemented into two different architectures. The watermarking architecture in uncompressed domain is in low-cost and low-complexity; the one in compressed domain with drift compensation has extra video compression and decompression modules.

4.1 Architecture of MPEG Watermarking in Uncompressed Domain

The watermarking in uncompressed domain is directly watermarking raw uncompressed video frames such that the watermark embedding can work at spatial domain or frequency domain (DFT, DCT, DWT, etc). The techniques can be quickly adapted from still image watermarking. The architecture is merged from two parts: one is MPEG video compressing; and another is still image watermarking. The watermarking works at Y (brightness) frames only for human visual perception is sensitive to them if the watermark image is monochrome. For a color watermark image, the Cb and Cr color space must be watermarked with same techniques for Y frames as well. The top level simplified view of watermarking in uncompressed domain is follows:

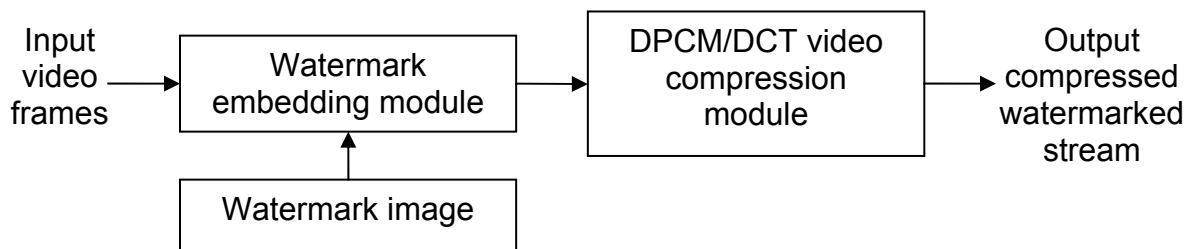


Figure (4.1) Block level view of MPEG video compression and visible watermark embedding module in uncompressed domain.

The high level architecture of the module is tested with Simulink™ firstly, and the prototyping implementation is created with VHDL. The system architecture for FPGA implementation is as:

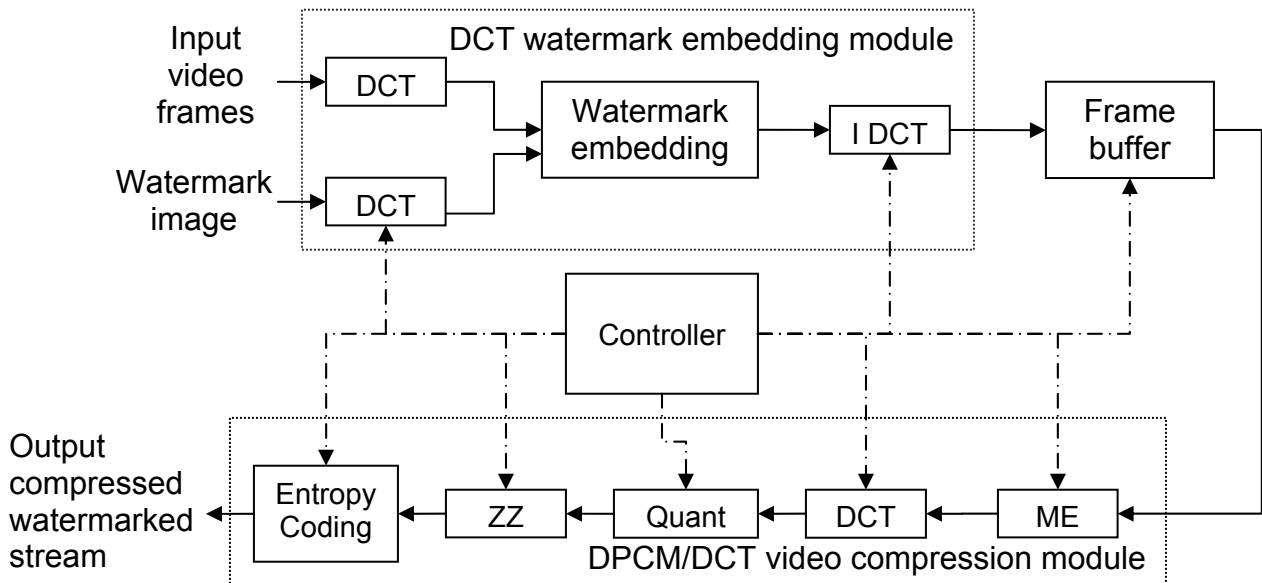


Figure (4.2) System architecture of MPEG video compression and watermarking in uncompressed domain.

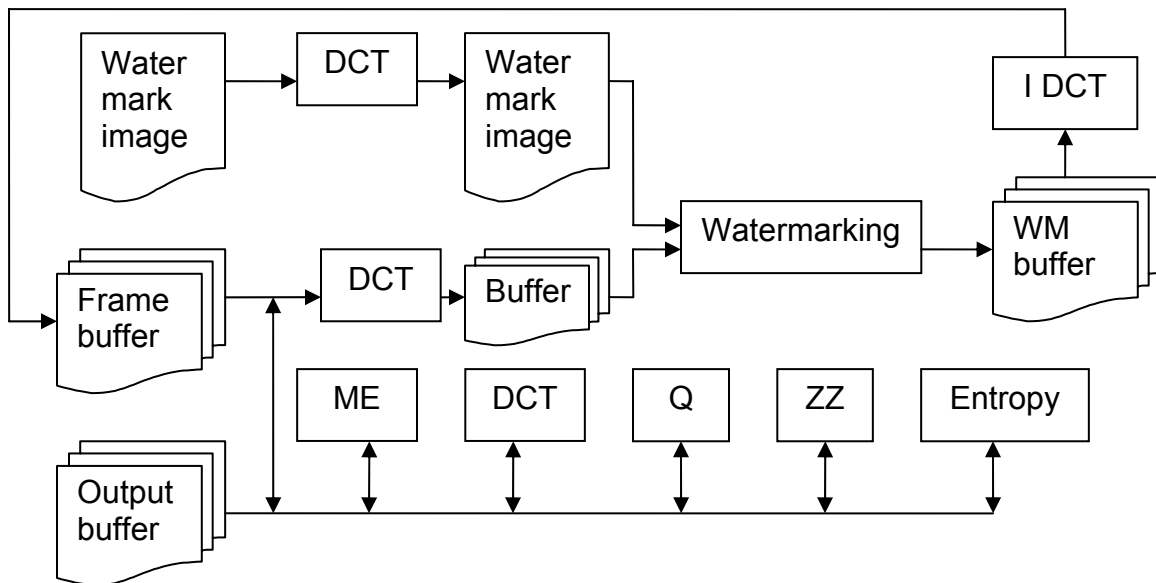
In above system architecture, “DCT watermark embedding module” processes watermark embedding. After that procedure, the watermarked video frames are resulted. By simply replacing it with other watermarking technique modules, spatial, DFT or DWT watermarking can be achieved. “DPCM/DCT video compression module” processes watermarked video frames to generate MPEG video stream. The data bus length is 12-bits. Each block in above figure is detailed as follows:

- Watermark embedding: watermark algorithm processing. It embeds a watermark image into a video frame with watermarking equation (2.25). The input and output are buffered to frame buffer.
- Frame buffer: It buffers the frames for every block procedure. Its size capacity is enough for one input GOP (for example, 15 frames for each

color, so 45 frames in total for Y, Cb and Cr color spaces), output motion vectors, and output stream.

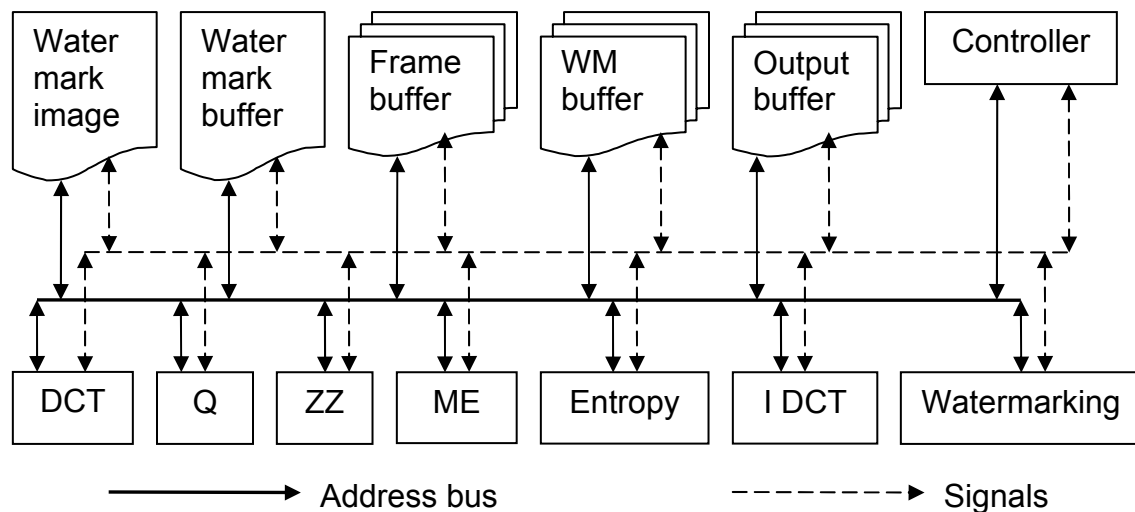
- DCT/IDCT: 2-D DCT with 12-bits data bus and 6-bits address bus for 64 bytes internal buffer. The input data is 8-bits unsigned integer, the output is a 12-bits unsigned integer. For further higher precision, greater bit length could be considered. The detail algorithms are in Table (3.1), (3.2), and Figure (3.4). The input and output are buffered to frame buffer.
- ME: motion estimate searches exhaust a 48X48 block for a 16X16 block match. The detail flow chart is as Figure (3.1) and (3.2). The input and output motion vector and prediction error for motion compensation are buffered to frame buffer.
- Quant: quantization procedure. It quantizes 8X8 DCT coefficients according to quantization Table (2.4) with quantization Equation (2.14) and (2.16). The input and output are buffered to frame buffer.
- ZZ: zigzag scanning procedure. It re-orders 8X8 DCT coefficients according to the Table (2.5). The input and output are buffered to frame buffer.
- Entropy: entropy coding procedure. Actually, it is Huffman coding table look up processing. The input and output are buffered to frame buffer.
- Controller: It generates addressing and control signals with clock for each individual component module in the system to synchronizes the system working functions. It is a finite state machine.

The MPEG video compression and visible watermarking in uncompressed domain system data path and its block diagram is:



Figure(4.3) System data path in uncompressed domain (data bus width is 12-bits).

The system has a controller which generates addressing signals and control signals to synchronize all components. The address bus and signals diagram is:



Figure(4.4) System address and signals of watermarking in uncompressed domain.

4.2 Architecture of MPEG Watermarking in Compressed Domain

Unlike watermarking in uncompressed domain, the watermarking in compressed domain is following DCT module inside a DPCM/DCT video compression component module. The watermarking subjects here is not independent frames as still images, they are correlated frames with each other in temporal mode, i.e., inter frames (P or B) predicated from intra frame. So, every object in base intra frame is inherited by predicted inter frames (P or B) such that the watermark in intra frame appears in inter frames (P or B) even though they are not embedded with the watermark. However, if it overlaps with any moving objects in the video scene, the watermark drifts around with the moving objects. To obtain a stable watermark, drift compensation is propose to cancel the side effect [20]. The concept is extracting drift watermark in inter frames (P or B), and cancel it subtracting. Generally, the watermarking here works at DCT domain for sharing same DCT component with video compression module. Extra video decode module is required for drift compensation procedure. Similarly, a monochrome watermark image is embedded into Y color space only. For the color watermark image embedding, all Y, Cb and Cr color spaces needs to be inserted with the watermarks respectively. The top level simplified view of watermarking in compressed domain is follows:

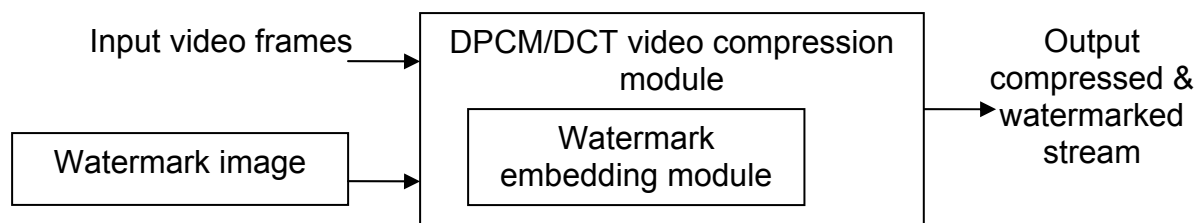


Figure (4.5) Block level view of MPEG video compression and visible watermark embedding module in compressed domain.

The high level architecture of the module also is tested with Simulink™ firstly, and the prototyping implementation is generated with VHDL. The system architecture in compressed domain for FPGA implementation is as:

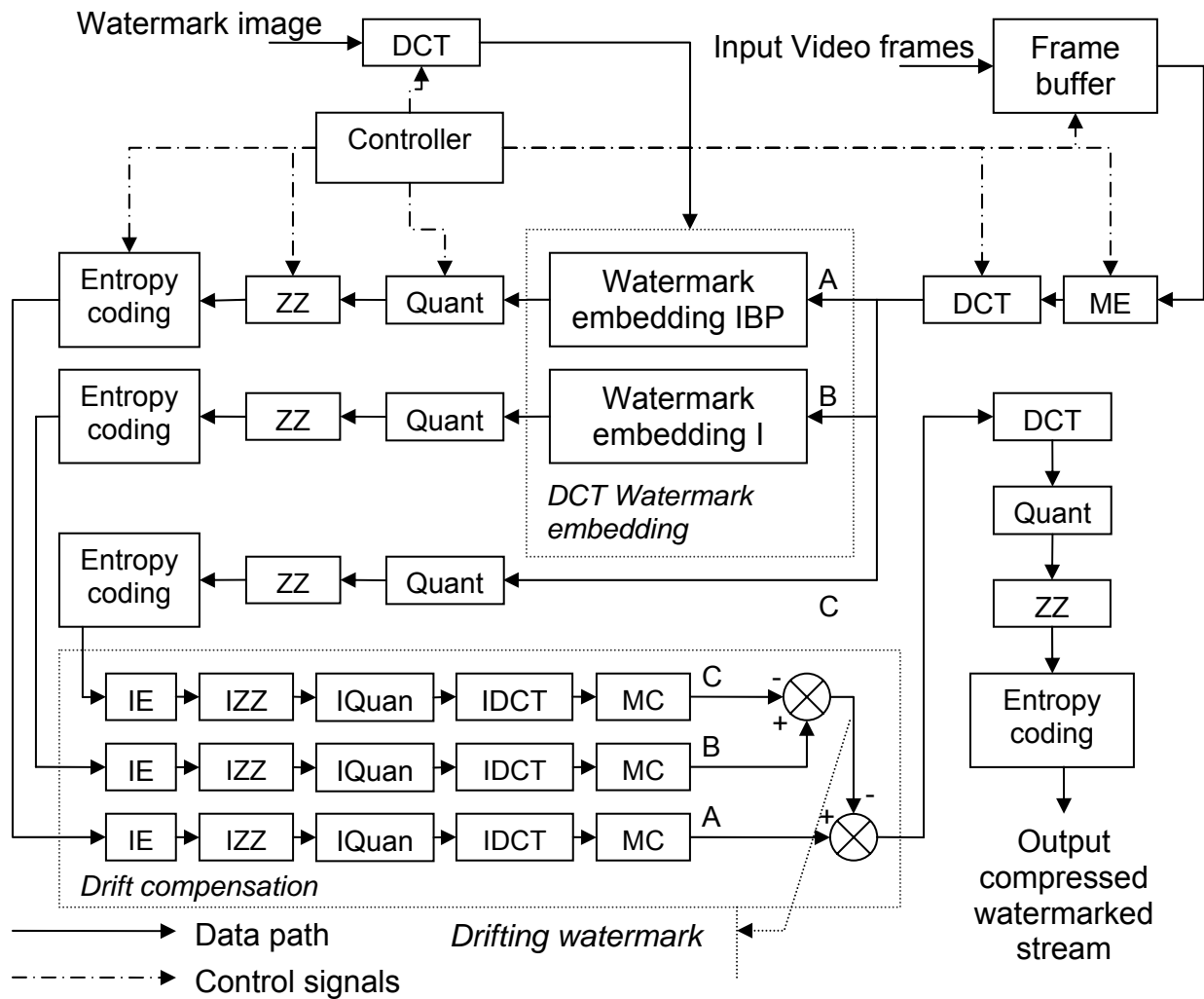


Figure (4.6) System architecture of MPEG video compression and watermarking in compressed domain.*

* Every block receives control signals from controller but not all of them are depicted

The architecture of compressed domain watermarking is much more complex than the one in uncompressed domain. The new components not existing in uncompressed domain one are: IE, IZZ, IQuant, MC and the watermarking embedding modules as follows:

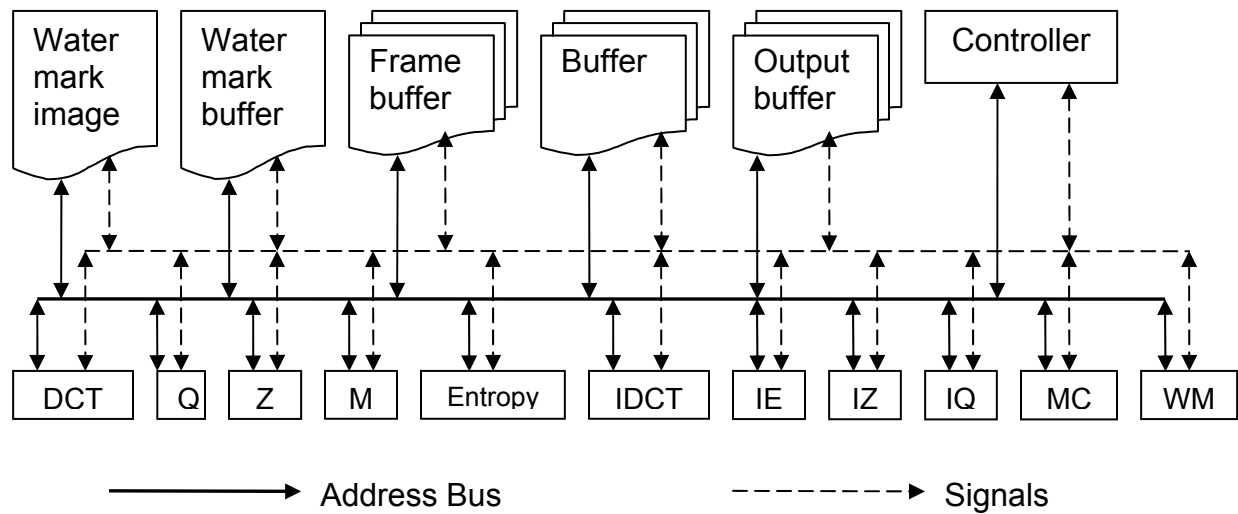
- IE: inverse entropy coding, or decoding. It applies Huffman pre-calculating table as decoding lookup table similarly as encoding. The input and output are buffered to frame buffer.
- IZZ: inverse zigzag scanning. It also applies zigzag table to resume the original order of 8X8 DCT coefficient matrix. The input and output are buffered to Frame Buffer.
- IQua: Inverse quantization. It applies quantization table and inverse quantization Equation (2.15) and (2.17) to resume the original 8X8 DCT coefficient matrix. The input and output are buffered to frame buffer.
- MC: motion compensation. With reference frame and motion vectors, prediction errors, a new frame is rebuilt resemble with original one. If it is intra frame, this block is skipped. The input and output are buffered to frame buffer.
- Watermark embedding IBP: the block embeds a watermark to every frame, I, B, P, sequentially, inter frames as B and P have two watermarks. One inherited from intra frame, one is embedded by the component module. The one inherited is the one drifting in inter frames (B and P).
- Watermark embedding I: the block embed a watermark to intra frame only. The inter frames (B and P) have the same one watermark in intra frame by predicating. If the watermark overlaps with moving objects, it will drift back and forth with the moving objects.

In the Figure (4.6), there are three coding branches: branch A, B and C. In branch A, the watermarking is embedded to all frames, i.e., I, B and P frames. So,

in this branch, inter frames B and P have two watermarks: one is predicted from intra frame, and one is embedded. In branch B, the watermark is inserted to intra frame only. However, inter frame B and P have the same watermark by prediction. This watermark is the drift one and need to be cancel in inter frames. In branch C, the frames are compressed without any watermark. So after decompressing, branch A has two watermarks, one is stable, another is drifting; branch B has one drifting watermark; branch C has no watermark. By subtracting branch B with branch C, the drifting watermark is extracted, and furthermore, by subtracting branch A with the extracted drifting watermark, the drifting watermark effect in inter frames is cancelled.

The purpose of branch C is canceling encoding noise in the drift compensation result. But by inspecting above drifting compensation architecture, one could consider that branch C is not essential because it could be replaced with original video frame directly. It could be removed to simplify drift compensation component's complexity if encoding procedure does not generate too noticeable noise.

Similarly to architecture of watermarking in uncompressed domain, the one in compressed domain has architecture as follows after adding IE, IZZ, IQuan, MC, and modified watermarking module:



Figure(4.7) System address and signals in compressed domain.

Other components are same as those in the model for uncompressed domain. But the controller is different.

Comparing two watermarking architectures, the conclusion is that the architecture complexities are different. As estimate, the time delay is different as well.

CHAPTER 5

PROTOTYPE DEVELOPMENT AND EXPERIMENTS

5.1 System Level Modeling with MATLAB/Simulink™

To verify algorithm and architecture, firstly, a fast prototyping module is built with MATLAB/Simulink™ in function block sets. The methodology at this high level system modeling is top-down: with MATLAB/Simulink™ building-in functions or block sets to create a top level conceptual system module, then each functions will be tuned in details, or add new functional blocks. Both watermarking in uncompressed domain and compressed domain are investigated at this stage.

5.1.1 System Level Modeling Methodology

MATLAB/Simulink™ has already offered video and image processing functions and modules for building fast prototype. The available function units are: DCT/IDCT, SAD for motion estimate, block processing (split), and delay (buffer). With minor work, quantization, zigzag scanning and entropy coding are built. Then the system level-modeling is accomplished as sub-tasks as follows:

Sub-task 1: Color conversion and sampling rate compression

Sub-task 2: DCT domain compression in each frame

Sub-task 3: Quantization and zigzag scanning re-order

Sub-task 4: Entropy coding by looking up Huffman coding table

Sub-task 5: Motion estimate and motion compensation only on I and P frames

Sub-task 6: Interpolating B frames

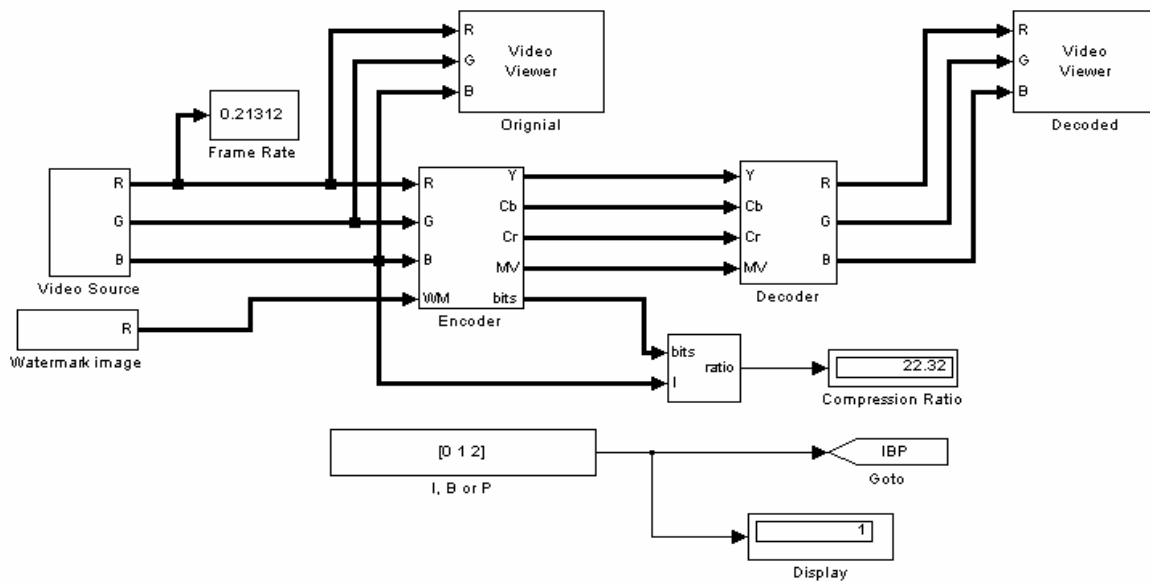
Sub-task 7: Uncompressed domain watermarking

Sub-task 8: Compressed domain watermarking without drift compensation

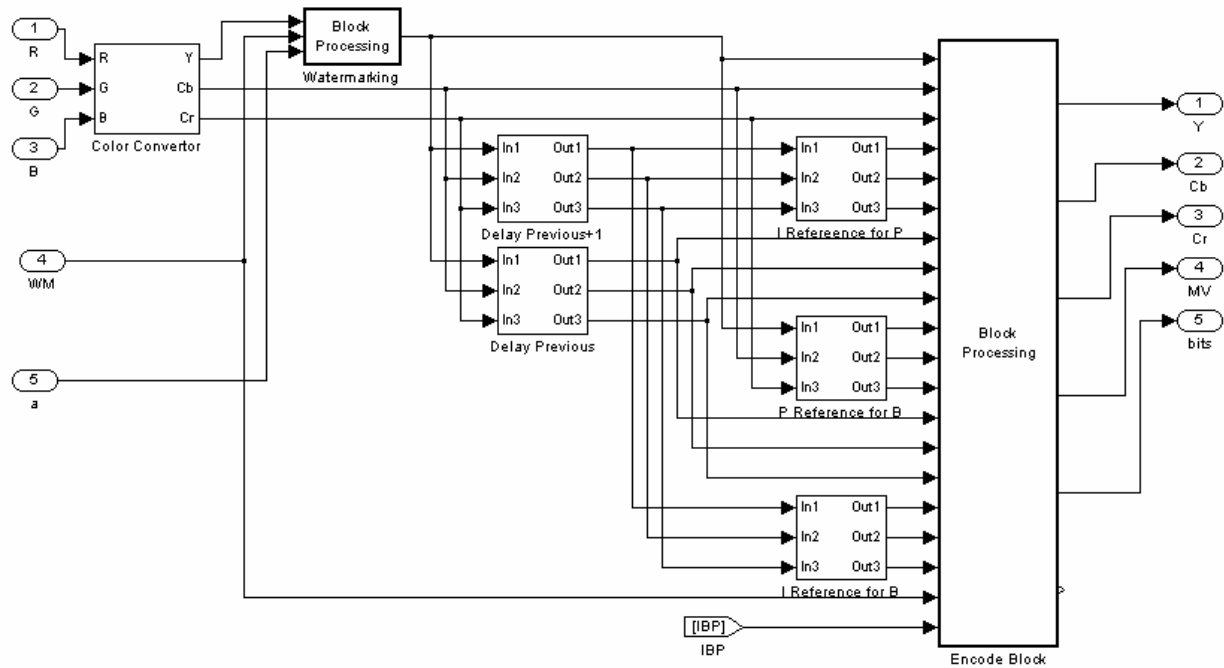
Sub-task 9: Drift compensation in compressed domain watermarking

5.1.2 Modeling Watermarking in Uncompressed Domain

The system block diagrams in Simulink™ are [12]:



(a) Top level block set diagram.



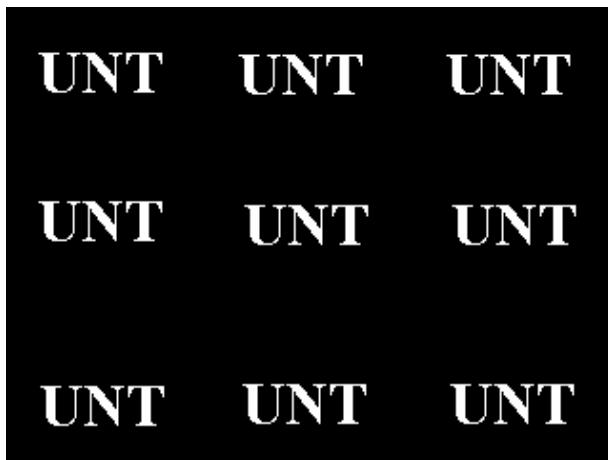
(b) Block set inside "Encoder" in (a).

Figure (5.1) Simulink™ system block set diagram for MPEG watermarking in uncompressed domain.

From Figure (5.1) (b), the video frames are watermarked at DCT domain before being compressed. For the three Y, Cb and Cr color frames, only Y color frame is watermarked for the following reasons:

- The watermark image which is black-white monochrome or gray scale should only modify brightness of picture. If the watermark is color, Cb and Cr must be watermarked as well.
- Y color space is more sensitive to human perception such that any unauthorized modification is easily detected so that it makes watermarking Y color frames ideal for copyright protection.
- To avoid too much redundancy added to frames, the watermark is not embedded into Cb or Cr.

To protect against frame interpolating attacks on watermarking, all I, B, P frames must embed the watermark. The results of watermarking on uncompressed frames are:



(a) Watermark image 1.



(b) Watermark image 2.



(c) Watermarked video 1 with image 1.



(d) Watermarked video 1 with image 2.



(e) Watermarking video 2 with image 1.



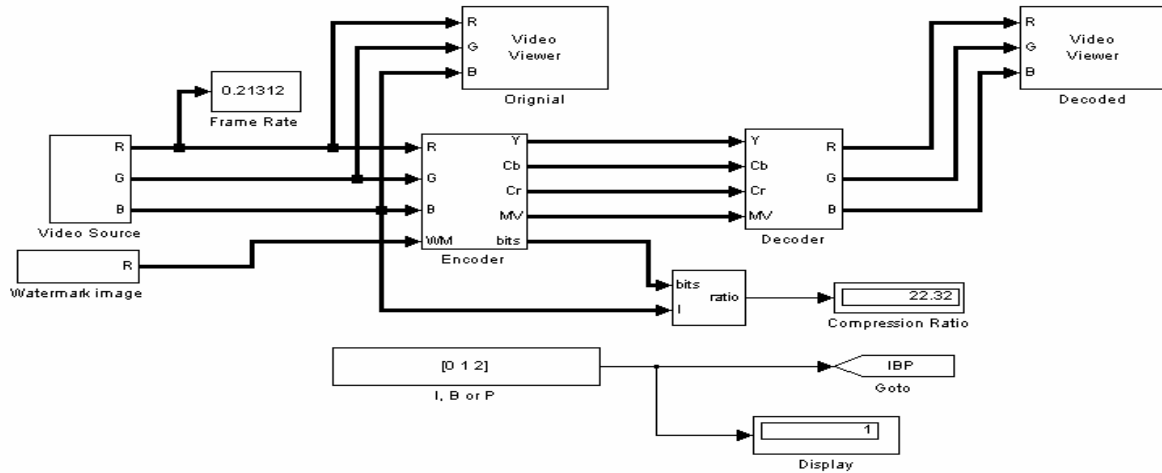
(f) Watermarking video 1 with image 2.

Figure (5.2) Watermarking in uncompressed domain results (resolution 240X320).

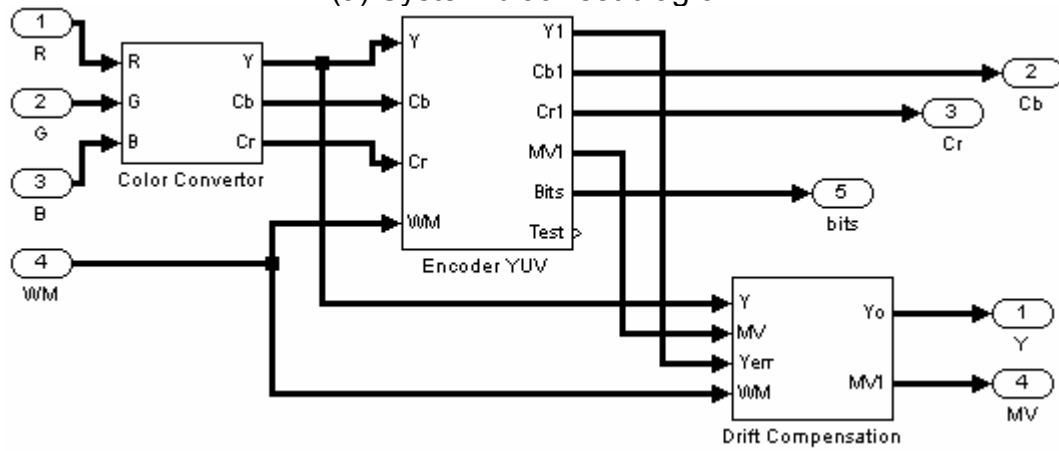
In testing, two different types of watermark images are considered: one is small size font but covers different locations as (a) in Figure (5.2); one is big size font but covers only one location as (b) in Figure (5.2). Similarly, two different types of video clips are under testing: one is a complex but slowly changing scene as (c) and (d) in Figure (5.2); one is a simple but quickly changing scene as (e) and (f) in Figure (5.2). The same testing methodology is applied to other tests during the design.

5.1.3 Modeling Watermarking in Compressed Domain

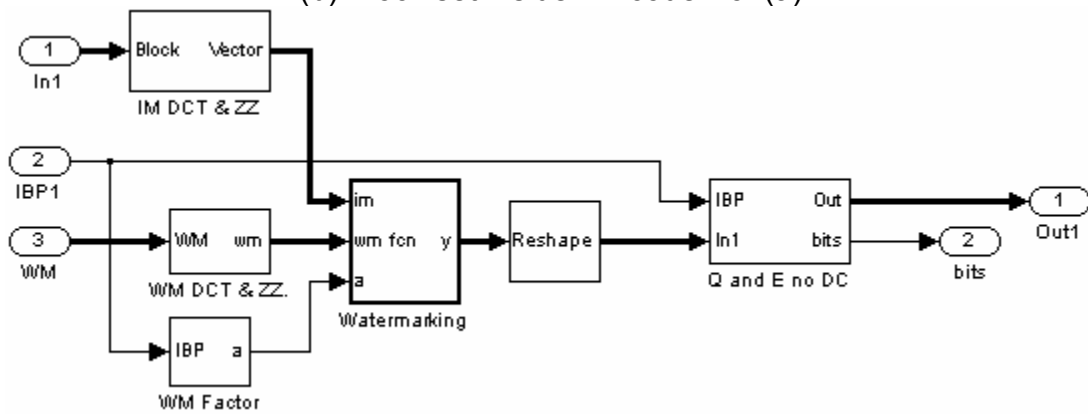
The system block sets in Simulink™ are [12]:



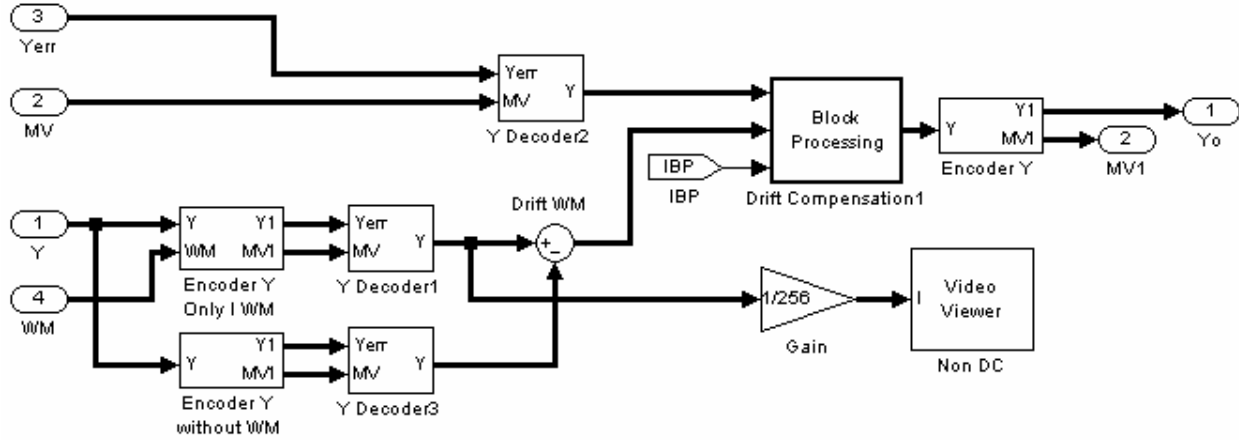
(a) System block set diagram.



(b) Block set inside "Encoder" of (a).



(c) Watermark embedding block set inside "Encoder YUV" in (b).



(d) Drift compensation block set inside “Drift Compensation” in (b).

Figure (5.3) Simulink™ system block set diagram for MPEG watermarking in compressing domain.

The watermarking block in Figure (5.3) (c) embeds the watermark in all I, B and P frames. As estimate, the watermark in I frame also appears in B and P because they are predicted from I frame. It will result in two watermarks in non-intra frames. The watermark predicted from I frame will drift if it overlaps with moving objects in the scene. So the drift compensation is applied to cancel the B and P’s watermark predicted from I frame. In Figure (5.3) (d), the block “Encoder Y only I WM” compresses the original video and watermarks I frame only. Another block “Encode Y without WM” just compresses original video, but does not embed watermark. The two encoders’ difference is the drifting watermark. After decoding two video compression codes, the drifting watermark can be extracted by subtracting above two videos. The “Drift Compensation1” block cancels the drifting watermark on B and P by subtracting. From the above description, the conclusion is that above drift compensation works at spatial domain.

The video compression and watermarking in compressed domain with drift compensation results are:



(a) No drift compensation.



(b) Drift compensation.



(c) No drift compensation.



(d) Drift compensation.



(e) No drift compensation.



(f) No drift compensation.



(g) Drift compensation.



(h) Drift compensation.

Figure (5.4) Watermarking in compressed domain results (resolution 240X320).

Comparing two video clips and two watermark images in uncompressed domain or in compressed domain, and with or without drift compensation, the result demonstrates that the drift compensation cancels the drifting watermark effect, especially for quickly moving objects.

But one phenomenon is also observed: if the moving object is very fast, while drift compensation canceling the drifting watermark, it also causes another side effect of blur shape moving object or even totally erased area. It is shown as figure follows:



(a) Stable watermark but moving bird blur. (b) Drifting watermark but moving bird clear.

Figure (5.5) Side effect of drift compensation of blur moving object.

This phenomena is not observed in intra frames, only inter (P and B) frames. By monitoring extracted watermark, no extra video object is found with the extracted watermark. The suspicious one could be motion estimate failure.

5.2 System Level Modeling with VHDL and FPGA Performances

Unlike previous modeling with MATLAB/Simulink™, the high-level synthesis in FPGA applies a different bottom-up methodology: the low level components are built and verified, then, with functional components, the whole system is created. At this

system-level prototyping development, the video compression and watermarking working module are implemented in FPGA with VHDL. The modules are controller, frame buffer, watermark buffer, DCT/IDCT, quantization, zigzag, Huffman coding and watermarking.

5.2.1 Controller Performance

The controller generates address and control signals to synchronize other components. It is a finite state machine and its states are:

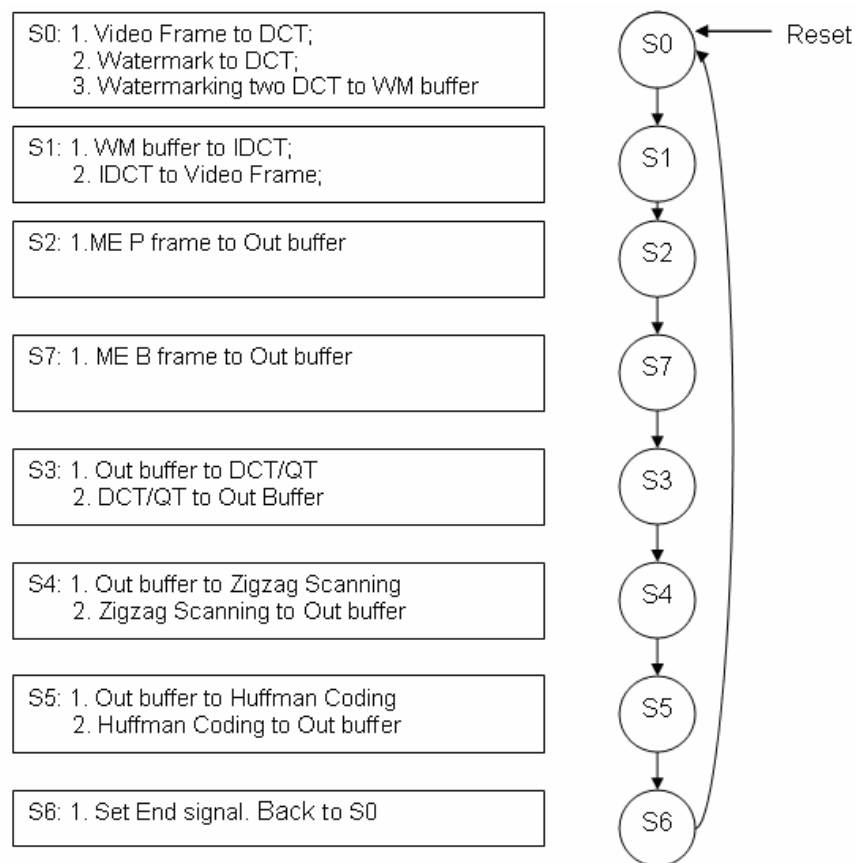


Figure (5.6) Controller's FSM states diagram.

If using traditional FSM design, the controller has more states than those in above Figure (5.6) because the video compression and watermarking procedures are complicated. The solution is to merge several sub-states into one state, however, the

inside structure of each state become complex. The simulation of the controller by Altera Quartus II is:

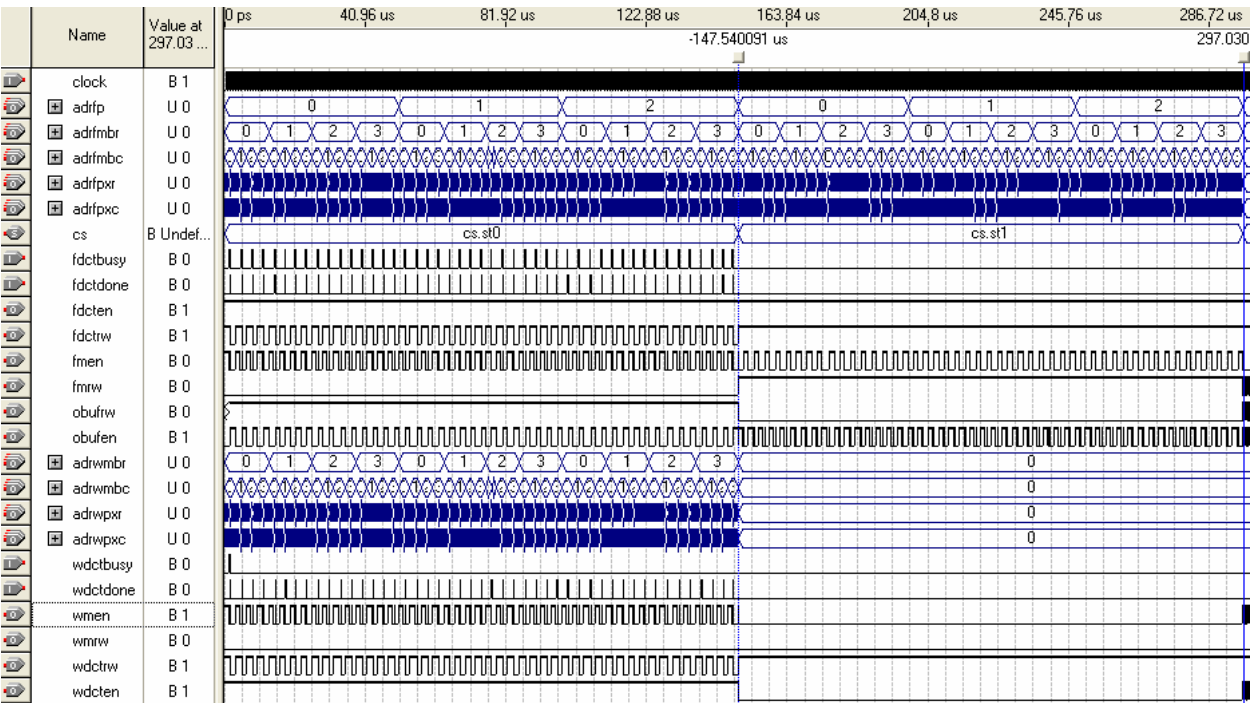


Figure (5.7) Controller simulation. S0 and S1 for 297us in clock 50Mhz.

5.2.2 2-D DCT Performance

The 2-dimensioal DCT is implemented with Loeffler’s fast 1-dimensional DCT algorithm [16]. The simulation in Xilinx© ISE is as:

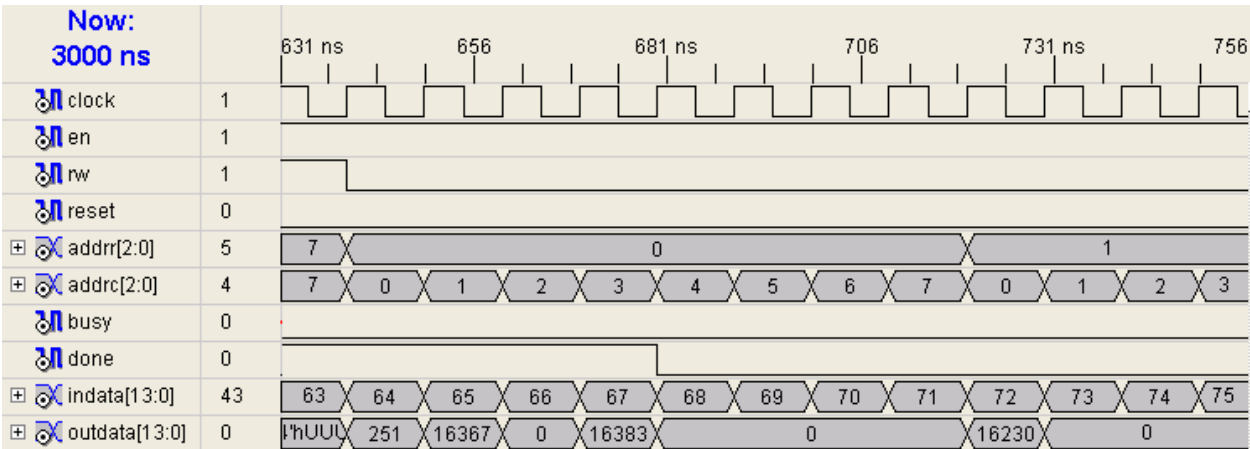


Figure (5.8) 2D DCT simulation. Total processing time: 1281ns in clock 100Mhz.

The simulation result comparing with MATLAB™ function dct2:

Table (5.1) Comparison of first 20 coefficients of simulation and MATLAB™ dct2.

<i>Index</i>	<i>dct2</i>	<i>Simulation</i>	<i>Index</i>	<i>dct2</i>	<i>Simulation</i>
0	252	251	10	0	0
1	-18.2	-17	11	0	0
2	0	0	12	0	0
3	-1.9	-1	13	0	0
4	0	0	14	0	0
5	-0.56	0	15	0	0
6	0	0	16	0	0
7	-0.14	0	17	0	0
8	-145.78	-152	18	0	0
9	0	0	19	0	0

5.2.3 Motion Estimation

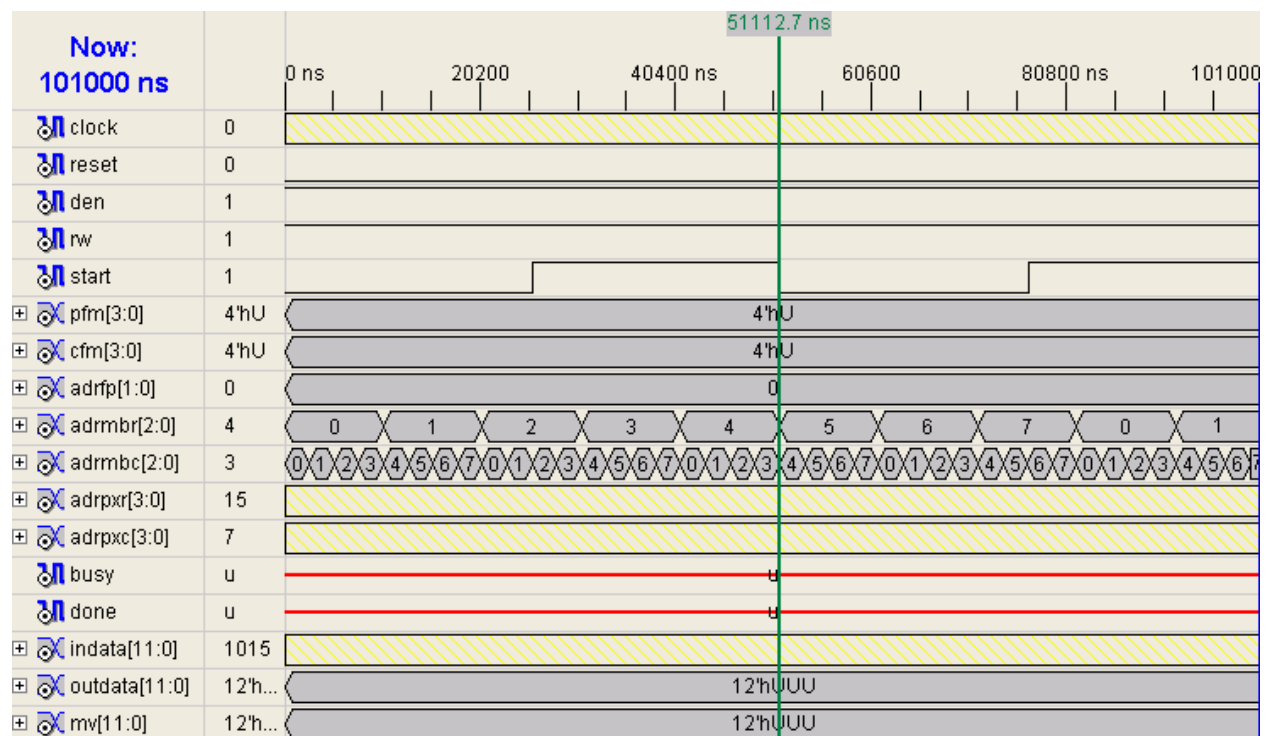


Figure (5.9) Motion estimate simulation. Total processing time: 51112.7ns in 100Mhz.

5.2.4 Quantization Performance

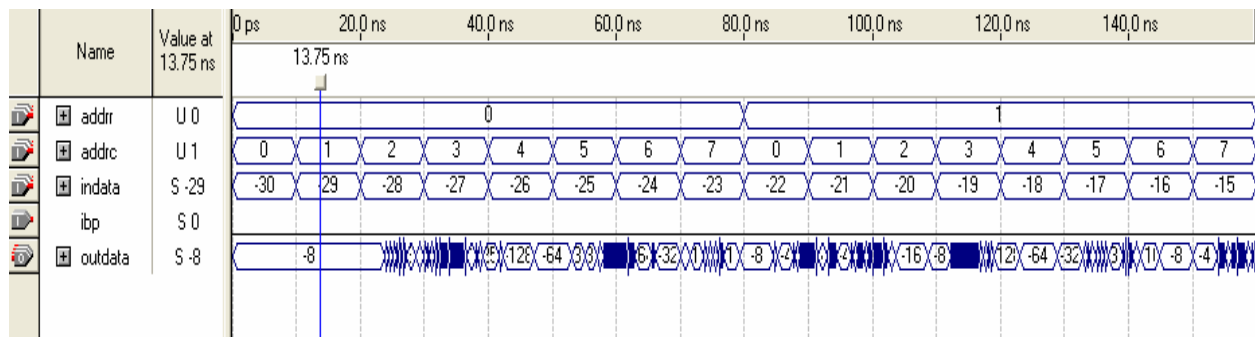


Figure (5.10) Quantization simulation.

5.2.5 Zigzag Scanning Performance

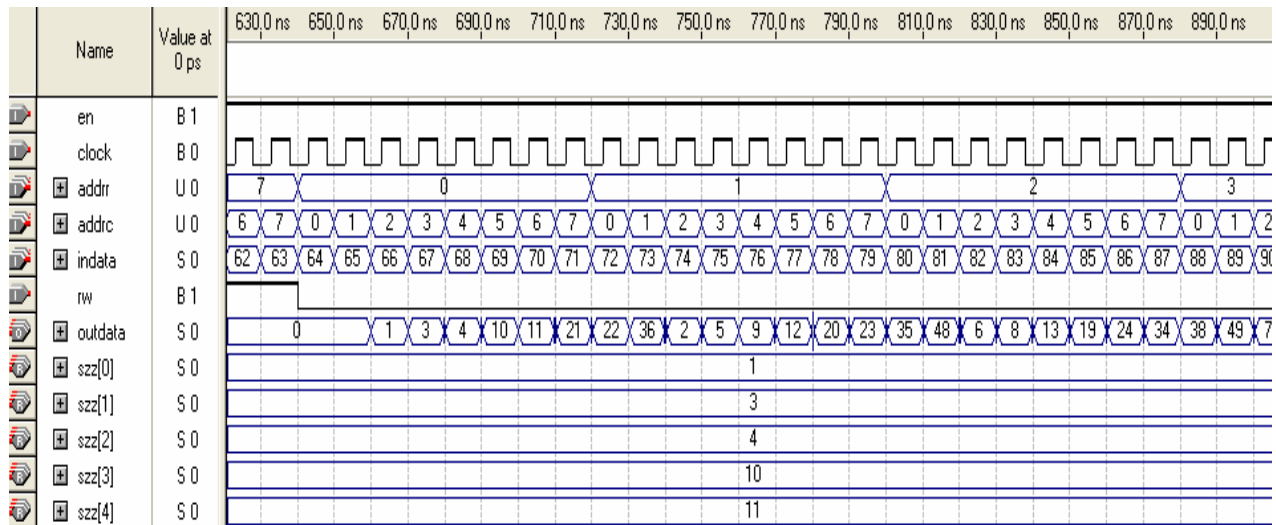


Figure (5.11) Zigzag scanning simulation. Total processing time: 1281ns in clock 100Mhz.

The VHDL compilation report of components by Altera© Quartus II™ is:

Table (5.2) Compilation and timing report of 128X128 Y frame processing in 100Mhz clock.

Component	Logic elements	Registers	Pins	Multipliers	Time(ns)
Controller	588	157	69	0	75+30X2+150X3
2D DCT X 4	80459X4	1006X4	40X4	70X4	1281ns
Quantization	2363	0	31	1	0
Zigzag	1028	780	35	0	1281
Watermark	24	37	0	0	0
Frame buffers	7701	6156	43	0	0

Motion vector buffer	667	520	31	0	0
Watermark buffer	4043	3048	41	0	0
RGB to YCbCr	1501	0	48	0	0
Motion estimate	n/a	n/a	n/a	n/a	51112+4194304
Total	339754	14722	457	281	4248563

The above FPGA compilation and timing report is generated by Altera© Quartus II™ high level simulation and synthesis IDE tools Quartus II™. The module is DE2 Cyclone II™ module board. The clock 27 MHz is applied to verify FPGA performance.

5.3 Discussions

With the performance of above functional components and integrated system, the whole all performance of system is estimated. The video quality metrics are applied to verify the system's performance.

5.3.1 The Video Quality of Video Compression and Watermarking

The video quality metrics are the RGB color image mean square error (MSE) [3] and peak-signal-noise-ratio (PSNR) as equations [2]:

$$MSE = \frac{\sum_{m=1}^M \sum_{n=1}^N \sum_{k=1}^3 |p(m,n,k) - q(m,n,k)|^2}{3MN}, \quad (5.1)$$

$$PSNR = 10 \log_{10} \left(\frac{(2^i - 1)^2}{MSE} \right). \quad (5.2)$$

Where, m is the image pixel row from 1 to M , n is the image pixel column from 1 to N , and k is from 1 to 3 as RGB color pixels. $p(m,n,k)$ and $q(m,n,k)$ are images' pixels after and before processing. i is the bit length of image pixel, in common RGB 24-bits digital video system, it is 8. From the above MSE and PSNR equations, the quality metrics results of video compression and watermarking in the working model are:

Table (5.3) Video quality metrics of video compression and watermarking.

<i>Video processing type</i>	<i>PSNR (dB)</i>	<i>MSE</i>	<i>Compression ration</i>		
			<i>Average</i>	<i>Range</i>	<i>Estimate</i>
Video compression only	30	71	27	(16~39)	16
Video compression and watermarking in uncompressed domain	20	616	26.7	(15~38)	16
Video compression and watermarking in compressed domain	19	812	26	(15~38)	16

The criteria of video quality are: PSNR between 40dB to 50dB, the noise is beyond human perception; 10dB to 20dB, the noise can be detected by human visual system [13]. The video compression working module has PSNR 30dB, which implies that the current implementation of MPEG video compression generates noticeable noises. The procedures of MPEG video compressions are lossy unnoticeable color space sample rate compression, the motion estimate with great errors, lossy DCT compression, quantization with noise by different step sizes, and lossless Huffman coding. More improvement should come from the motion estimation and the quantization procedures.

The PNSR of watermarking at uncompressed and compressed domain is about 20dB. It satisfies the fact that the watermarks are visible. The 1dB difference in PNSR for two different watermarking schemes indicates that they are effectively same in watermarking even though their complexities are different. The extra complexity of the watermarking in compressing is caused by the drift compensation.

The video compression rate is contributed from two categories, the constant one like 4:2:0 color space sample rate whose compression rate is always 2:1, and the content adaptive compression whose compression rate is variable and depends on its content data in the motion estimation, DCT coefficients quantization and Huffman coding procedures. Here the variable compression rate is estimated as: assume half

DCT coefficients are truncated so the compression rate is 2:1. The redundancy of two frames is removed in 75% by the motion estimate or compression rate is 4:1, and in the working module, one GOP is constituted with one I frame, one B frame and 1 P frame or IBP structure. The motion estimation compression could be $(1+1+1)/(1+1/4+1/16) \approx 2:1$. The DCT coefficients quantization and Huffman coding could have compression rates are 2:1. So the estimated average compression rate of the video compression working module is: $2 \times 2 \times 2 \times 2 = 16:1$. The observed average compression rate in the experiment is 26:1. To achieve higher compression rate, one way is to interpolate more B frames and more P frames in one GOP. After tuning, the average compression rate could be greater than 100:1.

5.3.2 Physical and Timing Analyzing.

From Table (5.2), the physical and time parameters of the working module are estimated by simply adding extra Cb and Cr processing. Since the motion estimation and watermarking only occur in Y color space, the physical structure of Y processing is more complicated than Cb and Cr such that it is safe to estimate the total logic elements by tripling with Y color processing branch. However, the Y, Cb and Cr processing are concurrent; the time delay in Y color processing could be considered as the total delay in the whole working module. The physical and timing results of the working module are:

Table(5.4) Physical and timing results for 128X128 YCbCr frames at 400Mhz.

<i>Elements</i>	<i>Registers</i>	<i>Multipliers</i>	<i>Time(us)</i>	<i>Frame/s</i>
1,019,262	44,166	843	1,063	940

Above result is upon the ideal assumption that there is no physical delay in logic gates, but only processing delay. The total elements are a great amount because only the high-level modeling and simulation in behave has been achieved. The structure and

performance need to be optimized in algorithms and RTL level. If the model is utilized in resolution 720X486 applications, like NTSC television video broadcasting system, the working model processing speed could reach 44 frame/s, which exceeds with the required 29.97 frame/s.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

The working module has demonstrated the basic algorithms of MPEG-4 visual part advanced simple profile. Two visible watermarking schemes, watermarking in uncompressed domain and compressed domain, have been proved achieving similar visual output result (difference in 1dB), however, the compressed domain watermarking is more complex in structure than the uncompressed one. The similar result in the two working modules is based on the fact that video compression processing with PSNR 30 dB; otherwise, the watermarking in uncompressed domain could have better PSNR than that in compressed domain.

The uncompressed and compressed domain watermarking also demonstrate the robust of DCT domain watermarking because after every compression procedures, the watermark is still noticeable and integrity.

The simulation reports of working module with FPGA performance confirm that 2-D DCT/IDCT is more complex in structure, and motion estimation has the greatest time delay. If not considering physical constraints, the encoding speed could satisfy the standard real time NTSC video encoding/watermarking applications in clock 400Mhz, but the video encoding PSNR 30 dB is still lower than normal requirement 40~50 dB.

6.2 Future Work

The present model just implements basic MPEG and watermarking algorithms, further optimization needs to utilize for minimize physical parameters like logic gates

number in RTL level and pipeline to reduce time delay. The robust of watermark should be tested further by emulating more different attacks attempting to remove watermark.

6.2.1 MPEG-4 Video Compression

The prototype only implements the integer pixel motion estimation, to reach the goal of a fine resolution as MPEG-4, the half and quarter- pixel motion estimation must be implemented. The motion estimation is exhaustive square searching, but in MPEG-4, it is diamond 3-steps algorithm which can greatly improve searching speed. The prototype module only has an average compression rate 26:1 because an IBP GOP model is applied, but most commercial video compression could be more than 100:1. To gain higher compression rate, the GOP could be interpolated with more B and P frames. In the prototyping module, the video compressed stream is raw, unformatted, however, if a standard MPEG-4 decoder is a received ender, the prototyping module must generate MPEG-4 stream headers.

6.2.2 Watermarking

The prototyping module only achieves the basic watermarking embedding at DCT domain, and it could be fragile under some attacks [53], [56], [58], [59], [60], [61] and [62]. More watermarking algorithms could be considered for copyright protection such as [53], [54], [56], [57], [59], and [62].

The watermarking techniques discussed in this paper also can embed color or animation watermark even though just an implement of still monochrome watermark image is discussed for simplifying reasons.

The error of blur or even disappearing fast moving video object after the drift compensation could be caused by failure of motion estimate. Further testing should be conducted, and a solution upon testing will be proposed.

6.2.3 Hardware Implementation

The working module FPGA performance is investigated with simulation. More optimization on the lower RTL levels and physical structures are needed.

REFERENCES

- [1] www.xvid.org
- [2] Iain E.G. Richardson, *H.264 and MPEG-4 Video Compression*, John Wiley & Sons, Ltd. England, USA, 2003.
- [3] Jie Chen, Ut-Va Koc Koc, and K.J. Ray Liu, *Design of Digital Video Coding Systems – A Complete Compressed Domain Approach*, Marcel Dekker, Inc. New York, USA, 2002.
- [4] S. P. Mohanty, *Digital Watermarking: A Tutorial Review*,
<http://www.csee.usf.edu/~smohanty/research/Reports/WMSurvey1999Mohanty.pdf>
- [5] S. Emmanuel, M.S. Kankanhalli, “Security and Copyright Protection for Broadcast Video”, *International Conference on Multimedia Modeling (MMM 2000)*, pp. 123-140, Nagoya, Japan, Nov 2000.
- [6] W.Q. Yan and M.S. Kankanhalli, “Erasing Video Logos Based on Image Inpainting”, *Proc. IEEE International Conference in Multimedia and Expo (ICME 2002)*, Lausanne, August 2002.
- [7] W. Q. Yan, J. Wang, and M.S. Kankanhalli, “Automatic Video Logo Detection and Removal”, *ACM/Springer-Verlag Multimedia Systems Journal*, Volume 10, No. 5, 2005, pp. 379–391.
- [8] M. Awrangjeb and M.S.Kankanhalli, “Reversible Watermarking Using a Perceptual Model”, *Journal of Electronic Imaging*, Vol. 14, No. 1, Jan-Mar 2005.
- [9] W.Q. Yan and M.S. Kankanhalli, “Detection and Removal of Lighting & Shaking Artifacts in Home Video”, *Proc. ACM International Conference on Multimedia (ACMMM 2002)*, Juan Les Pins, France, December 2002.

- [10] www.altera.com
- [11] www.xilinx.com
- [12] www.mathworks.com
- [13] C. John Russ. *The Image Processing Handbook*. 4th Edition, CRC Press. USA, 2002.
- [14] Joan L. Mitchell, William B. Pennebaker, Chad E. Fogg, and Didier J. LeGall. *MPEG Video Compression Standard*, pages 33-49, Chapman & Hall, New York, USA, 1996.
- [15] W. W. A. Chen, C. Harrison, and S. C. Fralick, "A Fast computational Algorithm for the Discrete Cosine Transform," *IEEE Transactions on Communications*, Vol. COM-25, No. 9, pp. 1004-1011, Sept. 1977.
- [16] C. Loeffler, A. Lightenberg, and G. Moschytz, "Practical fast 1-D DCT algorithms with 11 multiplications," *Proc. IEEE ICASSP*, vol. 2, pp.988–991, Feb. 1989.
- [17] C.-Y. Pai et al., "Low-power data-dependent 8x8 DCT/IDCT for video compression", *IEEE Pmc.-vis. Image Signal Process.*, Vol. 150, Iss. 4, pp. 245 – 255, 22 Aug. 2003.
- [18] D. Huffman, "A method for the construction of minimum redundancy codes", *Proc. Of the IRE*, 40, pp. 1098-1101, 1952.
- [19] C.E. Shannon. *The Mathematical Theory of Communication*. The University of Illinois Press, 1949.
- [20] F. Hartung, B. Girod, "Watermarking of uncompressed and compressed video", *Signal Processing*, v 66, n 3, May 1998, p 283-301

- [21] Stefan Katzenbeisser, Fabien, A.P. Petitcolas, *Information Hiding Techniques for Steganography and Digital Watermarking*, Idea Group, USA, 2000.
- [22] Michael Arnold, Martin Schmucker, Stephen D. Wolthusen, *Techniques and Applications of Digital Watermarking and Content Protection*, Artech Hosue, Inc, 2003, England.
- [23] S. P. Mohanty, K. R. Ramakrishnan, and M. S. Kanakanhalli, "A Dual Watermarking Technique for Images", in *Proceedings of the 7th ACM International Multimedia Conference (ACMMM) (Vol. 2)*, pp.49-51, 1999.
- [24] S. P. Mohanty, K. R. Ramakrishnan, and M. S. Kanakanhalli, "A DCT Domain Visible Watermarking Technique for Images", in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME) (Vol. 2)*, pp.1029-1032, 2000.
- [25] P. Premaratne, C.C., "A novel watermark embedding and detection scheme for images in DFT domain", *Image Processing and Its Applications*, 1999. *Seventh International Conference on (Conf. Publ. No. 465) (Vol. 2)*, pp.780-783, 1999.
- [26] X. G. Xia, C. Boncelet, and G. Arce, "Wavelet transform based watermark for digital images," *Opt. Express* 3, pp.497-511, 1998.
- [27] O. B. Adamo, S. P. Mohanty, E. Kougianos, M. Varanasi, and W. Cai, "VLSI Architecture and FPGA Prototyping of a Digital Camera for Image Security and Authentication", in *Proceedings of the IEEE Region 5 Technology and Science Conference*, pp. 154-158, 2006.

- [28] S. P. Mohanty, K. R. Ramakrishnan, and M. S. Kanakanhalli, "An Adaptive DCT Domain Visible Watermarking Technique for Protection of Publicly Available Images", in *Proceedings of the International Conference on Multimedia Processing and Systems (ICMPS)*, pp.195-198, 2000.
- [29] Y.H. Wu, X. Guan, M.S. Kankanhalli, Z.Y. Huang, "Robust Invisible Watermarking of Volume Data", *27th International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2000)*, Sketches & Applications, pp. 252, July 2000.
- [30] S. P. Mohanty, N. Ranganathan, and R. K. Namballa, "VLSI Implementation of Invisible Digital Watermarking Algorithms Towards the Development of a Secure JPEG Encoder", *Proceedings of the IEEE Workshop on Signal Processing System (SIPS)*, pp.183-188, 2003.
- [31] S. P. Mohanty, N. Ranganathan, and R. K. Namballa, "VLSI Implementation of Visible Watermarking for a Secure Digital Still Camera Design", *Proceedings of the 17th IEEE International Conference on VLSI Design (VLSID)*, pp.1063-1068, 2004
- [32] S. P. Mohanty, N. Ranganathan, and R. K. Namballa, "A VLSI Architecture for Visible Watermarking in a Secure Still Digital Camera (S²DC) Design", *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, Vol. 13, No. 7, July 2005, pp. 808-818
- [33] S. P. Mohanty, N. Ranganathan, and K. Balakrishnan, "A Dual Voltage-Frequency VLSI Chip for Image Watermarking in DCT Domain", *IEEE Transactions on Circuits and Systems II (TCAS-II)*, Vol. 53, No. 5, May 2006, pp. 394-398.

- [34] M. Bansal, W.Q. Yan and M.S. Kankanhalli, "Dynamic Watermarking of Images", *Proc. IEEE Pacific-Rim Conference On Multimedia (PCM 2003)*, Singapore, December 2003.
- [35] M. Barni, F. Bartolini, V. Cappellini, A. Piva, "A DCT-domain system for robust image watermarking". *Signal Processing*, v 66, n 3, May 1998, p 357-72.
- [36] S. Pranata, Y.L. Guan, H.C. Chua, "Improved bit rate control for real-time MPEG watermarking", *2004 International Conference on Image Processing (ICIP) (IEEE Cat. No.04CH37580)*, 2004, pt. 4, p 2619-23 Vol. 4
- [37] K. Wiatr, P. Russek, "Embedded zero wavelet coefficient coding method for FPGA implementation of video codec in real-time systems", *Proceedings International Conference on Information Technology: Coding and Computing (Cat. No.PR00540)*, 2000, p 146-51
- [38] A. Piva, R. Caldelli, A. De Rosa, "A DWT-based object watermarking system for MPEG-4 video streams", *Proceedings 2000 International Conference on Image Processing (Cat. No.00CH37101)*, 2000, pt. 3, p 5-8 vol.3
- [39] D. He, Q. Sun, Q. Tian, "A secure and robust object-based video authentication system", *EURASIP Journal on Applied Signal Processing*, v 2004, n 14, 15 Oct. 2004, p 2185-200.
- [40] S. Biswas, S.R. Das, E.M. Petriu, "An adaptive compressed MPEG-2 video watermarking scheme", *IEEE Transactions on Instrumentation and Measurement*, v 54, n 5, Oct. 2005, p 1853-61
- [41] Tian-Hang Chen, Shao-Hui Liu, Hong-Xun Yao, Wen Gao, "Robust video watermarking based on DC coefficients of selected blocks", *Proceedings of*

- 2005 International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 05EX1059)*, 2005, pt. 9, p 5273-8 Vol. 9
- [42] Mauro Barni, Franco Bartolini, Nicola Checcacci, "Watermarking of MPEG-4 video objects", *IEEE Transactions on Multimedia*, v 7, n 1, February, 2005, p 23-31
- [43] M. J. Garrido, C. Sanz, M. Jimenez, J.M. Meneses, "An FPGA implementation of a flexible architecture for H.263 video coding" *2002 Digest of Technical Papers. International Conference on Consumer Electronics (IEEE Cat. No.02CH37300)*, 2002, p 274-5
- [44] Jianhao Meng, Shih-Fu Chang, "Embedding visible video watermarks in the compressed domain", *Proceedings 1998 International Conference on Image Processing. ICIP98 (Cat. No.98CB36269)*, 1998, pt. 1, p 474-7 vol.1
- [45] L. Qiao, K. Nahrstedt, "Watermarking Methods for MPEG Encoded Video: Towards Resolving Rightful Ownership", *1998 IEEE International Conference on Multimedia Computing and Systems (ICMCS'98)*, 1998, p. 276
- [46] Wei Zhang, Sen-Ching S. Cheung, Minghua Chen, "Hiding privacy information in video surveillance system", *Proceedings - International Conference on Image Processing, ICIP*, v 3, *IEEE International Conference on Image Processing 2005, ICIP 2005*, 2005, pp 868-871
- [47] Wen-Nung Lie, Guo-Shiang Lin, Ta-Chun Wang, "Digital watermarking for object-based compressed video", *ISCAS 2001. The 2001 IEEE International Symposium on Circuits and Systems (Cat. No.01CH37196)*, 2001, pt. 2, p 49-52 vol. 2

- [48] M.S. Kankanhalli and T.G. Teo, "Compressed Domain Scrambler/Descrambler for Digital Video", *IEEE Transactions on Consumer Electronics*, Vol. 48, No. 2, pp. 356-365, May 2002.
- [49] S. Emmanuel and M.S. Kankanhalli, "A Digital Rights Management Scheme for Broadcast Video", *ACM Multimedia Systems Journal*, Vol. 8, No. 6, pp. 444-458, 2003.
- [50] E.C. Chang, M.S. Kankanhalli, X. Guan, Z.Y. Huang and Y.H. Wu, "Robust Image Authentication Using Content-based Compression", *ACM Multimedia Systems Journal*, Vol. 9, No. 2, pp. 121-130, 2003.
- [51] C.M. Chew and M.S. Kankanhalli, "Compressed Domain Summarization of Digital Video", *Proc. Second IEEE Pacific-Rim Conference on Multimedia (PCM 2001)*, Beijing, October 2001.
- [52] S. P. Mohanty, R. Kumara C., and S. Nayak, "FPGA Based Implementation of an Invisible-Robust Image Watermarking Encoder", *Lecture Notes in Computer Science (LNCS), CIT 2004*, Springer-Verlag, Vol. 3356, pp. 344-353, 2004.
- [53] S.D. Lin, Chin-Feng Chen, "A robust DCT-based watermarking for copyright protection", *IEEE Transactions on Consumer Electronics*, v 46, n 3, Aug. 2000, p 415-21.
- [54] P.A.A. Assuncao, M. Ghanbari, "Transcoding of MPEG-2 video in the frequency domain", *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing (Cat. No.97CB36052)*, 1997, pt. 4, p 2633-6 vol.4.

- [55] L. Agostini, I. Silva, and S. BAMPI. "Pipelined Fast 2-D DCT Architecture for JPEG Image Compression", *SBCCI2000 – XIII Symposium on Integrated Circuits and System Design*, GO-Brazil, 2001.
- [56] Seong-Whan Kim, Hyun Jin Park, Hyunseong Sung, "A natural modification of autocorrelation based video watermarking scheme using ICA for better geometric attack robustness", *Lecture Notes in Computer Science*, v 3611, n PART II, *Advances in Natural Computation: First International Conference, ICNC 2005. Proceedings*, 2005, p 451-460.
- [57] Guo-Zua Wu, Yi-Jung Wang, Wen-Hsing Hsu, "Robust watermark embedding/detection algorithm for H.264 video", *Journal of Electronic Imaging*, v 14, n 1, Jan. 2005, p 13013-1-9.
- [58] P. Vinod, P.K. Bora, "Motion-compensated inter-frame collusion attack on video watermarking and a countermeasure", *IEE Proceedings-Information Security*, v 153, n 2, 12 June 2006, p 61-73.
- [59] K. Su, D. Kundur, D. Hatzinakos, "A content dependent spatially localized video watermark for resistance to collusion and interpolation attacks", *Proceedings 2001 International Conference on Image Processing (Cat. No.01CH37205)*, 2001, pt. 1, p 818-21 vol.1.
- [60] B. Vassaux, P. Nguyen, S. Baudry, P. Bas, J.-M. Chassery, "Survey on attacks in image and video watermarking", *Proceedings of the SPIE - The International Society for Optical Engineering*, v 4790, 2002, p 169-79.

- [61] Jiang Du, Choong-Hoon Lee, Heung-Kyu Lee, Youngho Suh, "BSS: a new approach for watermark attack", *Proceedings Fourth International Symposium on Multimedia Software Engineering*, 2002, p 182-7.
- [62] Chun-Shien Lu, Jan-Ru Chen, Kuo-Chin Fan, "Resistance of content-dependent video watermarking to watermark-estimation attacks", *2004 IEEE International Conference on Communications (IEEE Cat. No.04CH37577)*, 2004, pt. 3, p 1386-90 Vol.3.
- [63] Iain E.G. Richardson. H.264 and MPEG-4 Video Compression, John Wiley & Sons, Ltd. England, USA, 2003.
- [64] Chun-Shien Lu, *Multimedia Security: Steganography and Digital Watermarking Techniques for Protection of Intellectual Property*, Idea Group Inc, 2005, USA.
- [65] M.S. Kankanhalli and K.F. Hau, "Watermarking of Electronic Text Documents", *Electronic Commerce Research*, Vol. 2, No. 1/2, pp. 169-187, Kluwer Academic Publishers, 2002.
- [66] S. Emmanuel and M.S. Kankanhalli, "Mask-based Interactive Watermarking Protocol for Video", *Proc. SPIE International Symposium on the Convergence of Information Technologies and Communications (ITCOM 2001)*, Denver, August 2001.
- [67] S. Emmanuel and M.S. Kankanhalli, "Mask-based Fingerprinting Scheme for Digital Video Broadcasting", *Multimedia Tools and Applications*, 2006.
- [68] Thomas Sikora, "The MPEG-4 Video Standard Verification Model", *IEEE Transactions on Circuits and Systems for Video Technology* Vol.7, No.1, 1997, p19-31.

- [69] X.D. Sun, M.S. Kankanhalli, Y.W. Zhu and J.K. Wu, "Content-based Representative Frame Extraction for Digital Video", *Proc. IEEE International Conference on Multimedia Computing and Systems (ICMCS 1998)*, pp. 190-193, Austin, Texas, July 1998.
- [70] M.S. Kankanhalli, Rajmohan and K.R. Ramakrishnan, "Content-based Watermarking of Images", *Proc. of The 6th ACM International Multimedia Conference*, pp. 61-70, Bristol, UK, September 1998.
- [71] M.S. Kankanhalli, Rajmohan and K.R. Ramakrishnan, "Adaptive Visible Watermarking of Images", *IEEE International Conference on Multimedia Computing and Systems (ICMCS 1999)*, Florence, Italy, June 1999.
- [72] T.S. Chua, Y. Lin and M.S. Kankanhalli, "A General Framework for Video Segmentation based on Temporal", *Multi-resolution Analysis, Proc. International Workshop on Advanced Image Technology (IWAIT 2000)*, pp. 119-124, Fujisawa, Japan, January 2000.
- [73] S. Emmanuel, M.S. Kankanhalli, "A System for Security and Copyright Protection in Broadcast Video", *Proc. International Workshop on Advanced Image Technology (IWAIT 2001)*, Taejon, Korea, February 2001.
- [74] S. Emmanuel and M.S. Kankanhalli, "Copyright Protection for MPEG-2 Compressed Broadcast Video", *IEEE International Conference on Multimedia and Expo (ICME2001)*, Tokyo, Japan, August 2001.
- [75] P.K. Atrey, W.Q. Yan and M.S. Kankanhalli, "A Scalable Signature Scheme for Video Authentication", *Multimedia Tools and Applications*, 2007.

- [76] M. Awrangjeb and M.S. Kankanhalli, "Lossless Watermarking Considering the Human Visual System", *Proc. International Workshop on Digital Watermarking (IWDW 2003)*, Seoul, October 2003.
- [77] Mauro Barni, Franco Bartolini, *Watermarking Systems Engineering Enabling Digital Assets Security and Other Applications*, Marcel Dekker Inc, New York, USA, 2004.