University of Mosul

College of Electronic Engineering

FPGA Implementation of H.264/ AVC Decoder

A Thesis Submitted by

Rusul Nabeel Ahmed

To

The Council of College of Electronic Engineering

University of Mosul

In Partial Fulfillment of the Requirements

For the Degree of Master of Sciences

In

Computer and Information Engineering

Supervised by

Dr Mohammed Hazim Aljammas

2018 A.C. 1439 A.H.

﴿ وَلَوْ أَنَّمَا فِي الْأَرْضِ مِنْ شَجَرَةٍ
أَقْلَامٌ وَالْبَحْرُ يَمُدُّهُ مِنْ بَعْدِهِ سَبْعَةُ أَبْحُرٍ
مَا نَفِدَتْ كَلِمَاتُ اللَّهِ إِنَّ اللَّهَ عَزِيزٌ
حَكِيم ﴾

لقمان: ٢٧

Supervisor's Certification

I certify that the dissertation entitled (**FPGA Implementation of H.264 /AVC Decoder**) was prepared by **Rusul Nabeel** under my supervision at the Department of Computer and Information Engineering, University of Mosul, as a partial requirement for the Master of Science Degree in Computer and Information Engineering.

Signature:

Name: Dr. Mohammed Al.Jammas.

Department of Computer and Information Engineering

Date:

Report of Linguistic Reviewer

I certify that the linguistic reviewer of this dissertation was carried out by me and it is accepted linguistically and in expression.

Signature:

Name:

Date: : / /2018

Report of the Head of Department

I certify that this dissertation was carried out in the Department of Communication Engineering. I nominate it to be forwarded to discussion.

Signature:

Name:

Date: / /2018

Report of the Head of Postgraduate Studies Committee

According to the recommendations presented by the supervisor of this dissertation and the linguistic reviewer, I nominate this dissertation to be forwarded to discussion.

Signature:

Name:

Date: / /2018

Committee Certification

We the examining committee, certify that we have read this dissertation entitled (**FPGA Implementation of H.264 AV\C Decoder**) and have examined the postgraduate student (**Rusul Nabeel**) in its contents and that in our opinion; it meets the standards of a dissertation for the degree of Master of Science in Communication Engineering.

Signature:	Signature:
Name:	Name:
Head of committee	Member and Supervisor
Date: / /2018	**Date: / /2018**

Signature:	Signature:
Name:	Name:
Member	Member and Supervisor
Date: / /2018	**Date: / /2018**

The college council, in its …… meeting on / /2018, has decided to award the degree of Master of Science in Communication Engineering to the candidate.

Signature:

Name:

Dean of the College

Date: / /2018

Dedication

My God is not the night except with your thankfulness and the daytime to your obedience. And the moments are not forgotten except your memory . Do not wait for the hereafter but Affuk . Do not pray paradise only to see you.

(God almighty).

To this who reached the message and led the Secretariat. And advised the nation. To the prophet of mercy and light of the worlds.

(Prophet Muhammad peace be upon him).

To whom God has given glory and glory. To those who taught me tender without waiting . To whom I carry his name with all pride . I ask God to extend in your age to see the fruit has come harvested after a long waiting and will remain your words as stars I promise today and tomorrow and forever.

(My dear father).

To my angel in life. to the meaning of love and to the meaning of compassion and dedication .. To the smile of life and the secret of existence and the secret of my success.

(To my beloved mother).

With all love ... to the companion of my husband's path which is my soul and all my ambition.

"Who walked with me towards the dream. step by step

We sowed it together ... and harvested it together.

And we will stay together . God willing.

(To My husband and my lover).

To a burning candle illuminates the darkness of my life .To those who have gained strength and love without limits .To whom you know the meaning of life.

(To my lovely sister).

To those who have learned the science of siphons of knowledge, including the professors of the Department of Computer and Electronics.
(To my distinguished lecturers)

To my department in which my future flourished.

(Department Of Computer and Information Engineering).

To my college with which my future was built and laid the foundation stone for my scientific career.

(Electronics Engineering College).

To this young and mighty scientific edifice university.

(University Of Mosul)

Thanks and praise

After thanking Allah for his graces by putting me in this degree of studying, I would like to give my deepest thanks for the one who supports me and guides me to complete the thesis by giving me precious advices during the research period , Dr. Mohammed Al-jammas.

I also thanks all who help me in the Computer and Information engineering department and electronics department lecturers specially T.A ' Khalid Fazaa 'in the Electronics Department and T.A 'Mamoun abd-aljabbar' in Computer and Information Department.

Abstract

The H.264/ AVC considers as a good technical method for encoding and decoding most types of video formats and provides very well results. Its power comes from its algorithm in work and how it is encoding the video frames and it is trying to reach to the ideal access to gain original video. Our work involves applying the encoding and decoding process of the standard using MATLAB (2013Ra) program. The work is focusing in inter frame prediction using the (IBBB) frame pattern with (28) frames and encoded them in a very record time equal to (0.2142) sec, and a decoded time equal to (0.0994) sec. The video that was subjected to encoding and decoding processing was (Xylophone video name) with (240x320) size. In this way, we reached to compression rate =71%, this is a very good value.

In the second part of this thesis, involve a practical applying of the H.264/ AVC in FPGA kit (Spartan 6). Intra prediction is using in the second part because of the determinants that are within the design of the kit. We were able to achieve the conditions by working within the appropriate frequency for the kit and within the inner silicon area of the kit and this is clear from the results we have obtained.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Name
ALU	Arithmetic Logic Unit
AVC	Advanced Video Coding.
B frame	Bi-directional frame
CABAC	Context-adaptive binary arithmetic coding
CAVLC	Context-adaptive variable-length coding
CD	Compact disc
Chroma	Chrominance
CPU	Central Processing Unit
DCT	Discrete Cosine Transform
DCM	Digital Clock Manager.
DDR2	Double data rate synchronous Dynamic Random-access memory .
DVD	Digital Video Disc.
FPGA	Field-Programmable Gate Array
GOP	Group Of Picture
HD	High Definition.
I frame	Intra frame.
IDCT	Inverse Discrete Cosine Transform

ISE	Institute of Software Engineers
ISO	International Organization for Standardization
ITU-T	International Telecommunication Union Telecommunication Standardization Sector.
JVT	Joint Video Team
Luma	Luminance
LUT	Look Up Table
MAD	Mean Absolute Difference
MATLAB	Matrix Laboratory
MPEG	Moving Picture Experts Group
MSE	Mean Squared Error
NAL	Network Abstraction Layer
P frame	Predicted frame
PAL	Phase Alternating Line
RAM	Random Access Memory
RTL	Radio Television Luxembourg.
SAD	Sum Absolute Difference
SP601	Spartan 601 kit
TSS	Three Step Search
TV	Television

UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
UWB	Ultra Wideband
VCEG	Video Coding Experts Group
VHDL	Hardware Description Language

Chapter One

Introduction

1.1 Background

In our technological world, the technology and multimedia are increasing exponentially day by day. Video conferencing and telephony, TV, streaming Video/Audio online, etc. are demand in video industry. These applications require a high bandwidth, large storage, and high latency time to send on network. Achieving these requirements, we must reduce this huge data by compressing the original video at the transmitter before sending. At the receiver, it required to decompress the video file to retrieval original video. Many different video codec standards such as H.261, MPEG-1, MPEG-2, H.263, and H.264 are implement.

H.264 is the aim of this research and it is the international video coding standard. This protocol was developed jointly by International Telecommunication Union–Telecommunication Standardization Sector (ITU-T), and International Organization for Standardization (ISO). Video coding has become an integral part for every computing device from televisions and computers to portable devices such as cell phones. Achieving high quality resolution over limited bandwidth has led to the development of the H.264 video standard providing greater encoding performance [1][2].

1.2 Application of H.264

H.264 / AVC standard enters in almost necessary life joints that need to use high-resolution videos. This indicates that it has a flexibility in applying it, because of its ease and the richness of the results obtained through it. These are some applications of the standard:

- Used to monitor the movement of the child inside the house if it contains cameras dedicated or any place was installed surveillance cameras inside. The cameras are connect via the internet, if a large size of video file then, we will need a large bandwidth, for this will reduce the size by the H.264/ AVC standard.
- Used in a smart house that controls remotely through the internet, the cameras installed all over the house to monitor.
- Used in medical application. In last some years and through the improvement and growing in technology the turned attention is to the medical healthcare field. Easy access to the medical videos and the three dimensional (3D) medical data sets, such as computed tomography (CT), magnetic resonance imaging (MRI), echocardiograph and so on, provides doctors with best capability to analyze and diagnose the patient conditions. Medical videos and 3D medical data sets require large amounts of memory for storage and large transmission bandwidth in telemedicine applications. This calls for video compression and decompression to reduce the amount of data needed to represent the video. The best choice for this reasons, is H.264/ AVC.
- In internet, the security is the magic word of sending and receiving through the internet. The security of video communication is a challenging task especially for wireless video applications. The perceptual video encoding scheme is proposed by exploiting the special feature of the entropy coding and decoding in H.264. The encoding scheme is compose of coded block pattern changing, sign of trailing ones scrambling and levels of the non-zero coefficients encryption. The important syntax elements and sensitive coded elements are chose to encode using mathematical operations, permutation

and etc., so the H.264 is suitable for the security multimedia services like a mobile device and wireless applications.

1.3 Literature Survey

The bright aim of this work on H.264 is to reduce the huge amount of the data in a video during the transmission or storage. Video compression is aimed to make the digital storage or transmission at this time at the best state because the huge information of many digital videos became difficult to transmit or storage with a limited bandwidth or resource for storage. At the transmit side there is a video compression unit, and on the other side there is a decompression unit to inverse the operation and to retrieval the original video. Because of the huge benefits of this talent, produce many researchers work on H.264/AVC (encoder and decoder):

- In **August 2011 YIM, Ka Yee [3],** in his thesis, they worked in video decoder for H.264/AVC, which is supports full HD resolution, by modulation the baseline profile decoder. Developed and integrate the CABAC decoder to the baseline profile. The new profile, baseline, was reused to provide the full HD.

- In **May 2010 Samia Sharmin Shimu [4],** she present how to use the H.264 video codec in HD video transmission by Ultra Wide Band (UWB). The UWB used because of its short range, low power and cost. She used the JM 15.1 simulation program to reduce some of the H.264 encoder parameters, which are important to deal with them in communications applications. These parameters like in–loop de-blocking filter, GOP length and the quantization parameters for the frames (I, P and B).

- In **June 2008 Kermin Fleming, and others [5],** are dealing how to use the H.264 video codec in multiple applications from the phone cell to the HD TV. They designed the major H.264 and they showed how many applications could share the design benefits. The practical worked implemented by Bluespec System Verilog. They designed a variety of H.264 decoder designs with the resolutions ranges from (176X144 at 15 fr/sec.) of QCIF to the (1280X1080 at 60 fr/sec.). They generated multiple RTL design for (IP frames in H.264 codec).

- In **July 2003 Michael Horowitz and others [6],** they are present a H.264 decoder in baseline profile. They measured some of basic computational operations that required by the decoder to do well results, and measured the frequency of the decoder by using bit streams that generated from two different encoders of a variety contents, resolutions, and bit rates**.** They compared the results of the decoder to that in Pentium 3 hardware platform**.** The results from this comparison takes to compare it with the H.263 baseline decoder profile. They found that the H.264 profile decoder is more complex from the H.263 by (2.5 times)**.**

- In **October 2011 A. Ben Atitallah, H. Loukil , N. Masmoudi [7],** they designed H.264 encoder by using FPGA. They tried to reduce the critical path length and increase the throughput using pipeline architecture. This design implement by Altera Stratix III FPGA and VHDL language. The throughput of the FPGA architecture reaches a processing rate higher than 177 million P/sec at 130 MHz.

1.4 The Aim of the Project

Major manufacturers of video cameras are seeking to improve the quality and accuracy of videos, the quality and accuracy of the videos

have evolved to obey the users desire to take best videos and images. Everyone need to share his own videos, while maintaining the quality and accuracy of the video. The aim of this thesis is to implement and design the main profile of the H.264 decoder. The main profile used to produce good results. The theoretical work is about programming the H.264 encoder of (IBBB frame types and series) to encode the original video. We used video named (Xylophone) of size (644) KB, frame size (320X240) and frame rate equal to (30 fr/sec), implemented using MATLAB as good environment to explain the results. In the practical work, the Intra frame coded by VHDL using Spartan-6 SP601 evaluation kit [3].

1.5 Thesis Layout

This thesis includes five chapters as following:

- Chapter Two includes the general definition of the H.264 encoder and decoder. It includes the history of the beginning about video coding and decoding, and explaining how it is important.
- Chapter Three, design H.264 encoder and decoder in MATLAB program and explain in details how the video frames are encoding and decoding them.
- Chapter Four, is a practical design of the H.264 /AVC in FPGA using (VHDL) as a programming language. Our design in the encoder and decoder include designing each parts individually and gathering them in main program. This step is a smart step to exploit the specifications of encoder design. This leads to reduce the area space and stay in the limited range of the kit storage space.
- Chapter Five, the conclusion of the designing in MATLAB and in FPGA kit, and it include the useful suggestion for future works.

Chapter Two

H.264 Encoder and Decoder

2.1 Introduction

Video coding as a term, is a process of converting the original video frames in a format which is compatible with many video player applications. The past decade, videos are stored in magnetic tape or at an analogue format. With a development in science wheel, the (CD) Compact Disc was been invented, then (DVD) Digital Video Disc was set. For the approval of this scientific development, the analogue format of the video was been replaced by the digital one. It becomes possible for all to watch and store the high-definition videos thanks to this scientific progress. The Internet has become an integral part of our daily lives and anyone can upload or download the high-quality videos. There are many things to consider when downloading or uploading videos via the network like video size, video resolution and as these criterion was increased, we need a high bandwidth, longer time to reach and etc. For these reasons, video coding technique is a more important part at sender side of the network and at the receiver, the video decoding is necessary. Data compression term is the process of reducing the information we need to send them over the bandwidth and the size of the disk space if it is stored.

2.2 Video Coding.

Video encoding (compressing) is an operation to reduce the desired video size to be able to send, store and make modifications to the user's wishes. Uncompressed video, video clip, was taken from the camera contained a large amount of data. For example, the video clip was been recorded with a resolution of (720x576) (PAL) Phase Alternating Line, with a frame rate of (25 fps) and (8-bit) color depth takes:

(720 x 576 x 25 x 8) + (2 x (360 x 576 x 25 x 8)) = 1.66 Mb/s. PAL is a color encoding system for analogue television used in Europe. Another video clip was HD with resolution (1920X1080) with (60 fps) and (8 bits) color depth it takes: (1920 x 1080 x 60 x 8) + (2 x (960 x 1080 x 60 x 8)) = 1.99 Gb /s.

In YUV color system, each pixel has one brightness value (luminance) and two colors values (chrominance). From these two examples, we notice there are amounts of information to be process in different applications so we need a large amount of storage (hard disk) and high processor to deal with them. In every codec system, there is a complementary element, encoder and decoder [10]. Encoder reduces the size of the video through the stages built into its design otherwise the decoder decoding the encoded video by stages built into its design.

2.2.1 Video Coding Standard History

With the development of technology and the emergence of different and modern types of cameras that competed for the manufacture of major international companies to obtain the best quality and the image of the video taken. With the entry of Internet in all joints of life increased the desire of users to capture videos and send them to those who want, regardless of the size or quality of the video. It was necessary to invent a way to shrink the size of this video, whatever its quality and then send it, but it must be takes into account not to lose the original video information when encoded it and get the original video information at decoder side. Since the early 1990s, when the technology was in its infancy, international video coding standards , H.261, MPEG-1, MPEG-2 / H.262, H.263 , and MPEG-4 (Part 2) or H.264 have been the engines behind the commercial success of digital video compression [9].

ITU-T H.264 MPEG-4 (Part 10) advance video coding is the newest series of international video coding standards. The basic of this standard lie in the series of ITU-T/H.26L digital video compression standards. It is currently the most powerful and the best in terms of output standard, and developed by a Joint Video Team (JVT) consisting of experts from ITU-T's Video Coding Experts Group (VCEG) and ISO/IEC's Moving Picture Experts Group (MPEG). Figure (2.1) shows the idea of the encoder and decoder.

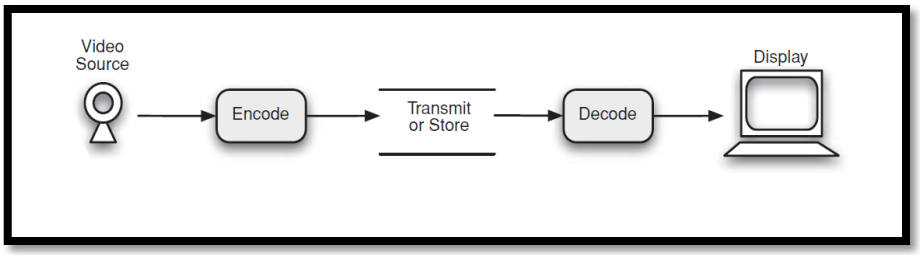


Figure (2.1) Encoder and Decoder

2.2.2 Video Coding Basic Principles

The video clip is an expression of a set of footage taken at successive intervals and continuously consists our video. It consists of a set of frames, these frames are a group of blocks and therefore they are composed of a set of pixels. Digital video is a representation of a real world visual scene that sampled spatially and temporally. The scene of the video contains shapes, textures, height, depth, colors, and illuminations. These characteristics determine the smoothness, quality and clarity of the video [10]. The characteristics of a natural or real video scene that are relevant to the video processing include spatial characteristics such as (texture that is varying within scene, number and shape of objects, color) and temporal characteristics like (object motions in time, changes in illumination and movement of the camera) [11]. Figure (2.2) explains the temporal and spatial sampling.

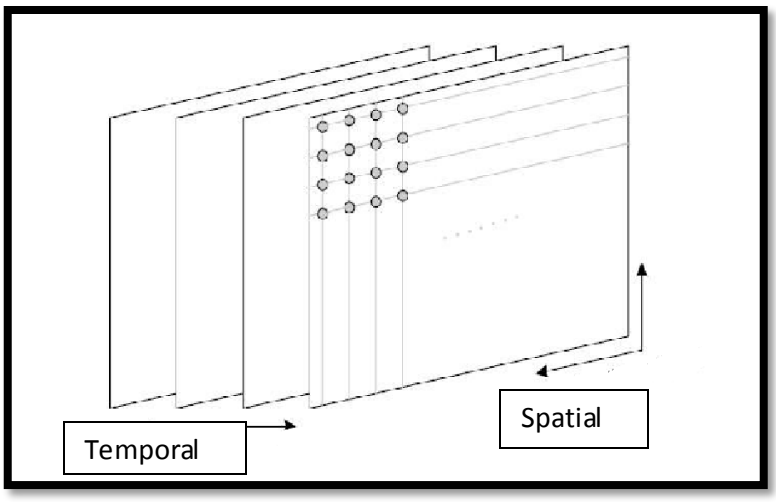


Figure (2.2) Temporal and Spatial Samples

2.3 Color Spaces

As we know, the light is a characterized by its wavelength (frequency) and intensity. The color is a visual perception of the light arriving at the photoreceptor cells in the retina of human eyes. The ability of the human eyes to distinguish colors is back to the varying sensitivity of different cells to the light of different wavelengths, and there are two kinds of photoreceptor cells in the human eyes (Rods and Cones). Rods are sensitive to the light intensity but insensitive to the colors, but cones are sensitive to the colors and insensitive to the light intensity. At very low light levels, visual experience just depends on the rods. For example, we cannot recognize the colors correctly in the dark rooms, because only one type of photoreceptor cell is active. The retina contains three types of cones that they can sense light with the spectral sensitivity peaks in short (420–440) nm, middle (530–540) nm, and long (560–580) nm wavelengths corresponding to (blue, green, and red) light respectively [13]. Figure (2.3) shows the colors (Red, Blue, Green) Sensitivity.

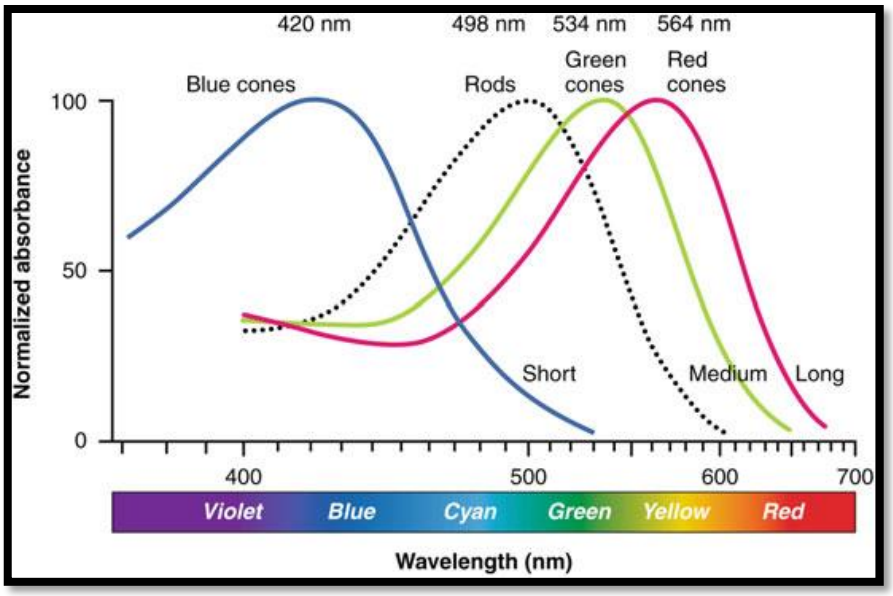


Figure (2.3) Color (Red, Blue, Green) Sensitivity

2.3.1 RGB, YUV and YCrCb Color Spaces

In the RGB color space, the image sampled into three values of equally distributed colors. These colors (Red, Green and Blue) are the basic color, by mixing them consist the other colors. In human visual system, it is sensitive to the luminance than color. If there is a video scene, the eye will be sensitive to the change of the intensity of the light more than the color changes and that is why in the dark, eyes don't distinguish colors, but be sensitive to the light and this is because of the physiology of the eyes. In RGB frame, each color sampled in equal rate but in other hand, the eye is less sensitive to the color so we can exploit this property by approved the YCrCb color space.

The YUV, Y is the lumnance (brightness) component while U and V are the chrominance (color) components. This model used in PAL composite color video (or in TV) where the Y is refer to the luminance and UV to the colors [13]. YUV and RGB conversion matrix shown in equations (2.1) and (2.2).

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.14713 & -0.28886 & 0.436 \\ 0.615 & -0.51499 & -0.10001 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \ldots\ldots\ldots\ldots \text{Equation}(2.1)$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.13983 \\ 1 & -0.39465 & -0.58060 \\ 1 & 2.03211 & 0 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \end{bmatrix} \ldots\ldots\ldots\ldots \text{Equation (2.2)}$$

The YCrCb is the new color space and it is mainly based on principle of it is possible to represent a color image more efficiently by separating the luminance from the color information and representing luminance with a higher resolution than color. The Y:Cr:Cb color space is a popular way of efficiently representing the color images. Y is the luminance component and can be calculate as a weighted average of R, G and B in equation (2.3)

$$Y = (kr * R + kg * G + kb * B) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation}(2.3)$$

Where k are weighting factors. The color (Red, Blue and green) Information can be represent as color difference (chrominance) components, from these equations (2.4, 2.5, and 2.6).

$$Cr = R - Y \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation (2.4)}$$

$$Cb = B - Y \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation (2.5)}$$

$$Cg = G - Y \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation (2.6)}$$

The RGB image can be convert to YCrCb after capture to reduce storage and transmission requirements (channel bandwidth) at encoder side. At the decoder side and before displaying the image, it is necessary to convert it to RGB. The green color can be extracted from the YCrCb representation by subtracting *Cr* and *Cb* from *Y*, because it is not necessary to store or transmit a *Cg* component.[12][13].

2.3.2 YCrCb Sampling Formats

YCrCb has three pattern of resolution used in video coding standards (4:4:4, 4:2:2 and 4:2:0).

- In first one (4:4:4) or full resolution is meaning that the three components (Y, Cr and Cb) have the same resolution and the sampling of each component exists at each pixel position. The numbers indicate that the relative sampling rate of each component in the horizontal direction is equal which means for every Y component (4 Y), (4 Cr) and (4 Cb).
- In 4:2:2 or YUY2 the chrominance elements have the same resolution to the luminance in vertical but half in horizontal resolution. In every of 4 luminance samples in the horizontal direction there are (2 Cr) and (2 Cb) samples.
- The 4:2:0 or (YV12) is the most popular resolution format used by H.264 /AVC (Advanced Video coding). Here there is 4 luminance or Y components and half values of the chrominance components in vertical and horizontal direction. The 4:2:0 sometimes called (YU12) because there is 12 bits per pixel. Using 4:2:0 sampling, only 6 samples are required, 4 Y and one each of Cr, Cb, and 8 bits, requiring a total of (6×8)=48 bits, an average of (48/4)=12 bits per pixel. Figure (2.4) shows the sampling patterns format.

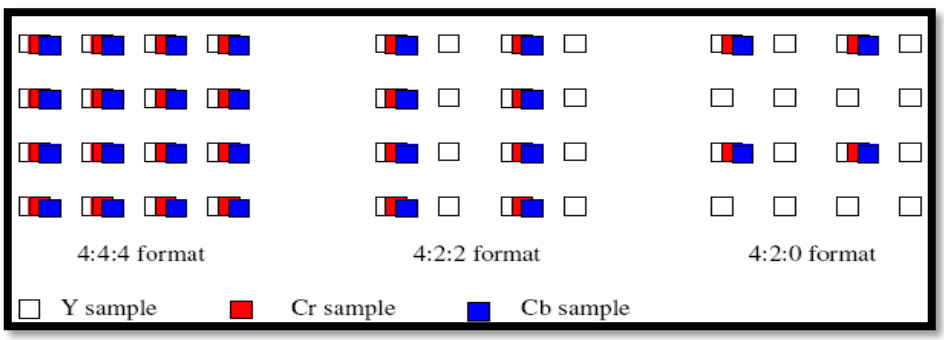


Figure (2.4) Sampling Patterns

2.4 H.264 /AVC Standard

The H.264 /AVC is the new standard improved by the ITU-T (VCEG) Video Coding Experts Group sharing with the ISO/IEC JTC1 (MPEG) or moving picture experts group. The project cooperation effort known as the Joint Video Team (JVT). The final drafting work on the first version of this standard was complete in May 2003, and the other extensions of its capabilities have been add in subsequent editions. H.264 supposedly best known as being one of the video encoding/decoding standards for Blue ray discs. It is also widely used by the Internet sources, like videos from YouTube, I Tunes store, Tube mates and Vimo, Web software such as the Adobe Flash Player. The advantage of H.264\ AVC it can get best performance and high quality results compared to the standards that preceded it. Compared to the MPEG 4 and MPEG 2, the H.264 has a lower bit rate and high quality in the same compressed image. For single layer DVD, it is possible to store a movie about 2 hours in MPEG-2 and MPEG-4, and can store movie of 4 hours with lower bit rate and best compression, this means (1.5 or 2) times best compression rate compared with MPEG2 and MPEG4 video coding standards. Figure (2.5) explains the comparison between the H.264 and MPEG-4.

Figure (2.5) H.264 and MPEG Comparison

2.4.1 H.264 Profiles

H.264 defines numbers of profiles and levels for helping to complete the process with the best formula and the best results. Each profile uses techniques and tools to produce best results. The H.264 standard has many types of profiles like (baseline, main, extended, and high) [14]. Figure (2.6) shows the H.264 profiles types.

- The baseline profile mainly used in applications that need low delay and complexity, it used in applications like mobile or conversational video transmission. It use (I, and P) types (Intra and Inter frame processing) with CAVLC.
- Extended profile is the same to the baseline with different in adding new tools to it that be able to be useful for efficient network streaming of H.264 data.
- The main profile is a superset of the Constrained Baseline Profile and it adds a coding tools that can be suitable for an entertainment and broadcast applications like a digital TV and DVD playback, namely (CABAC) entropy coding and a bi-directional prediction or B slices with prediction modes for better coding efficiency.
- The high profile support s higher quality applications, high definition, extended bit depths, higher color depths. High Profile is a superset of the Main Profile with adding the following tools (8×8) transform and (8×8) inter prediction for better coding performance. It may be useful for high definition applications.
- The High 10 profile, the maximum number of bits per sample extended to 10 bits in the High10 profile and to 14 bits in the High444 profile.
- The High422 Profile adds support for 4:2:2 video, i.e. higher Chrominance resolution.

- High444 Profile extends this to 4:4:4 video giving equal resolution in luminance and chrominance components and adds separate coding for each color component.[12]

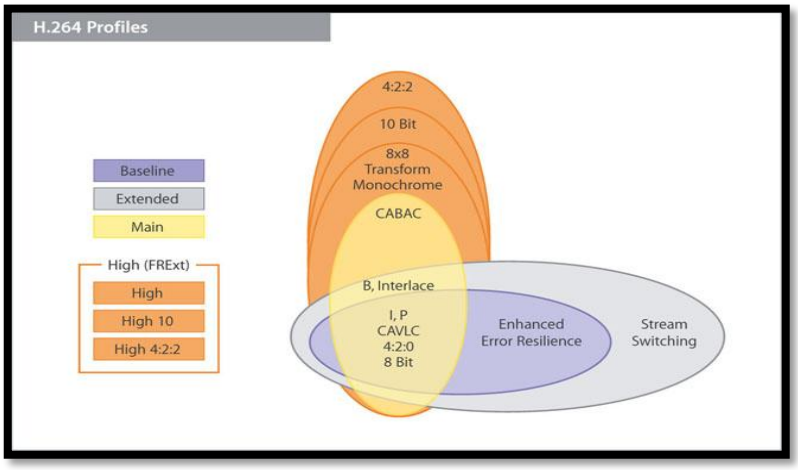


Figure (2.6) H.264 Profiles

2.5 H.264 Encoder and Decoder

H.264 encoder is the new technological way to encode and reduce the size of the indicated video to be able to send it over the network or store it in storage media like DVD, blue ray, and USB flash memory. It contents many stages as shown in Figure (2.7).

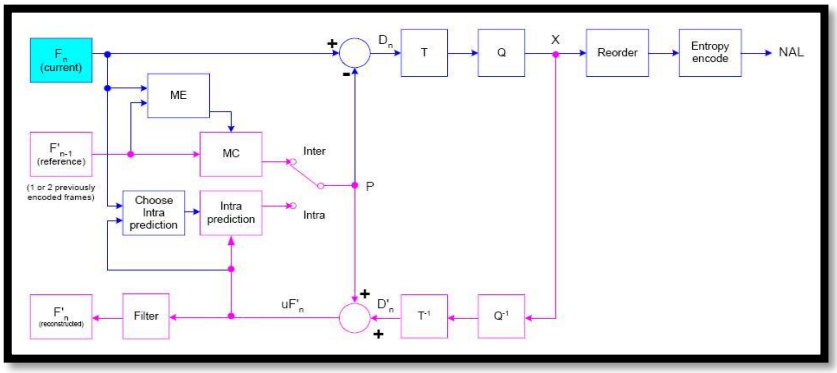


Figure (2.7) H.264 Encoder Stages.

The encoder has two paths (forward and reconstruction paths).

- In forward path, the input frame is process in units called macro blocks. Each macro block is encoded in an (intra or inter) mode and for each block in the macro block, a prediction is formed based on

reconstructed picture samples. In the Intra mode, prediction picture is formed from samples in the current slice in (I frame type) that have previously encoded, decoded and reconstructed. In the Inter mode, the prediction picture is formed by the motion compensated prediction from one or two previous reference picture(s), and the prediction reference for each macro block partition (in inter mode) may be chosen from a selection of past or future pictures (P or B frame types) that have already been encoded, reconstructed and filtered. The prediction picture is subtracted from the current block to produce a residual block that is transform (using a block transform) and quantized to a set of quantized transform coefficients, which are reorder and entropy encoded. The entropy encoded coefficients, together with side information required to decode each block within the macro block (prediction modes, quantize parameter, motion vector information, etc.) form the compressed bit stream which is passed to a Network Abstraction Layer (NAL) for transmission or storage.

- The reverse path in the encoder include decodes, (reconstructs) each block in a macro block to provide a reference frame for further predictions in the next. The coefficients are rescaled and inverse transformed to produce a difference block. The prediction block added to the previous decoded block to create a reconstructed block. The reconstructed reference picture is create from a series of blocks [12]. The H.264 decoder used to decode the encoding video come from the encoder. The decoder receives the compressed bit stream and decodes the data elements to produce a set of quantized coefficients, they are rescale and inverse transformed. Using the header information decoded from the bit stream, the decoder creates a prediction block [12]. Figure (2.8) shows the H.264 decoder path.

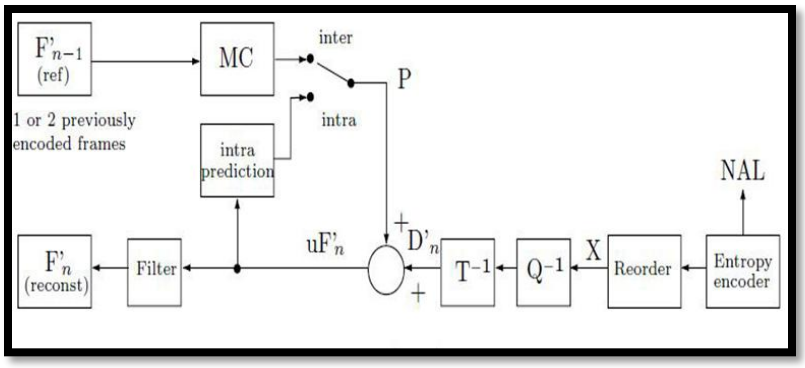


Figure (2.8) H.264 Decoder

2.6 The H.264 Working

The H.264/ AVC have many stages to help it to have best result (block matching, transform and quantization in the encoder side, and inverse transform and quantization at decoder side). The motion estimation (block matching) and composition is the power of H.264 because it reduces the amount of the data in the video overall and frame in special. A variety of methods used to reduce the video data, both within an image frame (I frame) and between a series of frames (I, P and B frame types). Within an image frame or (I frame), data may be reduced by removing the unnecessary data, which will have an impact on the image resolution. On the other hand, in a series of frames, the video data can be reduce by comparing a frame with a reference frame (I- P-or B frames) and only pixels that have changed with respect to the reference frame are code. In this way the number of pixels values that are coded and sent are reduced [17]. Figure (2.9) explains how the frame is encoded.

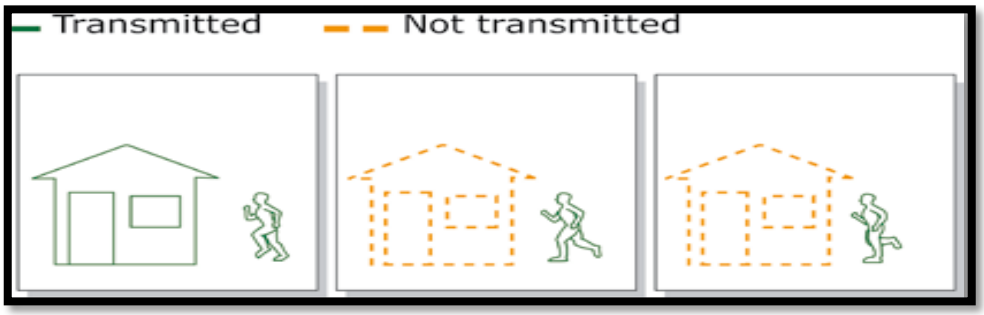


Figure (2.9) Frames to Encode.

2.6.1 Motion Estimation and Compensation

In H.264, Each frame divided into blocks (16x16) pixels called the macro blocks (MB). Each macro block encoded by using blocks of pixels encoded within the current frame or (Intra frame) coding or the micro blocks can be encoded by using blocks of pixels in previous or future encoded frames (Inter frame) coding like (P and B frames). The process of finding the best matching of pixels block in the inter frame is called the Motion Estimation (ME) and the displacement vector between two blocks is the Motion Vectors (MV) of the block [18][20]. The motion estimation is computationally expensive process when the search was doing at every pixel position over different reference frames [18]. There are several different integer search algorithm used to find the best matching like:

- Full search algorithm
- Fast integer search algorithms like (2D log algorithm, 3steps algorithm, diamond search, etc.)

The process of subtracting the two matching blocks after finding the best match and gain the motion vector called motion compensation**.**

2.6.1.1 Motion Estimation and Compensation Procedures

These procedures is execute in the MB and it contained:

1. For the reference frame, the search area is define to every block in the current frame. The search area size as (2 to 3) times the macro block size (16x16). After the searching process, the best matching value found in this collected area. The best matching value is determined based on the minimum displacement motion vector. This done by subtracting the candidate block in the search area from the current block located in the current frame. The process to find the best matching blocks called block based motion estimation. When the best matching is found, the motion vectors and the residues between the

reference and current blocks are computed and found, so the process of having the motion vectors and the residues is known as the motion compensation, this values is encoded by the DCT transformation and quantization. At the decoder, the process is reverse and decode by the inverse transformation and quantization. Figure (2.10) shows the block matching and motion vector.

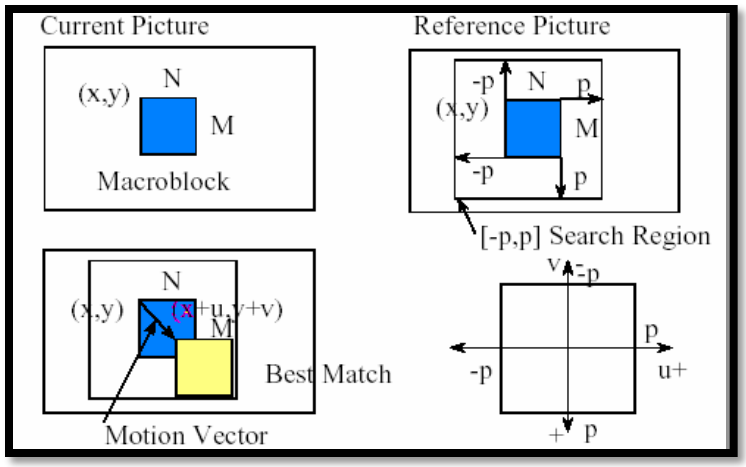


Figure (2.10) Block Matching and Motion Vector

2.6.1.2 Motion Estimation and Compensation Algorithms

There are many types of algorithms are used in H.264/ AVC to find the best matching between blocks in the search area of the reference frame and the current or previous frame, these are some algorithms:

1. Full Search or (Exhaustive Algorithm).

   It is the most computationally expensive block-matching algorithm. This algorithm calculates the cost function at each possible location in the search window in frame. As result, it finds the best possible matches and gives the highest PSNR amongst any block matching algorithm .The disadvantage of this algorithm need larger search window, which require more computations. [19][20].

   Figure (2.11) explains the full search algorithm where the shaded blue color represents search steps of the blocks within the frame.

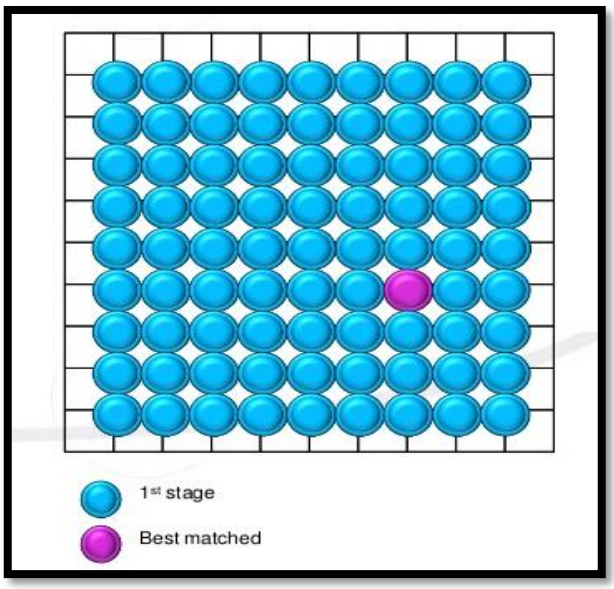


Figure (2.11) Full Search Algorithm

The total number of candidates windows are ((2P+1)^2) with (±Px, ±Py) search window, where Px, and Py are the sizes of the search window in the x and y direction respectively. For a (512x512) frame with (P16=number of blocks divisions) so the number of operation per frame is ((2X16)+1)^2=1089.

And (512x512x1089) = 2.85x10^8

so with a frame rate of (30fps), the number will be

(8.55x10^9) per minute and this is the number of process per minute and the complexity is (P^2).

2. The 2 D Log Algorithm

This algorithm has steps to find best matching.

- The first step includes (select an initial step size(s) and calculate the error for the block at the center of the search area and four points at (x) and (y) axis at distance (s) from center).
- The second step, if the position of the best match at the center of the candidate block, keep the center unchanged and reduce the step

size by half, otherwise the best match becomes the center so repeat the first step.

- In the last step, if the step size value becomes (1), then all the (8) neighbor blocks around the collected center will be checked to find the best match. The 2D-log search algorithm has lesser search points than TSS, yet its prediction is more accurate, it defines the step size at the beginning and terminates if the step size is equal to one.

  The 2D-log has the advantage of better prediction quality than the Full search and TSS algorithms. The complexity of this way is about ((log (p/2)) where p is the search area size [20]. Figure (2.12) show the 2D-Log algorithm.

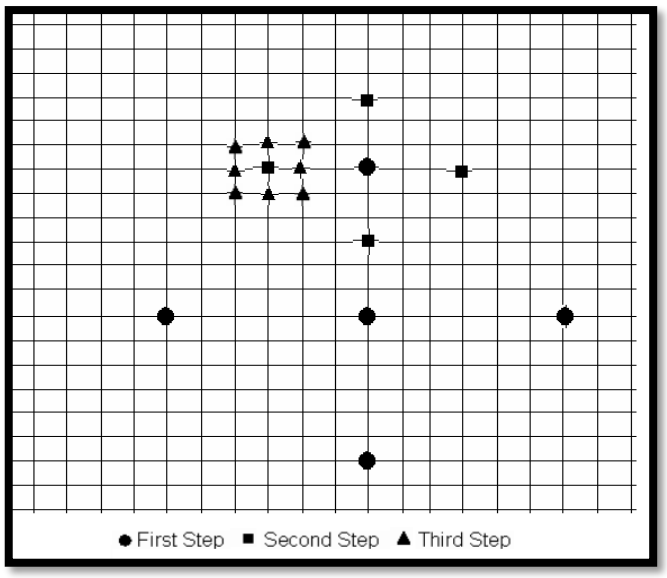


Figure (2.12) 2D-Log Algorithm

3. Three Steps Search Algorithm or (TSS)

   This is a good algorithm for the result to find the best matching and it is involving steps:

   - In the first step, we select the initial step size (s) equal or larger than half of the maximum search range, then calculate the error for

the block at the center and (8) square points neighbors at the distance of is from center.

- The second step includes moving the center to the point with minimum error and reducing step size by a factor of two. If the step size is greater than one, then repeat the first step, otherwise, implement the next step.
- In the last step, the final point with minimum error is the result. This method has low complexity but it has a high data bandwidth. The complexity, it is the same at the 2D algorithm (log(P/2)). Figure (2.13) show the TSS algorithm steps.

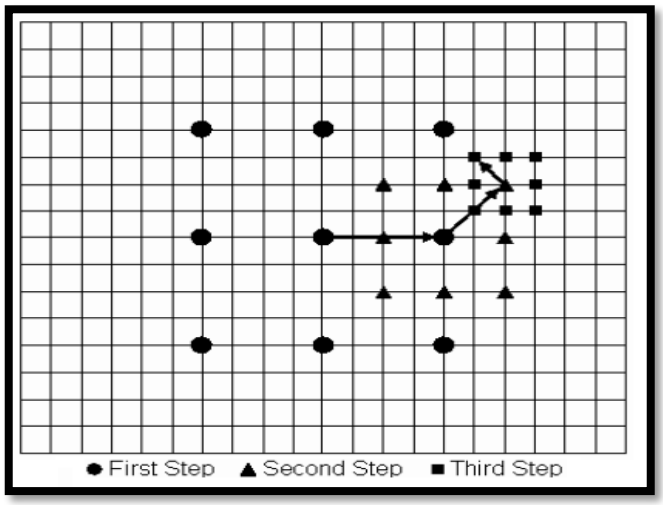


Figure (2.13) The TSS algorithm

4. Diamond Search Algorithm

Diamond Search (DS) algorithm uses a diamond search patterns, there is no limits of the number of steps. There are two different types of fixed patterns used for searching, the Large Diamond Search Pattern (LDSP) and the Small Diamond Search Pattern (SDSP). The search processing is done by the first (LDSP), and it stated with start with determination the search location at the center and choose the step size (S=2), then stating search for (8) locations pixels (X,Y) such that (|X|+|Y|=S) around location (0,0) using a diamond search point

pattern. Choose among the (9) locations searched, the one with minimum error value. If the minimum weight is founding at the center of search window, then go to the SDSP step, else if the minimum weight is founding at one of the (8) locations other than the center, then set the new origin to this location and repeat the LDSP. The SDSP is also contains steps for searching. It start with setting a new search origin point and new step size S = S/2=1, then repeating the processing in LDSP to find the minimum value of error in points. This algorithm finds the global minimum very accurately as the search pattern is neither too big nor too small. Diamond Search algorithm has a peak signal-to-noise ratio close to that of Full Search with less computational expense. The DS algorithm has a complexity of the order of (log (p/2)). Figure (2.14) explains the Diamond algorithm steps.

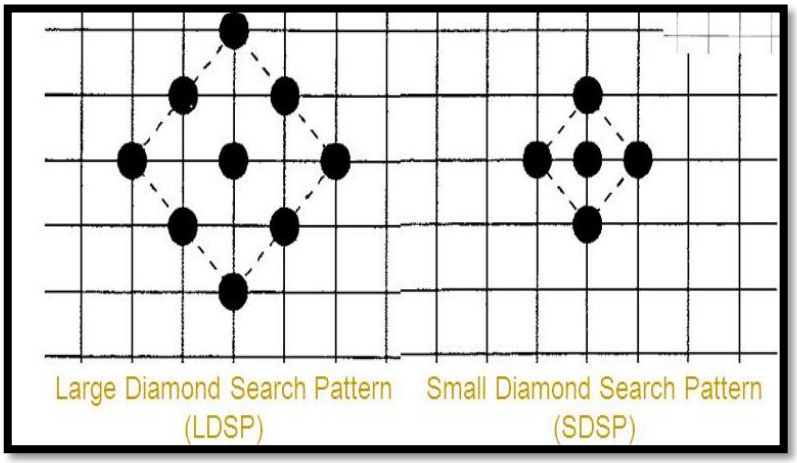


Figure (2.14) Diamond Search Algorithm

2.6.1.3 Motion Vectors (MV)

MV defined as the displacement between the current block and the best matching block in the search area in the reference frame. It is the successive key of motion estimation and it is a directional pair representing the displacement in the horizontal direction (x-axis) and

vertical direction (y-axis). The maximum value of MV determined by the search range. Traditionally one MV produced from each macro block in the frame.

2.6.1.4 Matching Criteria

Block based motion estimation gains the best matching by minimizing the cost function. Many cost functions have been used and analyzed to find the minimum value. The Sum of Absolute Difference (SAD), Mean Absolute Difference (MAD), Mean Square Error (MSE) are some models of calculation the cost function.

- **Sum of Absolute(SAD)**

  SAD is the most common matching method used in the video coding, because has good performance, low complexity ease in the hardware implementation. Equation (2.7) is the SAD equation

  $SAD = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |\mathrm{CB(i,j)\ RB(i\ m, j\ n)}| \ldots\ldots\ldots\ldots\ldots\ldots \mathrm{Equation(2.7)}$

  Where (NxN) is the block size.

  CB(i,j) is the current block intensity at location (i,j)

  RB(i m, j n) is the reference frame block intensity at location (i,j) [21].

- **Mean Absolute Difference (MAD)**

  The mean absolute difference has its equation and explained in equation (2.8)

$$MAD = \frac{1}{NxN} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} [CB(i,j)RB(i\ m, j\ n)] \ldots\ldots\ldots\ldots\ldots \mathrm{Equation(2.8)}$$

  Where (NxN) is the block size.

  CB(i,j) is the current block intensity at location (i,j)

  RB(i m, j n) is the reference frame block intensity at location (i,j).

  In the MAD (m,n) represents MAD at search position (m,n) and

(m,n) is the displacement vector of the micro block. The advantage of the MAD cost function is its simplicity and ease of implementation in hardware. Unfortunately, MAD tries to focus on differences of small values, giving a lower result to MSE. The only difference between SAD and MAD is that SAD takes the sum of all pixels in the candidate block, but MAD measures the absolute difference of the average pixel value in its candidate block.

- **Mean Square Error (MSE)**

  This method is measuring the energy in the remaining in the difference block. The MSE equation:

$$MSE = \frac{1}{NxN} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} [\mathrm{CB(i,j)RB(i\,m,j\,n)}]^2 \dots\dots\dots \mathrm{Equation(2.9)}$$

  Where (NxN) is the block size.

  CB(i,j) is the current block intensity at location (i,j).

  RB(i m, j n) is the reference frame block intensity at location (i,j).).

  In the MSE (m,n) represents MSE at search position (m,n) and (m,n) is the displacement vector of the micro block. The advantage of the MSE is its accuracy, but its disadvantage is the complexity is high for both software and hardware implementations.

2.6.2 Intra and Inter Frame Prediction

There is two types of prediction used in h264/ AVC to reduce the information in the frame, intra and inter frame prediction.

2.6.2.1 Intra Frame Prediction

Is the process to predict the blocks of an individual frame, and try to reduce the amount of data with minimum loss and high quality. Intra coding refers also to the case where only spatial redundancies within a video picture are exploited. In H.264 /AVC, the intra frame prediction is

an alternative to the inter frame prediction. If a block is part of a (P-slice or a B-slice), the encoder decides which prediction gives the best result. If there is no good motion prediction, then inter prediction is used. The macro block size may be (16x16) and divided into (16x8), and (8x16) or its size is (8x8) and divided into (8x4), and (4x8) and we can merge into size of (8x16), or (16x8). The (4x4) macro block size could not be divide into smaller size. A bigger macro block covers large area of frame and more information but low quality in there constructed image, and it decreases the computation cost and the complexity of algorithm. The bigger macro block used in the frame to code continuous area of picture. In the other side, the lower value of macro block used to predict the smallest details in the frame. The smaller macro block improves the quality of reconstructed image but increases the computational cost and complexity of algorithm. Figure (2.15) is about the (4x4) Luma prediction. There are several modes used to predict the Luma and Chroma samples in the frame:

1. Intra modes for Luma samples [4 modes for (16x16) blocks, 9 modes for (4x4) blocks, and 4 modes for (8x8) blocks].
2. Intra modes for Chroma samples [4 modes for (8x8) blocks].

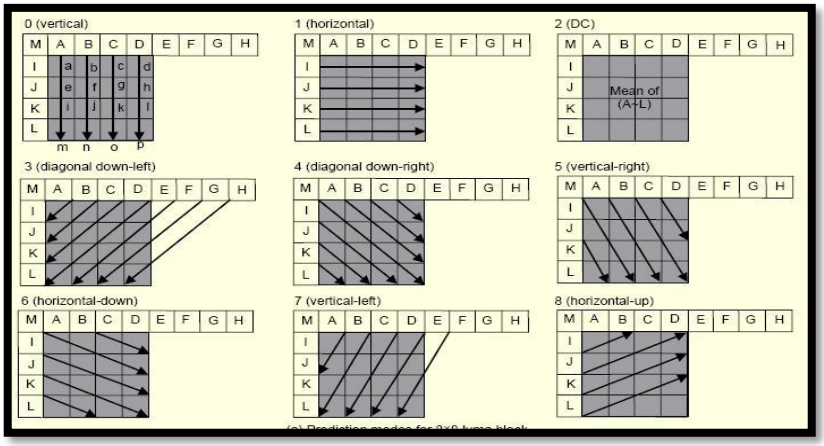


Figure (2.15) (4x4 Luma Prediction)

The details of each modes in intra mode prediction (Luma and Chroma):

1. The 9 Modes of (4x4) Luma prediction.
2. The 16 samples of the 4x4 block, which labeled as (a - p) are predicted using prior decoded samples in adjacent blocks labeled as (A - Q). For each (4x4) block, one of nine prediction modes can be utilize. Figure (2.16) is show the (4x4) Luma samples and direction.

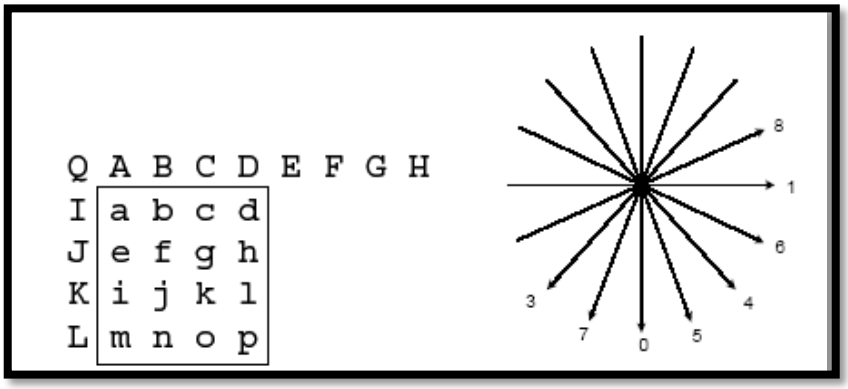


Figure (2.16) The 4x4 Luma Samples and Direction

- First mode (mode 0 or vertical prediction) and (mode 1 or horizontal mode), the samples (4x4) block are copied into the block as indicated by the arrows in mode 0 the arrows are to the down but in horizontal the arrows are to left.
- In mode 2 (DC prediction) the adjacent samples are averaged.
- The remaining 6 modes are (diagonal prediction modes, which are called diagonal down-left, diagonal-down-right, vertical-right, horizontal-down, vertical-left, and horizontal up prediction). As their names indicate, they are suited to predict the textures with structures. When samples E-H that are used for the diagonal-down-left prediction mode are not available (because they have not yet been decoded or they

are outside of the slice or not in an intra-coded macro block in the constrained intra-mode) [23].

3. The 4 modes for 16x16 block size

H.264 supports (16x16) Luma intra coding, in which one of four prediction modes is chose for the prediction of the entire macro block. These modes:

- Mode 0 or vertical prediction.
- Mode 1 or horizontal prediction.
- Mode 2or dc prediction.
- Mode3 or diagonal prediction) [23][24].

Figure (2.17) explains the 16x16 Luma prediction model.

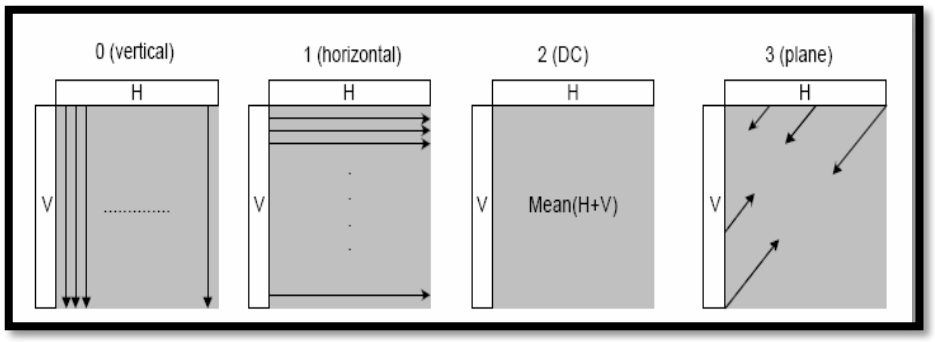


Figure (2.17) (16x16) Luma prediction

The intra Chroma has the same modes in (16x16) Luma prediction but they applied in the Chroma components.

2.6.2.2 Inter Frame Prediction

Inter prediction is using in the motion estimation and compensation because it takes the advantages of the temporal redundancies which found between the two successive frames and it is providing a very efficient coding. If the selected reference frame for is a previously encoded frame so is referred to a P frame and when a previously encoded frame and a future frame are chosen as reference frames, then the frame is referred to a B picture. [25]

2.6.3 H.264 Transformation, Quantization (Scaling).

The purpose of the transform in the video coding is to convert the motion compensated residual data into another domain. The transform used in this video coding standard must be computationally flexible which means low memory requirement and low number of arithmetic operations. Many transforms methods have been propose to do best work and gain best results. Discreet Cosine Transform or (DCT) is one of the best method in H.264. The DCT is revisable calculation method and it is apply the main condition of the transformation process. The blocks of residual samples that came from motion estimation and compensation stages are transformed using (4x4) or (8x8) integer transform method or DCT. The coefficients are the transformation process output and each of them are weighting values for standard basis patterns and when they are combined, the weighted basis patterns recreate the block of residual samples. The output of the transform, a block of transform coefficients, is quantize, means each coefficient is dividing by an integer value. The quantization process reduces the precision of the transform coefficients according to a quantization parameter (QP). There are some laws to choose the QP value. Setting the (QP) to the high value means that more coefficients are set to zero and it leads to high compression at the expense of poor decoded image quality. On the other hand choosing (QP) with low value means that more of non-zero coefficients remain even after quantization, resulting in lower compression but better image quality at the decoder . The DCT operates on block X, a block of (N $\times$ N) samples to create block Y, the coefficients to quantized. The equation (2.10) is a DCT equation, and the IDCT equation (2.11):

$$Y = [AXA^T] \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation}(2.10).$$

$$X = [A^TYA] \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation}(2.11).$$

Where A is a stander matrix of the evaluated from cosine function
Y is matrix of coefficients, and X is matrix of samples.

$$A=\begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation}(2.12)$$

Where a = $\frac{1}{2}$, b = $\sqrt[2]{\frac{1}{2}}\cos\left(\frac{\pi}{8}\right)$, and c= $\sqrt[2]{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right)$.

Figure (2.18) show the idea of transformation and quantization.

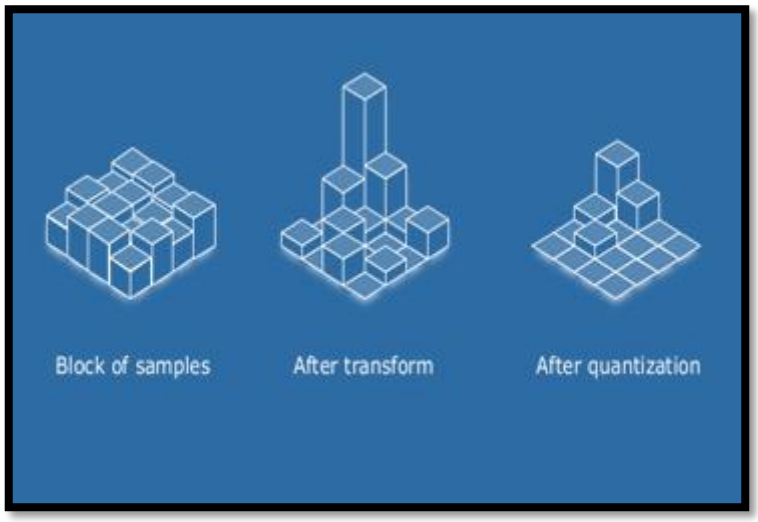


Figure (2.18) Transform and Quantization

2.6.4 H.264 Decoder Overview

The decoder receives the compressed bit stream data, decodes each of the syntax elements and extracts the information. The decoder re-scale the quantized transform coefficients and reconstruction the frame to get the original one.

2.6.4.1 Rescaling and Inverse Transform

The rescaling process is involving multiplying the quantized transform coefficient by an integer value to restore its original scale. The integer value is the same in the encoder at quantization step.[26]

Figure (2.19) explains the rescaling process.

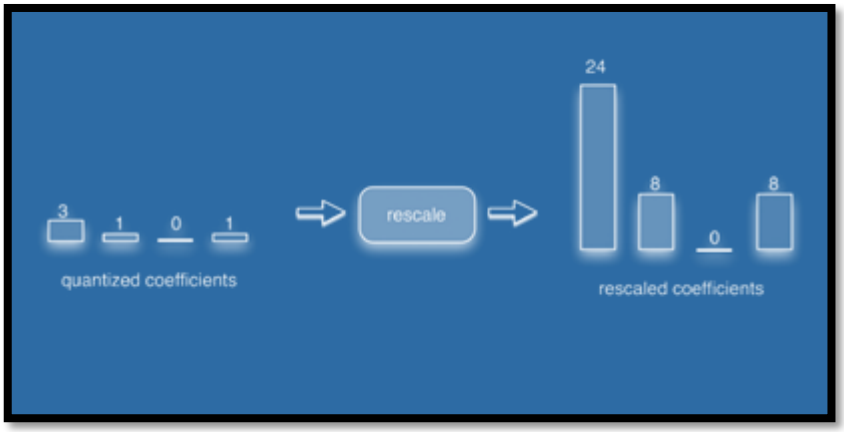


Figure (2.19) Re-Scaling Process

The inverse transform combines the standard basis patterns, weighted by the re-scaled coefficients to re-create each block of residual data. Figure (2.20) shows the inverse transform process.

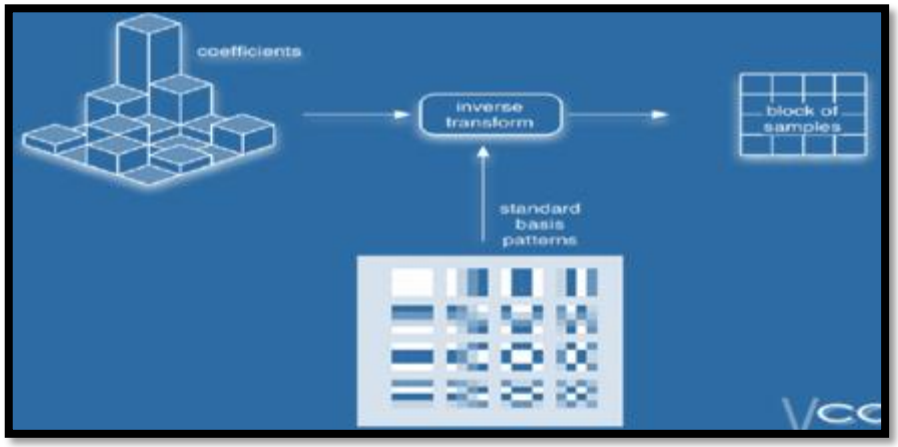


Figure (2.20) Inverse Transform

2.6.4.2 Reconstruction

In the reconstruction step each macro block, the decoder forms an identical prediction to that one created by the encoder using inter

prediction from previous or future decoded frames or intra prediction from previously decoded samples in the current frame. The decoder then adds the prediction to the decoded residual to reconstruct a decoded macro block.[27]. Figure (2.21) shows the reconstruction process.

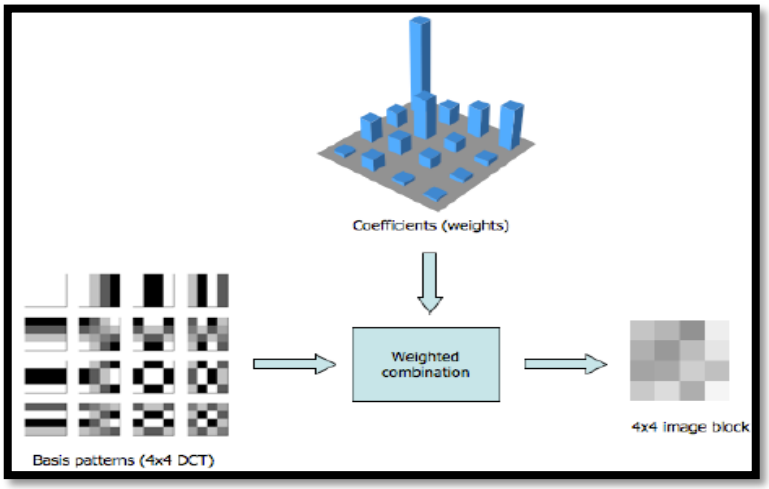


Figure (2.21) Reconstruction Process.

Chapter Three
Implementation of H.264/AVC Encoder and Decoder in MATLAB

3.1 Introduction

In order to achieve good results in compression and decompression, it was necessary to design the H.264 encoder and decoder. The H.264 decoder receives the data that send over network, the encoder encodes the frames of the video by divide them into blocks. At the encoder, the predictions we explained are intra prediction and inter prediction. At the decoder, the original video is obtain.

Matlab considered the best program available to simulate the reality and used to read In Matlab, sub-program is used to read the video (Xylophone video), 664 KB, (320x240), 30 fps .The laptop we used has specifications of (core i3 with RAM of 4GB). In this chapter, we designed the decoder by relying on our Encoder. This encoder has very good specifications with (IBBB) patterns of frame.

3.2 Encoder Process

In order to begin coding process through the encoder there are several stages to follow which is presented in figure (3.1) . The original video is in (RGB) color space and it is good idea to convert it into (YUV) color space because the YUV contracts the size of the frame. It depends on the idea that the human eye is affect by lighting more than gradient change colors. The colors components (Chroma) and lighting components (Luma) are sampled based on weight that specific by (4:2:0) sample format. In real world, videos are in RGB where each colors (Red, Green,

and Blue), are sampled equally in same resolution. In new space of color, the luminance is separated from the colors, and representing luminance with a higher resolution than colors.[28][29]

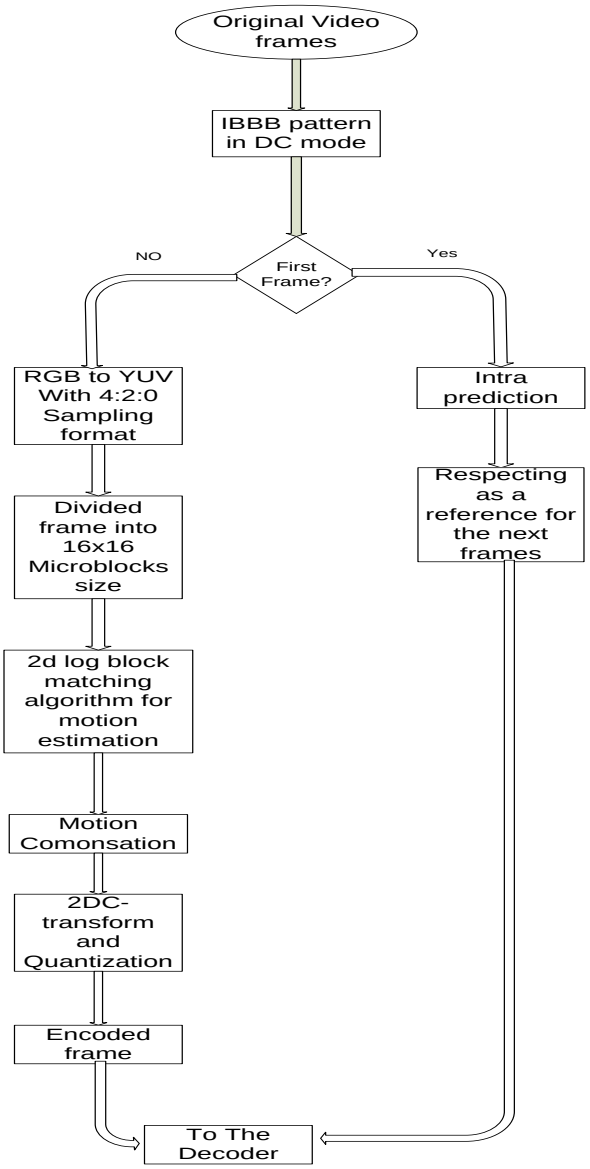


Figure (3.1) The H.264 Encoder steps

From equation (2.1), we gain the frames in YUV color space. Figure (3.2) and Figure (3.3) show the YUV component of the (Xylophone) video and another video called (Foreman), as another model.

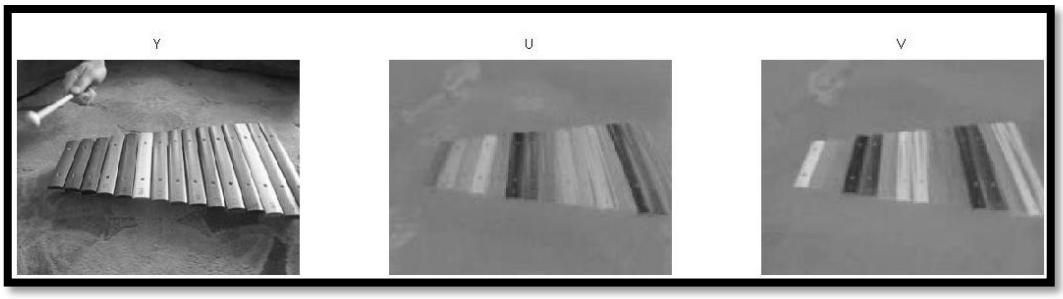


Figure (3.2) YUV of the Xylophone Video

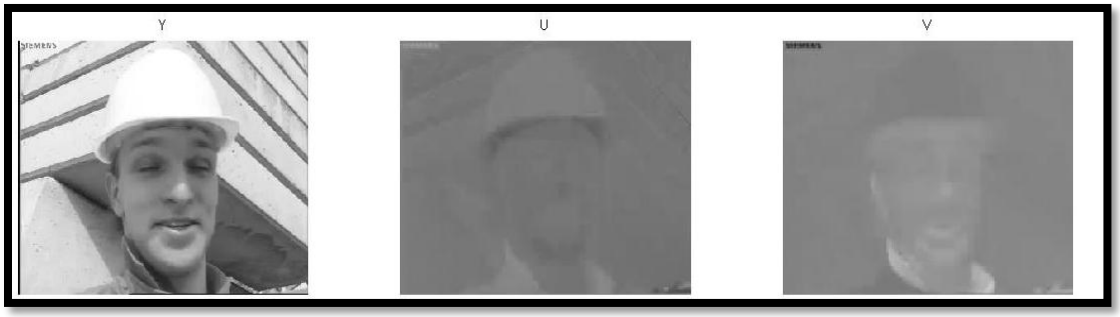


Figure (3.3) YUV of the Foreman Video

After color space transformation the frames of the video is divide in blocks of size (16*16).

3.3 Encoding Initialization

The video is initialize and it is ready to be encode by several steps. The video frame is predicted either (intra prediction) or (inter prediction) as the encoder design. If there is small details and they need to be predict, intra prediction is the solution. Supposing there is an action video or football matching video, there are many details and information inside it, so intra is best predicted way. The intra prediction is produce using the neighboring samples of the previously encoded and reconstructed blocks in the same frame. In this theses, we used it to puncture it with inter prediction and prove that the inter prediction is best choice for this video. The (16*16) (Luma) intra prediction and with (mode 2 or DC mode) is used, where every pixel of the macro block is predicted from the mean of upper and left neighboring samples of the macro block. [30][31]. Figure (3.4) explains the different between the original and intra frame.

Original Frame **Intra Frame**

Figure (3.4) Original Frame vs. Intra Frame

We notice that, there is a difference in the vision between original frame and the intra prediction one .This unclear vision back to that the video has details that are stayed stable like the xylophone but the hand is change its place, so for this reason the temporal redundancy must be utilized by using inter frame prediction.

The Inter frame prediction aimed is to remove the temporal redundancy of the video. The work is aiming to find a good match for the current block from the previously coded image, and the block from the future and previous coded image, this is the work of the (IBBB) frame pattern. The inter prediction tools contribute to the improved the compression efficiency of the H.264/AVC standard. The inter prediction have steps to find the best matching blocks and having a good compression efficiency [25].

3.3.1 Inter Prediction Steps

The inter prediction includes steps that are representing the power of the H.264/ AVC because they have the key of compression. They exploit the high redundancy that happened between successive frames in the video. To achieve this goal, a block matching algorithms are used. The real world objects can move, jumping, rotate, etc. These movements cannot be observe directly, but the light reflected from the object surfaces

and projected onto an image. There are some noises happened when the image is taken by a video camera. Motion compensation is the technique that uses the redundancy between frames in a video sequence to compress the data. Once the motion estimation has done, the algorithm of the block matching only transmits the difference between the successive frames by applying the motion compensation [26].

In the thesis, we used the (2D log algorithm) as a best result of block matching algorithm. We choose the search area and divide it into (16*16) of block size. After that, the search algorithm will done using the 2D-log algorithm.The 2D-Log algorithm is a good way to use, because it have better compression, rather than other ways. The search algorithm done in the Luma frame only and for the frame size of (320X 240) and search area of (16X16). The complexity in this way is Log (P/2) where, P=16, the number of blocks division. It is equal to Log (16/2) = 0.9 .The block matching criteria use the SAD, because it is a good performance and low complexity.

Figure (3.5) shows the frame after motion estimation only without Appling motion compensation. From the figure (3.5), we notice that the vision is unclear because there is some redundancy and noise that happened during the division process and doing searching algorithm. This noise or distortion can be removed or reduced by apply (motion compensation). Motion compensation is the next step after motion estimation process. It based in process to subtract the original frame with the predicted frame to construct the residue frame. This residual frame with the motion vector passed to the next stages, transformation and quantization.

Figure (3.5) Original Frame vs. Motion Estimated

Figure (3.6) shows the frame after applying motion compensation.

Figure (3.6) Motion Estimation and Compensation of the Video

Motion vectors is the power key of the motion estimation process. It used to represent a macro block in the search area of the predicted frame based on the position of this macro block in reference frame.

Figures [3.7(a), 3.7(b), 3.7(c)] show the frame after motion estimation and compensation. Figures [3.8(a), 3.8(b), 3.8(c)] show the motion vector of frames number (2, 3, 4) after motion estimation only and without motion compensation. Table (3.1) shows the YUV values of the frame. The table (3.2) shows the YUV Frame after motion estimation. The table (3.3) shows the YUV frame, after motion estimation and compensation.

Table (3.1) Frame in YUV

Frame in YUV

56.0740	55.0440	55.0440	50.9240	50.9240	52.9840	52.9840
54.0140	54.0140	54.0140	54.0140	54.0140	54.0140	54.0140
52.9840	52.9840	52.9840	52.9840	52.9840	52.9840	52.9840
54.0140	54.0140	52.9840	51.9540	50.9240	50.9240	50.9240

Table (3.2) Frame in YUV after Motion Estimation

YUV Frame after Motion Estimation						
52.0109	52.1985	52.1729	51.5311	50.2864	48.8870	47.8295
51.6961	51.7589	51.5368	51.4274	51.9408	52.2160	51.1189
53.0722	35.0415	52.2337	51.7522	53.1107	54.5225	53.1264
55.6375	55.7398	54.3402	52.4194	52.4170	53.1848	51.5799

Table (3.3) Motion Estimation and Compensation of TUV Frame.

YUV Frame after Motion Estimation and Compensation						
4.0631	2.8455	2.8711	2.4829	3.7276	4.0968	5.1545
2.3179	2.2551	2.4772	2.5866	2.0732	1.7980	2.8951
-0.0882	-0.0611	0.7503	1.2318	-0.1267	-1.5385	-0.1424
-1.6235	-1.7258	-1.3592	-0.4654	-1.4930	-2.2608	-0.6559

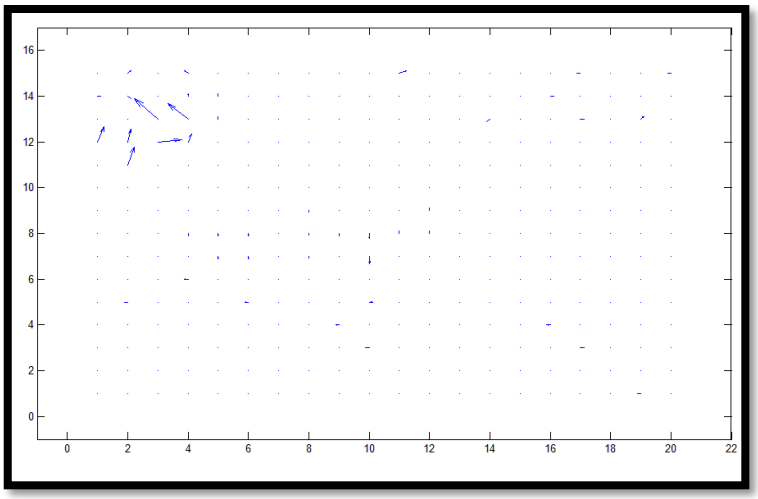


Figure 3.7(a) Motion Vector of Frame #2

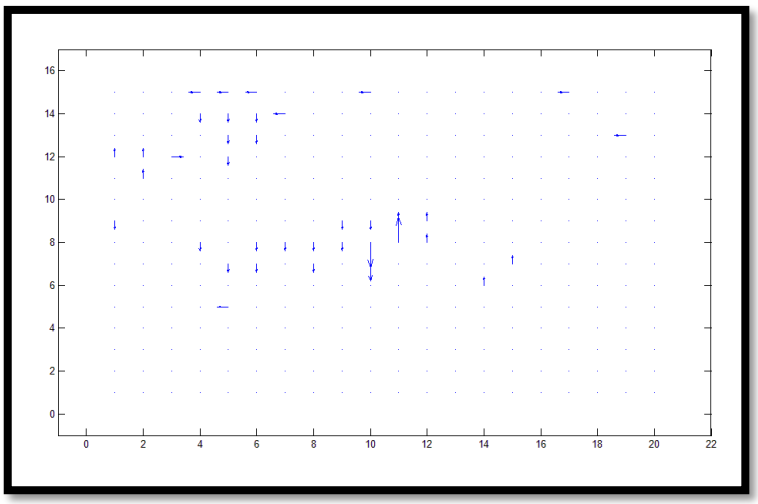


Figure 3.7(b) Motion Vector of Frame #3

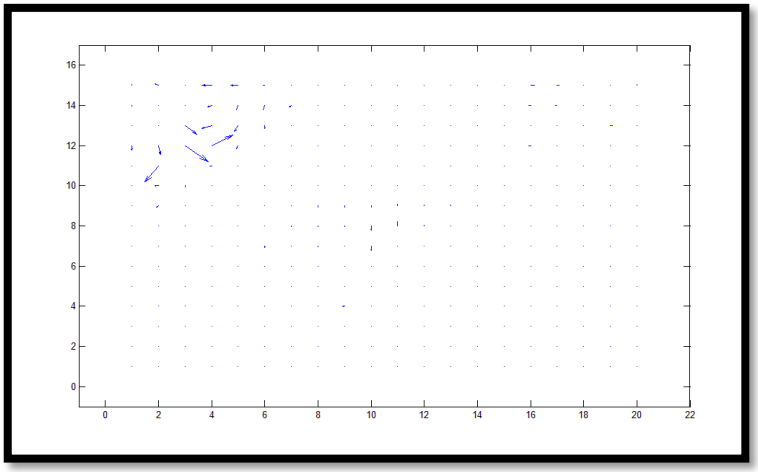


Figure 3.7(c) Motion Vector of Frame #4

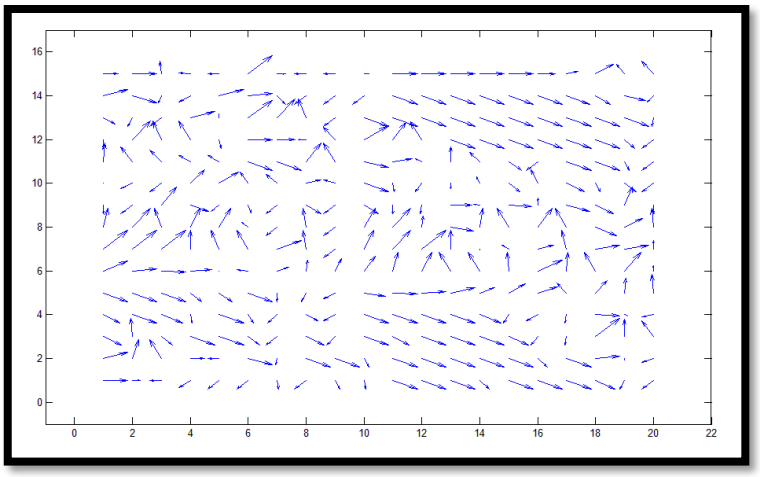


Figure 3.8(a) Motion Vector of Frame #2

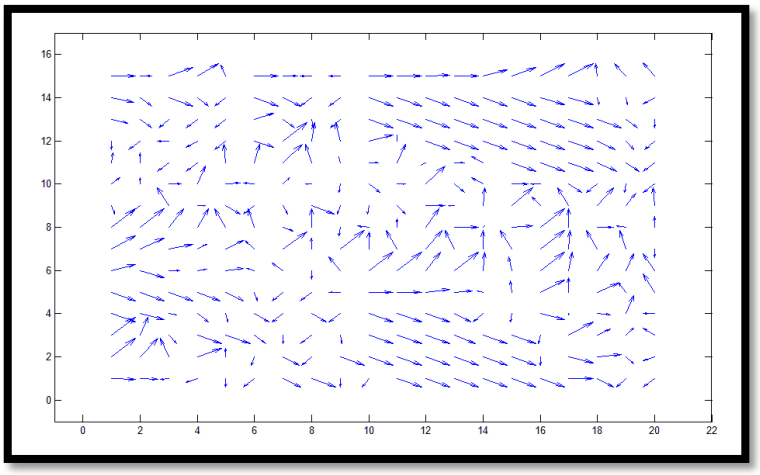


Figure 3.8(b) Motion Vector of Frame #3

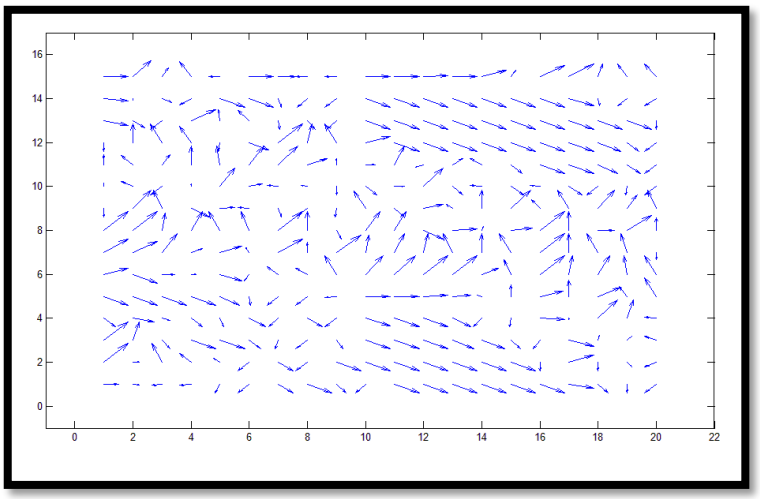


Figure 3.8(c) Motion Vector of Frame #4

3.3.2 Transformation and Quantization Units

In the encoder, after applying the motion estimation and compensation in the data and gain residual frame with motion vectors, transformation and quantization units receive the residual frames and process them. The transform unit reduces the temporal redundancy (inter frame) in the residual frames. The residual frame has a high correlation between pixel in one frame (intra frame) or a high correlation between the frames future and previous (inter frame of B frame type). The data after transformation is easy to compress rather than untransformed one. The results are called (transform coefficients) and they quantized by the quantization unit. There are many types to achieve the transformation like (2D-DCT).[27][28]

The range of the QP in H.264 from (1 to 100), where QP=1 represents less quality in coding, but high quality at the decoder, where QP=100 represents a higher quality at the encoder but less quality at the decoder. These reasons lead more companies to choose the threshold of QP at (50). So the q in inter prediction =50 and it have a matrix to divide the coefficients by it, at the encoder and multiply the coefficients by it.

$$q(50)=\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

The 2D-DCT equation that used in calculation is

$$Y = [A\ X\ A^T] \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation}(3.1)$$

This equation is done in Luma and Chroma components so the loop is over (6) times, (4) Luma components, and (2) Chroma components (red and blue).

Table (3.4) shows the residual frame values, table (3.5) shows the coefficients after 2D-DCT and table (3.6) shows the coefficients after quantization process.

Table (3.4) Residual Frame Valuse

Residual Frame Values						
52.9840	51.9540	51.9450	50.9240	50.9240	48.8640	48.8640
51.9540	51.9540	51.9540	51.9540	51.9540	51.9540	51.9540
52.9840	52.9840	52.9840	52.9840	52.9840	52.9840	52.9840
55.0440	55.0440	54.0140	52.9840	50.9240	48.8640	48.8640

Table (3.5) Residual frame after 2D-DCT

Residual frame after 2D-DCT						
12.6505	8.6638	3.3989	-2.4672	2.5121	3.6925	-8.2057
-4.6662	-2.8936	-4.6566	-3.5417	-7.2930	-1.5950	-8.6112
3.5197	8.936	1.0166	1.52886	-8.1026	-1.0530	7.5416
-5.1561	-1.5854	5.2176	-1.8712	1.3957	4.9691	1.7447

Table (3.6) Transformed frame after quantization

Transformed frame after quantization						
206	11	-1	1	0	0	0
2	-3	-1	0	-1	0	0
-3	-2	0	0	0	0	0
-1	2	0	-1	0	0	0

It is good to mention that the 2D-DCT is better than 1D-DCT. Although they are similar in work but there is a different that the 2D-DCT is repeating a mathematical process twice and this is leading to better results. Figures (3.9, 3.10) show the difference between 1D-DCT and 2D-DCT.

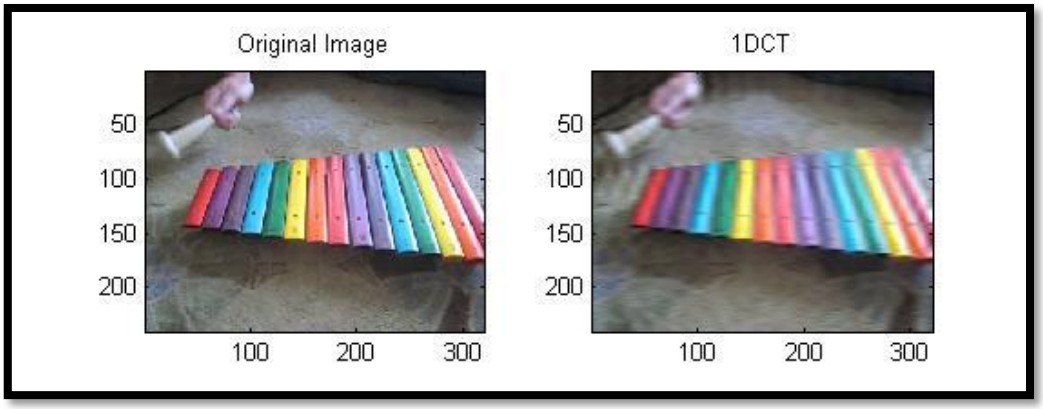


Figure (3.9) 1D-DCT

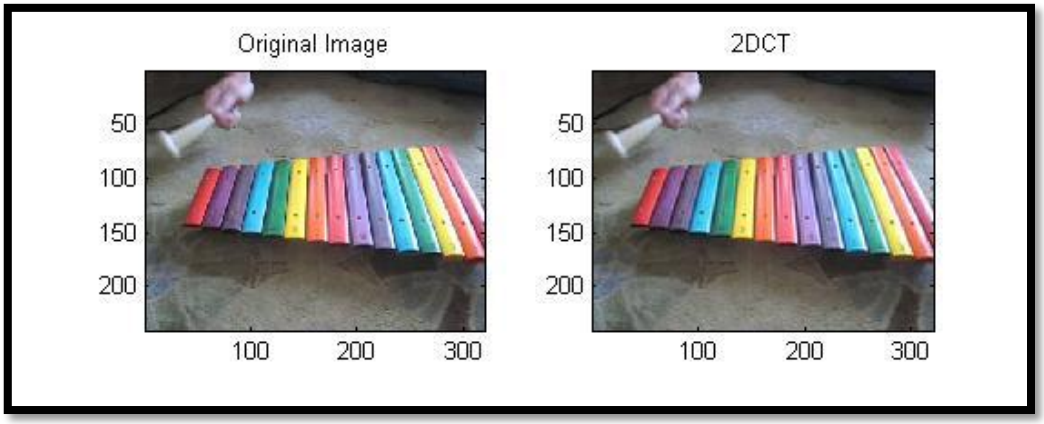


Figure (3.10) 2D-DCT

3.4 H.264 /AVC Decoder

The H.264 decoder receives the compressed video and decode it to gain the original video. The decoder receives the quantized data, it rescale it by multiplying by the same value of QP (50) matrix, doing inverse transform, reconstruct it and convert the result to the RGB color space then the video is return.[29][30]. Figure (3.11) shows the decoder steps.

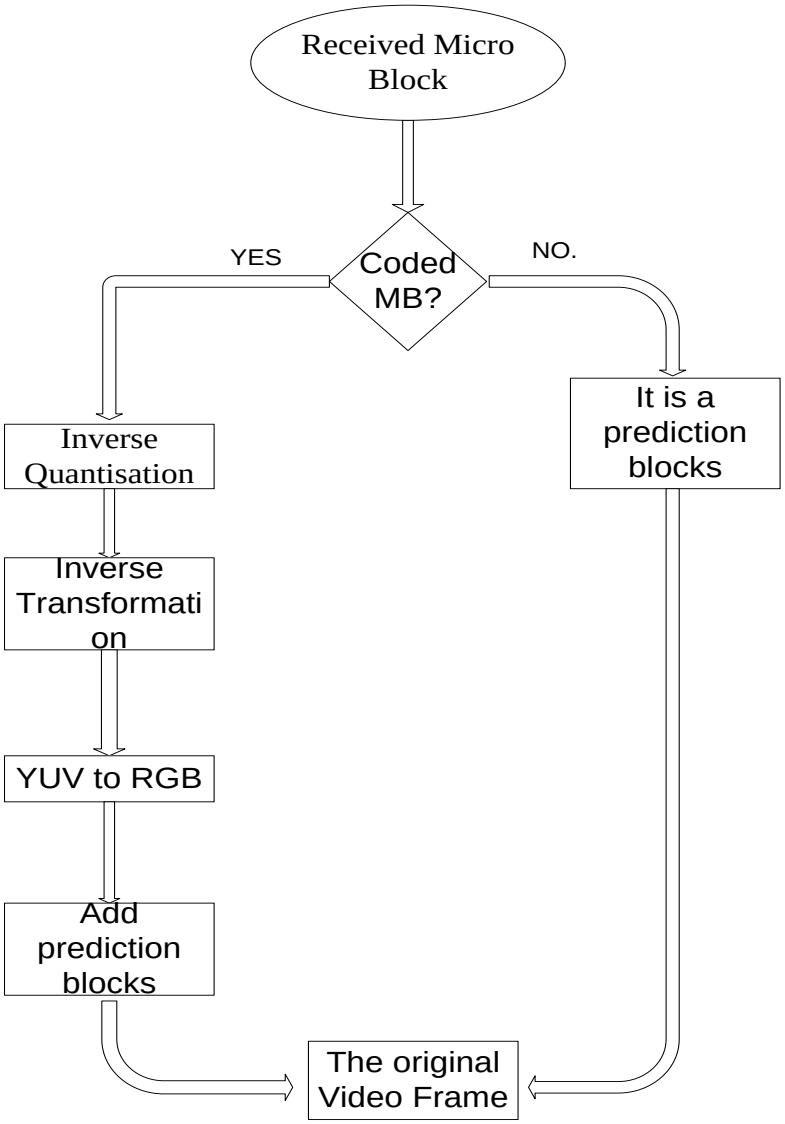


Figure (3.11) H.264 Decoder Steps

3.4.1 Rescaling and 2D- IDCT

The residual frame is rescaling in inverse quantization unit by Multiplying it with the same QP matrix value at the encoder (intra frame) or by (q=16) for inter frame. The receive frame is tailed with motion vector that indicate to the micro block with best matching in the search area. The (MB) is a new micro block that store motion vectors in both X and Y axis. After storing the motion vector and adding them to the blocks in frame, the rescaling and inverse DCT are applying.[32]

The inverse 2D-DCT equation explained in Equation (3.2)

$$X = [A^T Y A] \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \text{Equation}(3.2)$$

Table (3.7) shows the results of the received frames after rescaling (multiplying with 16) process.

Table (3.7) Rescaling Frame Values.

Rescaling Frame Values.						
412	15.1250	-1.2500	2	0	0	0
3	-4.500	-1.7500	0	-3.2500	0	0
-5.2500	-3.2500	0	0	0	0	0
-1.7500	4.2500	0	-3.6250	0	0	0

The inverse transformed results is shown in Table (3.8)

Table (3.8) Inverse Transform Frame Values.

The Inverse Transform Frame Values.						
51.5561	53.1186	53.3263	51.5919	50.2933	50.1092	49.3218
51.7891	52.4069	52.2352	51.5572	51.8601	52.4600	51.3932
53.5538	53.0438	52.1943	52.0437	53.0908	53.7066	52.1459
55.9915	55.0386	53.7442	52.9326	52.3749	51.9800	50.2166

Reconstructing frame is the inverse step to the one in the encoder (inter prediction). It adds the previous frame predicted by the encoder and stored in the decoder with the forward one because the decoder is designed as (IBBB), where (B) is used previous and the future frames. This operation is do depending on the information that attached with encoding frame. Motion vectors and residual frame information are attach with the sending encoding video.

Figure (3.12) shows the frame after reconstructing process but it still in YUV color space. After frame reconstructed, its color space must be change to RGB again to be ready to show.

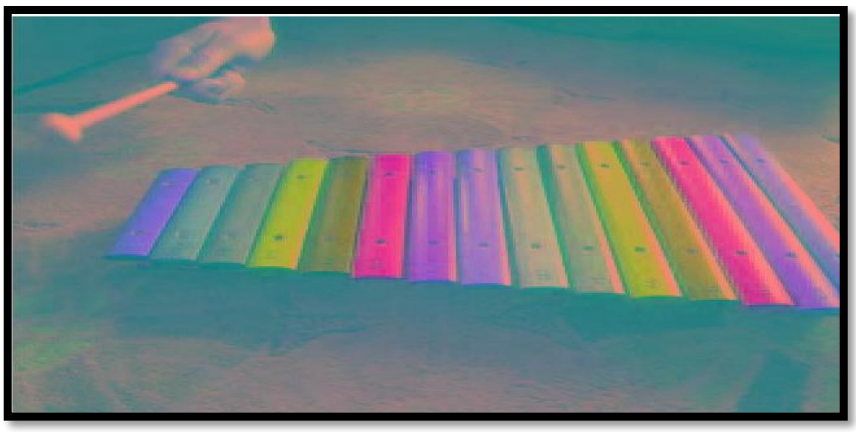


Figure (3.12) Reconstructing Frame

The last step is to format the decoded frame into formula worthily to the viewer. The decoder form the blocks into frames and view them. Table (3.9) shows the values of the decoding frame and table (3.10) the original frame.

Table (3.9) Decoding Frame

The Decoding Frame						
51.5561	53.1186	53.3263	51.5919	50.2933	50.1092	49.3218
51.7891	52.4069	52.2352	51.5572	51.8601	52.4600	51.3932
53.5538	53.0438	52.1934	52.0347	53.0908	53.7066	52.1459
55.9915	55.0386	53.7442	52.9325	52.6749	51.9800	50.2166

Table (3.10) Original Frame.

The Original Frame.						
52.9840	51.9540	51.9540	50.9240	50.9240	48.8640	48.8640
51.9540	51.9540	51.9540	51.9540	51.9540	51.9540	51.9540
52.9840	52.9840	52.9840	52.9840	52.9840	52.9840	52.9840
55.0440	55.0440	54.0140	52.9840	51.9540	51.9540	51.9540

3.5 Compression Rate and Bit Error

To check the quality and the efficiency of H.264/ AVC design , compression rate and bit error calculations are used .

3.5.1Compression rate

The term that represent the compression power. It used to calculate the amount of reduction in video size after using the compression algorithm, here the H.264 encoder. Equation (3.3) explains the compression rate calculation

$$\text{Compression rate} = 1 - \frac{compressed\ size}{uncompressed\ size} \ldots\ldots\ldots \text{Equation (3.3)}$$

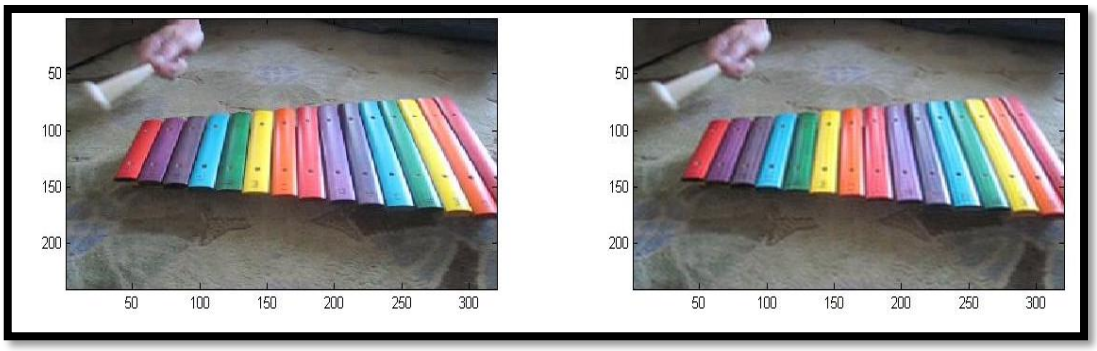
The uncompressed video size is about = (522KB) and the compressed size is about = (150K). The compression ratio = 71%, and this is a good result.

3.5.2 Bit Error Rate

In H.264 /AVC, quantization and inverse quantization, units are the only parts with lossy compression because they used (round) in their calculation. This rounding calculation is lead to lose some of the information. The number of frames we used are (28) frame, (7) I frames, and (21) B frames, I frame are not encoded, there is no losing in data, but the B frames are only having some bits loosed.

The (X and Y) are the original video frame with decoded one and they have (4838400 bit). The value of the bit errors are (51204). The bit error rate is = (0.0106), which means 99% of bits are not lost. It is good result and it means only (0.0106) of the total frames are lost.

Figure (3.13) shows the video frame after decoding it and with the original one.



Original video **Decoding video**

Figure (3.13) Original and Decoding video

3.5.3 Timing Calculations

As we know, the compression and decompression algorithm take time and this time determines one of the properties for calculating the efficiency of the design. Good design with low storage capacity, low time for calculations the processing of the design, and high compression and de-compression, all these are our aims in design.

The time of compression and decompression for the 28 frames and divided in to two stages (time for I frames and time for B frames).

- The I frame or intra frame takes a time as clear in the table (3.11)

Table (3.11) I Frame Time

No. of frames	Frame size	Frame type	Encode time	Decode time
7	(320X240)	I frame	0.015**(fr\sec)**	0.013**(fr\sec)**

- The B frame or inter frame takes a time as clear in table (3.12)

Table (3.12) B Frame Time

No. of frames (21)	Frame size	Frame type	Encode time	Decode time
21	(320X240)	B frame	0.0052**(fr\sec)**	0.0004**(fr\sec)**

The total time for encoder and decoder is equal to:

$$total\,time\ for\ H.264\ encoder\ = (7 * I) + (21 * B);$$

$$total\ time\ for\ h.264\ decoder = (7 * I) + (21 * B);$$

Total for encoder $(7 * 0.015) + (21 * 0.052) = 0.2142\ S$;

Total for decoder$(7 * 0.013) + (21 * 0.0004) = 0.0994\ S$;

This is a good time to encode 28 frame of (IBBB) pattern frames. The encoder time is about twice the decoder time because the encoder contains a decoder in its design, which results in a doubling of time. On the other hand, the decoder only read the data from the encoder and decode them.

We must take on respect in time calculating, the specifications of the personal Laptop we are working on and the size of the video we are trying to encode and decode it.

Chapter Four

Implementation of H.264/ AVC Encoder and Decoder in VHDL

4.1 Introduction

After designing the H.264/ AVC (encoder and decoder) in Matlab and get the results as described in chapter 3, they had to be design and obtained as a hardware. This design help us to calculate the total storage, number of operations, time etc. they have been utilized. This can be achieved by using the (ISE design of version 14.1), program which is produced by (Xilinx) company. ISE (Integrated Synthesis Environment), Is Xilinx software tool that produced by Xilinx company. It is used for synthesis, analysis of designs, enabling the programmers to synthesize or ("compile") their designs, perform the timing analysis, view the RTL diagrams, simulate its design to see the results, and finely to configure the target device with the programmer. The programming process on the kit is doing using (Spartan601). In this chapter we will talk about the intra prediction decoding and how to design our own encoder where its output we depend.

4.2 FPGA Chip Expression

FPGA (field programmable gate array) is an integrated circuit design to be configure by a customer or a designer after manufacturing. Semiconductor device containing programmable logic components which are called (logic blocks), and the programmable interconnects. We can program it for any digital function so we can apply the idea in practice. The programming on

FPGA is precedent by steps to be precede by steps taken on the computer programs to ensure the validity of the idea and the possibility of practical application. In our work, Matlab program was the best application program to get the results before implemented in practice (FPGA kit). This step is necessary in order not to waste the space inside the FPGA kit and in order not to waste a lot of time in the experiment and correction while the mistakes are made in the design and because the kit loses its functionality in case of power outages.[23][38]

4.3 FPGA Basic Parts

The FPGA kits, although there are differences but they are common in most parts [24].

Table (4.1) explains the specifications of the parts in FPGA kit.

Table (4.1) FPGA Parts specifications

Parts	Specifications
Programmable Logic blocks	Use to provide the basic computations and the storage elements that used in the digital systems.
Programmable Interconnect(routing)	They provide connections between (logic blocks) and (I/O blocks).
Programmable I/O(input/output)	They are necessary because they interface the logic blocks and the routing architectures to the wide range of external components to FPGA, which are called programmable I/O).
The ALU(arithmetic logic unit)	The (ALU) is the most important component of FPGA kit because all the

	arithmetic and logical computations are performed inside it.
RAM Blocks	RAM (random access memory) is the form that is used to store the data and functions in the computer

4.4 Spartan 601 Overview

The (SP601) kit board enables the hardware and software programmers to create or evaluate designs targeting the Spartan 6 XC6SLX16-2CSG324 FPGA. The SP601 provides a board features that the programmers used like the some commonly used features are DDR2 memory controller, UART, a tri-mode Ethernet PHY, general-purpose I/O (GPIO), and a parallel linear flash. The Spartan-6 FPGA Family is offering an optimal balance of power, cost and performance. Figure (4.1) shows the board of the Spartan 6.

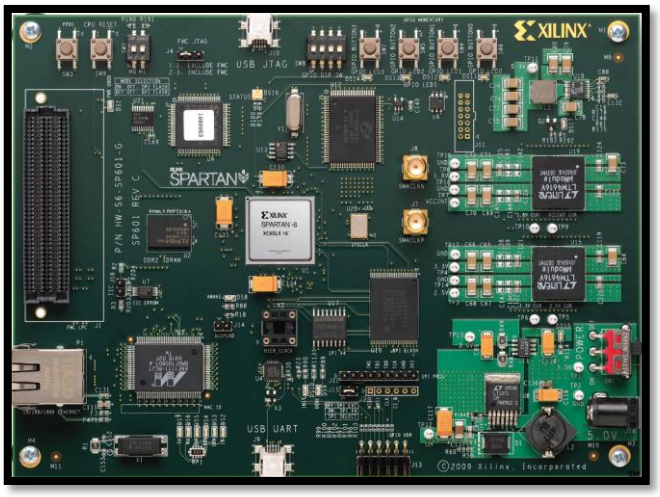


Figure (4.1) Spartan6 Kit

It has some key features:

- Memory component of DDR2 (128MB).
- JTAG for configuration and serial UART for communication.
- 200MHZ oscillator for clock supporting.

- 4Xled for display.
- 4X push buttons and 4X DIP switches for controlling.

4.5 H.264 Encoder and Decoder in FPGA

The H.264/ AVC has many features in terms of compression and decompression and this is why it is consider as a global standard. This standard applies the encoding process in two ways; inter prediction, or intra prediction**.** The intra frame consists groups of micro blocks and everyone is encode from previous micro block in the same frame. Due to some of the intra prediction properties and determinants of this type, we have indicated its use. Intra prediction has three types of prediction (16x16), (8x8), and (4x4) for Luma prediction and (8x8) only for Chroma prediction. Depending on the type of choice, when smaller prediction size (4x4) is chosen, it tend to have more accurate prediction and less number of residual data at the encoder but less data to reconstruct to original frame at the decoder, and vice versa at (16x16). The intra prediction sometimes used when there is more details in the video and needs to be accurate at the encoder but here the inter predation is unusable way because it takes more time to calculate this details . Spartan 6 is designed to perform this type of coding process (encoder and decoder) because in inter prediction the encoded frame must be store in an external memory to be used to encode the frame following it in (P type) or at least 3 frames store to predict the third one at (B type) .[33][34][40]

4.6 The H.264 /AVC Encoder Design

As we explained, the H.264 encoder has steps to perform the encoding. The dedicated frame read from the external memory of type (RAMB16BWER). The frame must be read to execute at least one bit (at the transform and quantize units) and the residual data

must be decoded to gain the original frame. The proceeding is intra prediction, the frame has two processes (4x4 Luma prediction) and (8x8 Chroma prediction) .This separation between the two processes is to accelerate the implementation process. The Luma components applied the calculation at one CLK but the Chroma components applied the calculation at two CLK. At the end, they assemble and go to the transformation and quantization units. Figure (4.2) shows the overall encoder processes.

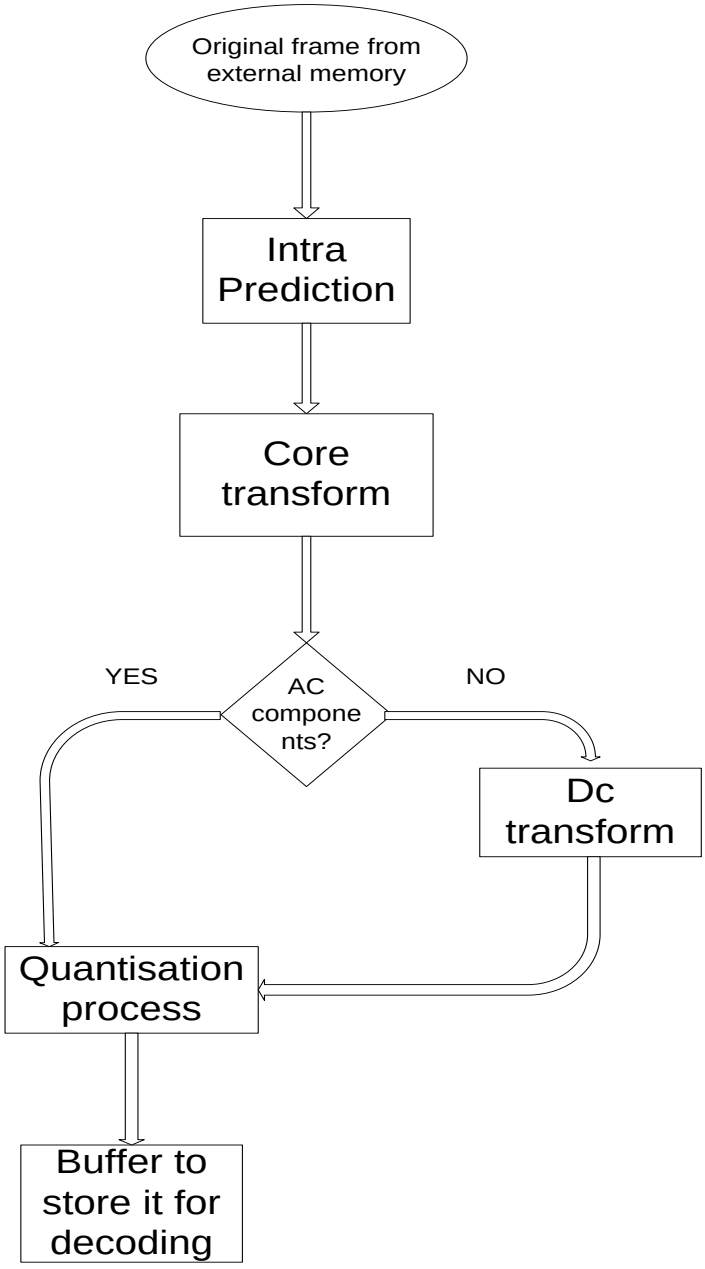


Figure (4.2) Encoder Processes.

4.7 H.264 Intra Prediction FPGA Design

The video frame is stored in an external memory, in every clock there is (4) byte ready to inter and process. The external memory is the best solution than LUT because the last one exploits space within chip and this has a negative impact in terms of storage and the time it takes to read the data. The memory that is used is type of (8*32bit) RAMB16_S9 and we used (4) of them because at every clock there is 4 byte reads. Figure (4.3) shows the external design of the memory in FPGA.

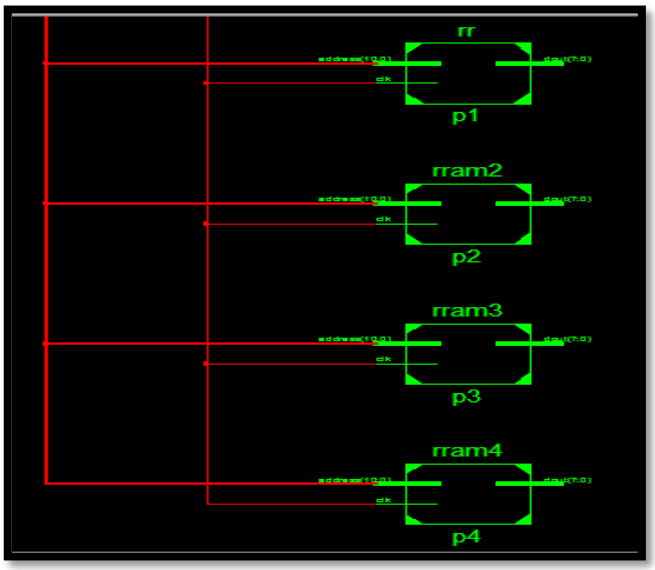


Figure (4.3) The four RAMs

Figure (4.4) shows the data output after getting address the memory.

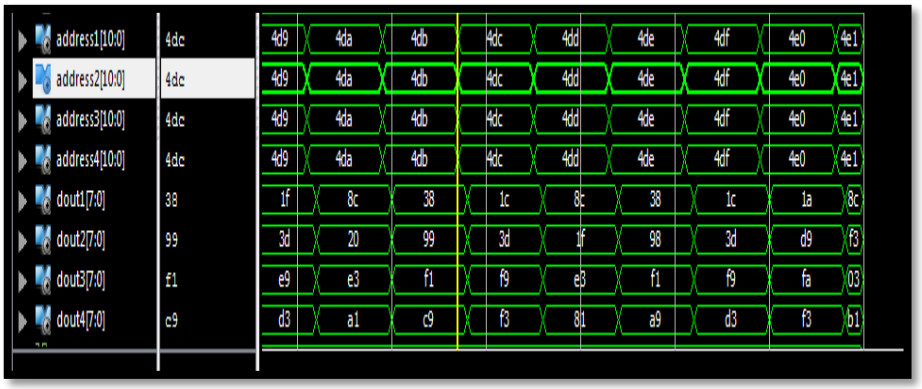


Figure (4.4) Reading from the Output Memory

The architecture of the intra prediction model depends on the (Luma and Chroma). Every mode has its own equation and after analyzing, every equation we can simplify them because they have common parts and this lead to reduce the architecture design area. [24][32]

Therefore, instead of dividing the predictor calculations into individual blocks for each mode to calculate them, the calculation are divide into two stages: base and derived prediction equations as shown in Table (4.2) and Table (4.3) respectively.

Table (4.2) Basic Equations in Intra mode

Output	**Equation**
Eq_0 =	A + B + 1
Eq_1 =	B + C + 1
Eq_2 =	C + D + 1
Eq_3 =	D + E + 1
Eq_4 =	E + F + 1
Eq_5 =	F + G + 1
Eq_6 =	G + H + 1
Eq_7 =	I + J + 1
Eq_8 =	J + K + 1
Eq_9 =	K + L + 1
Eq_10 =	M + A + 1

Eq_11 =	M + I + 1
Eq_12 =	2H + 1
Eq_13 =	2L + 1

Table (4.3) Derived Equations in Intra mode

Output	Equation	Derived Equation
Eq_14	M + 2*A + B + 2	Eq_0 + Eq_10
Eq_15	A + 2*B + C + 2	Eq_0 + Eq_1
Eq_16	B + 2*C + D + 2	Eq_1 + Eq_2
Eq_17	C + 2*D + E + 2	Eq_2 + Eq_3
Eq_18	D + 2*E + F + 2	Eq_3 + Eq_4
Eq_19	E + 2*F + G + 2	Eq_4 + Eq_5
Eq_20	F + 2*G + H + 2	Eq_5 + Eq_6
Eq_21	M + 2*I + J + 2	Eq_7 + Eq_11
Eq_22	I + 2*J + K + 2	Eq_7 + Eq_8
Eq_23	J + 2*K + L + 2	Eq_8 + Eq_9
Eq_24	A + 2*M + I + 2	Eq_10 + Eq_11
Eq_25	G + 3*H + 2	Eq_6 + Eq_12
Eq_26	K + 3*L + 2	Eq_9 + Eq_13
Eq_27	A + B + C + D + 2	Eq_0 + Eq_2
Eq_28	I + J + K + L + 2	Eq_7 + Eq_9

Eq_29	A+B+C+D+I+J+K+L+4	Eq_27 + Eq_28

The prediction calculator captures all the reconstructed pixels and then calculates the equations needed to create all predicted values for all nine modes in parallel. The final operation is to apply the equation results to predicted values as shown in Table (4.4). The prediction calculator needs at least one clock to generate all the predicted pixels.

Table (4.4) Equation Results to Predicted Values of Each Pixel

Pixel	**Modes**								
	0	1	2	3	4	5	6	7	8
A	A	I	Eq_29	Eq_15	Eq_24	Eq_10	Eq_11	Eq_0	Eq_7
B	B	I	Eq_29	Eq_16	Eq_14	Eq_0	Eq_24	Eq_1	Eq_22
C	C	I	Eq_29	Eq_17	Eq_15	Eq_1	Eq_14	Eq_2	Eq_8
D	D	I	Eq_29	Eq_18	Eq_16	Eq_2	Eq_15	Eq_3	Eq_23
E	A	J	Eq_29	Eq_16	Eq_21	Eq_24	Eq_7	Eq_15	Eq_8
F	B	J	Eq_29	Eq_17	Eq_24	Eq_14	Eq_21	Eq_16	Eq_23
G	C	J	Eq_29	Eq_18	Eq_14	Eq_15	Eq_11	Eq_17	Eq_9
H	D	J	Eq_29	Eq_19	Eq_15	Eq_16	Eq_24	Eq_18	Eq_26
I	A	K	Eq_29	Eq_17	Eq_22	Eq_21	Eq_8	Eq_1	Eq_9
J	B	K	Eq_29	Eq_18	Eq_21	Eq_10	Eq_22	Eq_2	Eq_26
K	C	K	Eq_29	Eq_19	Eq_24	Eq_0	Eq_7	Eq_3	L

L	D	K	Eq_29	Eq_20	Eq_14	Eq_1	Eq_21	Eq_4	L
m	A	L	Eq_29	Eq_18	Eq_23	Eq_22	Eq_9	Eq_16	L
n	B	L	Eq_29	Eq_19	Eq_22	Eq_24	Eq_23	Eq_17	L
o	C	L	Eq_29	Eq_20	Eq_21	Eq_14	Eq_8	Eq_18	L
p	D	L	Eq_29	Eq_25	Eq_24	Eq_15	Eq_22	Eq_19	L

4.7.1 (4x4) Luma Prediction in FPGA

The H.264 standard exploits the spatial correlation between the adjacent micro blocks to achieve the Intra prediction. The current micro block predicted using adjacent pixels in the upper and left micro blocks that are decoded earlier. The H.264 /AVC Provides a rich sets of prediction patterns that useful for predicting within i.e. nine prediction modes for 4x4 Luma blocks. The standard provides equation at each mode but the DC mode is the best and its equation represents by **(**equation 29**),** in table (4.2)**.** The data received from external memory and the receiving frame divided into (16x16) block size. The blocks contain Luma and Chroma. The blocks of data divided into (4x4) micro blocks. The components are transform, then the transformed are contain (AC and DC) components. The DC components values tend to be a highly correlated and they transformed again by using (4x4) Hadamard transformed so, the all components gathered again to be quantized .The input data is (32 bits) because we take 1 bytes at each clock so we use 4 memories to get the data. Figure (4.5) shows how the data is read from the memory and its output from the (4x4) Luma prediction unit.[35][37]

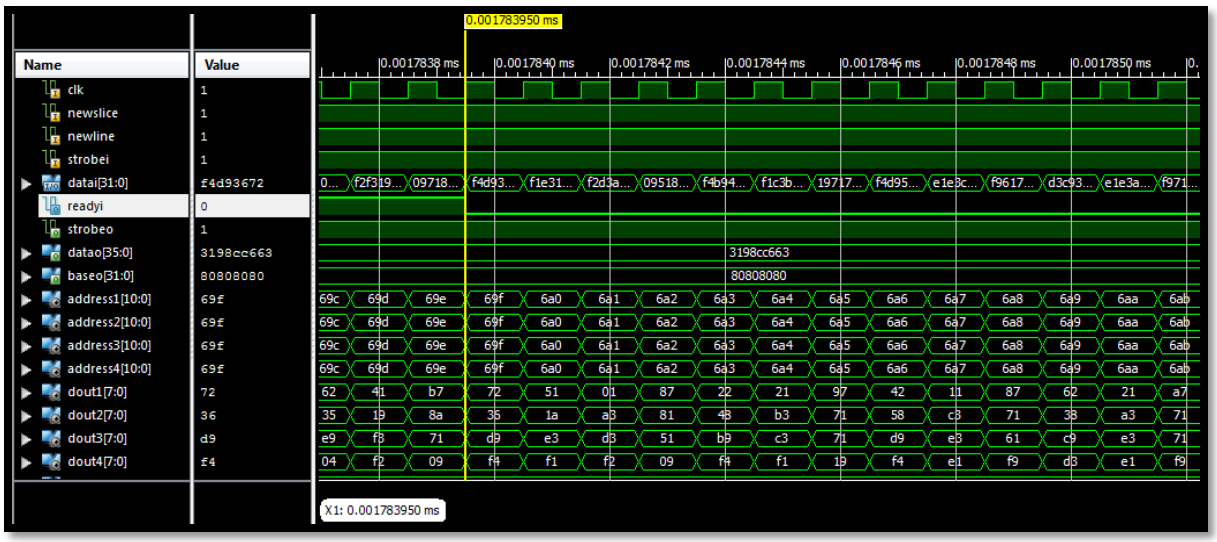


Figure (4.5) (4x4) Luma Prediction.

Figure (4.6) shows the architecture of the (4x4) Luma unit.

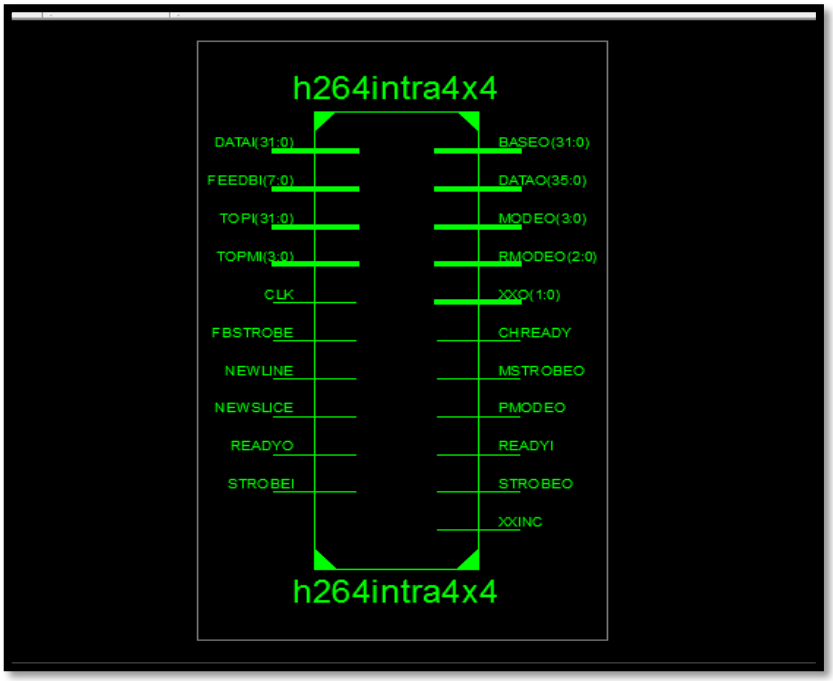


Figure (4.6) (4x4) Luma Unit Architecture .

4.7.2 Chroma Intra Prediction in FPGA

The greatest idea that used in H.264 is to convert the colors from RGB to YUV of (4:2:0) domain to reduce the memory needed to store the

pixels. The color components is process as well as the Luma components. The Chroma components divided into (AC and DC) components, for each red component and blue component. Each component of Chroma is gone to be (2x2) block size for DC components and (4x4) block size for AC component. The Hadmard transform is apply over DC components and integer transform over rest components. Figure (4.7) and (4.8) respectively show the data input and output to the units and the architecture of the 8x8 Chroma unit.

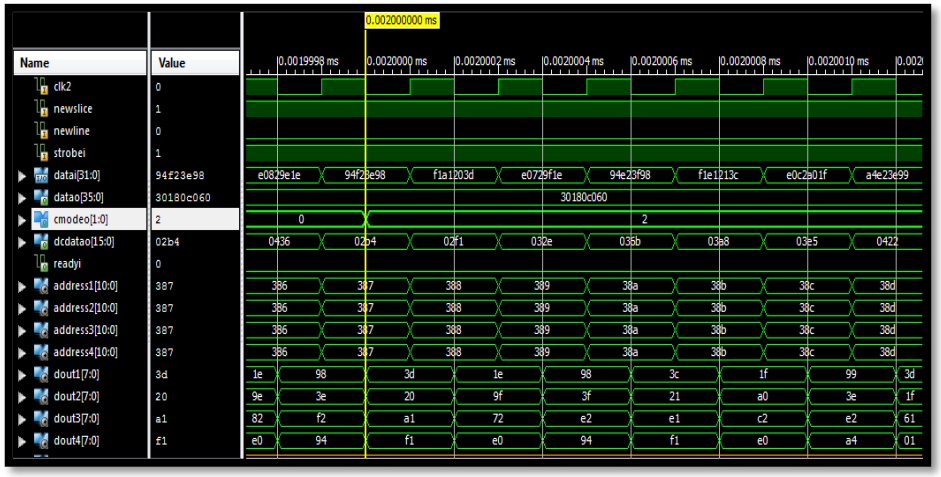


Figure (4.7) Data In and Data Out, the (8x8) Chroma Unit

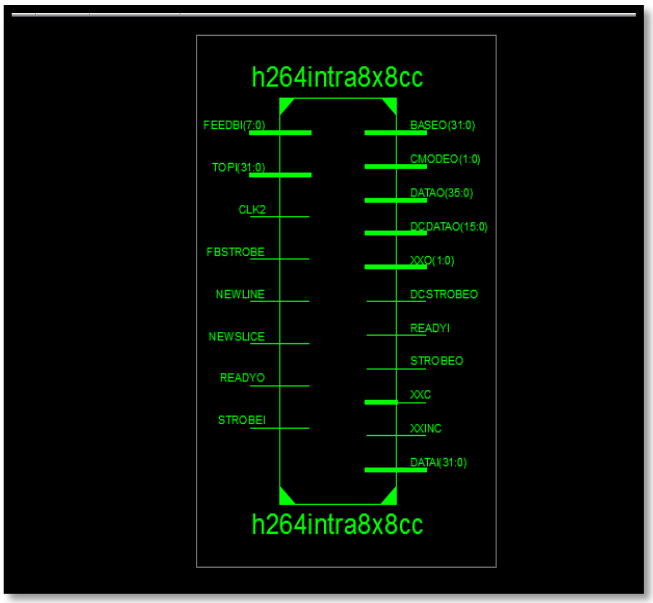


Figure (4.8) Architecture of the (8x8) Chroma Unit.

4.7.3 The Transformation Unit in FPGA

The transformation term use to transform data from one domain to another. In this proposal, design used two types of transforms:

- DCT transform or (core transform) for each (4x4) Luma components or (8x8) Chroma components.
- Hadamard transform for the DC components of (4x4) Luma components.
- Hadamard transform for (2x2) the DC Chroma components (cb, cr).

In the default process over Luma components, each (4x4) block within the (16x16) Luma of the micro block transformed to form coefficients from (0 to 15). If the micro block predicted by using (16x16) intra prediction, a second transform must apply to the DC frequency coefficients of the first transform. This second transform is (4x4) DC transform or Hadamard transformation. The Chroma components are also have a part of transformation. The (16x16) micro block has (8x8) cb and (8x8) cr coefficients and each (4x4) block of cb, cr is transformed using (DCT transform or core transform), the residual components (DC) are further transformed by (2x2 Hadamard transform). [39]

Figure (4.9) and (4.10) show respectively each kind of transform over Luma and Chroma components.

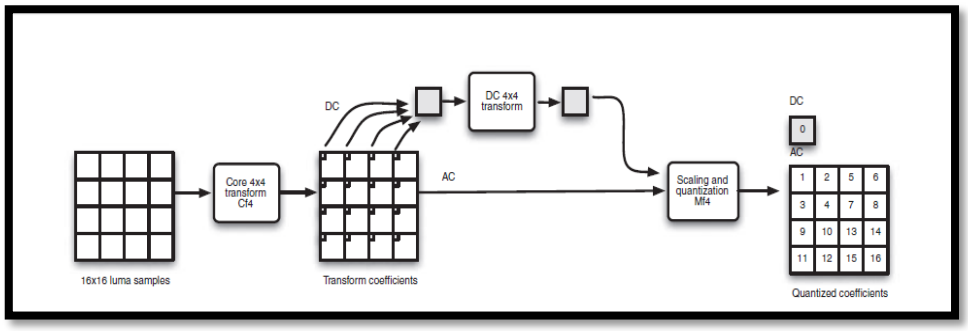


Figure (4.9) Forward Luma Transform

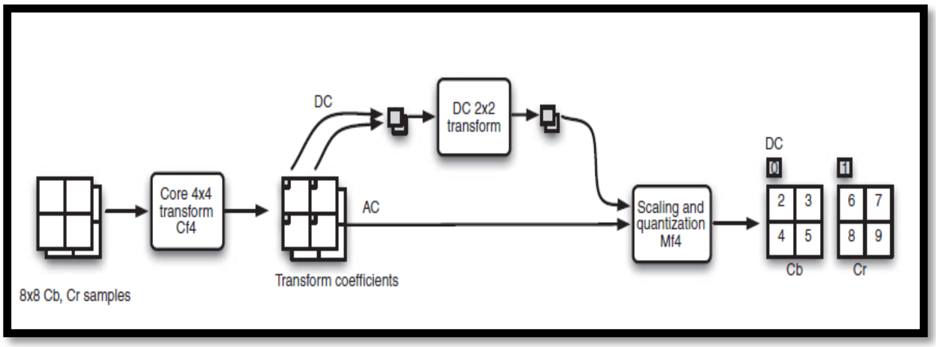


Figure (4.10) (8x8) Forward Chroma Transform

4.7.3.1 DCT Transformation or (Core Transform)

For each Luma and Chroma components, DCT transform must be apply. The Discrete Cosine Transform is operating on X matrix, which is a block of (NxN) samples and creates a block of Y matrix of the same dimension.[39], the transformation equation:

$$Y = AXA^T \text{ ……………………………………………….Equation (4.1)}$$

Where A= $\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}$

$$Y = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} . [X]. \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix} \text{……………Equation (4.2)}$$

Where X is the components matrix. After intra prediction, the dynamic range of the transform inputs data is 9 bits, i.e. from (-256 to +255) and because we used arithmetic operations like (additions, subtractions and shifts), the dynamic range of the pixel data is extended to (16 bit). So, the 4x4 residual data are process in parallel by the transform block, which consists of two cascade (one 1D row transform) and (one 1D column

transform). Figure (4.11) shows the architecture of the (DCT or core transform) design.

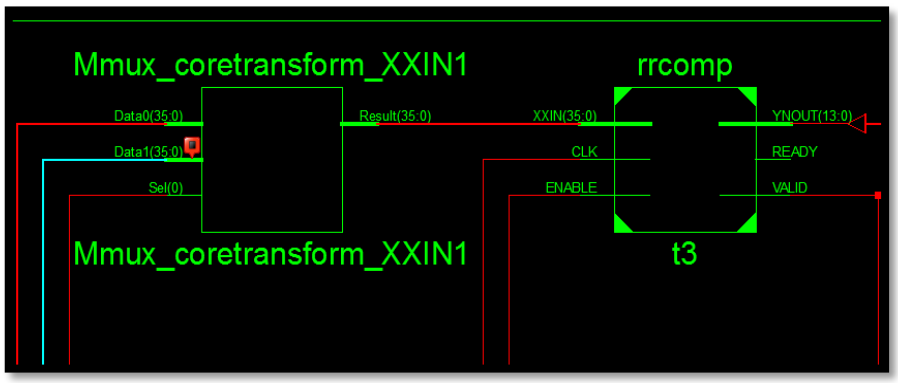


Figure (4.11) The DCT or Core Transform Design.

Figure (4.12) shows the signals in the design that comes from the (4x4) Luma prediction unit.

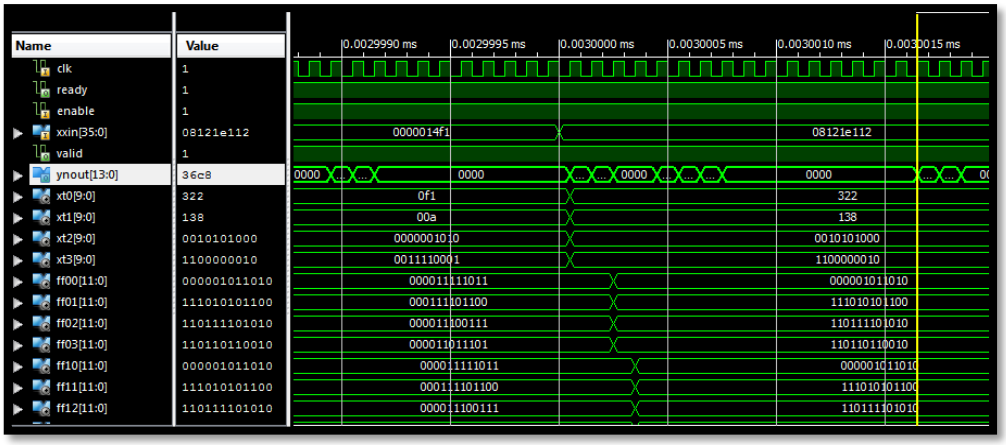


Figure (4.12) Signals of the DCT Design.

4.7.3.2 The DC Transform (Hadamard Transform)

If the micro block is encoded in (16x16) Intra prediction mode, in which the entire (16x16) Luma component is predicted from the neighbors pixels. Each residual of (4x4) block size is first transform using the DCT or Core transform described above. The DC coefficients of this (4x4) micro block transformed again by using (4x4) Hadamard transform.

The DC blocks gathered after the DCT transformation. Equation (4.3) shows the Hadamard transform.

$$Y = {(BXB^T)}/{2} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation}(4.3)$$

Where X is the blocks of (4×4) DC coefficients

And B=$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}$.

The equation (4.3) becomes as describe in equation (4.4)

$$Y = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} . [X] . \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \right) \Big/ 2 \quad \ldots\ldots \text{Equation (4.4)}$$

The DC coefficients of each transformed (4x4) Chroma blocks are grouped and transformed for a second time. The video format is (4:2:0) and there are four blocks of (4x4) in each Chroma coefficients and the DC coefficients form a (2x2) block, which is then transformed using the Hadamard transform equation.[25]

$$Y = CXC^T \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation } (4.5)$$

Where X is the matrix of the DC coefficients.

And C is the matrix with values of

C=$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$. So the previous equation becomes:

$$Y = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} . [X] . \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \text{.... ....Equation (4.6)}$$

Figure (4.13) and Figure (4.14) respectively show the architecture design and simulation results of the (DC or Hadamard) transform.

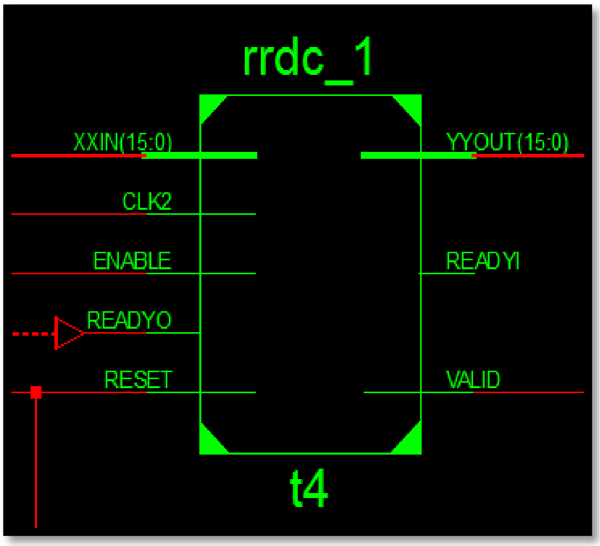


Figure (4.13) Hadamard (DC) Transform Architecture

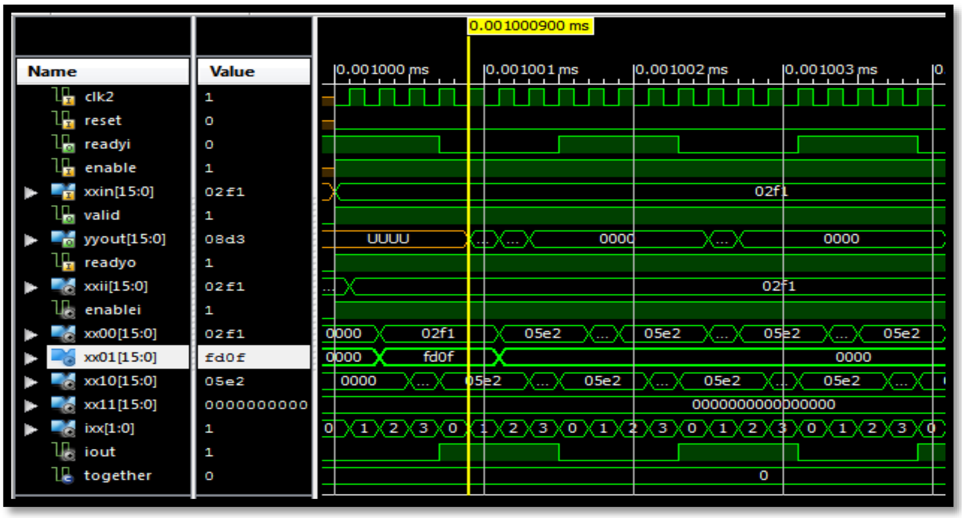


Figure (4.14) Hadamard (DC) Transform signals

4.7.4 The Quantization unit in FPGA

The quantization process is the mathematical operation that used in compression algorithms. The bright aim of the quantization unit is to

reduce the range of the coefficients and mapping them in to specific ranges. In the video CODECs standards, quantization can take place in two steps. First the forward quantization process that is built in the encoder and an inverse quantization unit in the decoder [25]**.** The quantization unit in the H.264 controlled by the Quantization Parameter (QP). It is the step size between two successive values. If it is large, the range of quantized value is small which giving a higher compression and vice versa. The output of the forward quantization unit (at the encoder) is an array of coefficients mostly converging to zero value, the quantization general equation:

$A_{ij} = round(Bij / Qstep)$…. ……Equation(4.7)

Where (B) is data after the transformation process. It is good to mention there are (52) of QP values, each one is having its corresponding Q step value as shown in Table (4.5).

Table (4.5) QP and Q Step Relationship

QP	0	1	2	3	4	5	6	7	8……	51
Qstep	0.63	0.59	0.81	0.88	1	1.13	1.25	1.38	1.625 ...	224

To avoid division in our work, which leads to a lack of values due to rounding operations and this negatively effects on the decoding process, although the values are few in encoder, we modified the equation (4.7) to equation (4.8) .

$A_{ij} = round\ (Bij.^{PF}/_{Qstep})$ …… Equation(4.8)

Where (PF) is varying according to coefficient position in matrix and its values can be Obtain from Table (4.6) that changes according to t matrix index [4.1].

Table (4.6) The Value of (PF)

PF	**Position (i,j)**
0.25	(0,0), (0,2) ,(2,0),(2,2)
0.4	(1,1), (1,3),(3,1),(3,3)
0.32	Others

Where $\frac{PF}{Qstep} = \frac{MF}{2^{qbits}}$ …… Equation (4.9)

The MF is the combination matrices of coefficients matrix and the scaling values and its represents the multiplication factor.

$\text{qbits} = 15 + floor\left(\frac{QP}{16}\right)$ …… Equation (4.10)

So the

$AA_{i\,j} = round\,(B_{ij}\,X\,MF + F)_{\gg qbits}$ …… Equation (4.11)

In intra prediction F= (2^{qbits}/3). For DC values, DC quantization is applied and the process has slightly changing.

$A_{ij} = round\left(B_{ij} \times MF_{zero} + 2F\right) << (q_{bits} + 1)$ …… Equation (4.12)

Where *(MF zero)* is the multiplication factor at index of (0,0), so value of MF is depending on QP only and not on the position in the matrix. Table (4.7) shows the relationship between MF and QP.

Table (4.7) Relationship between MF and QP

MF / QP	Position (0,0),(0,2), (2,0) , (2,2)	Position (1,1) , (1,3) , (3,1) , (3,3)	Other positions
0	13107	5243	8066
1	11916	4660	7490
2	10082	4194	6554
3	9362	3647	5825
4	8192	3355	5243
5	57282	2893	4559

Figure (4.15) and (4.16) respectively show the architecture design and the signal in the simulation of the quantization unit. It is needs only 4 clocks for (latching, multiplying, scaling and clipping) and get the output.

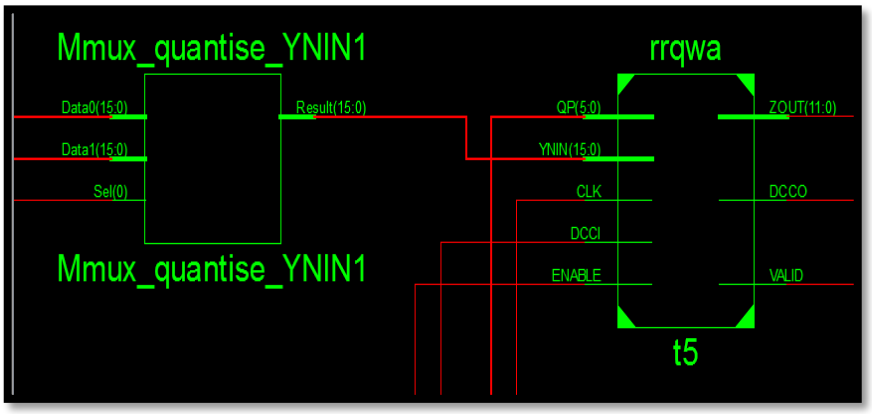


Figure (4.15) Quantization Unit Design

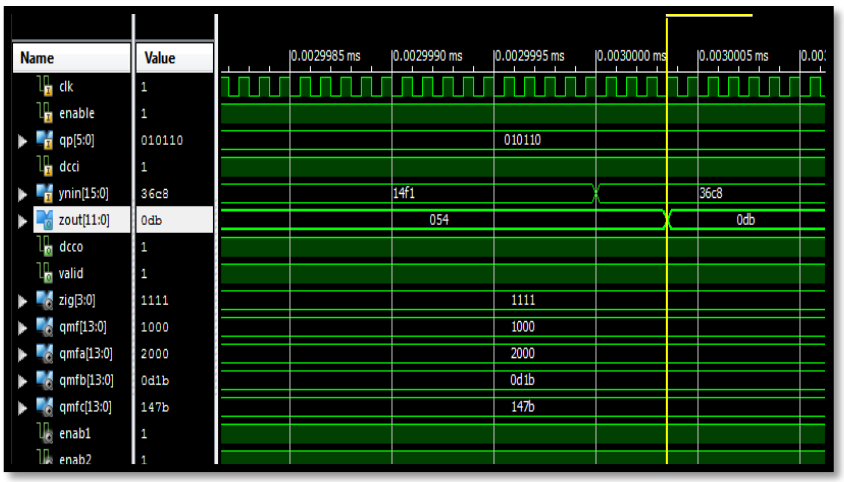


Figure (4.16) Quantization Unit Signals and Results

4.8 The H.264 Decoder Design in FPGA.

As already mentioned, the work of the decoder includes the reverse of what is in the Encoder and therefore all processes will be reflected and become (rescaling or de-quantized), inverse transform and reconstruction. The output of the encoder is the input to the decoder, so the coefficients a set of operations applied to restore the original values. The encoded coefficients (DC and AC), the DC coefficients is first inverse transform (inverse Hadamard transform) and they together with the quantized coefficient are inversed by inverse core transform. The set of decoded coefficients are reconstruct to gain the original data. Figure (4.17) shows the operation of the decoder. Figure (4.18) and Figure (4.19) show respectively the operation of the inverse Luma and Chroma transform.

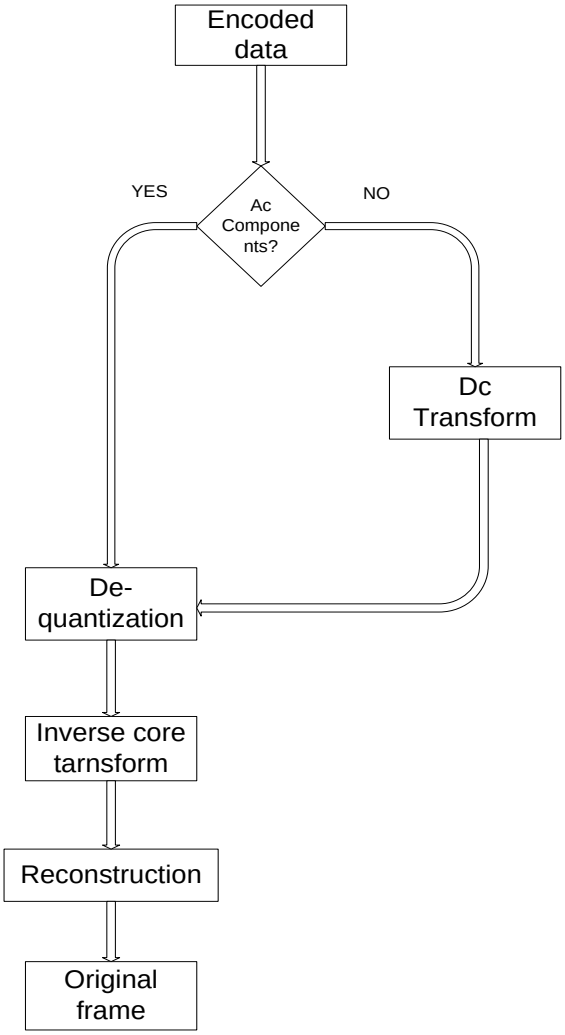


Figure (4.17) Decoder Process

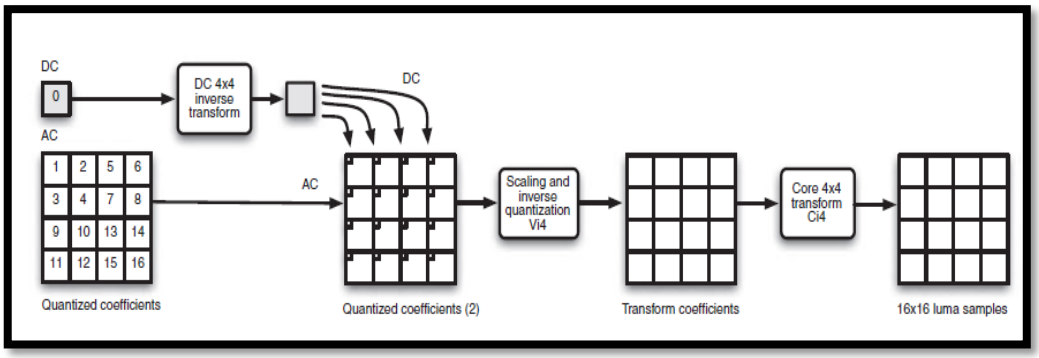


Figure (4.18) (4x4) Luma Inverse Transform

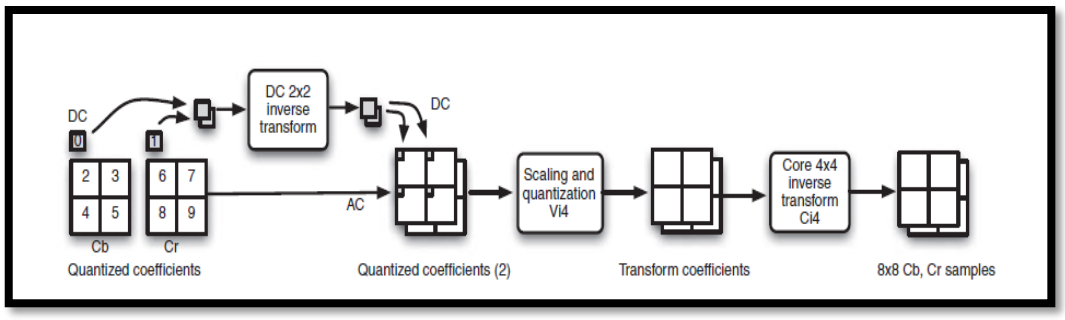


Figure (4.19) (8x8) Chroma Inverse Transform

4.8.1 Inverse Hadamard Transform.

The DC coefficient (Luma and Chroma) suffered from inverse Hadamard transform, which is the same in the encoder. For (4x4) Luma coefficients the equation (4.12) explain the inverse Hadamard equation

$Y = B\ X\ B^T$ ……..Equation (4.13)

where B is the matrix and equal

$B = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}$, so the equation (4.12) becomes

$Y = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} . [X] . \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}$ …… Equation(4.14)

For Chroma components, the inverse Hadmard is equal to equation (4.15)

$Y = C\ X\ C^T$ ……..Equation (4.15)

Where $C=\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$. So the equation (4.15) becomes

$Y= \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} . [X] . \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$……..Equation (4.16)

The operation is same to that used in the encoder. Figure (4.20) and Figure (4.21) respectively show the simulation results, and architecture design of the inverse Hadamard transform.

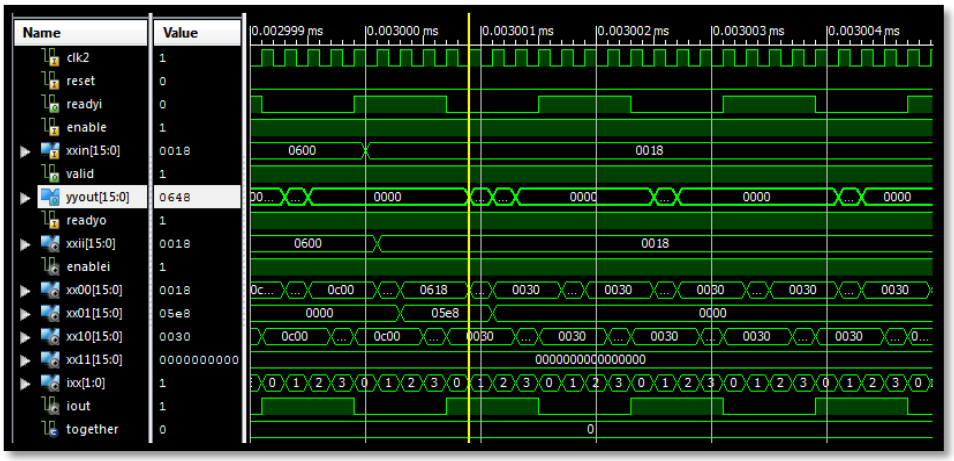


Figure (4.20) Simulation Results of Inverse Hadamard Transform.

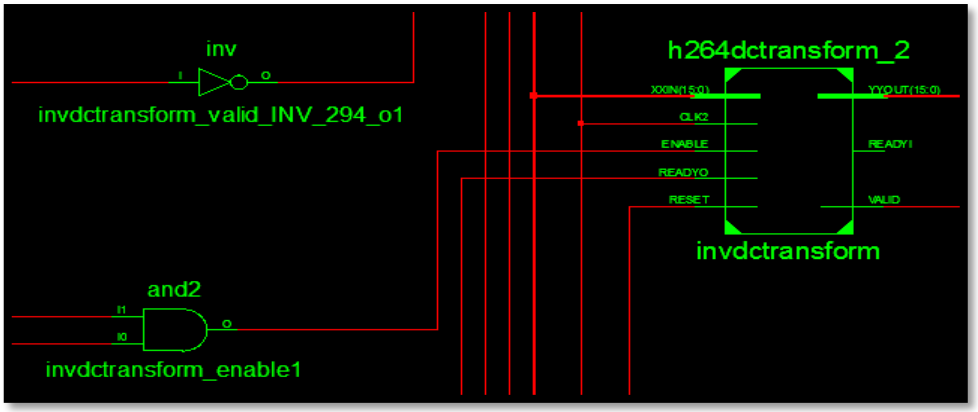


Figure (4.21) Architecture Design of Inverse Hadamard Transform.

The data input is the encoded coefficients that is coming from the transformation and quantization units are separate to AC and DC. The DC coefficients are inversed by Hadamard inverse transformed (for both Chroma and Luma). The results are gather with the quantized coefficients to be rescaling (de-quantized) and inverse core transform.

4.8.2 The Rescaling or De-Quantization Unit Design

The input of the de-quantization are sets of quantized and transformed coefficients. These coefficients (Luma or Chroma) are rescale. For (4x4), DC Luma matrix, the inverse quantization is taking place according to equation (4.11)

$$\text{Aij} = \text{round}\,(\,Bij \times V(0,0) \times 2\text{^ floor}\,(\text{Qp}/6) + 2\,) \ldots\ldots \text{Equation(4.17)}$$

for QP ≥ 12. For (2x2) DC Chroma matrix, the inverse quantization is taking place according to equation (4.12)

$$\text{Aij} = \text{round}\,(\,Bij \times V(0,0) \times 2\text{^ floor}\,(\text{Qp}/6) + 1\,) \ldots \text{Equation(4.18)}$$

For QP $\geq$ 6. The V is given by table (4.8) which is represents the rescaling factor.

Table (4.8) Represents the Recalling Factor

V / **QP**	**Position (0,0),(0,2),(2,0),(2,2)**	**Position (0,0),(0,2),(2,0),(2,2)**	**others**
0	10	16	13
1	11	18	14
2	13	20	16
3	14	23	18
4	16	25	20
5	18	29	23

The Figure (4.22) and Figure (4.23) respectively show the simulation results of the design and the architecture design.

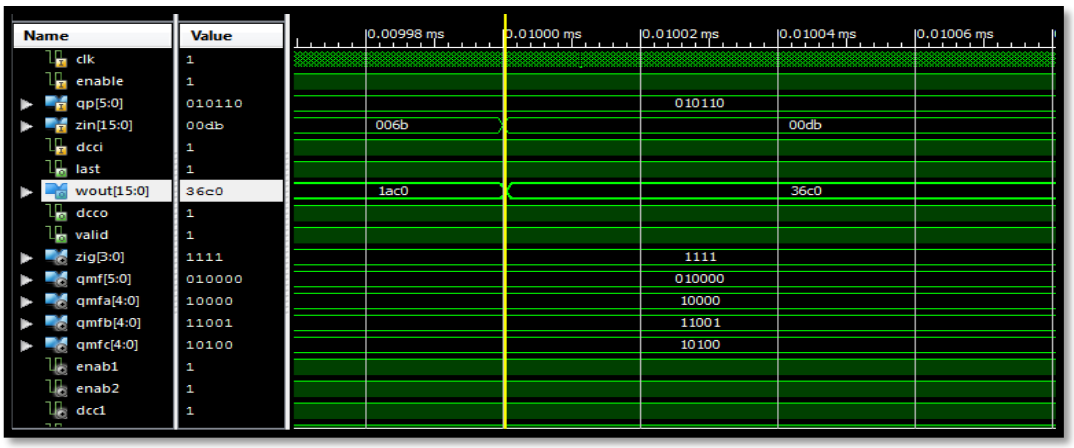
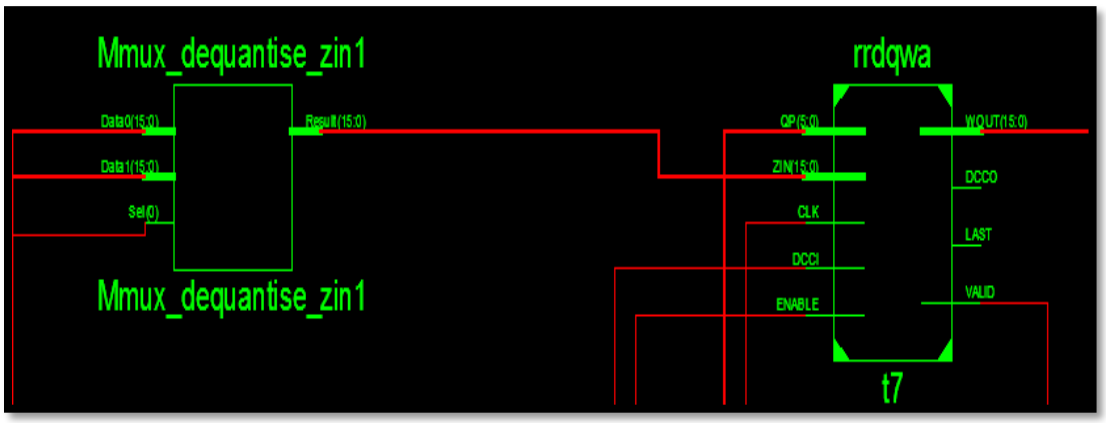


Figure (4.22) Simulation Results of de-Quantization Unit.



Figure(4.23) Architecture Design of de-Quantization Unit.

The design needs only (3) clocks of latency (latch, multiply, and scale) to gain the results. The data input is (data coefficients after core transform and quantization, and the inverse DC transform for DC coefficients). By comparing the results with Figure (4.16), the quantization input is (36c8) and the output of the de-quantization unit is (36c0), there is a difference in one bit. This different is normal because the (quantization and de-quantization units) are lossy unit operation and there is a round in their operation. The one bit different is a good results.

4.8.3 Inverse Core Transform

The Inverse Discrete Cosine Transform operates on (Z) matrix which is a block of $(N \times N)$ samples and creates a block X of same dimensions.

$X = A^T Y A$ ..... Equation(4.19)

Where $A = \begin{bmatrix} 1 & 1 & 1 & \frac{1}{2} \\ 1 & \frac{1}{2} & -1 & -1 \\ 1 & -\frac{1}{2} & -1 & 1 \\ 1 & -1 & 1 & -\frac{1}{2} \end{bmatrix}$

So the equation (4.19) becomes

$X = \begin{bmatrix} 1 & 1 & 1 & \frac{1}{2} \\ 1 & \frac{1}{2} & -1 & -1 \\ 1 & -\frac{1}{2} & -1 & 1 \\ 1 & -1 & 1 & -\frac{1}{2} \end{bmatrix} . [Y] . \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \frac{1}{2} & -\frac{1}{2} & -1 \\ 1 & -1 & -1 & 1 \\ \frac{1}{2} & -1 & 1 & -\frac{1}{2} \end{bmatrix}$ ....Equation (4.19).

The input is the de-quantized coefficients (AC and DC) and the output are sets of components that are ready to be reconstruct and to obtain the original frame. Figures (4.24) and (4.25) show the architecture design and the simulation results.

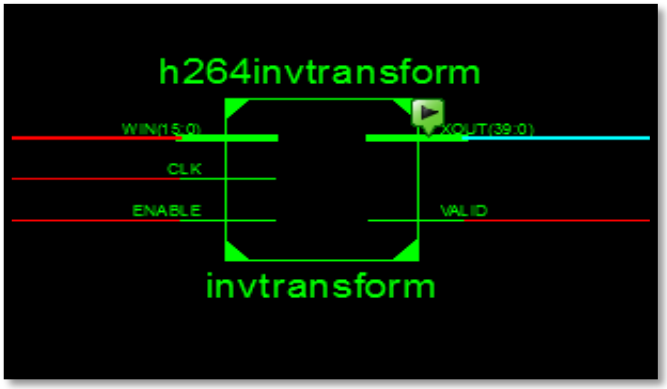


Figure (4.24) Design of the Inverse Core Transform.

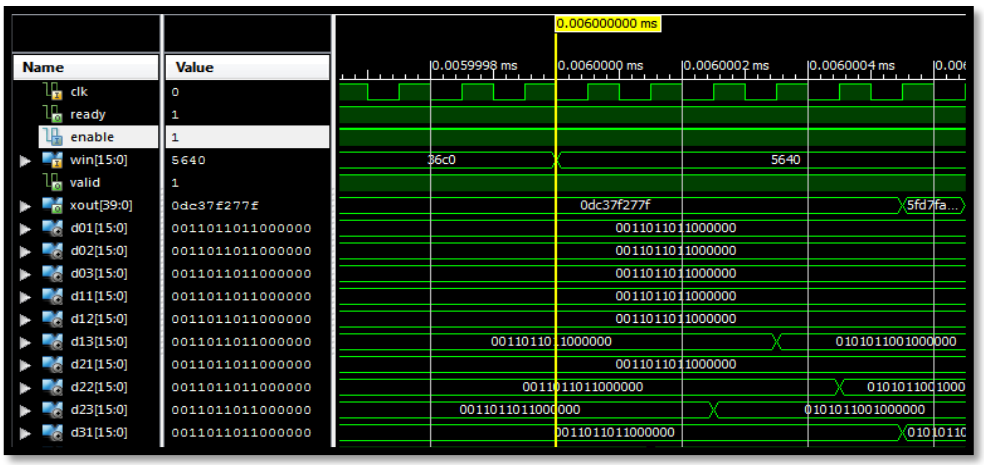
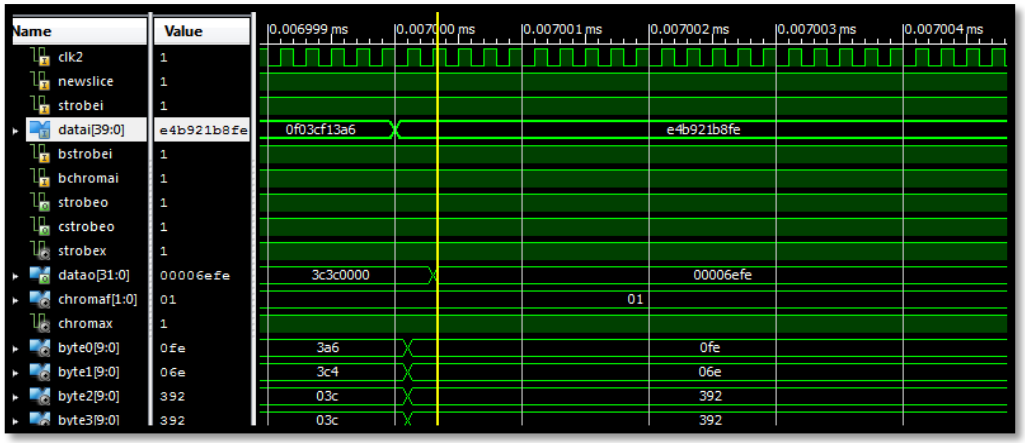


Figure (4.25) Simulation Results of Inverse Core Transform. .

4.8.4 The Reconstruction of the Frame

For each micro block, our decoder forms an identical prediction to the one created by the encoder using intra prediction from previously previous decoded samples in the current frame or same frame. The decoder adds the prediction blocks or samples to the decoded residual to reconstruct a decoded micro block which can then be a part of the original video .The data input is the inverse transformed coefficients and its values and the reconstructed values are explains in the figure (4.26) . The figure (4.27) shows the reconstruction units design.



Figure(4.26) Simulation Results of the Reconstruction Unit.

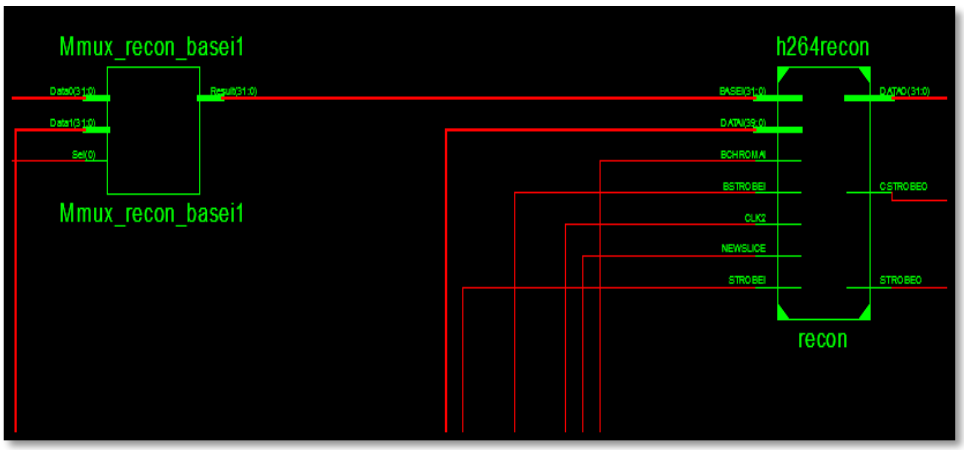


Figure (4.27) Reconstruction Design Unit

The output of the reconstruction unit is representing the Chroma and Luma values, which they are combine to represent the original value of the data in the frame.

4.9 Overall Design Results

Through our results, we found that the complete design of the encoder and decoder utilized a reasonable area space of the entire kit and the parameters set by the manufacturer of the Spartan 6 kit. Figure (4.28) shows the representing of the full exploitation of the design (encoder and decoder) from the original area space.

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	1914	18224	10%
Number of Slice LUTs	1894	9112	20%
Number of fully used LUT-FF pairs	1277	2531	50%
Number of bonded IOBs	85	232	36%
Number of Block RAM/FIFO	5	32	15%
Number of BUFG/BUFGCTRLs	3	16	18%
Number of DSP48A1s	2	32	6%

Figure (4.28) Overall Design Utilization.

Chapter Five

Conclusion and Future Works

5.1 Conclusion Work

In our work, we touched on design the model by two ways, Virtual programming use Matlab, and actual by implement model use FPGA with VHDL language. Although there is a decoder, encoder must be design for comparing the results and ensuring that our work is correct. For this, we design our encoder based on our decoder properties.

5.1.1 H.264/ AVC Encoder and Decoder Design Using Matlab.

- Through our design of the H.264 encoder and the decoder, we found that there is a decoder inside the encoder to get the reference frames for next compression frame.
- The Compression rate in (IBBB) is = 71%, this is a good result we gain and better than in (IPPP) which is equal to 50%.
- The decoded frame is mostly same to original one with little different.
- The time it takes to complete the overall encoding process is about 0.2142 sec. The time it takes to complete the overall decoding process is about 0.0994 sec.
- We can use intra prediction if there are precise details and we do not want to lose them. The results are accurate but at the expense of time to encode it and decode it.
- There is view errors and lose of data at the encoder and decoder because the quantization and de-quantization units are the only lossy units in H.264 /AVC.

5.1.2 The H.264 /AVC Encoder and Decoder Design Using (FPGA)

- Through the design, we used the Spartan 6 kit, because, it is support clock of (200MHZ) which is mean fast execution to the design.
- Inter frame prediction, means two frames or more are needs to reconstruct the reference frame or the next frames which is the motion estimation and compensation units for encoder and the inter frame reconstruction units at the decoder.
- The inter frame prediction of (IBBB) needs at least two frames to store them for reconstruct the third frame except that the first one stored at the encoder and sent to the decoder without any encoding process on them.
- This operation ends high speed or frequency and high storage to store this high amount of data at the encoder and decoder.
- Our design is fox only in intra frame and it utilized approximately 50% of the storage. If we want to design only first pattern of (IBBB), sure we need over 100% of the storage.

5.2 Future works

Through our own design and implementation in Matlab program and FPGA kit, we have found it possible to design and add specifications to this design**:**

- The inter frame prediction (motion estimation and compensation) can be implemented practically using the Vertex 5 type chip because the last has frequency up to 300 MHZ.
- We can design the (IBP) frame along patterns, and we will gain more quality but in return, we will take more time to implement and more storage to store the encoded data with high bandwidth in transmission.

- The adaptation may be taking advantage and the design becomes work like the high efficiency video coding or H.265.
- We can use another algorithm in motion estimation like (four steps search algorithm, diamond search algorithm, fast three steps algorithm and so on.).
- The networking application needs more for our design, so the entropy encoder and decoder is very important to use it. The advantage for using the entropy is to provide more compression to the data through the network and decompress the data at the decoder. Its advantage is appear if there is a noise or distortion over the network.

References

[1] Darshankumar Shah, B.E, **H.264 MOTION ESTIMATION AND MOTION COMPENSATION**, *Thesis of Master of Science in Electrical and Electronic Engineering at California State University, Sacramento, 2011*.

[2] By STEPHEN, **A Fine Grained Many-Core H.264 Video Encoder**, *Thesis of Master of Science in Electrical and Computer Engineering in the University of California, THE UY LE B.S., March, 2007*.

[3] YIM, Ka Yee, **Video Decoder for H.264/AVC Main Profile, Power Efficient Hardware Design**, *A Thesis of Master of Philosophy in Electronic Engineering, Chinese University of Hong Kong, August 2011*.

[4] Samia Sharmin Shimu, **Performance Analysis of H.264 Encoder for High-definition Video Transmission over Ultra-Wideband Communication Link**, *Thesis of Master of Science in the Department of Electrical and Computer Engineering ,University of Saskatchewan Saskatoon, May 2010.*

[5] Kermin Fleming, Chun-Chieh Lin, Jamey Hicks , **H.264 Decoder: A Case Study in Multiple Design Points,** *6th ACM/IEEE International Conference on Formal Methods and Models for Co-Design PP.165-174, 20JUNE 2008*.

[6] Michael , Anthony Joch, Faouzi Kossentini and Antti Hallapuro, **H.264/AVC Baseline Profile Decoder Complexity Analysis**, *IEEE Transactions on Circuits and Systems for Video Technology, VOL. 13, NO. 7, PP 704-716, JULY 2003*

[7] A. Ben Atitallah, H. Loukil ,Nouri, **FPGA Design for H.264/AVC Encoder**, *International Journal of Computer Science, Engineering and Applications (IJCSEA) Vol.1, No.5, PP 119-138, October 2011*

[8] Jignesh Patel, Haresh Suthar, Jagrut Gadit, Parul ,**VHDL Implementation of H.264 Video Coding Standard** , *International Journal of Reconfigurable and Embedded Systems (IJRES),Vol. 1, No. 3, pp. 95-102, November 2012*.

[9] Gary J. Sullivan, Pankaj Topiwala, and Ajay Luthra, **The H.264/AVC Advanced Video Coding Standard, Overview and Introduction to the Fidelity Range Extensions**, *SPIE Conference on Applications of Digital Image Processing XXVII Special Session on Advances in the New Emerging Standard H.264/AVC, August 2004.*

[10] LI Man Ho, **Variable Block Size Motion Estimation Hardware for Video Encoders**, *Thesis of Master of Philosophy in Computer Science and Engineering, The Chinese University of Hong Kong, November 2006.*

[11] Iain E. Richardson, **The H.264 Advanced Video Compression Standard**, *John Wiley & Sons, Ltd., Second Edition***,** *Vcodex Limited, UK, 2010.*

[12] Iain E. G. Richardson, **H.264 and MPEG-4 Video Compression Video Coding for Next-generation Multimedia**, *John Wiley & Sons Ltd, First Edition, The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England ,2003.*

[13] W. Gao and S. Ma, **Advanced Video Coding Systems**, *Springer International Publishing Switzerland, Edition Number 1***,** *Book, 2014.*

[14] **MPEG-4 Part 10 AVC (H.264) Video Encoding**, *PP.1-19, Scientific-Atlanta ,June 2005*.

[15] H**.264 Video Compression Standard, New Possibilities Within Video Surveillance**, *White Paper, Axis Communications, WWW.Axis.com*.

[16] Rahul Vanam, **Motion Estimation and Intra Frame Prediction in H.264/AVC Encoder**, *Lecture, University of Washington,*

[17] Wissal Hassen and Hamid Amiri , **Block Matching Algorithms For Motion Estimation**, *e-Learning in Industrial Electronics (ICELIE), 2013 7th IEEE International, IEEE Transactions Evolution Computation, Vienna, Austria, PP.1-4, 10-13 Nov. 2013*.

[18] Sid-Ahmed, Prof. Ahmadi and Elham Shahinfard, 2**-Dimensional Motion Estimation**, *Research Centre for Integrated Microsystems, University of Windsor,PP.1-39, December 2006*.

[19] R.Mohamed Niyas et **,Implementation of SAD Architecture for Motion Estimation in H.264/AVC**, *International Journal of Engineering and Technology (IJET), Vol 5 No 2, PP.1726-1730, Apr-May 2013*.

[20] Artur Gromek, **H.264/MPEG-4 -Advanced Video Coding**, *Warsaw University of Technology, Institute of Electronic Systems, Nowowiejska 15/19, 00-650 Warsaw, PP.1-10, 15/7/ 2016.*

[21] Sandya Basavanahalli Sheshadri, **Optimization of H.264 Baseline Decoder on Arm9tdmi Processor**, *Master of Science in Electrical Engineering, University Of Texas, December 2005*.

[22] Johotech Solutions, **H.264 Baseline Profile Video Encoder**, *Texas Instrument, PP.1-21, September 11, 2012*.

[23] Urvish Lakadiwala1, Soham Hirapara2, Raj Ramani3, Niket chaudhary**, Implementation of ALU on FPGA**, *International Research Journal of Engineering and Technology (IRJET)*, *Volume: 03 Issue: 04, PP 421-423,Apr 2016*.

[24] Umair Aslam, **H.264 CODEC Blocks Implementation on FPGA**, *Institutionen for systemteknik Department of Electrical Engineering, Linksping, PP.1- 72, Sweden 2014.*

[25] Chaminda Sampath Kannangara, **Complexity Management of H.264/AVC Video Compression,** *Open Access Institutional Repository at Robert Gordon University, October 2006*.

[26] Jigar Ratnottar1, Rutika Joshi2, Manish Shrivastav3, **Comparative Study of Motion Estimation & Motion Compensation for Video Compression**, *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)*, *Volume 1, Issue 1, PP 33-37, May June 2012*.

[27] Taheni Damak, Hassen Loukil, Ahmed, Nouri Masmoudi, **Software and Hardware Architecture of H.264/AVC Decoder,** *International Journal of Computer Applications,* Volume 59, *No.19, PP 20-27, December 2012.*

[28] CARREIRA, J.F.M., **A Two-Stage Approach for Robust HEVC Coding and Streaming,** *IEEE Transactions on Circuits and Systems for Video Technology, PP.1-14, 2016.*

[29] Michael Horowitz, Anthony Joch, Faouzi Kossentini, and Antti Hallapuro, **H.264/AVC Baseline Profile Decoder Complexity Analysis**, *IEEE Transactions on Circuits and Systems for Video Technology , Volume 13, Issue 7, PP.704-716, JULY 2003*.

[30] LI, X., EDIRISINGHE, E.A.and BEZ, H.E., **Shape Adaptive Integer Transform for Coding Arbitrarily Shaped Objects in H.264/AVC,** *Visual Communications and Image Processing, 2006.*

[31] Fernando Pereira, Thomas Stockhammer, and Thomas Wedi, **Video Coding with H.264/AVC, Tools, Performance, and Complexity,** *IEEE Circuits and Systems Magazine , Volume 4, Issue 1, PP. 7-28, 2004.*

[32] Senay Amanuel Negusse, **Improving Intra Pixel Prediction for H.264 Video Coding,** *Thesis of Master of Science in Electrical Engineering, Multimedia Technology, Ericsson research, Blekinge Institute of Technology (School of Engineering, Department of Signal Processing), Blekinge Institute of Technology, May 2008.*

[33] A. Elyousfi, A. Tamtaoui, and E. Bouyakhf , **A New Fast Intra Prediction Mode Decision Algorithm for H.264/AVC Encoders,** *World Academy of Science, Engineering and Technology International Journal of Electrical and Computer Engineering, Vol 1, No 3, PP 89-95, 2007.*

[34] Priyanka P James1, Chirappanath B Albert2, Inbanila.K3, **Analysis of Integer Transformation and Quantization Blocks using H.264 Standard and the Conventional DCT Techniques**, *International Journal of Computer Science and Mobile Computing, IJCSMC, Vol. 3, Issue. 3, PP.873 – 878, March 2014.*

[35] S. Valarmathi, R. Vani and Dr. M. Sangeetha**, Hardware Software Co-Simulation of Motion Estimation on H.264 Encoder,** *Natarajan Meghanathan, et al. (Eds), ITCS, SIP, JSE-2012, CS & IT 04, PP. 263–271, 2012.*

[36] Bharathi S.H. and K. Nagabhushana Raju and S. Ramachandran, **Implementation of Horizontal and Vertical Intra prediction Modes for H.264 Encoder,** *International Journal of Electronics and Communication Engineering, PP.105-114, Number 1 (2011).*

[37] Gary J. Sullivan, Pankaj Topiwala, and Ajay Luthra, **The H.264/AVC Advanced Video Coding Standard, Overview and Introduction to the Fidelity Range Extensions,** *SPIE Conference on Applications of Digital Image Processing XXVII***,** *Special Session on Advances in the New Emerging Standard: H.264/AVC, PP 1-22, August, 2004.*

[38] Sara Hamdy, Mostafa E. A. Ibrahim, Abdelhalim Zekry, **VHDL Realization of Efficient H.264 Intra Prediction Scheme Based on Best Prediction Matrix Mode**, *International Journal of Computer Applications, Volume 77 - No. 13, PP 1-7, September 2013***.**

[39] Jin Li, **Advances on Video Coding Algorithms for Next Generation Mobile Applications**, *The Thesis of Doctor of Science in Technology***,** *Tampere University of Technology, PP. 1-70,5th of August 2011*.

[40] Bojun Meng, Oscar C. Au*, Chi-Wah Wong, Hong-Kwai Lam, **Efficient Intra-Prediction Mode Selection for 4x4 Blocks in H.264**, *Multimedia and Expo, 2003 ICME International Conference on, IEEE, PP 521-524,2003*.

APPENDIX A

Matlab Code: H.264/ AVC Encoder and Decoder

1. Encoder code in MATLAB

```
function myencoder
% function to encode the video frames
fprintf('\nencoder Project\n')
nf =28;
fprintf('%i frames\n',nf)
nb = getmov(nf);
tic
mpeg = encmov(nb);
fprintf('Encode time: %s\n',sec2timestr(toc))
  tic
mov2 = decmpeg(mpeg);
fprintf('Decode time: %s\n',sec2timestr(toc))
save ('N.mat','nb');
save ('D.mat','mpeg');
save('S1.mat','mov2');
save lastmov nb mpeg mov2
X=nb(:,:,:,:);
Y=mov2(:,:,:,:);
biterr=Biter(X,Y);
save lastmov3 biterr
%% Reading the frame information
function movdata = getmov(nf)
load nbb
if nf == 0
nf = length(nb);
```

```
end
movdata = repmat(uint8(0),[size(nb(1).cdata), nf]);
for i = 1:nf
movdata(:,:,:,i) = nb(i).cdata;
end
%%
function mpeg = encmov(nb)
fpat = 'IBBB';
k = 0;
pf = [];
progressbar
for  i = 1:size(nb,4)
f = rgb2ycbcr(f); % convert from RGB to YUV
k = k + 1;
if k > length(fpat)
 k = 1;
end
ftype = fpat(k);
[mpeg{i},pf] = encframe(f,ftype,pf);
progressbar(i/(2*size(nb,4)))
end
```

%% Encode the frame

```
function [mpeg,df] = encframe(f,ftype,pf)
 [M,N,i] = size(f);
mbsize = [M, N] / 16;
f=f(1:floor(mbsize(1))*16,1:floor(mbsize(2))*16,:);
 [M,N,i] = size(f);
  mbsize = [M, N] / 16;
mpeg = struct('type',[],'mvx',[],'mvy',[],'scale',[],'coef',[]);
```

```
mpeg(mbsize(1),mbsize(2)).type = [];
 pfy = pf(:,:,1);
df = zeros(size(f));
for m = 1:mbsize(1)
for n = 1:mbsize(2)
x = 16*(m-1)+1 : 16*(m-1)+16;
y = 16*(n-1)+1 : 16*(n-1)+16;
[mpeg(m,n),df(x,y,:)] = encmacroblock(f(x,y,:),ftype,pf,pfy,x,y);
end
end
```

%% Encode the micro block(luma and chroma)

```
function b = getblocks(mb)
  b = zeros([8, 8, 6]);
 b(:,:,1) = mb( 1:8,  1:8,  1);
b(:,:,2) = mb( 1:8,  9:16, 1);
b(:,:,3) = mb( 9:16, 1:8,  1);
b(:,:,4) = mb( 9:16, 9:16, 1);
 b(:,:,5) = 0.25 * ( mb(1:2:15,1:2:15, 2) + mb(1:2:15,2:2:16, 2) ...
              + mb(2:2:16,1:2:15, 2) + mb(2:2:16,2:2:16, 2) );
b(:,:,6) = 0.25 * ( mb(1:2:15,1:2:15, 3) + mb(1:2:15,2:2:16, 3) ...
              + mb(2:2:16,1:2:15, 3) + mb(2:2:16,2:2:16, 3) );
```

%% Transformation and quantisation

```
function [mpeg,dmb] = encmacroblock(mb,ftype,pf,pfy,x,y)
persistent q1 q2
if isempty(q1)
q1 = qintra;
q2 = qinter;
end
 % Quality scaling
```

```
% for inter
  scale = 1;
%for intra
%       scale = 32;
% Init mpeg struct
%    mpeg.type = 'I';
mpeg.type = 'IBBB';
mpeg.mvx = 0;
mpeg.mvy = 0;
if ftype=='B'
mpeg.type = 'B';
[mpeg,emb] = getmotionvec(mpeg,mb,pf,pfy,x,y);
mb = emb;
q = q2;
else
q = q1;
 end
b = getblocks(mb);
for i = 6:-1:1
mpeg.scale(i) = scale;
coef = dct2(b(:,:,i));
mpeg.coef(:,:,i) = round( 8 * coef ./ (scale * q) );
end
toc
dmb = decmacroblock(mpeg,pf,x,y);
```

%% Motion estimation and compensation

```
function [mpeg,emb] = getmotionvec(mpeg,mb,pf,pfy,x,y)
mby = mb(:,:,1);
[M,N] = size(pfy);
```

```
% Logarithmic search
step = 8; %
dx = [0 1 1 0 -1 -1 -1  0  1 ];
dy = [0 0 1 1  1  0 -1 -1 -1 ];
mvx = 0;
mvy = 0;
while step >= 1
minsad = inf;
for i = 1:length(dx)
 tx = x + mvx + dx(i)*step;
if (tx(1) < 1) || (M < tx(end))
continue
end
ty = y + mvy + dy(i)*step;
 if (ty(1) < 1) || (N < ty(end))
continue
end
sad = sum(sum(abs(mby-pfy(tx,ty))));
 if sad < minsad
ii = i;
minsad = sad;
end
if sad< minsad
ii = i;
minsad = sad;
 end
 end
mvx = mvx + dx(ii)*step;
mvy = mvy + dy(ii)*step;
```

```
step = step / 2;
end
mpeg.mvx = mvx;
mpeg.mvy = mvy;
emb = (mb-pf(x+mvx,y+mvy,:));
function nb = decmpeg(mpeg)
movsize = size(mpeg{1});
nb = repmat(uint8(0),[16*movsize(1:2), 3, length(mpeg)]);
pf = [];
for i = 1:length(mpeg)
f = decframe(mpeg{i},pf);
pf = f;
f= ycbcr2rgb(f);
f = min( max(f,0), 255);
nb(:,:,:,i) = uint8(f);
progressbar((i+length(mpeg))/(2*length(mpeg)))
end
%%
function fr = decframe(mpeg,pf)
mbsize = size(mpeg);
M = 16 * mbsize(1);
N = 16 * mbsize(2);
fr = zeros(M,N,3);
for m = 1:mbsize(1)
for n = 1:mbsize(2)
x = 16*(m-1)+1 : 16*(m-1)+16;
y = 16*(n-1)+1 : 16*(n-1)+16;
fr(x,y,:) = decmacroblock(mpeg(m,n),pf,x,y);
 end
```

```
end
%%
function mb = putblocks(b)
 mb = zeros([16, 16, 3]);
mb( 1:8,  1:8,  1) = b(:,:,1);
mb( 1:8,  9:16, 1) = b(:,:,2);
mb( 9:16, 1:8,  1) = b(:,:,3);
mb( 9:16, 9:16, 1) = b(:,:,4);
z = [1 1; 1 1];
mb(:,:,2) = kron(b(:,:,5),z);
mb(:,:,3) = kron(b(:,:,6),z);
%%
function mb = decmacroblock(mpeg,pf,x,y)
persistent q1 q2
if isempty(q1)
q1 = qintra;
q2 = qinter;
end
mb = zeros(16,16,3);
if mpeg.type == 'B'
mb = pf(x+mpeg.mvx,y+mpeg.mvy,:);
q = q2;
else
q = q1;
end
 for i = 6:-1:1
coef = mpeg.coef(:,:,i) .* (mpeg.scale(i) * q) / 8;
b(:,:,i) = idct2(coef);
end
```

```
mb = mb + putblocks(b);
%%
function q = qintra
q = 16;
%%
function q = qinter
q=[16 11 10 16 24 40 51 61;
    12 12 14 19 26 58 60 55;
    14 13 16 24 40 57 69 56;
     14 17 22 29 51 87 80 62;
     18 22 37 56 68 109 103 77;
     24 35 55 64 81 104 113 92;
     49 64 78 87 103 121 120 101;
     72 92 95 98 112 100 103 99  ];
%%
function y = dct2(x)
persistent d
if isempty(d)
d = dctmtx(8);
end
y = d * x * d';
 y = dct(x); % Columns
y = dct(y')'; % Rows
%%
function y = idct2(x)
persistent d
if isempty(d)
d = dctmtx(8);
end
```

```
y = d' * x * d;
y = idct(x); % Columns
y = idct(y')'; % Rows
% toc
```

2.Decoder code in MATLAB

```
function mydecoder % decoder the video frames
fprintf('\nDecoder Project\n')
nf =28 ;
fprintf('%i frames\n',nf)
fprintf('Decode time: %s\n',sec2timestr(toc))
tic
load N;
load D ;
mov2 = decmpeg(mpeg);
fprintf('Decode time: %s\n',sec2timestr(toc))
save('S1.mat','mov2');
save lastmov nb mpeg mov2
X=nb(:,:,:,3);
Y=mov2(:,:,:,3);
biterr=Biter(X,Y);
save lastmov4 biterr
%% for configrate the frame and reading the frame proparties
function nb = decmpeg(mpeg)
movsize = size(mpeg{1});
nb = repmat(uint8(0),[16*movsize(1:2), 3, length(mpeg)]);
pf = [];
for i = 1:length(mpeg)
f = decframe(mpeg{i},pf);
pf = f;
```

```
f= ycbcr2rgb(f);
f = min( max(f,0), 255);
nb(:,:,:,i) = uint8(f);
end
%%
function fr = decframe(mpeg,pf)
mbsize = size(mpeg);
M = 16 * mbsize(1);
N = 16 * mbsize(2);
fr = zeros(M,N,3);
for m = 1:mbsize(1)
for n = 1:mbsize(2)
x = 16*(m-1)+1 : 16*(m-1)+16;
y = 16*(n-1)+1 : 16*(n-1)+16;
fr(x,y,:) = decmacroblock(mpeg(m,n),pf,x,y);
end
end
%%
function mb = putblocks(b)
mb = zeros([16, 16, 3]);
mb( 1:8,  1:8,  1) = b(:,:,1);
mb( 1:8,  9:16, 1) = b(:,:,2);
mb( 9:16, 1:8,  1) = b(:,:,3);
mb( 9:16, 9:16, 1) = b(:,:,4);
z = [1 1; 1 1];
mb(:,:,2) = kron(b(:,:,5),z);
mb(:,:,3) = kron(b(:,:,6),z);

%% for reconstruct block
```

```
function mb = decmacroblock(mpeg,pf,x,y)
persistent q1 q2
if isempty(q1)
q1 = qintra;
q2 = qinter;
end
mb = zeros(16,16,3);
 if mpeg.type == 'B'
mb = pf(x+mpeg.mvx,y+mpeg.mvy,:);
q = q2;
else
q = q1;
end
for i = 6:-1:1
coef = mpeg.coef(:,:,i) .* (mpeg.scale(i) * q) / 8;
b(:,:,i) = idct2(coef);
end
mb = mb + putblocks(b);
%%
function q = qintra
q = 16;
%%
function q = qinter
q=[16 11 10 16 24 40 51 61;
    12 12 14 19 26 58 60 55;
    14 13 16 24 40 57 69 56;
     14 17 22 29 51 87 80 62;
     18 22 37 56 68 109 103 77;
     24 35 55 64 81 104 113 92;
```

```
    49 64 78 87 103 121 120 101;
    72 92 95 98 112 100 103 99  ];
%% for inverse dct transform
function y = idct2(x)
persistent d
if isempty(d)
d = dctmtx(8);
end
y = d' * x * d;
y = idct(x); % Columns
y = idct(y')'; % Rows
```

3.The Bit Error Rate Code in MATLAB

```
function [out]=Biter(X,Y)
[m, n, l]=size(X);
errate=0;
for i=1:m
for j=1:n
if(X(i,j,l)==Y(i,j,l))
errate=errate+0;
else
errate=errate+1;
end
end
end
err=errate;
out=(errate)/numel(X);
```

APPENDIX B

FPGA Design of H.264 /AVC encoder and decoder

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use ieee.numeric_std.ALL;
use std.textio.all;
use work.h264.all;
use work.misc.all;
entity h264encdec is
        generic (
                IMGWIDTH : integer := 352;
                IMGHEIGHT : integer := 288;
                 j: INTEGER :=8);
        port (
        signal rst : in std_logic:= '0';
          DIV_CLK : in std_logic;
                signal CLK : inout std_logic;                    --clock
                signal CLK2 : inout std_logic;
signal  Dataout : out std_logic_vector(31 downto 0);
                -- controls
                signal NEWSLICE : in std_logic;
                 signal NEWLINE : in std_logic;
                signal QP : in std_logic_vector(5 downto 0);
                signal xbuffer_DONE : out std_logic := '0';
               signal intra4x4_READYI : out std_logic := '0';
```

```
                signal intra4x4_STROBEI : in std_logic := '0';
               signal intra8x8cc_readyi : out std_logic := '0';
                signal intra8x8cc_strobei : in std_logic := '0';
                signal intra8x8cc_datai : in std_logic_vector(31 downto 0)
:= (others => '0');
                 reg_out :out std_logic_vector(3 downto 0)    );
end h264encdec;
architecture hw of h264encdec is
component rr is
        port(clk:in std_logic;
        address:in std_logic_vector(10 downto 0);
        dout: out std_logic_vector( 7 downto 0) );
        end component;
component rram2 is
        port(clk:in std_logic;
        address:in std_logic_vector(10 downto 0);
        dout: out std_logic_vector( 7 downto 0) );
        end component;
component rram3 is
        port(clk:in std_logic;
        address:in std_logic_vector(10 downto 0);
        dout: out std_logic_vector( 7 downto 0) );
        end component;
component rram4 is
        port(clk:in std_logic;
        address:in std_logic_vector(10 downto 0);
        dout: out std_logic_vector( 7 downto 0) );
        end component;
signal PAKET1:std_logic_vector(31 downto 0);
```

```
signal Delay: integer range 0 to j:= 0;
signal cd : std_logic_vector(31 downto 0) := (others => '0');
signal intra4x4_DATAI :std_logic_vector(31 downto 0) := (others => '0');
signal address1 :std_logic_vector (10 downto 0):="00000000000";
signal address2 :std_logic_vector (10 downto 0):="00000000000";
 signal address3 :std_logic_vector (10 downto 0):="00000000000";
signal address4 :std_logic_vector (10 downto 0):="00000000000";
signal dout1: std_logic_vector(7 downto 0);
signal dout2: std_logic_vector(7 downto 0);
signal dout3: std_logic_vector(7 downto 0);
signal dout4: std_logic_vector(7 downto 0);
signal n:integer :=0;
signal intra4x4_TOPI : std_logic_vector(31 downto 0) := (others => '0');
signal intra4x4_TOPMI : std_logic_vector(3 downto 0) := (others => '0');
signal intra4x4_STROBEO : std_logic := '0';
signal intra4x4_READYO : std_logic := '0';
signal intra4x4_DATAO : std_logic_vector(35 downto 0) := (others =>
'0');
signal intra4x4_BASEO : std_logic_vector(31 downto 0) := (others =>
'0');
signal intra4x4_MSTROBEO : std_logic := '0';
signal intra4x4_MODEO : std_logic_vector(3 downto 0) := (others =>
'0');
signal intra4x4_PMODEO : std_logic := '0'; --prediction type same
signal intra4x4_RMODEO : std_logic_vector(2 downto 0) := (others =>
'0');
signal intra4x4_XXO : std_logic_vector(1 downto 0) := (others => '0');
signal intra4x4_XXINC : std_logic := '0';
signal intra4x4_CHREADY : std_logic := '0';
```

```
signal intra8x8cc_TOPI : std_logic_vector(31 downto 0) := (others =>
'0');
signal intra8x8cc_STROBEO : std_logic := '0';
signal intra8x8cc_READYO : std_logic := '0';
signal intra8x8cc_DATAO : std_logic_vector(35 downto 0) := (others =>
'0');
signal intra8x8cc_BASEO : std_logic_vector(31 downto 0) := (others =>
'0');
signal intra8x8cc_dcstrobeo : std_logic := '0';
signal intra8x8cc_dcdatao : std_logic_vector(15 downto 0) := (others =>
'0');
signal intra8x8cc_CMODEO : std_logic_vector(1 downto 0) := (others
=> '0');
signal intra8x8cc_XXO : std_logic_vector(1 downto 0) := (others => '0');
signal intra8x8cc_XXC : std_logic := '0';
signal intra8x8cc_XXINC : std_logic := '0';
signal coretransform_READY : std_logic := '0';
signal coretransform_ENABLE : std_logic := '0';
signal coretransform_XXIN : std_logic_vector(35 downto 0) := (others
=> '0');
signal coretransform_valid : std_logic := '0';
signal coretransform_ynout : std_logic_vector(13 downto 0);
signal dctransform_VALID : std_logic := '0';
signal dctransform_yyout : std_logic_vector(15 downto 0);
signal dctransform_readyo : std_logic := '0';
signal quantise_ENABLE : std_logic := '0';
signal quantise_YNIN : std_logic_vector(15 downto 0);
signal quantise_valid : std_logic := '0';
signal quantise_zout : std_logic_vector(11 downto 0);
```

```
signal quantise_dcco : std_logic := '0';
signal dequantise_enable : std_logic := '0';
signal dequantise_zin : std_logic_vector(15 downto 0);
signal dequantise_last : std_logic := '0';
signal dequantise_valid : std_logic := '0';
signal dequantise_dcco : std_logic := '0';
signal dequantise_wout : std_logic_vector(15 downto 0);
signal invdctransform_enable : std_logic := '0';
signal invdctransform_zin : std_logic_vector(15 downto 0);
signal invdctransform_valid : std_logic := '0';
signal invdctransform_yyout : std_logic_vector(15 downto 0);
signal invdctransform_ready : std_logic := '0';
signal invtransform_valid : std_logic := '0';
signal invtransform_xout : std_logic_vector(39 downto 0);
signal recon_BSTROBEI : std_logic := '0';
signal recon_basei : std_logic_vector(31 downto 0) := (others => '0');
signal recon_FBSTROBE : std_logic := '0';
signal recon_FBCSTROBE : std_logic := '0';
signal recon_FEEDB : std_logic_vector(31 downto 0) := (others => '0');
signal xbuffer_READYI : std_logic := '0';
signal xbuffer_CCIN : std_logic := '0';
COMPONENT DIV
        PORT(
                clk : IN std_logic;
                rst : IN std_logic;
                clk1_100M : INOUT std_logic;
                clk2_200M : INOUT std_logic
                );
END COMPONENT;
```

```
        COMPONENT REG
        PORT(
                CLK : IN std_logic;
                rst : IN std_logic;
                Data_IN : INOUT std_logic_vector(31 downto 0);
                Data_out : OUT std_logic_vector(31 downto 0)
                );
        END COMPONENT;
begin
p1:rr
 port map (clk=>clk,
address=>address1,
dout=>dout1);
 p2:rram2
  port map (clk=>clk,
address=>address2,
dout=>dout2 );
p3:rram3
port map (clk=>clk,
address=>address3,
dout=>dout3);
 p4:rram4
port map (clk=>clk,
address=>address4,
dout=>dout4);
process1 :process(clk)
begin
if (CLK'event and CLK ='1')
```

```
then
if (n>0)then
intra4x4_DATAI(7 downto 0)<= dout1;
intra4x4_DATAI(15 downto 8)<= dout2;
intra4x4_DATAI(23 downto 16)<= dout3;
intra4x4_DATAI(31 downto 24)<= dout4;
address1 <= address1 + "00000000001";
address2 <= address2 + "00000000001";
address3 <= address3 + "00000000001";
address4 <= address4 + "00000000001";
else
n <= n+1;
end if;
end if;
end process;
reg_sav : reg
PORT MAP(
clk =>clk2 ,
rst =>rst ,
Data_IN =>cd,
 Data_out =>Dataout
        );
process(CLK2)
begin
if rst='1' then
PAKET1<=(others=>'0');
Delay<=j-8;
elsif(CLK'event and CLK='1') then
Delay<=Delay+1;
```

```
PAKET1<=cd;
PAKET1<=PAKET1(27 downto 0)&PAKET1(31 downto 28);
reg_out<=cd(31 downto 28);
end if;
end process;
        intra4x4 : h264intra4x4
        port map (
CLK => clk2,
NEWSLICE => NEWSLICE,
NEWLINE => NEWLINE,
STROBEI => intra4x4_strobei,
DATAI => intra4x4_datai,
READYI => intra4x4_readyi,
TOPI => intra4x4_topi,
TOPMI => intra4x4_topmi,
XXO => intra4x4_xxo,
XXINC => intra4x4_xxinc,
FEEDBI => recon_FEEDB(31 downto 24),
FBSTROBE => recon_FBSTROBE,
STROBEO => intra4x4_strobeo,
DATAO => intra4x4_datao,
BASEO => intra4x4_baseo,
READYO => intra4x4_readyo,
MSTROBEO => intra4x4_mstrobeo,
MODEO => intra4x4_MODEO,
PMODEO => intra4x4_PMODEO,
RMODEO => intra4x4_RMODEO,
CHREADY => intra4x4_CHREADY
        );
```

```
intra4x4_readyo <= coretransform_ready and xbuffer_readyi
intra4x4_TOPI <= toppix(conv_integer(mbx & intra4x4_XXO));
intra4x4_TOPMI <= topmode(conv_integer(mbx & intra4x4_XXO));
        intra8x8cc : h264intra8x8cc
        port map (
CLK2 => clk2,
NEWSLICE => NEWSLICE,
NEWLINE => NEWLINE,
STROBEI => intra8x8cc_strobei,
DATAI => intra8x8cc_datai,
READYI => intra8x8cc_readyi,
TOPI  => intra8x8cc_topi,
XXO => intra8x8cc_xxo,
XXC => intra8x8cc_xxc,
XXINC => intra8x8cc_xxinc,
FEEDBI => recon_FEEDB(31 downto 24),
FBSTROBE => recon_FBCSTROBE,
STROBEO => intra8x8cc_strobeo,
DATAO => intra8x8cc_datao,
BASEO => intra8x8cc_baseo,
READYO => intra4x4_CHREADY,
DCSTROBEO => intra8x8cc_dcstrobeo,
DCDATAO => intra8x8cc_dcdatao,
CMODEO => intra8x8cc_cmodeo
        );
intra8x8cc_TOPI <= toppixcc(conv_integer(mbxcc &
intra8x8cc_XXO));

coretransform : h264coretransform
```

```
        port map (
CLK => clk2,
READY => coretransform_ready,
ENABLE => coretransform_enable,
XXIN => coretransform_xxin,
VALID => coretransform_valid,
YNOUT => coretransform_ynout
        );
coretransform_enable <= intra4x4_strobeo or intra8x8cc_strobeo;
coretransform_xxin <= intra4x4_datao when intra4x4_strobeo='1' else
intra8x8cc_datao;
recon_bstrobei <= intra4x4_strobeo or intra8x8cc_strobeo;
recon_basei <= intra4x4_baseo when intra4x4_strobeo='1' else
intra8x8cc_baseo;
dctransform : h264dctransform
generic map ( TOGETHER => 1 )
port map (
CLK2 => clk2,
RESET => newslice,
ENABLE => intra8x8cc_dcstrobeo,
XXIN => intra8x8cc_dcdatao,
VALID => dctransform_valid,
YYOUT => dctransform_yyout,
READYO => dctransform_readyo
        );
dctransform_readyo <= intra4x4_CHREADY and not
coretransform_valid;
        quantise : h264quantise
        port map (
```

```
CLK => clk2,
ENABLE => quantise_ENABLE,
QP => qp,
DCCI => dctransform_VALID,
YNIN => quantise_YNIN,
ZOUT => quantise_zout,
VALID => quantise_valid
        );
quantise_YNIN <= sxt(coretransform_ynout,16) when
coretransform_valid='1' else dctransform_yyout;
quantise_ENABLE <= coretransform_valid or dctransform_VALID;
invdctransform : h264dctransform
        port map (
CLK2 => clk2,
RESET => newslice,
ENABLE => invdctransform_enable,
XXIN => invdctransform_zin,
VALID => invdctransform_valid,
YYOUT => invdctransform_yyout,
READYO => invdctransform_ready
        );
invdctransform_enable <= quantise_valid and quantise_dcco;
invdctransform_ready <= dequantise_last and xbuffer_CCIN;
invdctransform_zin <= sxt(quantise_zout,16);
dequantise : h264dequantise
generic map ( LASTADVANCE => 2 )
        port map (
CLK => clk2,
ENABLE => dequantise_enable,
```

```
QP => qp,
ZIN => dequantise_zin,
DCCI => invdctransform_valid,
LAST => dequantise_last,
WOUT => dequantise_wout,
VALID => dequantise_valid
        );
dequantise_enable <= quantise_valid and not quantise_dcco;
dequantise_zin <= sxt(quantise_zout,16) when invdctransform_valid='0'
else invdctransform_yyout;
invtransform : h264invtransform
        port map (
CLK => clk2,
ENABLE => dequantise_valid,
WIN => dequantise_wout,
VALID => invtransform_valid,
XOUT => invtransform_xout
        );
recon : h264recon
        port map (
CLK2 => clk2,
NEWSLICE => NEWSLICE,
STROBEI => invtransform_valid,
DATAI => invtransform_xout,
BSTROBEI => recon_bstrobei,
BCHROMAI => intra8x8cc_strobeo,
BASEI => recon_basei,
STROBEO => recon_FBSTROBE,
CSTROBEO => recon_FBCSTROBE,
```

```
DATAO => recon_FEEDB
        );
Inst_DIV: DIV PORT MAP(
clk =>DIV_CLK,
rst =>rst,
clk1_100M =>clk,
clk2_200M =>clk2);
end hw;
```

الخلاصة

يعتبر H.264 /AV C بمثابة طريقة تقنية جيدة لتشفير وفك تشفير معظم أنواع تنسيق الفيديو ويوفر نتائج جيدة للغاية. تأتي قوتة و جودة اداء المشفر وفاتح التشفير h.264 من خوارزمية العمل وكيفية تشفيره لإطارات الفيديو ومحاولته للحصول على الناتج المثالي لكسب الفيديو الأصلي. يتضمن عملنا تطبيق عملية التشفير وفك التشفير للمعيار باستخدام برنامج MATLAB (2013Ra). و يركز العمل في التنبؤات بين الأطر باستخدام نمط إطار (IBBB)مع (٢٨) إطارًا ومشفرة في وقت قياسي جدً والذي يساوي (٠,٢١٤٢) ثانية ، ووقت فك التشفير يساوي (٠,٠٩٩٤) ثانية. كان الفيديو الذي خضع لعملية ترميز وفك التشفيراسمه (Xylophone)بحجم (٢٤٠ × ٣٢٠) . بهذه الطريقة وصلنا إلى معدل الضغط = ٧١٪ ، وهذه قيمة جيدة جدًا.

الجزء الثاني من هذه الأطروحة يتركز على التطبيق عملي للمعيار واستخدامنا لمصفوفة البوابات المنطقية المبرمجة حقليا FPGA. استخدمنا نمط التنبؤ نوع intra في هذا الجزء من العمل بسبب المحددات التي هي ضمن تصميم مصفوفة البوابات المنطقية المبرمجة حقليا.وقد تمكنا من تحقيق الشروط من خلال العمل في نطاق التردد المناسب للFPGAوضمن المساحة السيليكونية له وهذا واضح من النتائج العملية التي حصلنا عليها.

إقرار المشرف

أشهد بأن هذه الرسالة الموسومة (تمثيل فاتح التشفير H.264/AVC على مصفوفة البوابات المنطقية المبرمجة حقليا) والمعدة من قبل الطالبة (**رسل نبيل احمد**) تحت اشرافي في قسم / / جامعة الموصل، كجزء من متطلبات نيل شهادة الماجستير .

التوقيع:

الاسم:

التاريخ:

إقرار المقوم اللغوي

اشهد بأنه قد تمت مراجعة هذه الرسالة من الناحية اللغوية وتصحيح ماورد فيها من أخطاء لغوية وتعبيرية وبذلك أصبحت الرسالة مؤهلة للمناقشة بقدر تعلق الأمر بسلامة الأسلوب أو صحة التعبير.

التوقيع:

الاسم:

التاريخ:

إقرار رئيس

بناءً على التوصيات المقدمة من قبل المشرف والمقوم اللغوي أرشح هذه الرسالة للمناقشة.

التوقيع:

الاسم:

التاريخ:

إقرار رئيس لجنة الدراسات العليا

بناءً على التوصيات المقدمة من قبل المشرف والمقوم اللغوي أرشح هذه الرسالة للمناقشة.

التوقيع:

الاسم:

التاريخ:

جامعة الموصل

كلية هندسة لالكترونيات

تمثيل فاتح التشفير H.264/AVC على مصفوفة البوابات المنطقية المبرمجة حقليا

رسل نبيل احمد

رسالة ماجستير

هندسة الحاسوب والمعلوماتية

بإشراف

الأستاذ الدكتور

محمد حازم الجماس

١٤٣٩ هـ ٢٠١٨م