

KOCAELI UNIVERSITY  
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION ENGINEERING



MASTER'S THESIS

IMPLEMENTATION OF 8x8 LUMA INTRA PREDICTION MODULE FOR H.264/AVC  
STANDARD.

BUNJI ANTOINETTE RINGNYU

KOCAELI 2019

**KOCAELI UNIVERSITY  
GRADUATE SCHOOL OF NATURAL AND APPLIED  
SCIENCES**

**DEPARTMENT OF ELECTRONICS AND COMMUNICATIONS  
ENGINEERING**

**MASTER'S THESIS**

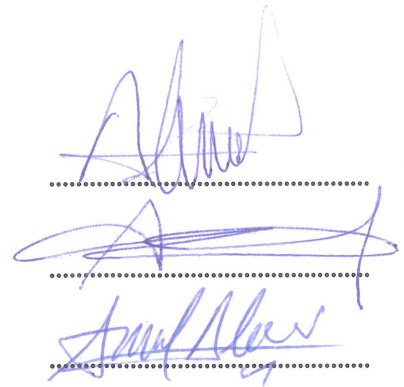
**IMPLEMENTATION OF 8x8 LUMA INTRA PREDICTION  
MODULE FOR H.264/AVC STANDARD.**

**BUNJI ANTOINETTE RINGNYU**

**Prof.Dr. Ali TANGEL  
Supervisor, Kocaeli Üniv.**

**Prof.Dr. Aşkın DEMİRKOL  
Jury Member, Sakarya Üniv.**

**Assist.Prof.Dr. Anıl ÇELEBİ  
Jury Member, Kocaeli Üniv.**



**Thesis Defense Date: 15.01.2019**

## ACKNOWLEDGEMENTS

Firstly, I will like to thank The Turkish Government and Türkiye Bursları Organisation (YTB) for giving me the opportunity to study in Turkey, gain enormous experience and knowledge about new technologies. In addition, I will like to thank the entire staff of the Department of Electronics and Communication Engineering, Kocaeli University for their tireless efforts in imparting knowledge to, and exposing us to new Technology.

Secondly, I will like to thank my supervisor, Prof.Dr. Ali Tangel, for helping me to transition from studies in a less developed world to all what I have experienced so far. With the guidance and courage he gave me, I was able to understand more on research, and my stay at Kocaeli University has been very memorable, all thanks to him. Also, I will like to thank Prof. Dr.Oğuzhan Urhan and Muhammed Aslam, for introducing me to Image processing, and helping me to understand both basic and complicated image processing algorithms. In addition, I will like to thank the entire staff of KuanTek EleKtroniks and YongaTek, for helping me to understand FPGAs.

Last but not the least, I will to express immense gratitude to my dad, Bunji Mathias Kimbi (RIP), whose only wish was always for me to be able to pursue my dreams. Then, my beautiful mum Rachael Muncho Bunji, for her constant prayers, and words of encouragements especially on bad days. Also, I will like to extend a word of thank you to my sisters Bunji Beri(RIP), Bunji Dorothy, Bunji Maureen and BunjiGetrude, and my brother Bunji Emmanuel and step-brother Mark Ngeh. They have been a huge part of this journey and continue to inspire me every day. In addition to this, I will like to extend words of gratitude to the Cameroonian community as well as the International Community in Kocaeli.

January-2019

Antoinette Ringnyu BUNJI

## TABLE OF CONTENTS

PREFACE AND ACKNOWLEDGEMENTS .....	i
TABLE OF CONTENTS .....	ii
LIST OF FIGURES .....	iv
ÖZET .....	viii
ABSTRACT .....	ix
INTRODUCTION .....	1
1. BACKGROUND .....	3
1.1. Image/Video Basics. ....	3
1.1.1. Colour spaces .....	3
1.1.1.1. RGB colour space .....	3
1.1.1.2. YCbCr colour space .....	3
1.1.2. Video formats.....	4
1.2. AVC / H.264 Coding Standard.....	4
1.2.1. AVC/H.264 profiles and levels .....	5
1.2.1.1. Profiles.....	6
1.2.1.2. Levels .....	8
1.2.1.3. Chroma sampling formats .....	8
1.3. AVC/H.264 Encoder .....	9
1.3.1. Prediction .....	9
1.3.1.1. Inter prediction .....	9
1.3.1.2. Intra prediction .....	10
1.3.2. Forward transform.....	14
1.3.2.1. Luma transform processes .....	16
1.3.2.2. Chroma transform process .....	17
1.3.3. Forward quantization .....	18
1.3.4. Inverse quantization.....	19
1.3.5. Inverse transform.....	19
1.3.5.1. Luma inverse transform process .....	20
1.3.5.2. Chroma inverse transform process .....	22
1.3.6. Deblocking filter.....	23
1.3.7. Entropy coding.....	23
1.4. AVC/H.264 Decoder.....	24
2. OVERVIEW OF AVC/H.264 IMPLEMENTATIONS .....	25
3. 8X8 LUMA INTRA PREDICTION IMPLEMENTATION .....	28
3.1. Intra Prediction .....	29
3.1.1. H.2-4/AVC compatible intra-frame video encoder .....	29
3.1.2. Design of an 8x8 intra prediction module.....	29
3.2. General System Control. ....	30
3.2.1. Storing incoming pixels .....	31
3.2.2. Modes implementation.....	31
3.2.3. Loading data for processing .....	33
3.3. Best Mode Selection .....	34
3.4. Forward Transform.....	35

3.5. Forward Quantization .....	36
3.6. Inverse Quantization .....	38
3.7. Inverse Transform.....	40
3.8. Reconstruction.....	40
3.9. Output Pixels .....	41
4. RESULTS.....	42
4.1. Synthesis Results .....	42
4.2. Simulation Results .....	42
4.3. The Output Frames .....	48
5. DISCUSSION.....	49
6. CONCLUSIONS AND SUGGESTIONS .....	52
REFERENCES .....	53
PERSONAL PUBLICATIONS.....	57
BIBLIOGRAPHY.....	59



## LIST OF FIGURES

Figure 1.1.	A representation of the YCbCr standard. ....	5
Figure 1.2.	The General AVC Encoder and Decoder Block Diagram.....	6
Figure 1.3.	A block Diagram of the AVC/H.264 Encoder.....	11
Figure 1.4.	Inter prediction showing the frames used.....	11
Figure 1.5.	The different block sizes for the AVC/H.264 Inter prediction .....	12
Figure 1.6.	The block diagram of Intra prediction the pixels to be used for prediction (orange) and predicted area in grey. ....	12
Figure 1.7.	The 16x16 Blocks, 8x8 blocks and 4x4 blocks for AVC Intra prediction. ....	13
Figure 1.8.	4X4 Block and Reconstructed Neighbors. ....	14
Figure 1.9.	4x4 Intra prediction modes .....	15
Figure 1.10.	6X16 Intra prediction Modes .....	15
Figure 1.11.	Different blocks that is used to create the residuals. ....	16
Figure 1.12.	T4 (4x4 Transform Matrix), H4(4x4 Hadamad Transform Matrix), and H2(2x2 Hadamad Transform Matrix). ....	16
Figure 1.13.	8x8 Integer DCT Matrix. ....	17
Figure 1.14.	Default AVC Forward Integer DCT.....	17
Figure 1.15.	AVC Frext(8x8) Forward Transform.....	17
Figure 1.16.	AVC 16X16 Luma Transform.....	18
Figure 1.17.	Chroma forward transform: 4:2:0 macroblock .....	18
Figure 1.18.	Chroma forward transform: 4:2:2 macroblock .....	19
Figure 1.19.	Sample output of Transform and Quantization with $Q_p = 4$ and $Q_p = 15$ , performed on an input, Residue. ....	20
Figure 1.20.	Inverse Transformed and Inverse Quantization performed on data. ....	21
Figure 1.21.	Default AVC Inverse Forward Integer DCT .....	21
Figure 1.22.	AVC 16X16 Luma Inverse Transform.....	22
Figure 1.23.	AVC Frext(8x8) Inverse Transform.....	22
Figure 1.24.	Chroma Inverse transform: 4:2:0 macroblock.....	22
Figure 1.25.	Chroma Inverse transform: 4:2:2 macroblock.....	23
Figure 1.26.	A block Diagram of the AVC/H.264 Decoder .....	24
Figure 3.1.	Modes of Prediction for 8x8 Luma Intra Predictions .....	28
Figure 3.2.	Complete Block diagram of Frext Luma Prediction .....	32
Figure 3.3.	Order of Block processing in the implementation. ....	33
Figure 3.4.	The Single Port Block Rams used in the 8x8 Luma Intra prediction .....	33
Figure 3.5.	Modes that can be processed depending on the part of the frame being processed .....	34
Figure 3.6.	Results of the mode selection implementation. ....	35

Figure 3.7. Four stage comparator to select best prediction mode .....	35
Figure 3.8. Block Diagram of 1D Forward Transform. ....	36
Figure 3.9. A sample of an 8X8 Matrix generated from the MF table .....	37
Figure 3.10. The stages through which Forward Quantization is realized.....	38
Figure 3.11. A sample of the 8x8 QI matrixes generated from the MI table. ....	39
Figure 3.12. Shift adder for the Inverse Quantization using multiplexer to select appropriate line. ....	40
Figure 3.13. The stages through which Inverse Quantization is realized. ....	40
Figure 3.14. Block diagram of 1D Inverse Transform.....	41
Figure 4.1. Waveform for BRAMs (1 to 8) for the incoming pixels (first eight lines on the wave form) and for reconstructed pixels (last eight lines on the waveform). ....	43
Figure 4.2. Waveform for BRAMs (9 to 16) for the incoming pixels (first eight lines on the wave form) and for reconstructed pixels (last eight lines on the waveform) .....	44
Figure 4.3. Outputs of mode_predict block.....	44
Figure 4.4. Outputs of the Sad_8 block .....	45
Figure 4.5. The results of the first forward Transform .....	45
Figure 4.6. The results of Quantization.....	46
Figure 4.7. The results of the second forward transform .....	46
Figure 4.8. The results of the inverse quantization.....	46
Figure 4.9. The results of first inverse transform .....	47
Figure 4.10. The results of second transform.....	47
Figure 4.11. Original and predicted frames with Quantization Parameters 0, and 15 using 8x8 Luma Intra prediction.....	48
Figure 4.12. Original and predicted frames with Quantization Parameters 0, and 15 using 4x4 Luma Intra Prediction.....	48

## LIST OF TABLES

Table 1.1.	Average bit-rate reduction compared to prior coding schemes .....	5
Table 1.2.	Video Formats with their resolutions and areas of use.....	6
Table 1.3.	AVC/H.264 Original standard profiles .....	7
Table 1.4.	AVC FRExt Standard profiles and their characteristics .....	8
Table 1.5.	The different levels of the AVC/H.264 Standard .....	10
Table 1.6.	Different modes of prediction with their respective angles .....	13
Table 1.7.	Different Intra prediction Block Sizes with the number of Predictions per MB and number of possible predictions per mode.....	13
Table 1.8.	Sampling formats with Chroma MB sizes and number of 4x4 Blocks. ....	15
Table 3.1.	Equations used for calculating the Forward Integer DCT .....	37
Table 3.2.	The MF (Multiplication Factors) used for quantization computation.....	38
Table 3.3.	The storage of pre-calculated values in the LUTs.....	38
Table 3.4.	The MF (Multiplication Factors) used for quantization computation.....	39
Table 3.5.	The MI used for inverse quantization computation .....	41
Table 4.1.	Synthesis Results for the independent blocks. ....	42
Table 4.2.	Synthesis Results for the for the complete block. ....	43
Table 5.1.	Synthesis Results for the independent blocks and for the complete block.....	50
Table 5.2.	Error of the Implemented Luma 8x8 Intraprediction and Luma 4x4 Intra prediction.....	51



## ABBREVIATIONS

1D	: 1 Dimension
2D	: 2 Dimension
ASIC	: Application-Specific Integrated Circuit
AVC	: Advanced Video Coding
BRAM	: Block RAM
CABAC	: Context-Adaptive Binary Arithmetic Coding
CAVLC	: Context-Adaptive Variable-Length Coding
CIF	: Common Intermediate Format
chroma	: Chrominance
CODEC	: Coder/Decoder pair
DC	: Mean (Prediction Mode)
DCT	: Discrete Cosine Transform
DDL	: Diagonal-Down Left (Prediction Mode)
DDR	: Diagonal-Down Right (Prediction Mode)
FPGA	: Field-Programmable Gate Array
H	: Horizontal (Prediction Mode)
H.264	: H.264/MPEG-4 Part 10 AVC (Advanced Video Coding)
HD	: Horizontal Down (Prediction Mode)
HU	: Horizontal Up (Prediction Mode)
HVS	: Human Visual System
IDCT	: Inverse Discrete Cosine Transform
IQ	: Inverse Quantization
ISO	: International Standards Organisation
ITU	: International Telecommunication Union
JPEG	: Joint Photographic Experts Group
Luma	: Luminance
LUT	: Look-Up-Table
MB	: Macroblock
Q	: Quantization
QCIF	: Quarter Common Intermediate Format
Qp	: Quantization Parameter
RAM	: Random Access Memory
RDO	: Rate Distortion Optimization
RGB	: Red/Green/Blue (Color Space)
ROM	: Read Only Memory
SAD	: Sum Of Absolute Differences
SATD	: Sum Of Absolute Transformed Difference
SNR	: Signal-To-Noise Ratio
SSD	: Sum Of Squared Difference
V	: Vertical (Prediction Mode)
VHDL	: VHSIC Hardware Description Language
VL	: Vertical Left (Prediction Mode)
VR	: Vertical Right (Prediction Mode)

## **H.264 / AVC STANDART İÇİN 8x8 LUMA INTRA PREDICTION MODÜLÜNÜN UYGULANMASI**

### **ÖZET**

AVC, 2003 yılındaki ilk tanıtımı ve ilk olarak 2004 yılında AVC Fidelity Range Extension'ın tanıtılmasından bu yana, dijital televizyonlar üzerinden popülerlik kazanmış ve sürekli olarak DVD-Video, mobil TV, video konferans ve internet video akışı gibi alanlarda yaygın olarak kullanılmaktadır. ITU-T VCEG ve ISO / IEC MPEG'in Joint Video Team(JVT)'in ortak çabası olarak H.264 / MPEG4-AVC standardının piyasaya sürülmesinden sonra, HD TV gibi alanlarda yüksek kaliteli video kodlama talebi nedeniyle bir değişiklik çağrısı yapıldı. Bu, AVC Fidelity uzantısının 8x8 Intra prediction ve 8x8 transform gibi değişikliklerle sunulmasına yol açtı.

Bu tezde, AVC FRExtCODEC'in çok önemli bir bileşeni olan 8x8 Luma Intra prediction tasarlandı. Bu amaçla verimli tasarım mimarileri ve bunların performans üzerindeki etkileri araştırıldı. Bu çalışmada, 4x4 Luma Intra tahmini ve 8x8 Luma Intra tahmini için daha önceden gerçekleştirilen iki tasarım referans olarak alınmıştır. Bunlar dönüşüm bloklarındaki Butterfly algoritmalarından, nicemleme bloğundaki DSP(Sayısal Sinyal İşleme)'lerden ve en iyi mod seçim bloklarında SAD (Mutlak Farkın Toplamı)'dan faydalanır. Tüm tasarımı önce test etmek ve algoritmaları daha iyi anlamak için MATLAB kullanıldı, daha sonra da tüm tasarım ZC70C kartı ile VHDL kullanılarak FPGA'de gerçekleştirildi.

**Anahtar Kelimeler:** FPGA, H.264, İçi Kestirim, Kuantalama, Tamsayı DCT, VHDL, Video Kodlama.

## **IMPLEMENTATION OF 8x8 LUMA INTRA PREDICTION MODULE FOR H.264/AVC STANDARD**

### **ABSTRACT**

Since the introduction of AVC in 2003 and first introduction of the AVC Fidelity Range Extension in 2004, it has gained popularity throughout the years with digital television, and it is continuously being used in areas like DVD-Video, mobile TV, video conferencing and internet video streaming. After the introduction of H.264/MPEG4-AVC standard as a collective effort of Joint Video Team (JVT) of ITU-T VCEG and ISO/IEC MPEG, there was a call for amendment due to the demand for the coding of higher-fidelity video in areas like HD TV. This led to the introduction of the AVC Fidelity extension with amendments like 8x8 Intra prediction and 8x8Transform.

In this thesis, a very important component of the AVC FRExt CODEC, the 8x8 Luma Intra prediction is designed, exploring efficient architectures and its effect on the design. This design is made, referencing two previous designs for 4x4 Luma Intraprediction and 8x8 Luma Intra prediction. These range from the use of the butterfly algorithms in the Transform Blocks, DSPs (Digital Signal Processing) in the Quantization Block and SAD(Sum of Absolute Difference) in the best mode selection blocks. The whole design is first implemented in matlab for testing purposes and better understanding of the algorithms, then implemented in FPGA using VHDL, targeting the ZC70C board.

**Keywords:** FPGA, H.264, Integer DCT, Intra Prediction, Quantization, VHDL, Video Coding.

## INTRODUCTION

Over the years, new image compression algorithms have been developed and existing ones have been greatly improved to achieve high quality videos, flexibility of implementation, and lower SNR. Even though HEVC has been recently developed and it is gaining popularity, AVC is still widely used in many systems today such as internet video streaming, television broadcasting, digital cinema applications and 4D medical image compression[1] as well because it offers higher efficiency in compression compared to previous standards[2]. This is due to its variety of issues, from the availability of different transform matrices, different prediction methods (Intra prediction and inter prediction), as well compression methods such as CABAC.H.264/AVC was developed by the ITU-T Video Coding Experts Group (VCEG) and ISO/IEC Moving Picture Experts Group (MPEG) [3]. It was officially accepted as an international standard in 2003 and since then, a number of additions have been made and of such was the introduction of the 8x8 Intra prediction officially known as the AVC Fxext (Fidelity Extension) Standard.

Over the years, a variety of hardware implementations have been achieved for the AVC compression standards most especially for the 4x4 intra prediction. These implementations include fast prediction algorithms (best and fast prediction mode selection), motion estimation algorithms, as well as different transform algorithms. The hardware implementation is aimed at achieving high speed, lower power consumption, lower area occupation and flexibility. These implementations have been done with FPGAs, ASICs, multimedia co-processors, and general-purpose processors. While general purpose processors have been unable to meet the requirements, multimedia co-processors have focused on smaller frame sizes. Hence, most hardware implementations are done on FPGA or ASICs due to their parallel processing architecture. However, due to the high cost of ASICs, FPGAS are generally preferred[3].

In this thesis, the design and implementation of 8x8 Luma Intra prediction which is part of the AVC/H.264 FRext standard is discussed. Some implementations are available but they cannot be used for real time video. This is because, in the attempt to reduce latency and processing time, some implementations [3] do not wait for 16 lines of video before starting processing. Also, in order to improve flexibility, some implementations use 2D transform matrices as opposed to the 1D implementations that were later introduced to increase maximum frequency. The implementation presented in this thesis can be used for real time systems and also uses the 1D (butterfly algorithms). The algorithm was first implemented in MATLAB for better understanding and testing purposes. The remaining part of the thesis consists of the Background, Efficient 8x8 Luma intra prediction Implementation, Results, Discussion and Conclusion.

## **1. BACKGROUND**

This chapter gives an overview about image processing and image compression, AVC compression for both the AVC standard and AVC FReExt standard.

### **1.1. Image/Video Basics.**

A video is a composition of many frames/images. Every video has specifications such as Frame rate which is the number of frames per unit time, frame resolution which specifies the number of pixels used to represent each frame, pixel depth which specifies the number of pixels used to represent each pixel.

#### **1.1.1. Colour spaces**

Pixels usually indicate colour and brightness. Depending on the type of image, a number or a set of numbers can be used to represent these properties. For monochrome images (YCbCr), one pixel used to represent brightness and for RGB formats, 3 different values are used to represent colour. Some of these colour spaces include:

##### **1.1.1.1. RGB colour space**

This colour space has three colour representations (Green, Red and Blue). The change in concentration in one of the colours reproduces a different colour. Video is typically captured and displayed using the RGB format but these components are highly correlated.

##### **1.1.1.2. YCbCr colour space**

This format is generally preferred in Digital Image processing. It has three components, Y (Luma), Cb (Blue chrominance), and Cr (Red chrominance). The human visual system is better matched to the luma (brightness) and chroma (hue and saturation) representations, rather than RGB.

A visual representation of these different formats is shown in Figure 1.1.

From the visual representation in, it can be seen that the Luma component has more information about the frame than other components. That is why most image processing operations are done on this component. The luma component(Y) can be calculated by a weighted average of the Red, Green and Blue components using equation 1.1.

$$Y = K_r R + K_g G + K_b B \quad (1.1)$$

Where  $K_r$ ,  $K_g$  and  $K_b$  are weighting factors.

$C_b$  and  $C_r$  components are calculated using the equations 1.2 and 1.3 respectively.

$$C_b = B - Y \quad (1.2)$$

$$C_r = R - Y \quad (1.3)$$

### 1.1.2. Video formats

Each video format has specific resolutions and each format is used for specific applications. For example, HD is used for high definition TV and 4CIF are used for standard Television. Table 1.2 shows the different formats, resolutions and areas used.

## 1.2. AVC / H.264 Coding Standard

The AVC standard like other standards has both the encoding and the decoding block. The main parts of this Codec system as shown in Figure 1.2 which include prediction, transform/inverse transform, quantization/inverse quantization and reconstruction. These powerful techniques lead to the robustness, less decoder complexity and high coding efficiency of the AVC with respect to previous standards [7].

This is evident from the data shown in Table 1.1, showing the performance comparison of AVC to MPEG2 and MPEG4 .From the block diagram, it can also be seen that the encoder has an internal decoder which is used in the concept of prediction. This is because pixels of predicted sub blocks or macro blocks are used to

predict subsequent blocks. Since AVC is a lossy compression method, there is generally some difference between the original frames and decoded frames.



Figure 1.1. A representation of the YCbCr standard. a) The original Photo b) The Y or Luma component of the Photo c) The Cr (Red Chroma) component of the photo d) The Cb(Blue Chroma) component of the photo.

Table 1.1. Average bit-rate reduction compared to prior coding schemes

<b>Standard</b>	<b>MPRG 4/ASP</b>	<b>H.263/HLP</b>	<b>MPEG-2</b>
H.264/AVC	38.62%	48.80%	64.46%
MPEG 4/ ASP	---	---	42.95%
H.263/HLP	---	---	30.61%

### 1.2.1 AVC/H.264 profiles and levels

AVC contains a rich set of video coding tools, and not all the tools are supposed to be used at the same time. For example sophisticated error resilience tools are not important for the networks with very little data corruption or loss. So, the tools can be implemented independently and the decoder decides on which set to use. Forcing



the decoder to implement all these tools at the same time can lead to unnecessary complexity.

Table 1.2. Video Formats with their resolutions and areas of use.

Format	Resolution	Areas of use
Sub-QCIF	128x96	streaming video on mobile phones
Quarter CIF	176x144	Video Conferencing
CIF	352x288	closed circuit television, DVD or online video design
4CIF	704x576	Format for H.264/AVC, Video-on-demand or multimedia streaming services over ISDN, Broadcast over cable and satellite.
720p	1280x720	HD television channels broadcast
1280x720HD	1920x1080	Appropriate for High Definition television
UHD	3840x2160	Specialized video Cameras for Military

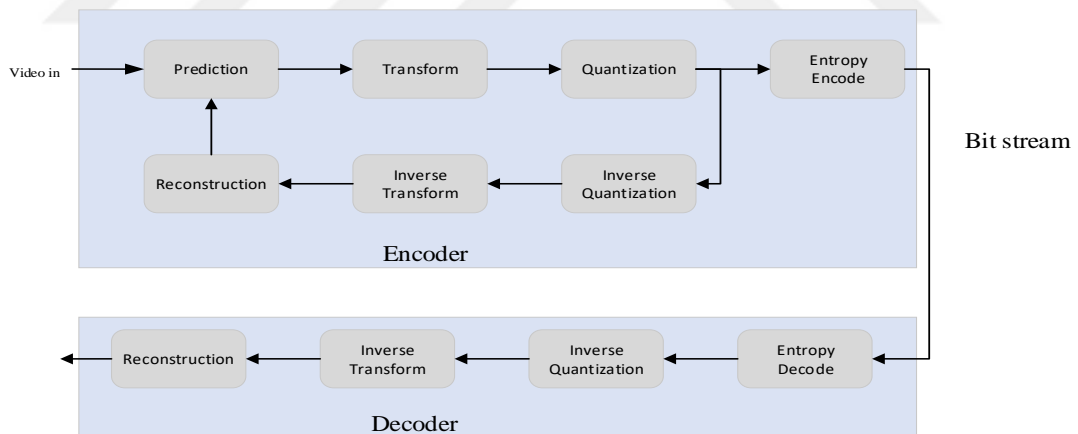


Figure 1.2. The General AVC Encoder and Decoder Block

Due to this issue, the subsets (profile) of tools can be implemented at the same time. The profiles can be separated into two groups.

### 1.2.1.1. Profiles

This section explains the profiles of the original standards and the profiles of the FRExt standard.

a) Profiles of the original standard

The profiles of the original standard can be divided into three profiles, Baseline (BP), Extended (XP), and Main (MP). Table 1.3 gives a summary of the tools of each profile [5].

Table 1.3. AVC/H.264 Original standard profiles [5].

<b>Coding Tools</b>	<b>Baseline</b>	<b>Main</b>	<b>Extended</b>
I and P Slices	X	X	X
CAVLC	X	X	X
CABAC		X	
B Slices		X	X
Interlaced Coding (PicAFF, MBAFF)		X	X
Enh. Error Resil. (FMO, ASO, RS)	X		X
Further Enh. Error Resil (DP)			X
SP and SI Slices			X

a) The profile of the FExt Amendment

The FExt defines four new levels which are built on the previous main profile. these profiles are listed below:

- High (HP)
- High 10 (Hi10P)
- High 4:2:2 (Hi422P)
- High 4:4:4 (Hi444P)

The four new profiles include 3 enhancements which are:

- Adaptive macroblock-level switching between 8x8 and 4x4 transform block size.
- Encoder-specified perceptual-based quantization scaling matrices.
- Encoder-specified separate control of the quantization parameter for each chroma component.

The main characteristics of these profiles can be found in the Table 1.4

### 1.2.1.2. Levels

Levels define the maximum data processing rate of a decoder. It puts constraints on some video parameters such as the maximum frame rate and the maximum frame size of a video. The Table 1.5 shows the characteristics of the different levels in AVC/H.264.

Table 1.4. AVC FReXt Standard profiles and their characteristics [9]

<b>Coding Tools</b>	<b>High</b>	<b>High 10</b>	<b>High 4:2:2</b>	<b>High 4:4:4</b>
Main Profile Tools	X	X	X	X
4:2:0 Chroma Format	X	X	X	X
8 Bit Sample Bit Depth	X	X	X	X
8x8 vs. 4x4 Transform Adaptivity	X	X	X	X
Quantization Scaling Matrices	X	X	X	X
Separate Cb and Cr QP control	X	X	X	X
Monochrome video format	X	X	X	X
9 and 10 Bit Sample Bit Depth		X	X	X
4:2:2 Chroma Format			X	X
11 and 12 Bit Sample Bit Depth				X
4:4:4 Chroma Format				X
Residual Color Transform				X
Predictive Lossless Coding				X

### 1.2.1.3. Chroma sampling formats

As mentioned in the previous sections, chroma samples contain less information about frames than the Luma sample. For this reason, representing chroma components with less data usually may give better results in some video compression applications. This is achieved as follows: Instead of using one Cb and one Cr pair for each Y component, same Cb and Cr pairs are used for more than one Y component.

In the 4:4:4 sampling format, one Cb and one Cr pair is used for each luma component. In the 4:2:2 sampling format, chroma components are sampled by two in

the horizontal axis which means the same Cb and Cr components are used for each two horizontal neighbour luma components. In the 4:2:0 sampling format, which is the commonly used format, chroma components are sampled by two both in the horizontal and vertical directions.

### **1.3. AVC/H.264 encoder**

The AVC/H.264 is a video compression standard which uses the macroblock system of processing previously used in the JPEG 2000. Each macroblock (MB) is 16x16 pixels. The compression is done macroblock by macroblock until the entire frame is covered. The Figure 1.2 shows a detailed block diagram of an AVC/H.264 encoder. This encoder consists of the following main blocks prediction, motion estimation and compensation, transform and quantization, inverse transform and inverse quantization, entropy coding and reconstruction. These blocks are explained in detail below.

#### **1.3.1. Prediction**

In AVC/H.264, prediction is performed by using previously coded pixels to predict pixels of the current frame. The aim of this operation is to construct a prediction block as close as possible to the original block and send the difference (error or residual) between these blocks instead of the original block as opposed to what is done in JPEG 2000. If the error is small, that means the residual block contains less information, the bitrate to transmit the error will be less. So, the compression efficiency increases. There are two types of prediction in AVC, Inter prediction and Intra prediction.

##### **1.3.1.1. Inter prediction**

Inter prediction uses previously decoded pixels in different frames to predict pixels of the current frame as shown in Figure 1.3. In AVC, several inter prediction block sizes are used. A macroblock can be divided into two 16x8 blocks or two 8x16 or four 8x8 blocks which are called as macroblock partitions. Further, an 8x8 macroblock partition can be divided into two 4x8 blocks or two 8x4 blocks or four

4x4 blocks which are called as sub-macroblock partitions. A macroblock can be predicted using macroblock partitions from different frames.

Table 1.5. The different levels of the AVC/H.264 Standard [9]

Level Number	Typical Picture Size	Typical frame rate	Maximum compressed bit rate (for VCL) in Non-FRExt profiles	Maximum number of reference frames for typical picture size
1	QCIF	15	64 kbps	4
1b	QCIF	15	128 kbps	4
1.1	CIF or QCIF	7.5 (CIF) / 30 (QCIF)	192 kbps	2 (CIF) / 9 (QCIF)
1.2	CIF	15	384 kbps	6
1.3	CIF	30	768 kbps	6
2	CIF	30	2 Mbps	6
2.1	HHR (480i or 576i)	30 / 25	4 Mbps	6
2.2	SD	15	4 Mbps	5
3	SD	30 / 25	10 Mbps	5
3.1	1280x720p	30	14 Mbps	5
3.2	1280x720p	60	20 Mbps	4
4	HD Formats	60p / 30i	20 Mbps	4
4.1	(720p or 1080i)	60p / 30i	50 Mbps	4
4.2	HD Formats	60p	50 Mbps	4
5	(720p or 1080i)	72	135 Mbps	5
5.1	1920x1080p	120 / 30	240 Mbps	5

However, sub-macroblock partitions of a macroblock partition must be in the same frame. Figure 1.5 shows the inter prediction block sizes used in AVC [3]. If a macroblock is inter predicted, the reference frame index or indexes and motion vector or vectors must be signalled to the decoder side to properly construct the decoded picture.

### 1.3.1.2. Intra prediction

As opposed to inter prediction, Intra prediction uses previously coded pixels in the current frame as shown in Figure 1.6. It uses predicted pixels on the top and left of the block to be predicted. There are three different blocks in AVC intra prediction,

4x4, 8x8 and 16x16 blocks are used for the luminance and 8x8 blocks used for chrominance and these blocks are explained in the subsequent sections.

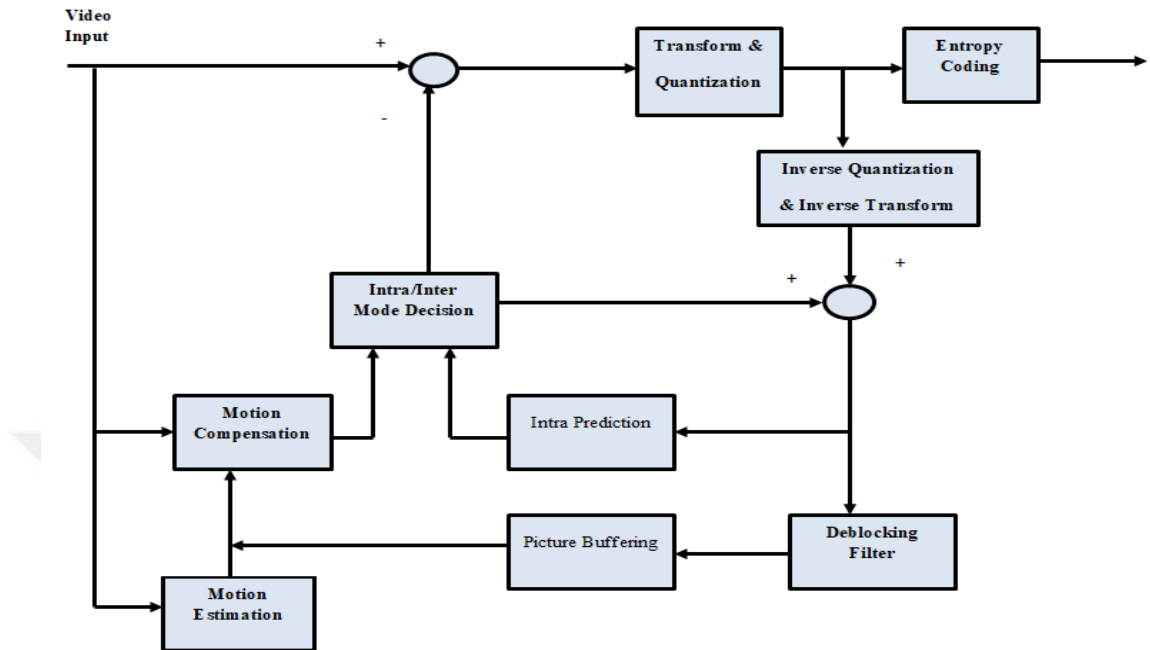


Figure 1.3. A block Diagram of the AVC/H.264 Encoder

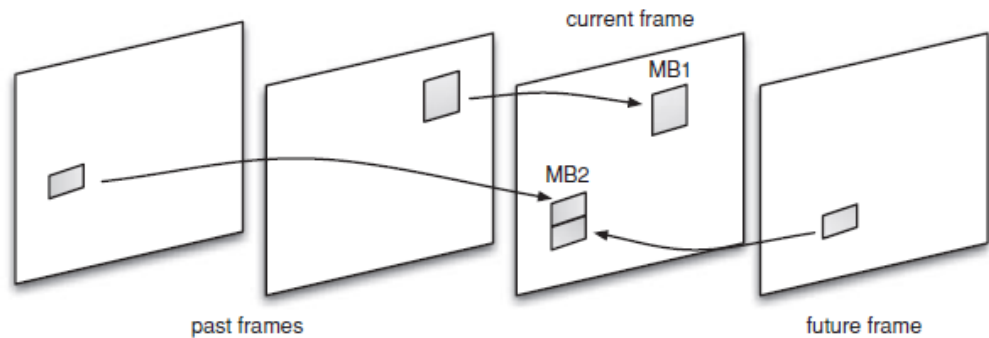


Figure 1.4. Inter prediction showing the frames used.

i) Luma (luminance) intra prediction.

For Luma intra prediction, 4x4, 8x8 and 16x16 block sizes are used. The different prediction modes used for the luma intra prediction can be seen in Table 1.6 with the direction in which the pixels are taken. The 8x8 blocks are derived from the 16x16 blocks and the 4x4 blocks are derived from the 8x8 blocks. This can be seen in Figure 1.7.

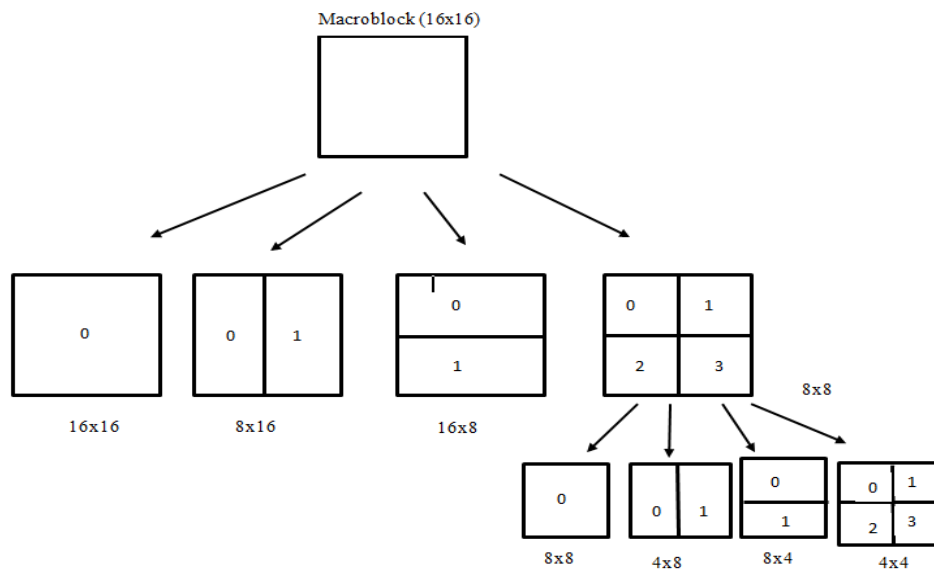


Figure 1.5. The different block sizes for the AVC/H.264 Inter prediction [2]

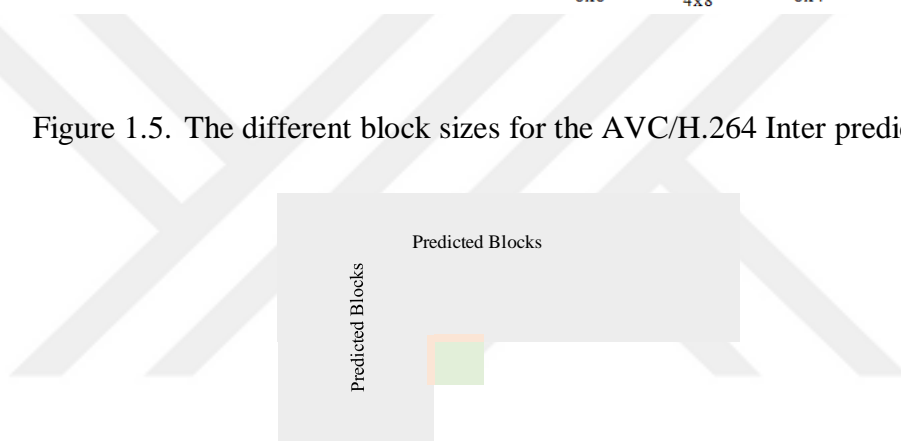


Figure 1.6. The block diagram of Intra prediction, showing the block to be predicted (green) the pixels to be used for prediction (orange) and predicted area in grey.

Figure 1.7 shows the different blocks of the AVC Luma Intra prediction. It also shows how the smaller blocks are derived from larger blocks. Table 1.7 shows the intra prediction blocks; number of prediction blocks in a macroblock and number of possible prediction modes. A detailed description of each prediction block size is presented in the next section.

#### a) 4x4 Luma Intra Prediction

This is performed using reconstructed top pixels and left pixels of every 4x4 block. The Fig.2.7 shows 4x4 blocks with pixels( a, b, ...,p) to be predicted , the top pixels(M,A,...,H), and left pixels (I,J,...,L). . This definitely gives a more accurate prediction compared to the 16x16 but, it takes a high level of signalling to achieve this. There are nine modes for this prediction as shown in Figure 1.9.

Table 1.6. Different modes of prediction with their respective angles

Mode number	Name	Direction
0	Vertical (V)	
1	Horizontal (H)	
2	Mean (DC)	
3	Diagonal down-left (DDL)	45°
4	Diagonal down-right (DDR)	45 °
5	Vertical-right (VR)	26.6° right of V
6	Horizontal-down (HD)	26.6° below H
7	Vertical-left (VL)	26.6° left of V
8	Horizontal-up (HU)	26.6° up from H

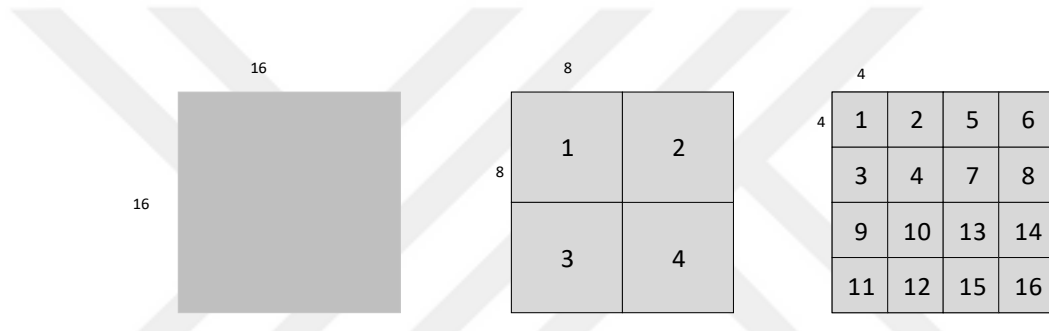


Figure 1.7. The 16x16 Blocks, 8x8 blocks and 4x4 blocks for AVC Intra prediction.

Table 1.7. Different Intra prediction Block Sizes with the number of Predictions per MB and number of possible predictions per mode.

Intra prediction block size	Number of prediction blocks in a MB	Number of possible prediction modes
16x16 luma	1	4
8x8 luma	4	9
4x4 luma	16	9

#### b) 16x16 Luma Intra Prediction

This prediction method uses 4 modes of prediction , vertical, horizontal, DC and planar. The vertical, horizontal and DC modes are similar to those of the 4X4 Luma Intra prediction. The Figure 1.10 shows a diagram of the four modes used for this prediction. The 16x16 block predicts the pixels of a complete macroblock at the same time.



M	A	B	C	D	E	F	G	H
I	a	b	c	d				
J	e	f	g	h				
K	i	j	k	l				
L	m	n	o	p				

Figure 1.8. 4X4 Block and Reconstructed Neighbors.

c) 8x8Luma Intra prediction

Since the 4x4 Luma produces better prediction results but having a problem of more signals involved in the implementation, 16x16 Luma Intra prediction solves the signalling problem but, the prediction is less accurate. The 8x8 Luma intra prediction introduces a part of the Frest Amendment which solves this problem. It is very similar to the 4x4 intra prediction, using the same equations with 8x8 blocks. More on this intra prediction block is presented in chapter 3.

ii) Chroma (chrominance)

Each chroma component of a macroblock is predicted from the previously encoded chroma samples above and/or to the left of the macroblock. One prediction block is generated for each chroma component. There are four possible intra prediction modes and they are very similar to the luma 16x16 intra prediction modes. The chroma Macroblock size varies with sampling format, and this can be seen in Table 1.8, showing the MB sizes and number of 4x4 Blocks per MB.

**1.3.2. Forward transform**

At the level of this block, residuals are converted from spatial domain to the frequency domain. Residuals are formed by taking the difference between the predicted pixels and the original pixels, as presented in Figure.1.11 which shows the original pixels, predicted pixels and the difference (residuals). This reduces the spatial redundancy of the prediction error signal. Former standards used floating point 8x8 DCT transform. AVC uses integer DCT. This is advantageous because it reduces computational complexity especially at the level of hardware. Also, it reduces encoder/decoder mismatch. The transform in AVC is done using the equation 1.4.

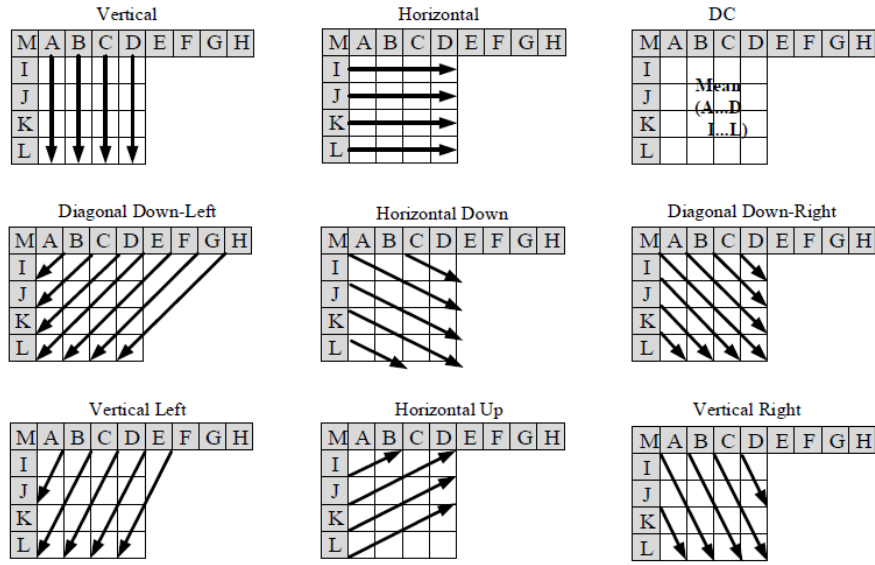


Figure 1.9. 4x4 Intra prediction modes [3]

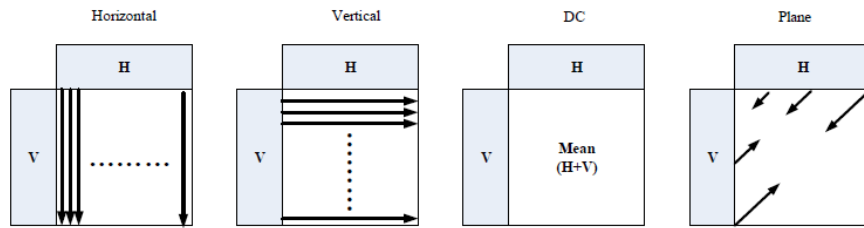


Figure 1.10. 16X16 Intra prediction Modes [3]

Table 1.8. Sampling formats with Chroma MB sizes and number of 4x4 Blocks.

Sampling Format	Chroma Macroblock Size	Number of 4x4 Blocks
4:2:0	8x8	4
4:2:2	16x8	8
4:4:4	16x16	16

$$Y = CXC^T \tag{1.4}$$

where X is the residual input of the transform block and C is the transform.

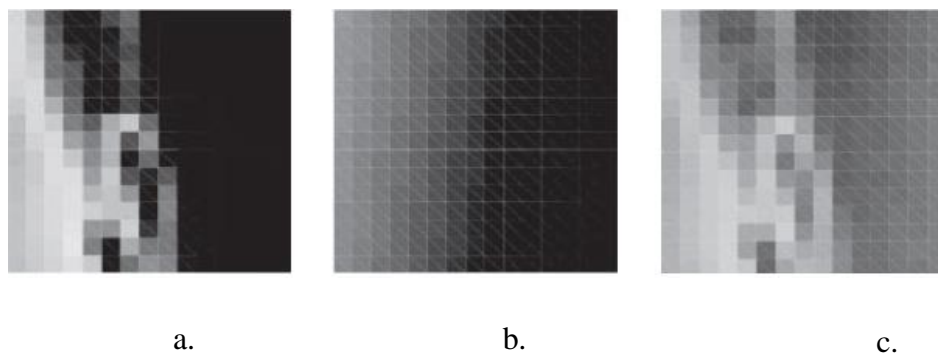


Figure 1.11. Different blocks that are used to create the residuals. a) the original block. b) the predicted block c) the residuals.

In AVC, C can be 4x4 Luma transform matrix or 8x8 luma transform (for FRExt). When the intra prediction 16x16 is performed, then another hadamard transform is applied to the DC components of the 4x4 Luma transform output. These matrices are shown in Figure 1.12.

$$T4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & -2 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{pmatrix} \quad H4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{pmatrix} \quad H2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Figure 1.12. T4 (4x4 Transform Matrix), H4(4x4 Hadamard Transform Matrix), and H2(2x2 Hadamard Transform Matrix).

Initially, the Integer 4x4 DCT was part of AVC standard and it helped to reduce blocking and ringing artifacts. In July of 2004, the Fidelity Range Extensions (FRExt, Amendment I) was added to the H.264 standard with the introduction of the 8x8 Integer DCT which is shown in Figure 1.13. It demonstrates higher code efficiency compared to the main H.264 standard [7].

### 1.3.2.1. Luma transform processes

The default transform process is usually a 4x4 Luma Transform, unless 8x8 Transform or 16x16 is selected. The Default 4x4 Luma Transform on an MB is shown on Figure 1.14 [3]. If it is an 8x8 Transform, the block diagram Figure 1.15 shows how the Transform process is carried out, using the 8x8 Transform matrix [3].

$$H = \begin{pmatrix} 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \\ 12 & 10 & 6 & 3 & -3 & -6 & -10 & -12 \\ 8 & 4 & -4 & -8 & -8 & -4 & 4 & 8 \\ 10 & -3 & -12 & -6 & 6 & 12 & 3 & -10 \\ 8 & -8 & -8 & 8 & 8 & -8 & -8 & 8 \\ 6 & -12 & 3 & 10 & -10 & -3 & 12 & -6 \\ 4 & -8 & 8 & -4 & -4 & 8 & -8 & 4 \\ 3 & -6 & 10 & -12 & 12 & -10 & 6 & -3 \end{pmatrix}$$

Figure 1.13. 8x8 Integer DCT Matrix.

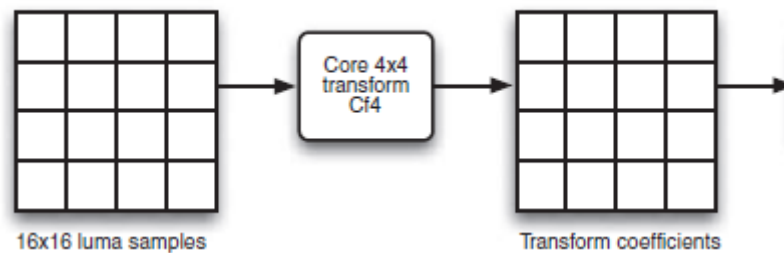


Figure 1.14. Default AVC Forward Integer DCT [3]

If the 16X16 Luma Intra Prediction is selected, the default Transform is first performed on the 4x4 blocks of the predicted pixels. Then, a second transform, the Hadamad is performed on the DC components.

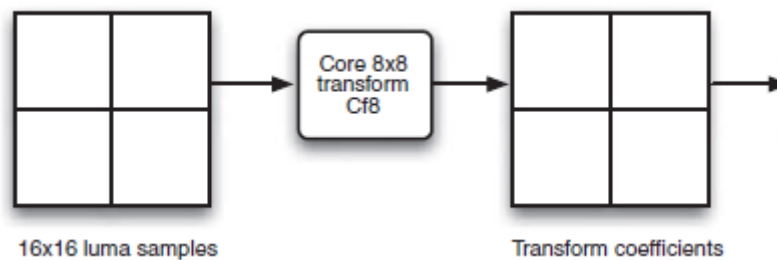


Figure 1.15. AVC Frext(8x8) Forward Transform [3]

This is because these DC components are highly correlated and the second transform improves the coding performance. Figure 1.16 shows how the transform for the 16x16 Luma is performed [2].

### 1.3.2.2. Chroma transform process

The chroma transform process is performed just like the 16x16 Luma Transform. The only difference is a 2x2 Hadamad Transform is performed on the DC

components of the blocks. Since the size of the MB and the number of 4x4 blocks depends on the sampling format, the Hadamad Transform for each sample is different. For the 4.2.0 format, it has 8x8 Cr MB and 8x8 Cb MB. This transform is performed by applying the default 4x4 Transform on four 4x4 blocks that make up the MB, then the DC components are further transformed using the 2x2 Hadamad Transform as shown in Figure 1.17 [3].

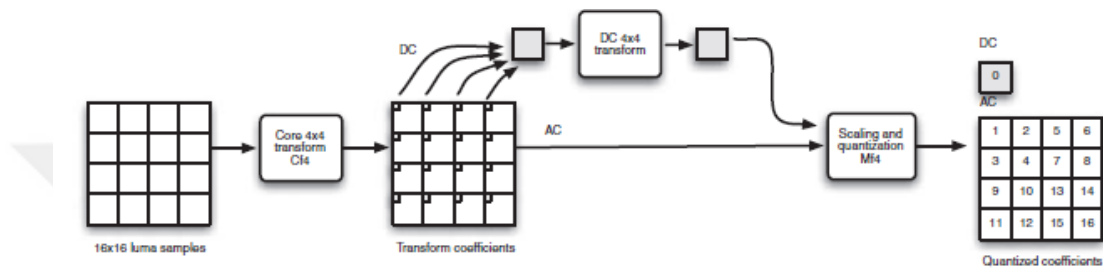


Figure 1.16. AVC 16X16 LumaTransform [3]

The 4.2.2 sample format has 8x16 MB Cb and 8x16 Cr MB. Just like the other formats, the 4x4 default transform is first performed on the 4x4 blocks of the MB and then the DC components are transformed using 2x4 Hadamad Transforms as shown in Figure 1.18. The 4.4.4 sample format has 16x16 MB Cr and 16x16 MB Cb. Hence, the prediction is sameas that of the 16x16 Luma Transform.

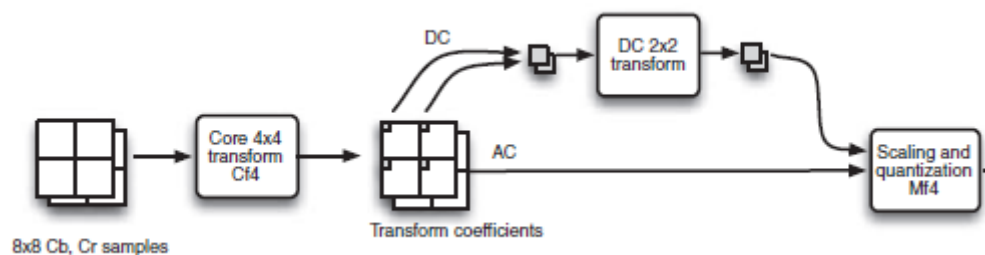


Figure 1.17. Chroma forward transform: 4:2:0 macroblock [3]

### 1.3.3. Forward quantization

After transform, quantization is performed to reduce precision of the transform coefficients according to a quantization parameter. At the level of the quantization block, intentional errors are added to the system. This helps to increase the

compression performance with a reasonable distortion. If the quantization step size increases, more quantized coefficients will be zero which means less data to represent. This leads to keeping only a few coefficients for efficient representation and results in more distortion. More details on AVC/H.264 Quantization are found in chapter 3. A representation of the effects of these processes are shown in Figure 1.19.

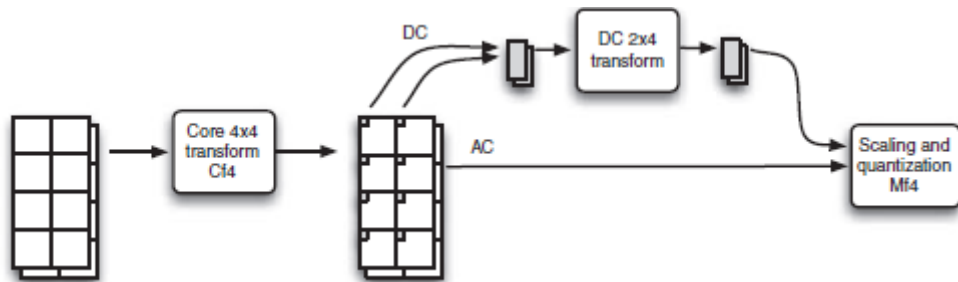


Figure 1.18. Chroma forward transform: 4:2:2 macroblock [3].

It can be seen from the figure that, Transform processes give rise to the largest positive element (DC) and the AC components. Then, quantization reduces most of the values to zero and when the Qp increases, more quantized coefficients become zero, hence less data to represent.

#### 1.3.4. Inverse quantization

The inverse quantization does the opposite operation of the quantization process; that is, multiplying by a quantization parameter and more on Quantization can be found in Chapter 3. Figure 1.20 shows the output of inverse quantization and inverse transform. It can be seen that, the error in the final output increases as Quantization parameter increases.

#### 1.3.5. Inverse transform

The reverse of the forward Transform block is performed at the level of this block and it is realized with the equation 1.5.

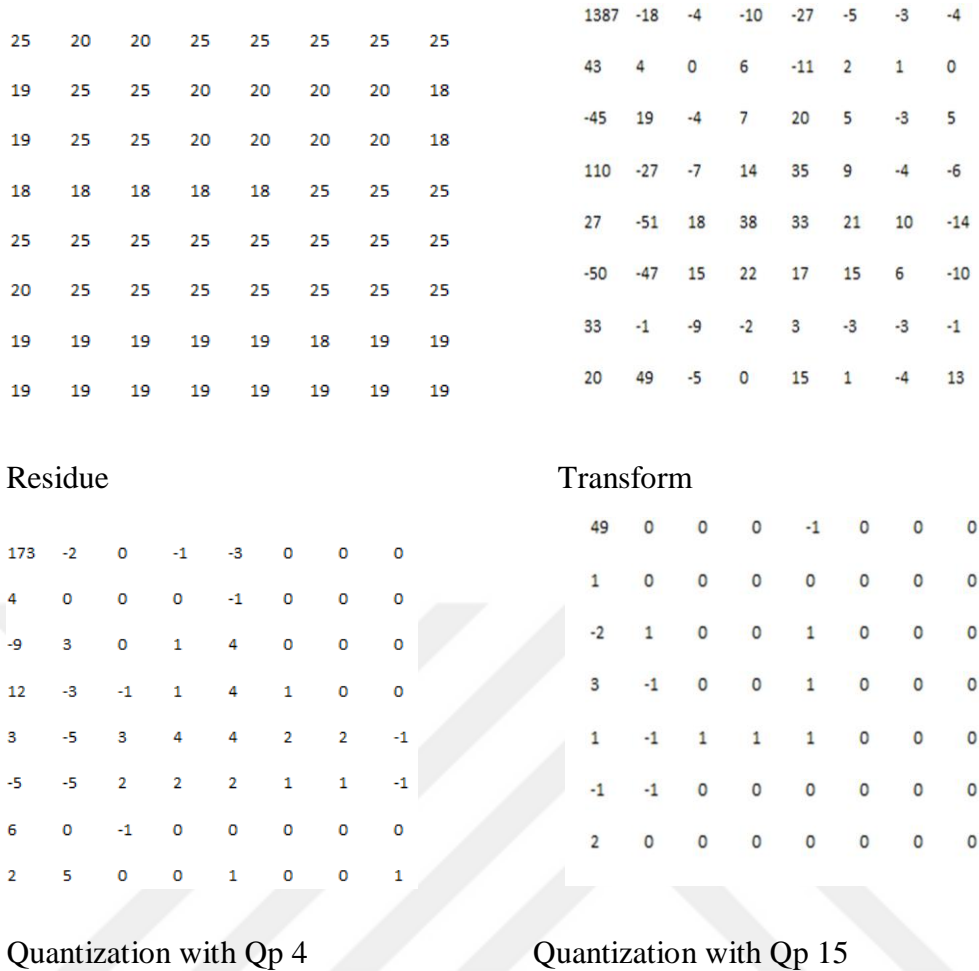


Figure 1.19. Sample output of Transform and Quantization with Qp = 4 and Qp 15, performed on an input, Residue.

$$X = C^T Y_q C \tag{1.5}$$

Where C is one of the matrices in Figure.1.12 and Y<sub>q</sub> is the inverse quantization. In addition to inverse transform process, rounding is also done at the level of this block to get the final output (in the range (-255,255)). Just like the forward transform process, the inverse transform process is carried out depending on the Intra prediction block size and these are illustrated in section 1.3.5.1.

**1.3.5.1. Luma inverse transform process**

The default inverse transform same is also uses the 4x4 Block, so it uses the 4x4 integer DCT, and follows the block diagram in Figure 1.21 [3].

5488	0	0	0	-112	0	0	0
104	0	0	0	0	0	0	0
-280	132	0	0	140	0	0	0
312	-100	0	0	104	0	0	0
112	-104	140	104	112	0	0	0
-104	-104	0	0	0	0	0	0
280	0	0	0	0	0	0	0
0	100	0	0	0	0	0	0

5536	-60	0	-30	-96	0	0	0
120	0	0	0	-30	0	0	0
-360	114	0	38	160	0	0	0
360	-84	-38	28	120	28	0	0
96	-150	120	120	128	60	80	0
-150	-140	76	56	60	28	38	0
240	0	-51	0	0	0	0	0
60	140	0	0	30	0	0	0

Inverse Quantization with Qp = 15

24	22	21	23	24	23	23	25
20	22	22	21	19	20	19	18
20	24	24	20	19	22	21	18
19	20	20	19	21	24	25	23
24	24	23	23	23	24	25	24
22	24	25	24	24	25	25	25
18	19	20	19	18	19	18	18
22	21	19	19	19	19	19	18

Inverse Quantization with Qp = 4

25	19	20	24	24	24	24	25
18	24	24	20	20	19	20	18
20	25	25	20	20	21	21	18
18	18	19	19	19	25	25	25
25	25	25	24	25	25	25	25
21	24	26	25	25	25	26	25
19	19	19	19	19	18	19	19
20	19	19	19	19	19	19	19

Inverse Transform for Qp = 15

Inverse Transform for Qp = 4

Figure 1.20. Inverse Transformed and Inverse Quantization performed on data.

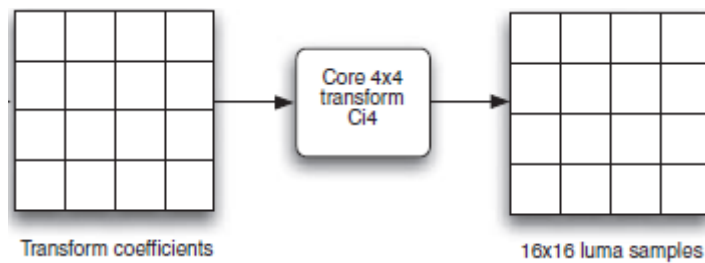


Figure 1.21. Default AVC Inverse Forward Integer DCT [3]

If the 16x16 Luma is selected then, DC coefficients first undergo the Hadamard transform then together with the other AC coefficients undergo inverse DCT as shown in Figure 1.22.



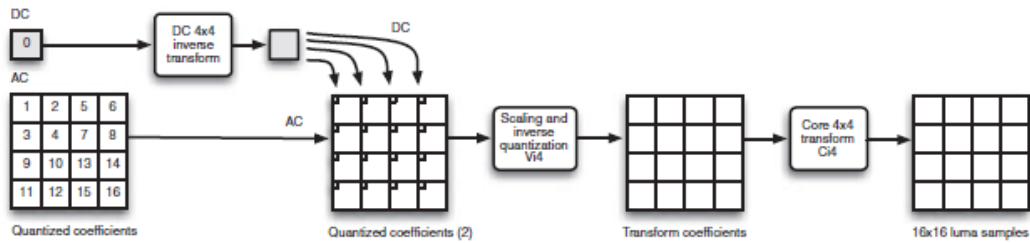


Figure 1.22. AVC 16X16 Luma Inverse Transform [3]

If the 8x8 Luma Intra prediction is selected, then the outputs of the inverse quantized coefficients undergo the 8x8 Integer DCT, as shown in Figure 1.23.

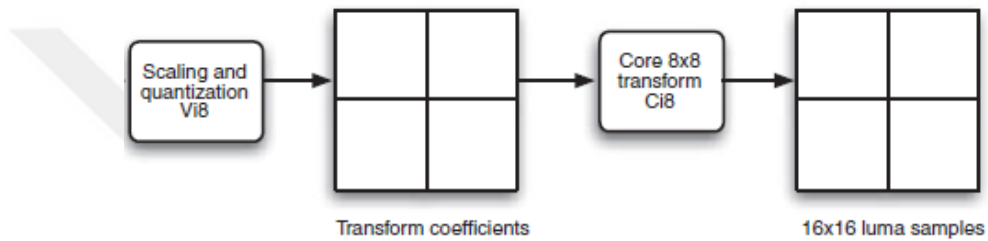


Figure 1.23. AVC Fxext(8x8) Inverse Transform [3]

### 1.3.5.2. Chroma inverse transform process

Just like the chroma forward transform, the inverse transform depends on the sample. For the 4.2.0, the inverse transform also has 8x8 Cr MB and 8x8 Cb MB. The Hadamard transform is performed on the DC coefficients, then the default Integer DCT is performed on the output of the Hadamard together with AC coefficients as shown in Figure 1.24. For the 4.2.2 sample, the same procedure is carried out except that, 2x4 Hadamard Transform is carried out as shown in Figure 1.25

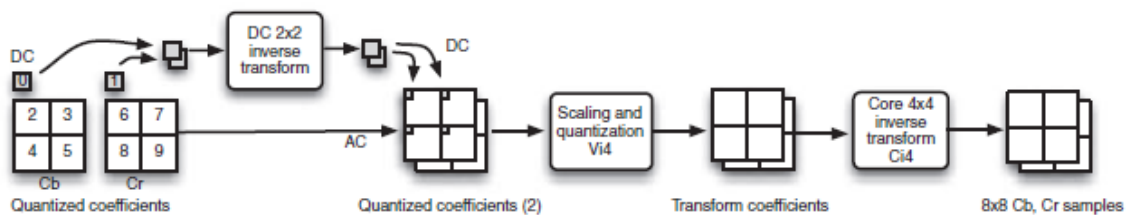


Figure 1.24. Chroma Inverse transform: 4:2:0 macroblock [2]

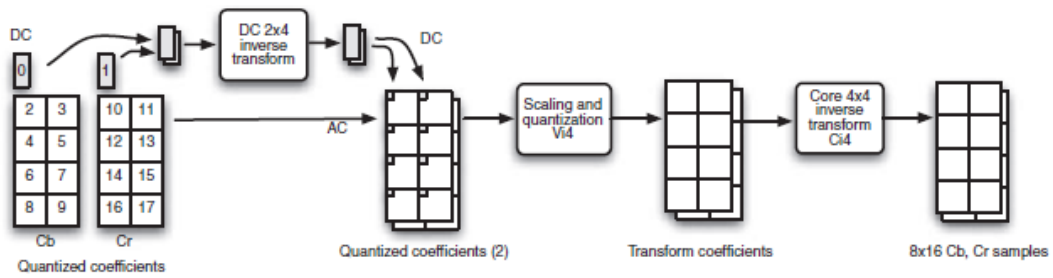


Figure 1.25. Chroma Inverse transform: 4:2:2 macroblock [2]

Since the 4.4.4 sample has 16X16 MB, it follows the same procedure as that of 16X16 Luma Intra predictions.

### 1.3.6. Deblocking filter

Although the filter block is an optional block, it greatly affects the quality of the decoded photo by improving its subjective visual and objective quality, by reducing blocking distortion [9]. This filter is applied after the inverse transform at the level of the encoders (before reconstruction and pixel storage for future use) and in the decoder (for reconstruction and displaying of macroblock). The filtered image is used for motion-compensated prediction of future frames and this can improve compression performance. The deblocking filter adjusts its strength depending on the compression mode of the macroblock (for both intra prediction and inter prediction), the quantization parameter, motion vector, frame or field coding decision and the pixel values. The effect of the filter is decreased when the quantization step size is decreased and completely shuts off when the quantization step size is very small [9]. The vertical edges are first filtered, then the horizontal edges and the bottom row and right column of a macroblock are filtered when decoding the corresponding adjacent macroblocks.

### 1.3.7. Entropy coding

This is the last stage of video compression before the bit stream is sent to the decoder. After the transform and quantization is performed, most coefficients become zero. The entropy coding helps to prevent the continuous transmission of these zeros and other recurrent elements. It is a lossless compression technique and there are generally two techniques used in AVC.

- Context-based adaptive variable length coding (CAVLC) or the quantized transform residues after ordering them by ZigZag scanning
- context-based adaptive binary arithmetic coding (CABAC) has a higher compression performance but unlike CAVLC, it is not supported in all H.264 profile like baseline and extended profiles

#### 1.4. AVC/H.264 decoder

The AVC decoder converts the bit stream (output received from the encoder) into frames. The bit stream is first entropy decoded, then inverse quantized, inverse transformed then reconstructed according to the parameters of the encoder in accordance with the profiles. The block diagram of the AVC Decoder is shown in Figure 1.26.

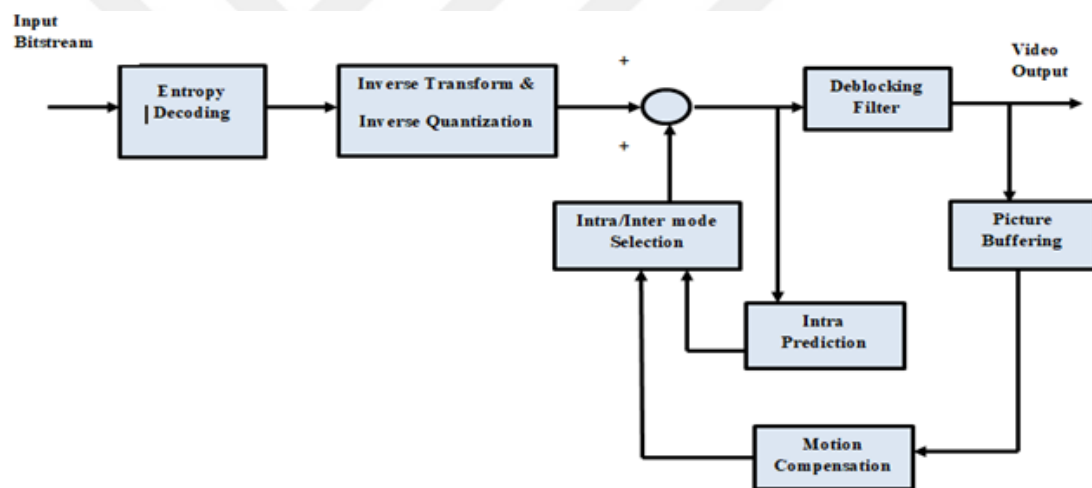


Figure 1.26. A block Diagram of the AVC/H.264 Decoder

## 2. OVERVIEW OF AVC/H.264 IMPLEMENTATIONS

Since the introduction of the AVC/H.264 standard in 2004 and the addition of the Fidelity Range Extensions in 2008, so much hardware related research has been done. The aim of these is to achieve the three most important aspects of every digital design. That is, speed (architectures that perform faster), low power consumption and low area. Some of those implementations are stated and briefly explained in this chapter.

H.264/AVC, adopts rate-distortion optimization (RDO) technique to obtain the best intra and inter prediction, while maximizing visual quality and minimizing the required bitrate. However, full RDO calculations, searches for optimal motion vectors for all block sizes, and the multiple references frame procedure considerably increase its computational complexity. In order to reduce the complexity, [7] proposes a new approach for both inter- and intra-mode decisions and implementing with JM, that takes into account the two effective parameters, image content type and the quantization parameter. The fast inter-prediction mode decision approach uses split/merge procedure based on correlation of motion vectors and motion details of video objects. With this proposition, encoding time is decreased by reducing the number of modes used for prediction. Apart from using the standard equations, implementations like [9], pixels within a block are a weighted sum of neighbouring pixels according to the  $n$ -th order of the Markov Linear Model. These pixels are obtained using the least-squares estimates of the reconstructed pixels. This method improves the compression for images that are rich in directional structures. Apart from implementing the fast prediction techniques, some implementations like [10], explore parallelism and pipelining techniques. These techniques enhance the speed and reduce the area. Most of these implementations mention very little or nothing about the transform and quantization blocks which affect the overall performance of the encoder, but [11] gives a detailed implementation of the Intra prediction, using all block sizes of an AVC FRT Intra prediction, using SAD for the best mode selection and it reduces the total number of clock cycles compared to

other designs, to complete one MB. More designs and implementations of the AVC Intra prediction can be found in the references [12, 13, 14, 15, 16, 17, and 18].

Transform, Quantization, Inverse Quantization and Inverse Transform are very important blocks of the AVC/H.264 CODEC, used as part of the reconstruction loop of the encoder and the decoder. The Integer DCT and Integer IDCT blocks for AVC are realised through matrix multiplication, and this is realised in this standard using the butterfly algorithm (1D implementation), to reduce complexity. The work presented in [19,20] sees the effect these different implementations (1D and 2D) have on the forward Integer DCT for both 4x4 Integer DCT and 8x8 Integer DCT. The 1D method reduces resource utilization and can be seen from [21] that use this method to implement the 8x8 forward Integer DCT and replaces large multipliers with adders and shifters for the 8x8 Quantization.

Also, [22] uses the butterfly method to calculate the 8x8 Forward Transform, but uses multipliers and shifters instead of adders and it helps reduce the complexity of the design. Some designs like [24], use reconfigurable multipliers for the transform and quantization implementations. Since it uses only one multiplier, cost can be considerably reduced. In the 4x4 AVC Encoder, Integer DCT and Hadamard DCT are used. These matrices increase area, and to reduce this, implementations like [25], propose a method that derives the Integer DCT coefficients from the Hadamard Coefficients. More work on the blocks can be found in the references, [26, 27].

Even though the decoder has many common blocks with the Encoder; some researches have also been done to make the decoder blocks less complex. Just like many other implementations, [28] implements 4x4 luma prediction modes using the required equations and the same is done for 16x16 Luma and 8x8 Chroma. The main aspect of this design that reduces complexity is calculating repetitive equations and making them available as signals. [29] Implementation uses two parallel pipelines, for 4x4 Block prediction and the other pipeline used to prepare data for MB loops and can achieve higher throughputs than other designs.

In order to reduce the artefacts caused by partition of a frame into Blocks, the deblocking filter is required by the standard to reduce these effects. These artefact

effects are more evident at lower data rates as shown by [30]. Instead of using the traditional filter required by the standard, [31] uses histogram statistics to analyse the correlation between offset and filtering performance which is mostly found at the position of three offsets. Then, the best offset can be searched for minimum SAE among the three candidates. This method reduces the number of computations and improves image quality. Due to the high data dependency in deblocking filters, parallelization can be complicated, but [32] overcomes the problems then exploits the implicit parallelism and reduces the synchronization overhead using the TILE64 platform.



### 3. 8X8 LUMA INTRA PREDICTION IMPLEMENTATION

Though the H.264 standard was revolutionary to image/video compression, there was need to improve the standards in the terms of quality and resolution. This led to the introduction of a new amendment to the standard called the Fidelity Range Extensions (FRExt, Amendment I). These high profiles in Table 1.4 that were introduced with these amendments support all the features and tools of the main standard. Two main coding tools too were introduced, 8X8 Transform and 8x8 Intraprediction. For Inter prediction, no new amendments are made but macroblocks larger than 8x8 are allowed to be coded by 8x8 Transform. For the high profiles, 8x8 Intra prediction is introduced which is just an extension of the 4x4 Intra prediction used in the main standard. It also has 9 modes of prediction and neighboring reconstructed reference pixels. The diagram in Figure 3.1 shows the different modes, as extension of the 4x4 intra prediction modes.

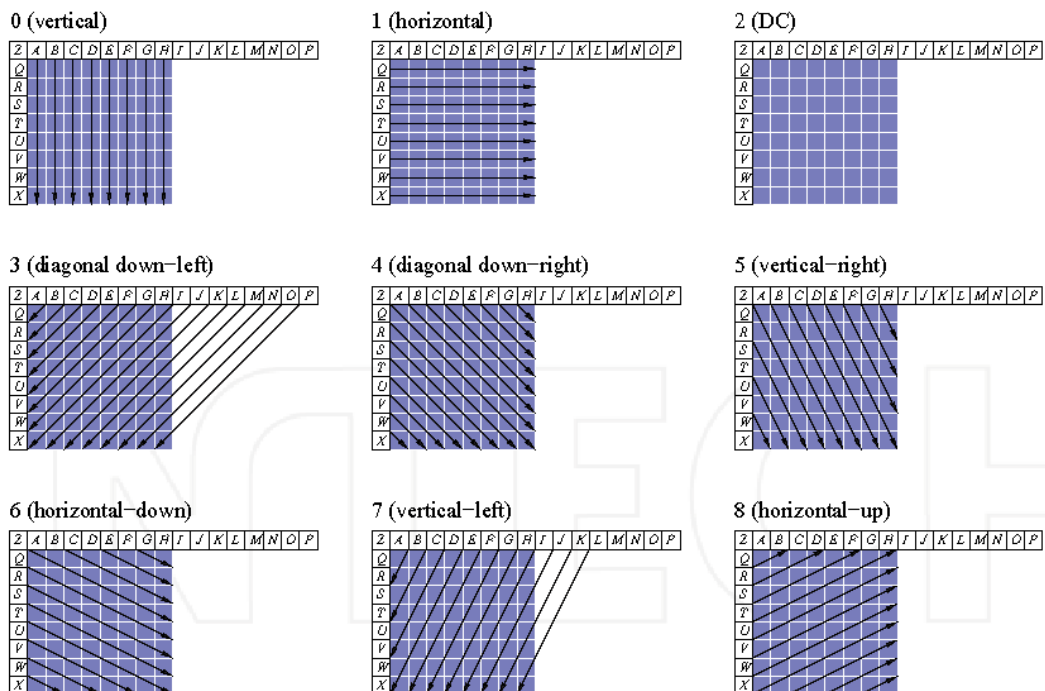


Figure 3.1. Modes of Prediction for 8x8 Luma Intra Predictions [35]

When the 8x8 Luma Intra prediction is used, the 8x8 Forward and Inverse Transform, and Forward and Inverse Quantization must also be used according to the amendment. This new intra prediction can improve I-Frame coding efficiency significantly.

### **3.1. Intra Prediction**

This design is based on two previous designs [2] and [3]. A brief review of the implementations, their advantages and shortcomings are explained below.

#### **3.1.1. H.2-4/AVC compatible intra-frame video encoder [2]**

This implementation is of the AVC/H.264 compatible Intra-frame video encoder, using 4x4 Luma Intra predictions, 16x16 Luma Intra predictions and 8x8 Chroma Intra predictions.

The main advantages of this implementation are:

1. It is a real-time implementation following the full specifications of the AVC/H.264 standard, having optional blocks like the Deblocking filter and can actually be used in the encoder.
2. Instead of using simple best mode selection algorithms like SAD, SSD, SADT, Rate Distortion Optimization is used because it is a complexity mode decision algorithm.

The short-comings of this implementation are:

1. For this real-time design to be complete, the 8x8 Luma intra prediction is not available.
2. It also tackles only the 4.2.0 chroma sample.

#### **3.1.2. Design of an 8x8 intra prediction module [3]**

This implementation is for 8x8 Luma Intra prediction for the FRExt standard. The main improvement of this implementation is to load 4x4 pixels in one clock cycle, instead of waiting for eight lines of pixels (half Macroblock) or sixteen lines of pixels (Macroblocks) to start loading pixels for best mode selection. For the best



mode selection, it also uses a simple SAD (Sum of Absolute Difference) method. Some of the main advantages of this implementation are stated below.

1. Given the fact that most equations are repeated for the prediction of one 8x8 block, most of the calculations are made available as signals, hence just reused. One of the main uses of this method is to reduce implementation time.

The main disadvantages are stated below:

1. Even though it loads 4x4 blocks of pixels, the signalling for this is too much, which complicates the design.
2. Also, there is very little presented about the implementation of Transform, Quantization, Inverse Quantization, and Inverse Transform. So, it cannot be fully known if 2D or 1D (Butterfly) was used for transform and if DSPs were used for Quantization and Inverse Quantization as clearly stated in our design.

The design implemented in this thesis solves some of the problems raised by the designs above and makes use of some of the aspects of the designs. Some of these are explained below:

### **3.2. General System Control.**

The general controller shows all the modules that make up the complete design, and how the blocks communicate with each other. The complete block diagram can be seen in Figure 3.2. The luma controller block stores all the incoming pixels in Block RAMs, generates signals and addresses to read and write pixels from and to Block RAMS. Also, it contains a shift register to shift read pixels to be sent to prediction models. The module also determines when every prediction mode is enabled with respect to the lines of frames being loaded.

The blocks Transform, Quantization, Inverse transform and Inverse quantization carry out their respective operations and more on these blocks is explained below. The mode predict block carries out prediction and sends the results to the SAD\_8 which then determines which of the predictions has the lowest sum of absolute difference (best prediction mode) and sends the results of the best prediction mode to the transform block. The reconstruction block stores all the reconstructed pixels,

reads the top and left reconstructed pixels for prediction and also reads pixels to be sent out. Finally, the reconstructed pixels (frames) are sent to the output with the help of the pixels out block.

### **3.2.1. Storing incoming pixels**

16 lines of buffer are used to store incoming pixels (one pixel per one clock cycle). 8 of these lines store first eight lines of a set of MBs and the next eight lines (9 to 16) store the last eight lines of every MB. This structure simplifies the control of the block RAMS for the reading of pixels for processing. Unlike the 4x4 implementation explained above, the processing of the MBs starts after 8 lines of pixels have been loaded. This is because, instead of processing the one macroblock before processing the next, we processed half Mbs all across before processing the other half MB. This method is possible because a block uses only top and left reconstructed pixels for its own prediction and the order of processing is as shown in Figure 3.3. The reconstructed pixels are also stored in Block Rams, so 16 RAMS are used to store these reconstructed pixels. This is as shown in the block diagram in Figure 3.4. The first 8 lines of every Macroblock are stored in the Block 1. The second 8 lines of the Macroblocks are stored in Block 2.

The first 8 lines of the reconstructed MBs are stored in Block 3 and the second 8 lines of the reconstructed MBs are stored in the Block 4. The first reconstructed pixels are released after 16 lines of pixels have been loaded. When prediction is being done on pixels on Block 1, then pixels are being released from Block 3 and when prediction is being done on Block 2, the pixels of Block 4 are being released.

### **3.2.2. Modes implementation**

From the Figure 3.1 showing the different modes of prediction, and as it was earlier on stated, the prediction modes used reconstructed pixels from the top and left of the block. But, it can be realised that, for the top of every frame and the left of the frame, some of the pixels are not available for all the modes to be used. So, for the top of the frame, only Horizontal, DC and DC UP can be used. For the left of every frame, Vertical, DC and some others can be used. But for the other parts of the frame, all other modes can be used and the details of this can be shown in Figure 3.5. This

selection takes one clock cycle. The waveform below shows the selection of modes to set, depending on the part of frame being loaded for processing. From the wave, it can be seen that, there is a signal MB\_cnt. This signal keeps track of the sets of MB being processed. It can be seen that, for the first 8x8 block, mode2\_0 is enabled and for the remaining 8x8 blocks of the top of the image, mode1, mode2\_2 and mode8. For the 8x8 blocks on the left of the image, only mode0, mode2\_1, mode3 and mode7. The remaining modes are used only at the centre of the image as required.

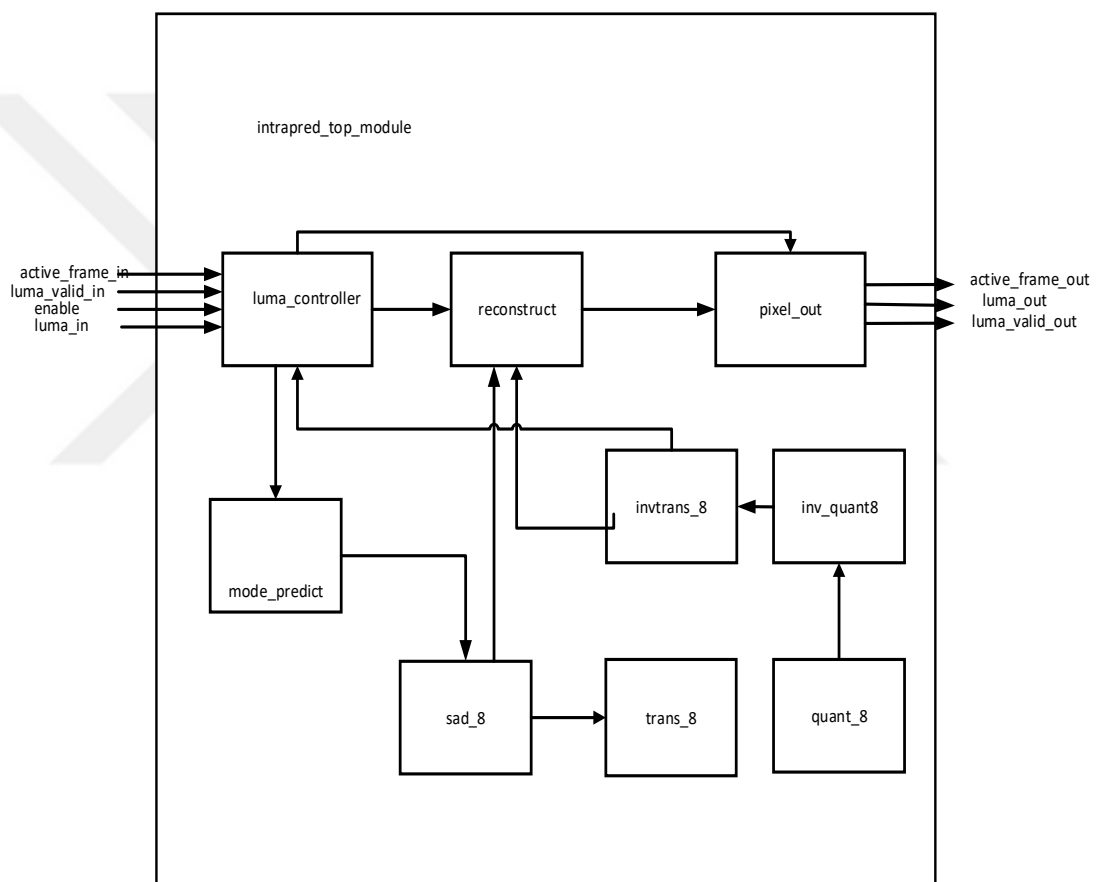


Figure 3.2. Complete Block diagram of Frext Luma Prediction

In total, there are 32 block RAMs used, 16 to store incoming pixels and 16 to store predicted reconstructed pixel.

1	2	3	4
5	6	7	8

Figure 3.3. Order of Block processing in the implementation.

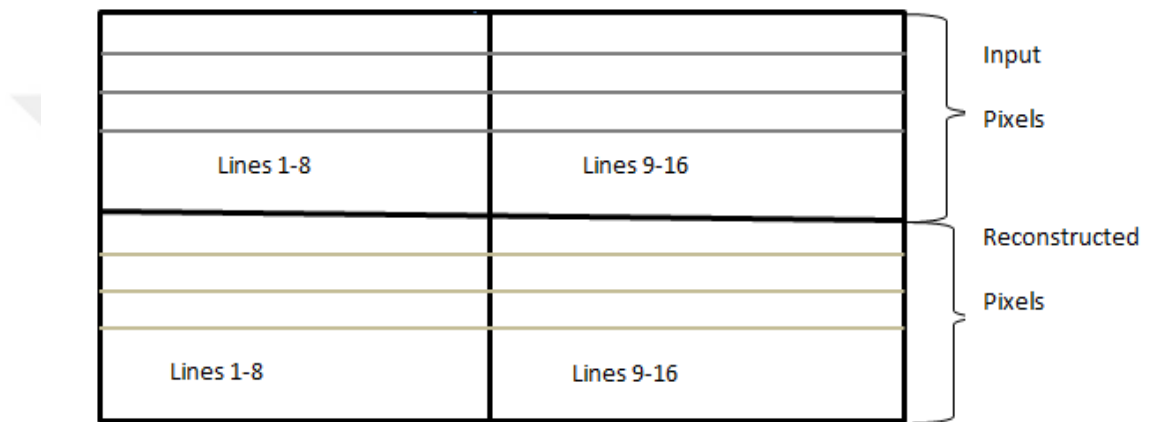


Figure 3.4. The Single Port Block Rams used in the 8x8 Luma Intra prediction

### 3.2.3. Loading data for processing

8 pixels are loaded from the single port block rams in one clock cycle, therefore 64 pixels loaded in 8 clock cycles. These pixels are loaded to a shift register to facilitate the loading and storing processes. Also in order to load reconstructed pixels, the left pixels are less complicated because before the reconstructed pixels are stored in the Block Rams (Blocks 3 and 4) of the Figure 3.4, the left pixels are extracted, stored in a flip flop and used in the next prediction sequence. As for the top reconstructed pixels, when the prediction of block 2 in Figure 3.4, is taking place and the release of the reconstructed pixels of line 1 of block 4 taking place, the pixels of line block 4 are also loaded for on-going prediction. The same cycle repeats itself when prediction of Block 1 is taking place and release of the pixels stored in block 3.

After 8 lines of pixels have been stored in the block Rams, then the loading of pixels starts, depending on which part of the MB is being processed. A signal bram\_cnt\_ff, which determines which block

RAM is being loaded with pixels, is also used to determine which pixels are being processed. When bram\_cnt\_ff is greater than 8, then pixels in the block RAMs numbers 1 to 8 are being processed and vice versa.

### 3.3. Best Mode Selection

After the modes to be predicted are selected, the prediction is done and it takes one clock cycle. Then for the best mode to be selected, the Sum of Absolute Difference (SAD) is used to calculate the best mode. The mode with the lowest SAD (minimal error,) is the best mode. SAD is calculated using equation 3.1. and is achieved in 4 clock cycles. For this to be achieved, a four stage comparator system is used, as shown in Figure 3.7. The lowest decides which of the modes will be used to make residue values available for subsequent blocks.

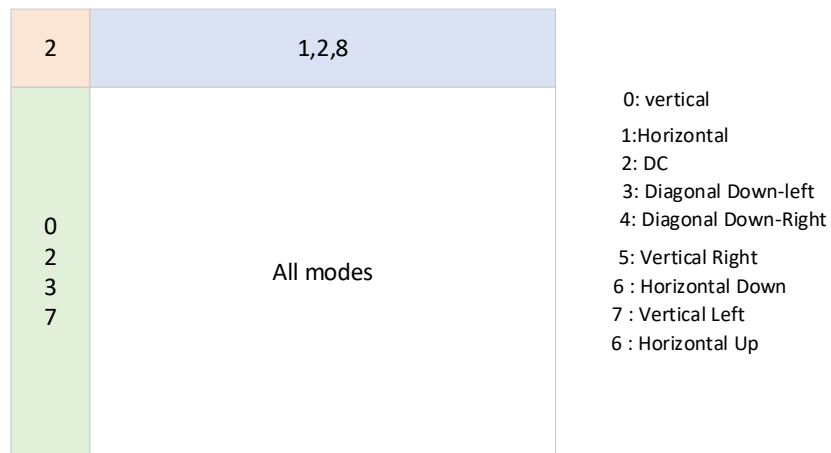


Figure 3.5. Modes that can be processed depending on the part of the frame being processed

$$SAD(\text{Orig}, \text{Pred}_m) = \sum_{x=1}^8 \sum_{y=1}^8 |\text{Orig}(x, y) - \text{Pred}_m(x, y)| \quad (3.1)$$

The residue is then used as the input to the Transform block, for the process to be continued.

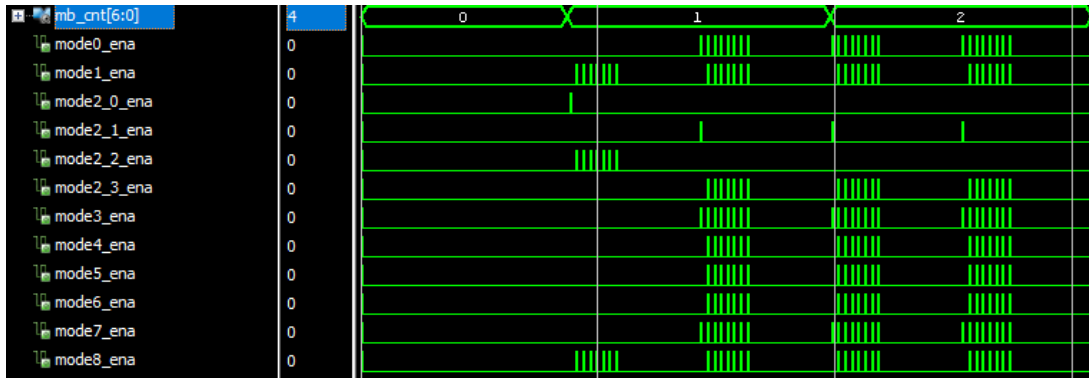


Figure 3.6. Results of the mode selection implementation.

### 3.4. Forward Transform

In the FRExt amendment, the 8x8 Integer DCT is used. Unlike other implementations that use 2D matrix multiplication or 2D full adders' implementation, the 1D butterfly method was chosen for this implementation. This is because, based on our earlier project done on these different 8x8 Integer DCT architectures, the 1D is preferable because expensive multipliers are not used and it achieves a higher operating frequency compared to the other implementations. The 1-D 8x8 Integer transform is implemented in 3 stages using the equations in the Table 3.1.

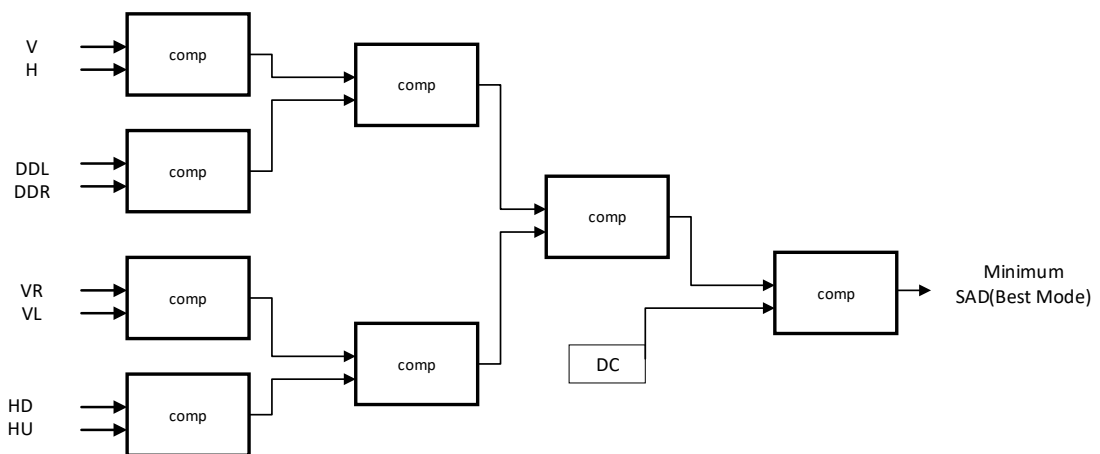


Figure 3.7. Four stage comparator to select best prediction mode

The 1D 8x8 integer DCT is accomplished with VDHL by first applying the butterfly to the columns, then taking a transpose. Secondly, the butterfly is applied to the rows and the final transpose is taken. This is as shown on the block diagram in Figure 3.8.

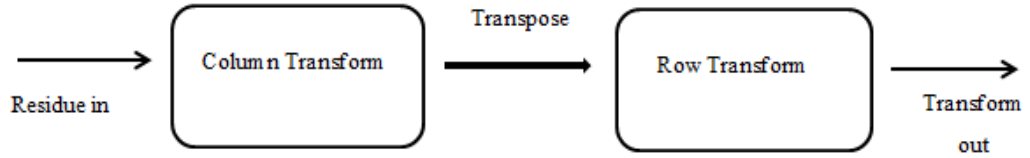


Figure 3.8. Block Diagram of 1D Forward Transform.

Hence, complete forward transform is accomplished in 6 clock cycles.

### 3.5. Forward Quantization

Quantization and post-scaling is generally achieved using the equations below. This is the point after which lossy compression is achieved. The loss of insignificant data starts from this point depending upon the parameter known as Quantization Parameter (QP). This process is achieved by using the equations below:

$$Z_{ij} = (|Y_{ij}| \cdot MF + f) \gg \text{qbits} \quad (3.2)$$

$$\text{sign}(Z_{ij}) = \text{sign}(Y_{ij}) \quad (3.3)$$

$$\text{qbits} = 15 + (P \bmod 6) \quad (3.4)$$

Where  $\gg$  represents the shift right operation, and  $Y$  is the output of forward transform. According to the software model,  $f$  is  $2^{\text{qbits}/3}$  for intra blocks or  $2^{\text{qbits}/6}$  for inter blocks.  $MF$  stands for multiplication factor and every  $MF$  depends on the quantization parameter as shown on the Table 3.2

Since the values of  $f$  change only when  $Qp$  changes, they are pre-calculated and stored in look up tables. So, there is a look up table that stores 51 values of  $f$ , and just selected when the  $Qp$  changes. The same method is applied to  $qbits$  calculation such that, instead of calculating them any time  $Qp$  changes, they are also pre-calculated and stored in the LUT and just selected when needed

Table 3.1. Equations used for calculating the Forward Integer DCT

Stage 1	Stage 2	Stage 3
$Y(0) = X(0) + X(7)$	$V(0) = Y(0) + Y(3)$	$Z(0) = V(0) + V(1)$
$Y(1) = X(1) + X(6)$	$V(1) = Y(1) + Y(2)$	$Z(1) = V(4) + (V(7) \gg 2)$
$Y(2) = X(2) + X(5)$	$V(2) = Y(0) - Y(3)$	$Z(2) = V(2) + (V(3) \gg 1)$
$Y(3) = X(3) + X(4)$	$V(3) = Y(1) - Y(2)$	$Z(3) = V(5) + (V(6) \gg 2)$
$Y(4) = X(0) - X(7)$	$V(4) = Y(5) + Y(6) + ((a4 \gg 1) + a4)$	$Z(4) = V(0) - V(1)$
$Y(5) = X(1) - X(6)$	$V(5) = Y(4) - Y(7) - ((a6 \gg 1) + a6)$	$Z(5) = V(6) - (V(5) \gg 2)$
$Y(6) = X(2) - X(5)$	$V(6) = Y(4) + Y(7) - ((a5 \gg 1) + a5)$	$Z(6) = (V(2) \gg 1) - V(3)$
$Y(7) = X(3) - X(4)$	$V(7) = Y(5) - Y(6) - ((a7 \gg 1) + a7)$	$Z(7) = -V(7) - (V(4) \gg 2)$

$$QF = \begin{pmatrix} kf_0 & kf_1 & kf_2 & kf_1 & kf_0 & kf_1 & kf_2 & kf_1 \\ kf_1 & kf_3 & kf_4 & kf_3 & kf_1 & kf_3 & kf_4 & kf_3 \\ kf_2 & kf_4 & kf_5 & kf_4 & kf_2 & kf_4 & kf_5 & kf_4 \\ kf_1 & kf_3 & kf_4 & kf_3 & kf_1 & kf_3 & kf_4 & kf_3 \\ kf_0 & kf_1 & kf_2 & kf_1 & kf_0 & kf_1 & kf_2 & kf_1 \\ kf_1 & kf_3 & kf_4 & kf_3 & kf_1 & kf_3 & kf_4 & kf_3 \\ kf_2 & kf_4 & kf_5 & kf_4 & kf_2 & kf_4 & kf_5 & kf_4 \\ kf_1 & kf_3 & kf_4 & kf_3 & kf_1 & kf_3 & kf_4 & kf_3 \end{pmatrix}$$

Figure 3.9. A sample of an 8X8 Matrix generated from the MF table

Finally, the values of MF are also stored in the look up table and just one row selected for processing when QP changes. This storage of values in the look up table is illustrated in Table.3.3. The Forward Quantization block is implemented in four clock cycles but only three of the clock cycles directly affect the prediction time. During the PARAM\_SEL state the parameters (appropriate Multiplication factors in accordance with the QF matrix) are selected from lut\_1 depending on the Quantization parameter. This is achieved when the enable is HIGH, so does not affect the prediction directly.



Table 3.2. The MF (Multiplication Factors) used for quantization computation

QPmod6	(i, j) ∈MF0	(i, j) ∈MF1	(i, j) ∈MF2	(i, j) ∈MF3	(i, j) ∈MF4	(i, j) ∈MF5
0	13107	11428	20972	12222	16777	15481
1	11916	10826	19174	11058	14980	14290
2	10082	8943	15978	9675	12710	11985
3	9362	8228	14913	8931	11984	11295
4	8192	7346	13159	7740	10486	9777
5	7282	6428	11570	6830	9118	8640

During the first clock cycle (MULT state), multiplication and sign bit extraction operations are carried with their results stored in flip-flops and the operation  $Y_{ij}.MF + f$  is performed. For the second clock cycle (BIT\_SHIFT state), right shift operations are achieved selecting the qbits value form the lut\_2.

Table 3.3. The storage of pre-calculated values in the LUTs

Qp	0	1	2	....	49	50	51
qbits(Qp)	16	16	16	...	24	24	24
f(Qp)	10923	10923	10923	...	2796203	2796203	2796203
MF(Qp)	Line0	Line1	Line2	...	Line0	Line1	Line2

Finally, during the third clock cycle (add\_sign state), the 2's operation is performed to restore the original sign of the quantized pixels. The block diagram in Figure 3.10 shows the states through which quantization is achieved.

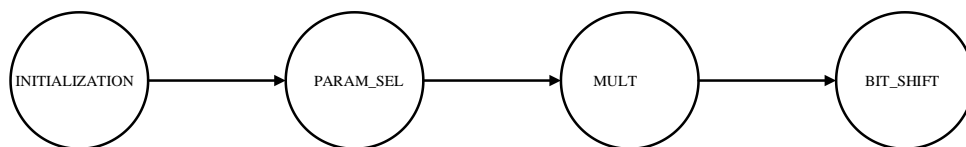


Figure 3.10. The stages through which Forward Quantization is realized.

### 3.6. Inverse Quantization

The inverse quantization is achieved using the equation 3.4.

$$W_{ij} = Z_{ij} \cdot V_{ij} \cdot 2^{\text{floor}(QP/6)} \quad (3.4)$$

Where  $Z_{ij}$  are the quantized pixels and  $V_{ij}$  are the rescaling factors which are shown on Table 3.4.

Unlike the Quantization block, the values of the matrix are not stored in look up tables. A series of multiplexers are used to implement the matrix selection with the quantization parameter as control signal and this selection is in accordance with the sample QI matrix in Fig.4.11. This block is achieved in 2 clock cycles and the first clock cycle achieves the  $Z_{ij} \cdot V_{ij}$  part of the equation. Instead of using DSPs, adders are used for this implementation as the Quantization block. During the second clock cycle, multiplication by  $2^{\text{floor}(QP/6)}$  is achieved by using an adder and multiplexer. The floor  $(QP/6)$  value changes as Qp changes with the multiples of 6. That is, for Qp values 0 to 5, floor  $(QP/6)$  is same, same goes for 6 to 12 and the pattern continues and this multiplexer decides on how the addition is done depending on the QP value. The block diagram showing the stages through which inverse quantization is achieved is shown in Figure 3.14.

Table 3.4. The MF (Multiplication Factors) used for quantization computation

QPmod6	(i, j) ∈MI0	(i, j) ∈MI1	(i, j) ∈MI2	(i, j) ∈MI3	(i, j) ∈MI4	(i, j) ∈MI5
0	20	19	25	18	24	32
1	22	21	28	19	26	35
2	26	24	33	23	31	42
3	28	26	35	25	33	45
4	32	30	40	28	38	51
5	36	34	46	32	43	58

The whole block is implemented with the block diagram in Figure 3.12.

$$QI = \begin{pmatrix} ki_0 & ki_1 & ki_2 & ki_1 & ki_0 & ki_1 & ki_2 & ki_1 \\ ki_1 & ki_3 & ki_4 & ki_3 & ki_1 & ki_3 & ki_4 & ki_3 \\ ki_2 & ki_4 & ki_5 & ki_4 & ki_2 & ki_4 & ki_5 & ki_4 \\ ki_1 & ki_3 & ki_4 & ki_3 & ki_1 & ki_3 & ki_4 & ki_3 \\ ki_0 & ki_1 & ki_2 & ki_1 & ki_0 & ki_1 & ki_2 & ki_1 \\ ki_1 & ki_3 & ki_4 & ki_3 & ki_1 & ki_3 & ki_4 & ki_3 \\ ki_2 & ki_4 & ki_5 & ki_4 & ki_2 & ki_4 & ki_5 & ki_4 \\ ki_1 & ki_3 & ki_4 & ki_3 & ki_1 & ki_3 & ki_4 & ki_3 \end{pmatrix}$$

Figure 3.11. A sample of the 8x8 QI matrixes generated from the MI table.

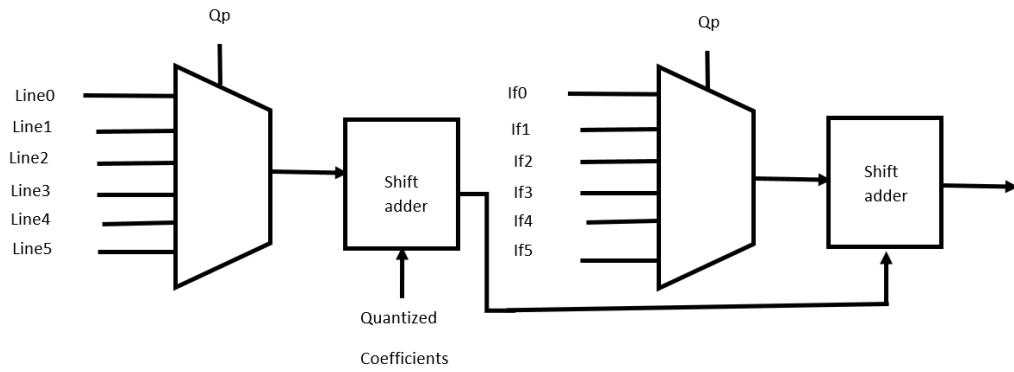


Figure 3.12. Shift adder for the Inverse Quantization using multiplexer to select appropriate line.

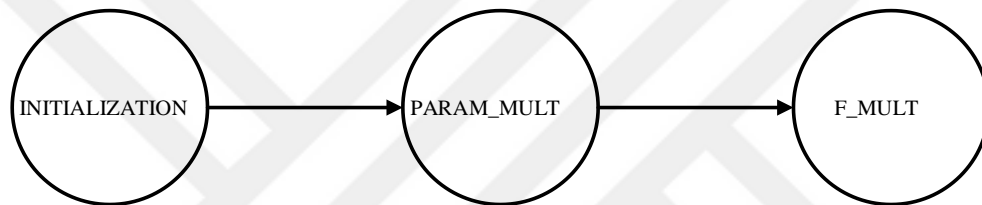


Figure 3.13. The stages through which Inverse Quantization is realized.

### 3.7. Inverse Transform

The inverse transform is calculated using the equations in Table 3.5 and achieved with the butterfly algorithm in 6 clock cycles.

3 clock cycles for the first inverse transform and 3 clock cycles for the second inverse transform. The equations for each stage are shown on the table below: The block diagram in Figure 3.15 shows the stages involved in the inverse transform calculation

### 3.8. Reconstruction

The reconstruction is done by adding the output of the inverse transform block to the output of the best predicted mode. In order to achieve this, the best predicted mode is stored in a flip flop and changes when a new prediction mode is taking place. This addition takes place in one clock cycle. The reconstructed pixels are stored in Block RAMs, so there is a total of 16 Block RAMS used to store the reconstructed pixels.

Just like the storage of the incoming pixels, the first 8 lines are used for the storage of reconstructed first 8 lines of a set of MBs and the last 8 lines are used for the reconstructed last 8 lines of every MB.

Table 3.5. The MI used for inverse quantization computation

Stage 1	Stage 2	Stage 3
$V(0) = Z(0) + Z(4)$	$Y(0) = V(0) + V(6)$	$X(0) = Y(0) + Y(7)$
$V(1) = -Z(3) + Z(5) - Z(7) - (Z(7) \gg 1)$	$Y(1) = V(0) + (V(7) \gg 2)$	$X(1) = Y(2) + Y(5)$
$V(2) = (Z(2) \gg 1) - Z(5)$	$Y(2) = V(4) + V(2)$	$X(2) = Y(4) + Y(3)$
$V(3) = Z(1) + Z(7) - Z(3) - (Z(3) \gg 1)$	$Y(3) = V(3) + (V(5) \gg 2)$	$X(3) = Y(6) + Y(1)$
$V(4) = Z(0) - Z(4)$	$Y(4) = V(4) - V(2)$	$X(4) = Y(6) - Y(1)$
$V(5) = -Z(1) + Z(7) - Z(5) - (Z(5) \gg 1)$	$Y(5) = (V(3) \gg 2) - V(5)$	$X(5) = Y(4) - Y(3)$
$V(6) = Z(2) + (Z(6) \gg 1)$	$Y(6) = V(0) - V(6)$	$X(6) = Y(2) - Y(5)$

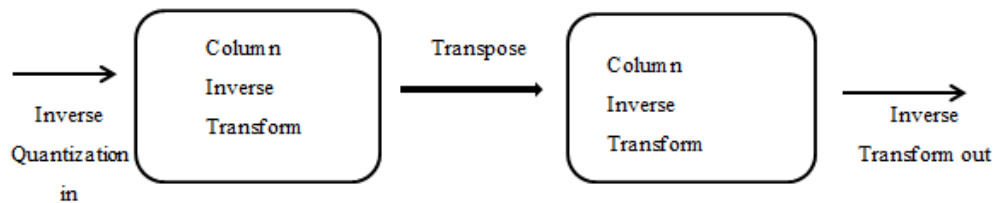


Figure 3.14. Block diagram of 1D Inverse Transform

### 3.9. Output pixels

At the level of this block, reconstructed pixels are sent from the system. As mentioned in the processes above, the release of pixels starts when the first pixel of the 17<sup>th</sup> line of the frame starts loading. When the pixels of the first eight lines of the sets of frames are loading, the pixels of the first eight frames of the reconstructed pixels are also being released. Also, when the last eight lines (9 to 16) of the frames are loading, the last eight lines of the reconstructed pixels are also release

## 4. RESULTS

The results are presented in three main groups; synthesis, simulation and visual results.

### 4.1. Synthesis Results

The synthesis results for the independent blocks are also presented in Table 4.1 and results of the completed design are presented in Table 4.2. This is because, the blocks can also be used in other designs are independent IPs or modules. It takes a total of 22 clock cycles for one 8X8 block to be predicted; hence it takes a total of 88 clock cycles for one MB to be predicted.

Table 4.1 Synthesis Results for the independent blocks.

BLOCK	LUT	FF	DSPs	BUFG	Maximum Frequency
Sad_8 (SAD calculation)	9018	1425	---	1	200MHz
Forward Transform	8155	4848	---	1	200MHz
Forward Quantization	4802	2538	62	1	200MHz
Inverse Quantization	1906	1885	---	1	200MHz
Inverse Transform	11429	5784	---	1	200MHz
Complete Block Design	---	---	---	---	195 MHz

### 4.2. Simulation Results

This section contains the waveforms for the main blocks of the design. The waveform in Figure 4.1 shows the loading of incoming pixels, for the first eight lines of the frames and also the release of predicted pixels. The first eight lines are for incoming pixels, enable and write and the last eight lines on the graph are for

released pixels enable. This same procedure for the release and loading of pixels for lines 9 to 16 is shown on Figure 4.2. For lines BRAMs 8 and 16 (last lines Figure 4.1 and Figure 4.2), it can be seen that, the read enable is also HIGH after the reconstructed pixels have been loaded. These are the reconstructed pixels that are used for further prediction of future blocks. The signals to enable prediction modes in accordance with the design that is, enabling only some modes for the top part of the frame and for the left part of the frame are as shown in Figure 4.6.

Table 4.2. Synthesis Results for the for the complete block.

Resource	Utilization	Available	Utilization
LUT	38016	218600	17.39%
LUT RAM	8	70400	0.01
FF	19351	437200	4.43
BRAM	16	545	2.94
DSP	62	900	6.89
IO	23	362	6.35
BUFG	1	32	3.13

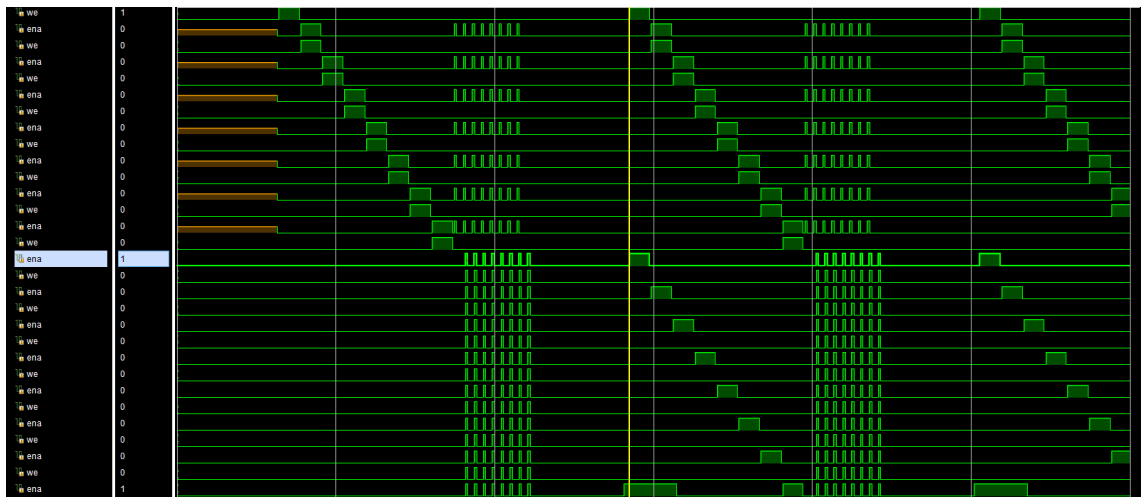


Figure 4.1. Waveform for BRAMs (1 to 8) for the incoming pixels (first eight lines on the wave form) and for reconstructed pixels (last eight lines on the waveform).

The waveform showing the predicted pixels is shown in Figure 4.3. It can be seen that, in accordance with Figure 4.8, not all prediction modes are active at the same time. They can only be active at the same time when pixels of the middle of the frame (luma\_in (W-8, L-8)) are being loaded. The Figure 4.4 shows the outputs of

inputs and outputs of SAD block. It can be seen that, the output (best prediction mode and the corresponding residue) are available after 4 clock cycles.

Figure 4.5 shows the output of the first (Forward) Transform block is available 3 clock cycles later after the residue is available. Also 3 clock cycles later after the first forward transform, the results of the second forward transform are available as shown in Figure 4.6. When transform process is done, quantization starts with selection of appropriate values for the MF

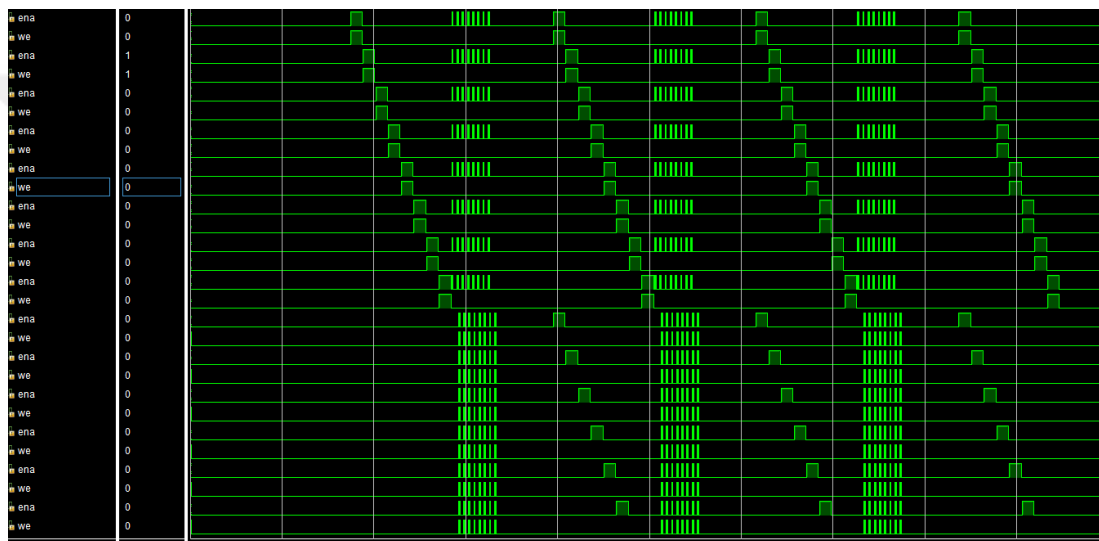


Figure 4.2. Waveform for BRAMs (9 to 16) for the incoming pixels (first eight lines on the wave form) and for reconstructed pixels (last eight lines on the waveform)

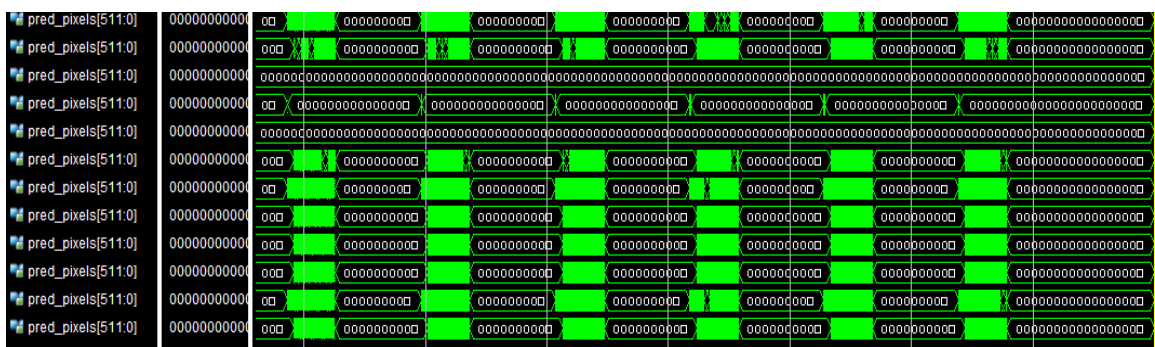


Figure 4.3. Outputs of mode\_predict block

matrix, the quantization process itself which takes and the whole process takes 3 clock cycles as shown in Figure 4.7. The Inverse Quantization takes 2 clock cycles as





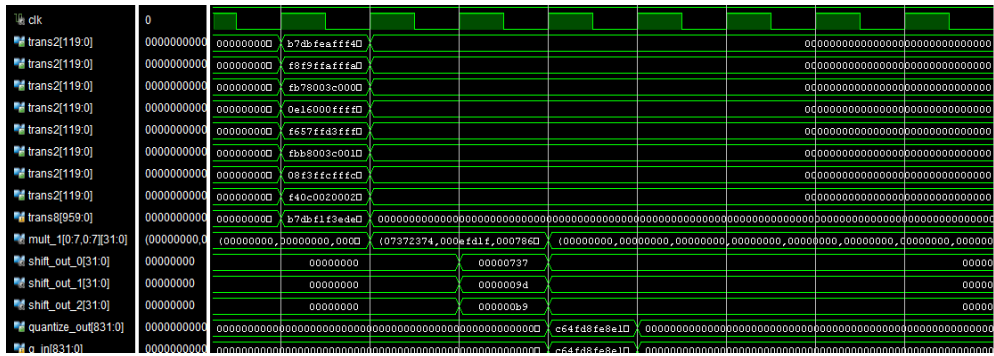


Figure 4.6. The results of Quantization

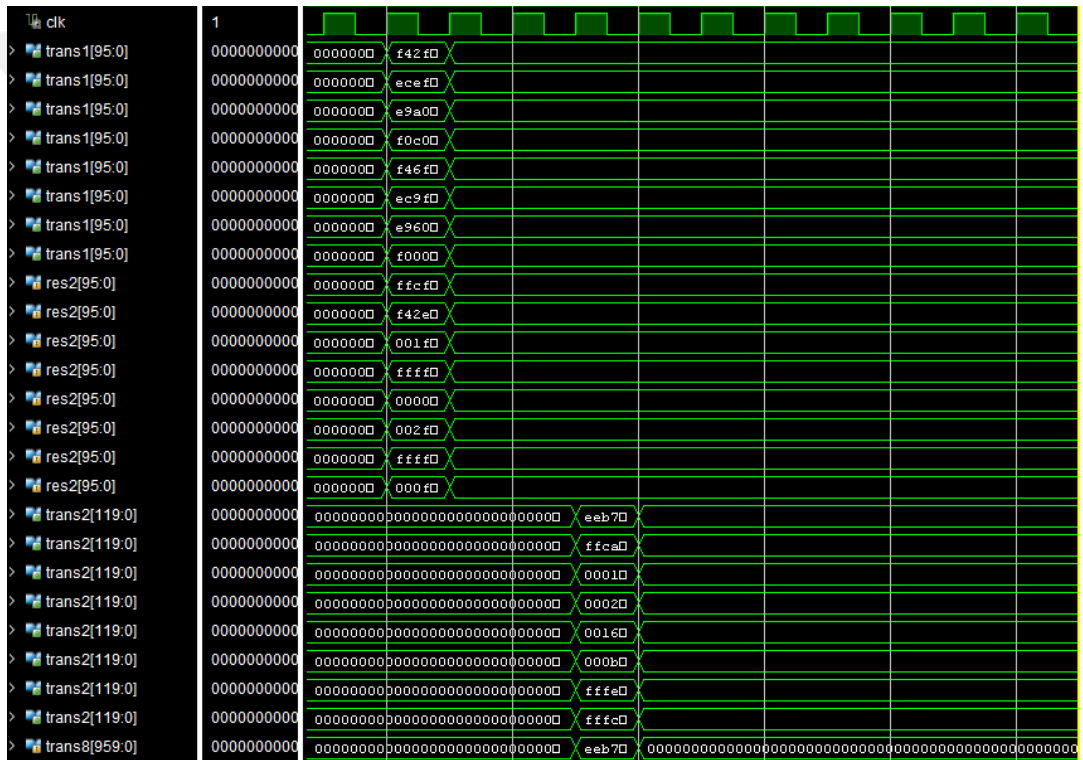


Figure 4.7. The results of the second forward transform

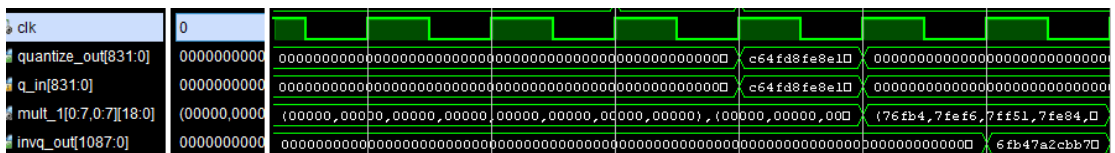


Figure 4.8. The results of the inverse quantization



## The output frames

The sample of output frames is shown in Figure 4.11 with different Quantization parameters.



Figure 4.11. Original and predicted frames with Quantization Parameters 0, and 15 using 8x8 Luma Intra prediction

Also, the output of a Luma 4x4 intraprediction is shown in Figure 4.12.



Figure 4.12. Original and predicted frames with Quantization Parameters 0, and 15 using 4x4 Luma Intra Prediction

## 5. DISCUSSION

The implementation of 8x8 luma presented in this project achieves a higher maximum frequency (195 MHz) compared to [4] with a maximum frequency of 129.34 MHz and [3] with the maximum frequency not specified.

The presented design uses 16 BRAMs for the storage of incoming pixels and reconstructed pixels, while [4] uses more since its prediction starts after 16 lines have been loaded. Compared to other designs, this design uses more flip-flops. This is partly caused by the fact that, other designs store all reconstructed pixels in BRAMs but, the presented design stores only top reconstructed pixels in BRAMs and left in Flip-flops. For the total number of processing clock cycles for one MB, the implemented design uses 88 clock cycles and [4] uses 72 clock cycles. For both blocks, the total number of clock cycles will increase in real time implementations due to the addition of extra blocks like the Deblocking filter and the use of more efficient algorithms for best mode selection. Also, our design uses more DSPs because part of the Quantization block is designed without shift adders. Table 5.1 gives a more detailed comparison between the three different implementations.

The total number of clock cycles for processing one MB is lower of [4] and this could partly be because of the Transform block and Quantization blocks implemented. Nothing is mentioned in [4] about the Transform, Quantization, Inverse Transform and Inverse Quantization blocks. That is, 2D implementations could have been used for the Transform and fewer clock cycles used for quantization block implementations.

Though our design achieves a high maximum frequency, it can be improved by using a more accurate efficient mode selection algorithm like RDO also used in the JM reference software. (Sorting algorithms). Also, the Quantization block can be implemented with shift add operators like in [23] instead of using DSPs, even though DSPs make the design less complicated. Our design uses the butterfly algorithm which is very efficient according to the analysis done in [20] and [21].

Table 5.1. Synthesis Results for the independent blocks and for the complete block.

Property	Implemented Design	8x8 Luma Design [4]	H.264 Intra-Frame Video Encoder [3]	An Efficient Intra Prediction Hardware [30]	High Through-Put and Low Complexity H.264 [20]
Maximum Frequency	195 MHz	129.34	Not mentioned	70MHz	94MHz
Clock Cycles used to Process one MB	88	72	--	384	48
Number of LUTs	38016	26109	--	--	4465
Number of Flip flops	19351	--	--	--	2412
Number of lines available before prediction starts	8	16	16	16	Not mentioned
Best Mode Selection Used	Sum of Absolute Difference (SAD)	Sum of Absolute Difference (SAD)	Rate Distortion Optimization (RDO)	Not mentioned	Sum of Absolute Difference (SAD)
Integer DCT algorithm used	Buffer Fly Algorithm	Not mentioned	Buffer Fly Algorithm	Not mentioned	Not mentioned
Quantization and Inverse Quantization Blocks	DSPs for Quantization	Not mentioned	Improved Multipliers for Quantization	Not mentioned	Not mentioned

The butterfly can be improved by using the pipe lining architectures (pipeline architectures). Also, Quantization and Inverse Quantization blocks can still be implemented with multipliers using efficient multiplication algorithms like the one used in [3].

Even though this design reduces the complexity for the storage of predicted pixels (incoming and predicted pixels), uses fewer number of BRAMs used, achieves a high maximum frequency, it still uses more resources compared to other designs.

The comparison between the outputs of the Luma 8x8 and Luma 4x4 intra predictions can be seen in Table 6.2. The mean square error is calculated by using the formula

$$MSE = \frac{1}{MN} \sum_{y=1}^M \sum_{x=1}^N (I(x,y) - I'(x,y))^2 \quad 5.1$$

Table 5.2. Error of the Implemented Luma 8x8 Intraprediction and Luma 4x4 Intra prediction

Prediction mode	Qp = 0 ,MSE	Qp = 15, MSE
Luma 8x8	36.0054	36.2290
Luma 4x4	0.0155	1.0587

The error of the 8x8 intra prediction is larger than that of the 4x4 luma intra prediction. This is because, the prediction matrix is larger and relationship with reconstructed neighbouring pixels is not fully explored. Compared to the Luma 4x4, signalling is less and the prediction is less complicated.

## 6. CONCLUSIONS AND SUGGESTIONS

A less complex  $8 \times 8$  luminance intra prediction module, which is part of the H.264/AVC CODEC, is implemented with respect to the specifications of the standard. The block-based prediction module exploits similarities between neighbouring image samples in the same video frame to reduce redundancy and compress the video.

The intra prediction block is made less complicated by first prediction half of all the macro blocks before predicting all the other halves. This does not only reduce complexity but also enables prediction to start after 8 lines of incoming pixels have been loaded into the Block Rams. The Forward and Inverse Transform Blocks use the butterfly algorithm which enables processing to be done only by shift adders. Forward Quantization uses DSPs which increases the percentage of resources used. The Inverse Quantization block does not use Look up tables and is implemented only with a series of Multiplexers and Shift adders. This reduces complexity but, the number of LUT used sharply increases. The frequency of the entire system is less than the frequency of the individual blocks, due to the primitive best mode selection algorithm used.

Even though this block can be used in a CODEC design and the individual designs can also be used in designs, it can be improved mainly by changing the mode selection algorithm, improving the butterfly using pipelining, avoidance of DSPs in the Quantization block and the introduction of a Deblocking filter to remove artefacts.

## REFERENCES

- [1] Sanchez V., Nasiopoulos P. and Abugharbieh R., Lossless Compression Of 4D Medical Images Using H.264/AVC,II. *International Conference on Acoustics, Speech, and Signal Processing*, Toulouse, France, 14-19 May 2006.
- [2] Richardson I.E., *Video Coding for next Generation Multimedia*. 2 ed., John Wiley & Sons Ltd, West Sussex, United Kingdom, 2003.
- [3] Günay Ömer, Design of H.264/AVC Compatible Intra-Frame Video Encoder on FPGA Programmable Logic Devices, Master Degree Thesis, Middle East Technical University, The Graduate School of Natural and Applied Sciences, Ankara,2014, 12617751.
- [4] Kim Trønnes, Design of an 8x8 Intra Prediction Module, Master Degree Thesis, Norwegian University of Science and Technology, Department of Electronics and Telecommunications, 2014.
- [5] Sullivan Gary J., Topiwala Pankaj N., The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions, *Optical Science and Technology, the SPIE 49<sup>th</sup> Annual Meeting*, Denver Colorado, United States, 2-6 August 2004.
- [6] Amer I., Wael Badawy, Graham Jullien, A High-Performance Hardware Implementation of The H.264Simplified 8x8 Transformation and Quantization, *International Conference on Acoustics, Speech, and Signal Processing*, Philadelphia, PA, USA , 23-23 March 2005.
- [7] Jafari Mehdi, Kasaei Shohreh, Fast Intra- and Inter-Prediction Mode Decision in H.264 Advanced Video Coding, *10th IEEE Singapore International Conference on Communication Systems*, Singapore, Singapore, 30 Oct.-1 Nov. 2006
- [8] Richardson Iain E. G., *H.264 and MPEG-4 Video Compression*, 1st ed., John Wiley & Sons Ltd, West Sussex, United Kingdom ,2004.
- [9] Garcia Diogo Caetano, De Queiroz Ricardo L., Least-Squares Directional Intra Prediction in H.264/AVC, *IEEE SIGNAL PROCESSING LETTERS*, 2010, **17**(10), 831 - 834.



- [10] Muralidha P., Vasundhara Devi R., Ramarao C.B., Murthy N.S., An Efficient Architecture for H.264 Intra Prediction Mode Decision Algorithm, *Recent Researches in Communications, Automation, Signal Processing, Nanotechnology, Astronomy and Nuclear Physics*, Cambridge, UK, 20-22 February 2011.
- [11] Kibum Suh, An Efficient Architecture of Intra Prediction and TQ/IQIT Module of Video Encoder, *Indian Journal of Science and Technology*, 2016, 9(43), 352-355.
- [12] IMRAN ULLAH KHAN, ANSARI M. A., Overview and Implementation of Intra predictions for H.264/AVC Video CODEC, *International Journal of Electronics and Communication Engineering*, 2014, 3(4), 177-186.
- [13] Al-Jammas Mohammed H., Hamdoon Noor N., FPGA Implementation of Intra Frame for H.264/AVC Based DC Mode, *International Journal of Computer Engineering and Information Technology*, 2017, 9(11), 264–270.
- [14] Patel Jignesh, Suthar Haresh, Gadit Jagrut, VHDL Implementation of H.264 Video Coding Standard, *International Journal of Reconfigurable and Embedded Systems* 2012, 1(3), 95-102.
- [15] Adda Chahrazed, Benyamina Abou Elhassen, Design of the H264 application and Implementation on Heterogeneous Architectures, *International Journal of Computer Applications*, 2017, 180 (7).
- [16] Loukil Hassen, Werda Imen, Masmoudi Nouri, Atitallah Ahmed Ben, Kadionik Patrice, FPGA Design of an Intra  $16 \times 16$  Module for H.264/AVC Video Encoder, *Circuits and Systems*, 2010, 1, 18-29.
- [17] Bharathi S.H, Nagabhushana Raju K., Ramachandran S., Implementation of Horizontal and Vertical Intra prediction Modes for H.264 Encoder, *International Journal of Electronics and Communication Engineering*, 2011, 4(1), 105-114.
- [18] Orlandić Milica, Svarstad Kjetil, A High-Throughput and Low-Complexity H.264/AVC Intra  $16 \times 16$  Prediction Architecture for HD Video Sequences, *21st Telecommunications Forum Telfor (TELFOR)*, Belgrade, Serbia, 26-28 Nov. 2013.
- [19] Ringnyu B., Tangel A., Implementation Of Forward  $8 \times 8$  Integer DCT For H.264/AVC Frext., *The Eurasia Proceedings of Science, Technology, Engineering & Mathematics* , 2017, 1(1), 353- 358.
- [20] Ringnyu B., Tangel A., Implementation of Different Architectures of forward  $4 \times 4$  integer DCT for H.264/AVC encoder, *10th International Conference on Electrical and Electronics Engineering (ELECO)*, Bursa, Turkey, 30 Nov.-2 Dec. 2017.

- [21] Park Jeong Sung, Ogunfunmi Tokunbo, A New Hardware Implementation Of The H.264 8x8 Transform And Quantization, *International Conference on Acoustics, Speech and Signal Processing*, Taipei, Taiwan, 19-24 April 2009
- [22] Ozgur Tasdizen, Ilker Hamzaoglu, A High Performance and Low-Cost Hardware Architecture for H.264 Transform and Quantization Algorithms, *13th European Signal Processing Conference*, Antalya, Turkey, 4-8 Sept. 2005.
- [23] Nadeem Muhammad, Wong Stephan, Kuzmanov Georgi, An Efficient Realization of Forward Integer Transform in H.264/AVC Intra-frame Encoder, *International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation*, Samos, Greece, 19-22 July 2010.
- [24] Lubobya Charles S., Dlodlo Mqele M., Jager Gerhard De., Ferguson Keith L., Optimization of 4x4 Integer DCT in H.264/AVC Encoder, *The 14<sup>th</sup> annual Southern Africa Telecommunication Networks and Applications Conference*, East London, South Africa, 4-7 September 2011.
- [25] Marques R., Silva V., Faria S., Navarro A., Assuncao P., Fast Conversion of H.264/AVC Integer Transform Coefficients into DCT Coefficients, *Proceedings of the International Conference on Signal Processing and Multimedia Applications*, Setúbal, Portugal, August 7-10, 2006.
- [26] Logashanmugam.E, Ramachandran.R, An Efficient Hardware Architecture for H.264 Transform and Quantization Algorithms, *International Journal of Computer Science and Network Security*, 2008, 8(6), 300-3012.
- [27] Atitallah Ben, Loukil H., Kadionik P., Masmoudi N., Advanced Design of TQ/IQT Component for H.264/AVC Based on SoPC Validation, *WSEAS Transactions on circuit and systems*, 2012, 11 (7), 211-223.
- [28] Sahin E. and Hamzaoglu I., An Efficient Intra Prediction Hardware Architecture for H.264 Video Decoding, *10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2007)*, Lubeck, Germany, 29-31 Aug. 2007.
- [29] He Xun, Zhou Dajiang, Zhou Jinjia, Goto Satoshi, High profile intra prediction architecture for H.264, *International SoC Design Conference (ISOCC)*, Busan, South Korea, 22-24 November 2009.
- [30] Raja Gulistan, Mirza Muhammad Javed, In-loop Deblocking Filter for H.264/AVC Video, *2<sup>nd</sup> International Symposium on Communications, Control and Signal Processing*, Marrakech, Morocco, 13-15 March 2006, 235-240.
- [31] Hsia Shih-Chang, Hsu Wei-Chih, Lee Sheng-Chieh, Low-complexity high-quality adaptive deblocking filter for H.264/AVC system, *Signal Processing: Image Communication*, 2012, 27(7), 749-759.

- [32] Yan Chenggang et al, Parallel Deblocking Filter For H.264/AVC Implemented On Tile64 Platform, *IEEE International Conference on Multimedia and Expo*, Barcelona, Spain, 11-15 July 2011.
- [33] S. Kwon, A. Tamhankar and K.R. Rao, Overview of H.264 / MPEG-4 Part 10, *Visual Communication and Image Representation*, 2006, 17, 186-216.
- [34] Chen Yi-Hau, Tsai Chen-Han, Chen Yu-Jen, Chen Liang-Gee, Algorithm and architecture design for intra prediction in H. 264/AVC high profile, *Picture Coding Symposium*, Lisboa, Portugal, 7-9 November 2007.
- [35] <https://www.intechopen.com/books/signal-processing/spatial-prediction-in-the-h-264-avc-frext-coder-and-its-optimization>, (Date Visited: 1<sup>st</sup> August 2018).



## PUBLICATIONS AND WORKS

- [1] **Ringnyu B. A.**, Tangel A., Implementation of Different Architectures of forward  $4 \times 4$  integer DCT for H.264/AVC encoder, *10th International Conference on Electrical and Electronics Engineering (ELECO)*, Bursa, Turkey, 30 Nov.-2 Dec. 2017.
- [2] **Ringnyu B. A.**, Tangel A., Implementation of Forward  $8 \times 8$  Integer DCT for H.264/AVC FRExt, *The Eurasia Proceedings of Science, Technology, Engineering & Mathematics*, 2017, **1**(1), 353 – 358.



## **BIBLIOGRAPHY**

Antoinette was born in Kumbo, in the Northwest Region of Cameroon on the 6<sup>th</sup> February 1991. She did her primary education in Tabenken , secondary and high school in Kumbo. In 2010, she was admitted to the Faculty of Engineering and Technology, University of Buea to study Electrical and Electronics Engineering. While studying there, she carried out internships at Matrix Telecoms where she eventually worked after graduation in December 2014. In 2015, she was awarded the Turkish Government scholarship, and studied the Turkish Language for one academic year. In September 2016, she started her course work at the Department of Electronics and Communications Engineering at Kocaeli University, Turkey. She did an Internship at YongaTek Elektronik, where she started working on FPGAs. From April 2018, she has been interning at Kuantek Elektronik, still working on FPGAs. In January 2019, she completed an MSC in Electronics and Communication Engineering.