

1-1-2008

Pipelined implementation of Jpeg image compression using Hdl

Arun Kumar Reddy Toomu
University of Nevada, Las Vegas

Follow this and additional works at: <https://digitalscholarship.unlv.edu/rtds>

Repository Citation

Toomu, Arun Kumar Reddy, "Pipelined implementation of Jpeg image compression using Hdl" (2008).
UNLV Retrospective Theses & Dissertations. 2387.
<http://dx.doi.org/10.25669/swja-6ku1>

This Thesis is protected by copyright and/or related rights. It has been brought to you by Digital Scholarship@UNLV with permission from the rights-holder(s). You are free to use this Thesis in any way that is permitted by the copyright and related rights legislation that applies to your use. For other uses you need to obtain permission from the rights-holder(s) directly, unless additional rights are indicated by a Creative Commons license in the record and/or on the work itself.

This Thesis has been accepted for inclusion in UNLV Retrospective Theses & Dissertations by an authorized administrator of Digital Scholarship@UNLV. For more information, please contact digitalscholarship@unlv.edu.

PIPELINED IMPLEMENTATION OF JPEG IMAGE COMPRESSION
USING HDL

by

Arun Kumar Reddy Toomu

Bachelor of Technology
J.N.T University, Hyderabad, India
2006

A thesis submitted in partial fulfillment
of the requirement for the

Master of Science Degree in Electrical Engineering
Department of Electrical and Computer Engineering
Howard R. Hughes College of Engineering

Graduate College
University of Nevada, Las Vegas
August 2008

UMI Number: 1460544

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI[®]

UMI Microform 1460544

Copyright 2009 by ProQuest LLC.

All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

ProQuest LLC
789 E. Eisenhower Parkway
PO Box 1346
Ann Arbor, MI 48106-1346



Thesis Approval
The Graduate College
University of Nevada, Las Vegas

JULY 18, 2008

The Thesis prepared by

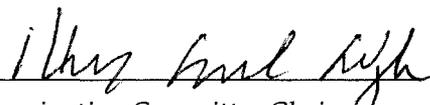
ARUN TOOMU

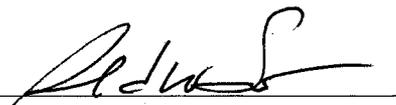
Entitled

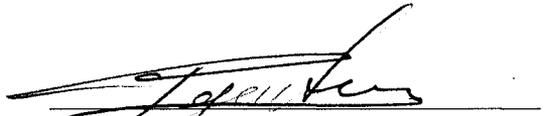
PIPELINED IMPLEMENTATION OF JPEG COMPRESSION USING HDL

is approved in partial fulfillment of the requirements for the degree of

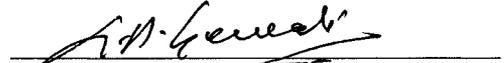
MASTER OF SCIENCE IN ELECTRICAL ENGINEERING


Examination Committee Chair


Dean of the Graduate College


Examination Committee Member


Examination Committee Member


Graduate College Faculty Representative

ABSTRACT

Pipelined Implementation of JPEG Image Compression using VHDL

by

Arun Kumar Reddy Toomu

Dr. Henry Selvaraj, Examination Committee Chair
Professor of Electrical and Computer Engineering
University of Nevada, Las Vegas

This thesis presents the architecture and design of a JPEG compressor for color images using VHDL. The system consists of major parts like color space converter, down sampler, 2-D DCT module, quantization, zigzag scanning and entropy coding. The color space conversion transforms the RGB colors to YCbCr color coding. The down sampling operation reduces the sampling rate of the color information (Cb and Cr). The 2-D DCT transform the pixel data from the spatial domain to the frequency domain. The quantization operation eliminates the high frequency components and the small amplitude coefficients of the co-sine expansion. Finally, the entropy coding uses run-length encoding (RLE), Huffman, variable length coding (VLC) and differential coding to decrease the number of bits used to represent the image. The JPEG compression is a lossy compression, since downsampling and quantization operations are irreversible. But the losses can be controlled in order to keep the necessary image quality.

Architectures for these parts were designed and described in VHDL. The results were observed using Active-HDL simulator and the code being synthesized using xilinx ise for vertex-4 FPGA. This pipelined architecture has a minimum latency of 187 clock cycles

TABLE OF CONTENTS

ABSTRACT.....	iii
LIST OF FIGURES	vi
ACKNOWLEDGEMENTS	vii
CHAPTER1 INTODUCTION.....	1
1.1 Thesis Outline	3
1.2 Background.....	3
CHAPTER2 THEORITICAL BACKGROUND.....	4
2.1 Data Compression Basics	4
2.2 Data Compression techniques.....	5
2.2.1 Lossless Vs Lossy Compression.....	5
2.2.2 Predictive Vs Transform Coding	5
2.2.3 Subband Coding.....	6
2.3 Loss less Compression.....	6
2.4 Lossy Compression techniques.....	7
2.4.1 Subband Coding.....	8
2.4.2 Transform Coding	10
2.4.2.1 Discrete Cosine Transform (DCT) Based Coding	11
2.4.2.2 Lapped Transforms (LT) Based Coding	14
2.4.2.3 Discrete Wavelet Transform (DWT) Based Coding.....	14
2.4.2.3.1 JPEG 2000	16
CHAPTER3 ARCHITECTURE.....	18
3.1 Outline of JPEG	18
3.2 Architectures of JPEG.....	21
3.2.1 Color Conversion.....	21
3.2.2 Discrete Cosine Transform	22
3.2.3 Quantization.....	27
3.2. 4 Zig zag Scanning	30
3.2.5 Entropy Coder.....	31
3.2.5.1 Differential Coder	34
3.2.5.2 Run Length Encoder	35
3.2.5.3 Size Calculator	37
3.2.5.4 Variable Length Coder.....	38
3.2.5.5 Huffman Encoder.....	39
3.2.5.6 Preassmebler	40

3.2.5.7 Assembler	41
CHAPTER4 RESULTS AND DISCUSSION.....	44
4.1 Simulation Waveforms	44
4.1.1 Color Conversion	44
4.1.2 DCT.....	44
4.1.3 Quantization.....	46
4.1.4 Zig zag Scanning.....	47
4.1.5 Differential Coder	48
4.1.6 Run Length Encoder	48
4.1.7 Size Calculator	49
4.1.8 VLC Coder.....	50
4.1.9 Huffman Encoder.....	50
4.1.10 Preassembler	52
4.1.11 Assembler	53
4.1.12 JPEG	54
4.2 Synthesis Report	55
CHAPTER5 CONCLUSION	57
REFERENCES	58
VITA	61

LIST OF FIGURES

Figure 2.1	Block Diagram of SBC.....	9
Figure 2.2	2-D DCT using Vector Processing.....	13
Figure 2.3	Level-3 dyadic DWT scheme used for Image Compression.....	16
Figure 2.4	General block diagram of the JPEG 2000 encoder	17
Figure 3.1	The JPEG Baseline Encoder.....	18
Figure 3.2	1-D DCT Implementation.....	25
Figure 3.3	2-D DCT Implementation.....	26
Figure 3.4	Quantization Architecture	27
Figure 3.5	Zigzag Scanning	31
Figure 3.6	Entropy Encoder.....	31
Figure 3.7	Pipelined Architecture for Entropy coder.....	33
Figure 3.8	Differential Coder.....	34
Figure 3.9	Run Length Encoder.....	45
Figure 3.10	Huffman Coder Architecture	39
Figure 3.11	Preassembler Architecture.....	41
Figure 3.12	Assembler Architecture	42
Figure 4.1	Simulation of Color Conversion.....	44
Figure 4.2	Simulation of DCT	45
Figure 4.3	Simulation of Quantizer	46
Figure 4.4	Simulation of Zig Zag Scanner.....	47
Figure 4.5	Simulation of Differential Coder.....	48
Figure 4.6	Simulation of Run Length encoder.....	48
Figure 4.7	Simulation of Size Calculator.....	49
Figure 4.8	Simulation of VLC Coder	50
Figure 4.9	Simulation of Huffman Coder for DC Components.....	50
Figure 4.10	Simulation of Huffman Coder for AC Components.....	51
Figure 4.11	Simulation of Preassembler.....	52
Figure 4.12	Simulation of Assembler	53
Figure 4.13	Simulation of JPEG Encoder.....	54
Figure 4.14	RTL Schematic of JPEG Encoder	55
Figure 4.15	Design Summary of JPEG for Vertex-4 FPGA.....	56

ACKNOWLEDGEMENTS

It is great pleasure for me to acknowledge the people who have helped me during the course of my thesis work. My special thanks to my advisor, Dr. Henry Selvaraj who has supported me in the right direction. I would specially acknowledge Dr. Emma Regentova, Dr. Yahia Baghzouz for serving as committee members and Dr Laxmi Gewali for serving as graduate college representative. I would like to thank my friends who gave me morale support in achieving this. I would like to acknowledge Aldec- ari people for supporting with the project “Real time system on Chip” # 2368-254-50YH.

CHAPTER 1

INTRODUCTION

The transition from magnetic film based image representation to digital representation has been primarily motivated by the ease of working with digital data and better special representation of the image. Over the years, the need for image compression has grown steadily and currently it is being recognized as an enabling technology. For example, image compression has been and continues to be crucial to the growth of multimedia computing. In addition, it is the natural technology for handling the increased spatial resolutions of today's imaging sensors, and evolving broadcast television standards. Furthermore, image compression plays a crucial role in many important and diverse applications, including videoconferencing, remote sensing, document and medical imaging, facsimile transmission (FAX), and the control of remotely piloted vehicles in military, space, and hazardous waste control applications. In short, an ever-expanding number of applications depend on the efficient manipulation, storage, and transmission of binary, gray-scale, or color images. One notable area of application which is greatly driving R&D in image compression is the enormous growth in the use of Internet and mobile communication devices that generated a revolution in the way human-beings communicate and exchange information. The necessity of efficient digital information

delivery (e.g. images) in those devices is imperative, and different methods to do that have been proposed. A digital image uses a big storage space and big bandwidth for transmission, in mobile devices this is a problem because the space and bandwidth can be spent or saturated rapidly. A possible solution to solve this problem is to find a representation that use less information to represent digital images, by this necessity image compression emerges in the field of video and digital images. Image compression addresses the problem of reducing the data amount required to represent a digital image and is made by a removal process of the image redundant information [1]. An ideal scheme is to make lossy image compression in order to save a lot of storage space but sacrificing the quality of an image. We can compress the image with lossless compression techniques (e.g. Run Length Coding, Huffman Coding) but the compression ratio is small. These techniques are highly useable in the areas like Medical and Military applications where highly accurate data is needed.

JPEG image compression is lossy compression technique which is based on transform coding. Basically the image compression techniques make use of following factors for the compression: One is majority of useful content changes relatively slowly across the image. So by transforming the image content to frequency domain we can represent data as frequency components. Usually low frequency components contain most of the image data than high frequency components. For compressing the image we can eliminate the high frequency components. The other is defects in Human visual system (HVS). Human eye is less perceptible to High frequency components than low frequency.

1.1 Background

Digital image compression is a very popular research topic in the field of multimedia processing. The main objective of research is to develop architecture for JPEG Compression schemes that give good visual quality and speed. The Compression technique was implemented hardware description language like VHDL, VERILOG. Hardware implementation speeds up image/video processing comparing to software.

1.2 Thesis Outline

This thesis is organized into five chapters. Chapter 1 gives the introduction. Chapter 2 gives an overview of image compression and classification of compression schemes. It discusses different compression methods such as subband coding, discrete cosine transform (DCT), lapped transform (LT) and discrete wavelet transform based coding. Chapter 3 describes architecture of JPEG image compression and its implementation. Chapter 4 gives simulated results of different modules in JPEG compression and discussion about the results. Chapter 5 describes the final conclusion of the thesis and presents some future work.

CHAPTER 2

2.1 DATA COMPRESSION BASICS

Data compression is the reduction or elimination of redundancy in data representation in order to achieve savings in storage and communication costs. It relies on the fact that image information, by its very nature, is not random but exhibits order and has some form of structure. If this order and structure can be extracted, the essence of the information often can be represented and transmitted using less data bits than would be needed for the original. We can then reconstruct the original or a close approximation of it at the receiving end. A common characteristic of most images is that the neighboring pixels are correlated and therefore contain redundant information. Image, Video and audio signals are amenable to compression due to the following factors: redundancy and irrelevancy reduction.

Redundancy reduction: Redundancy looks at “properties” of an image and reduces redundant data.

Irrelevancy reduction: Much of the data in an image may be irrelevant to a human observer so we can omit that data.

In general, three types of redundancy can be identified:

- Spatial Redundancy or correlation between neighboring pixel values.
- Spectral Redundancy or correlation between different color planes or spectral bands.

- Temporal Redundancy or correlation between adjacent frames in a sequence of images (in video applications).

Image compression research aims at reducing the number of bits needed to represent an image by removing the spatial and spectral redundancies as much as possible. Since we will focus only on still image compression, we will not worry about temporal redundancy.

2.2. Data Compression Techniques

2.2.1 Lossless vs. Lossy Compression

In lossless Compression schemes the reconstructed image, after compression is digitally identical to the original image. However, lossless compression can only achieve a modest amount of compression. On the other hand, lossy schemes are capable of achieving much higher compression but under normal viewing conditions no visible loss is perceived (visually lossless). Some of the lossy compression schemes used include differential pulse code modulation (DPCM), pulse code modulation (PCM), vector quantization (VQ), Transform and Subband coding. An image reconstructed following a lossy compression contains degradation relative to the original. Often this is because the compression scheme also discards non-redundant information.

2.2.2. Predictive vs. Transform Coding

In Predictive Coding, information already sent or available is used to predict future values, and the difference is coded. It removes redundancy between successive pixels. It only encodes residual between actual and predicted. Since this is done in the image or spatial domain, it is relatively simple to implement and is readily adapted to local image

characteristics. Differential Pulse Code Modulation (DPCM) is one particular example of predictive coding. Transform coding, on the other hand, first transforms the image from its spatial domain representation to a different type of representation using some well-known transforms such as DCT, DWT or Lapped transform, and then codes the transformed values (coefficients). This method provides greater data compression compared to predictive methods as transforms use energy compaction properties to pack an entire image or a video frame into

2.2.3 Subband Coding

In Subband Coding, information (image) is split in to frequency band of a signal in various subbands. To code each subband, we use a coder and bit rate accurately matched to the statistics of the subband.

2.3 Lossless Compression

In lossless compression schemes the reconstructed image, after compression is numerically identical to the original image. Through lossless compression we can only achieve a modest amount of compression [1].

The lossless methods are also called entropy-coding schemes, since there is no loss of information content during the process of compression. This type of compression is used in certain environments such as compression of text, database records, spreadsheets, word processing files, or medical and military imaging medical imaging where no loss of information is tolerated. Typical compression ratios for lossless data compression are around 3:1.

2.4 Lossy Compression Technique

In lossy compression, the reconstructed image is approximation of the original image. Lossy compression is generally used for video and sound, where a certain amount of information loss can be tolerated. The JPEG image compression is one of the examples of lossy compression. Using JPEG compression, one can decide how much loss to introduce and make a trade-off between file size and image quality. Depending upon the fidelity required, compression ratios of even up to 100:1 can be obtained.

The JPEG committee has created many standards since it was created in 1986. ISO had actually started to work on this 3 years earlier, in April 1983, in an attempt to find methods to add photo quality graphics to the text terminals of the time, but the 'Joint' that the 'J' in JPEG stands for refers to the merger of several groupings in an attempt to share and develop their experience. This is the collaboration between three international standard organizations, International Telegraph and Telephone Consultative Committee (CCITT), International Organization for Standardization (ISO), and the International Electrotechnical Commission (IEC).

The formal name of the standard that most people refer to as 'JPEG' is ISO/IEC IS 10918-1 | ITU-T Recommendation T.81, as the document was published by both ISO through its national standards bodies, and CCITT, now called ITU-T. IS 10918 has actually 4 parts

Part 1 - The basic JPEG standard, which defines many options and alternatives for the coding of still images of photographic quality

Part 2 - which sets rules and checks for making sure software, conforms to Part 1

Part 3 - set up to add a set of extensions to improve the standard, including the SPIFF file format

Part 4 - defines methods for registering some of the parameters used to extend JPEG [1].

JPEG has defined an international standard for coding and compression of continuous tone still images. The primary aim of the JPEG standard is to propose an image compression algorithm that would be generic, application independent and aid VLSI implementation of data compression. To meet the different applications, the JPEG standard includes two basic compression methods, each with various modes of operation. For lossy compression DCT (Discrete Cosine Transform) method is proposed and a predictive method for lossless Compression. The Baseline DCT method is most widely implemented JPEG method for many applications.

The compression ratio of the image is given by:

$$\text{Compression ratio} = \frac{\text{Source coder input data size}}{\text{Source coder output data size}} \quad (2.1)$$

Most widely used lossy compression techniques are

(i) Subband Coding

(ii) Transform Coding

2.4.1 Subband Coding

The fundamental concept behind Subband Coding (SBC) is to split up the frequency band of a signal (image in our case) into various frequency subband or subband signals and then to code each subband using a coder and bit rate accurately matched to the statistics of the band. SBC has been used extensively first in speech coding [10, 13] and

later in image coding [14] because of its inherent advantages like variable bit assignment among the subbands and coding error confinement within the subbands.

The simplest way to encode audio signals is Pulse Code Modulation (PCM), which is used on music CDs, DAT recordings, and so on. This produces a high quality signal, but at a high bit rate (over 700k bps for one channel of CD audio). To reduce the bandwidth we can use mu-law encoding. This is like PCM on a logarithmic scale, and the effect is to add noise that is proportional to the signal strength. Sun's au format for sound files is a popular example of mu-law encoding. Using 8-bit mu-law encoding we can reduce the bandwidth to 350k bps, which is better than PCM.

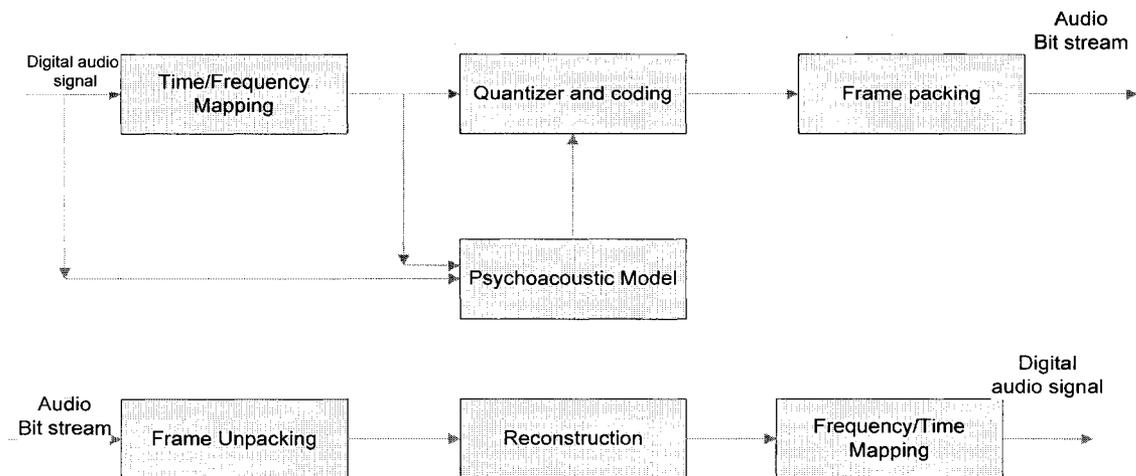


Figure 2.1 Block Diagram of SBC

Fig 2.1 represents general subband encoder. First, a time-frequency mapping (a filter bank, or FFT, or something else) decomposes the input signal into subbands. The psychoacoustic model looks at these subbands as well as the original signal, and determines masking thresholds using psychoacoustic information. Using these masking

thresholds, each of the subband samples is quantized and encoded so as to keep the quantization noise below the masking threshold. Finally all these encoded bits are packed as a frame and sent through communication channel.

At the decoder end, frames are unpacked, subband samples are decoded, and a frequency-time mapping turns them back into a single output audio signal.

Disadvantages of Subband Coding

- One of the major problems with the subband coding is to resolve the bit allocation problem or the number of bits assigned to each individual subband to get the best performance. One way is to use the idea of optimal bit allocation to each quantized subband output individually. This is mostly valid for higher bit rates of approximately 1 bit/sample or more.
- In Subband coding method it is difficult to determine optimal coding system for low bit rate applications.
- If the overall bit rate changes the optimal bit allocation change which requires repetition of entire coding process again.
- As the filters are not ideal filters it is not possible to perfectly decorrelate all the frequency Subbands and there is slight overlapping between adjacent frequency Subbands.
- It is very difficult to use Subband coding scheme for motion compensated video because of frequency Subbands.

2.4.2 Transform Coding

Transform Coding is converting information from one set values to another using mathematical functions.

Different types of transform coding techniques are

- (a) Discrete Cosine Transform (DCT) based coding
- (b) Lapped Transforms (LT) based coding
- (c) Discrete Wavelet Transform (DWT) based coding.

2.4.2.1 Discrete Cosine Transform (DCT) Based Coding

Discrete cosine transform (DCT) translates the image information from spatial domain to frequency domain to be represented in a more compact form. DCT properties are similar to Fourier transform.

By simple analogy we can illustrate how DCT works. Consider an unsorted list of 15 numbers between 0 and 4 (2, 3, 1, 4, 2, 2, 0, 1, 4, 1, 0, 1, 4, 0, and 0). The transformation involves two steps one is sorting the list and second is counting the frequency of occurrence of each number -> (4, 4, 3, 1, and 3). Through this transformation we lost the spatial information but captured the frequency information.

Neighboring pixels within an image are highly correlated. So it is required to use any transform to exploit this correlation and representing information with fewer number of bits. The Discrete Cosine Transform (DCT) has been shown to be near optimal for a large class of images in energy concentration and de-correlating (Karhunen Loeve Transform {KLT} is the optimal transform but it isn't used because its difficulty to practically implement) [7]. The DCT decomposes the signal into spatial frequencies, which then allow further processing techniques to reduce the precision of the DCT coefficients consistent with the Human Visual System (HVS) model.

Discrete Cosine Transform (DCT) is a lossy compression scheme where an $N \times N$ image block is transformed from the spatial domain to the DCT domain. DCT convert the

input image into spatial frequency components called DCT coefficients, in such a way that lower frequency components appear at left hand corner and high frequency components at right hand side of DCT matrix. As we know Human Visual System is less sensitive to high frequency components than low frequency components, we can further process the coefficients by quantization like process to represent data with less number of bits.

Advantages of DCT

- DCT is the near-optimal for signal processing
- Efficient and wide acceptability
- Parallel processing capability
- Less complex comparing to other transform algorithms
- DCT can be done block by block level.

Mathematical equations of DCT

The 2-D DCT is give as

$$XC_{PQ} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} XN_{mn} \cdot \frac{C(p)C(q)}{4} \cdot \cos \frac{\pi(2m+1)p}{2M} \cdot \cos \frac{\pi(2n+1)q}{2N} \quad (2.2)$$

First 1-D DCT for rows is calculated and then the 1- D DCT of columns is calculated.

The above equation is divided into rows and column parts as follows:

$$C = K \cdot \cos \frac{(2 \cdot \text{colnumber} + 1) \cdot \text{rownumber} \cdot \pi}{2M} \quad (2.3)$$

$$K = \frac{\sqrt{1}}{N} \quad \text{for row} = 0$$

$$K = \frac{\sqrt{2}}{N} \quad \text{for row} \neq 0$$

$$C^t = K \cdot \cos \frac{(2 \cdot \text{rownumber} + 1) \cdot \text{colnumber} \cdot \pi}{2N} \quad (2.4)$$

$$K = \frac{\sqrt{1}}{M} \quad \text{for column} = 0$$

$$K = \frac{\sqrt{2}}{M} \quad \text{for column} \neq 0$$

For the 8X8 blocks, a one dimensional DCT/IDCT followed by an internal buffer memory followed by one-dimensional DCT is used to perform 2-D DCT. This way we can reduce the computation complexity of DCT for the 2-D Image.

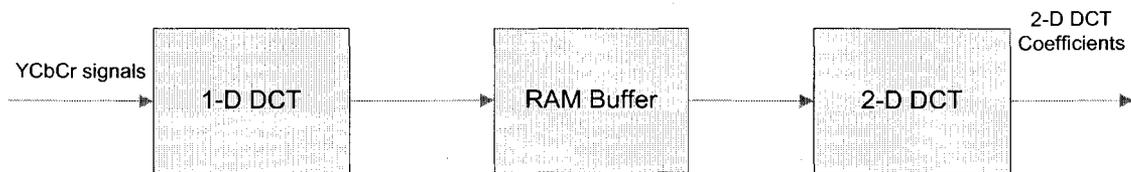


Figure 2.2 2-D DCT using Vector Processing

Disadvantages of DCT

- In JPEG, we divide an image into 2-D non-overlapping blocks of 8X8 and apply 8-point 2-D DCT on them to obtain fewer transformed coefficients. By this process we will only exploit spatial correlation between pixels but not correlation between blocks.
- The second disadvantage is blocking artifacts, discontinuities at the block boundaries (because of using 8X8 blocks) resulting from reconstruction mismatches at low bit-rate situations.

2.4.2.2 Lapped Transforms (LT) Based Coding

The lapped transform was developed to solve the problem of blocking effect in DCT based coding schemes. Instead of non-overlapping 2-D blocks, the process uses overlapping 2-D blocks of an image spatially. One of the special types of lapped transforms is called lapped orthogonal transform (LOT).

Advantages of lapped transform

- No need to use block based coding.
- Coding efficiency can be improved by taking into account of inter block spatial correlation.
- Blocking artifacts are eliminated
- Pre- and post-filter are can be constructed in modular cascaded stages, to minimize hardware/software modifications.

By lapped transform blocking effects are reduced but other effects like ringing around edges of blocks will appear due longer basis functions. LOT is extension of DCT but due to its complexity compared to improved advantages, so it is less popular for image compression [2].

2.4.2.3 Discrete Wavelet Transform (DWT) Based Coding

Discrete wavelet transform (DWT) is one of the latest coding techniques used instead of DCT. Its main advantage over DCT is that there is no need to divide the image into non overlapping blocks. Because of their inherent multi resolution nature, wavelet-coding schemes are especially suitable for applications where scalability and tolerable degradation are important. After JPEG image compression JPEG committee has released its new image coding standard, JPEG-2000, which has been based upon DWT.

By Fourier transform (DCT based) we can represent signal as sum of sine and cosine functions. By this we can know frequency spectrum of the signal, but we do not know when and where they are present. To overcome this problem we should be able to represent signal in frequency as well as time domain. This is done by wavelet transform.

By time-frequency joint representations one has to cut the signal of interest into several parts and then analyze the parts separately, by this we can get more information about the signal. In wavelet transform the use of a fully scalable modulated window solves the signal-cutting problem. The window is shifted along the signal and for every position the spectrum is calculated. Wavelet transform is convolving input signal with particular instances of the wavelet (window) at various time scales and positions. Then this process is repeated many times for every new cycle. By this we can get signal in time as frequency domain, all with different resolutions [3].

Performing these convolutions at every position and every characteristic scale is called the continuous wavelet transform. By Nyquist's theorem the highest frequency we can model with discrete signal data is half that of the sampling frequency. So in the worst case we have to use the transform at every other point [4].

The continuous wavelet transform is generally expressed as:

$$CWT_x^\psi(\tau, s) = \frac{1}{\sqrt{s}} \int x(t) \psi\left(\frac{t-\tau}{s}\right) dt \quad [4] \quad (2.5)$$

In CWT, the signals are analyzed using a set of basis functions which relate to each other by simple scaling and translation. In the case of DWT, digital filtering techniques are used for the time-scale representation. The signal to be analyzed is passed through filters with different cutoff frequencies at different levels [12].

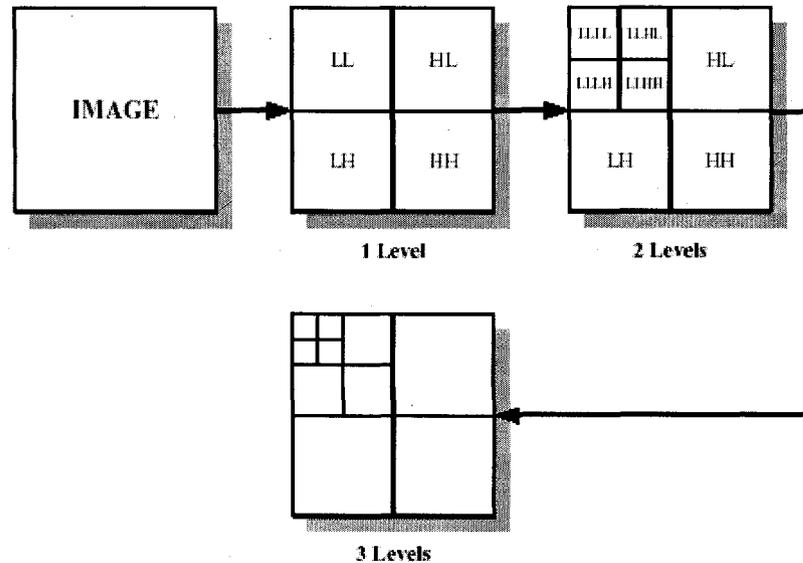


Figure 2.3 Level-3 dyadic DWT scheme used for Image Compression [5]

2.4.2.3.1 JPEG 2000

JPEG2000, the new standard for still image coding, better addresses the problems of still image compression by previous methods. It offers a wide range of functionalities such as lossless and lossy coding, embedded lossy to lossless coding, progression by resolution and quality, high compression efficiency, error resilience and region-of-interest (ROI) coding. Comparative results have shown that JPEG2000 is indeed superior to established image compression standards [5].

In JPEG-2000 compression first the image is preprocessed by tiling the image i.e. partitioning the original image into non-overlapping blocks. Tile components are decomposed into various decomposition levels by using separable wavelet transform, than a scalar

quantization is used to quantize than each block is entropy encoded. EBCOT process is used for Entropy coding.

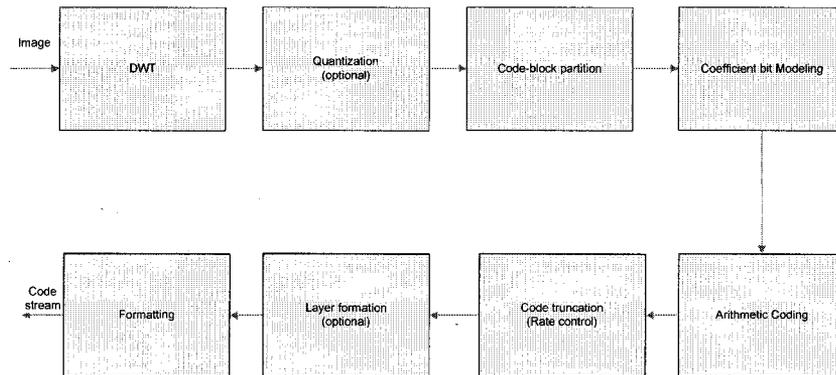


Figure 2.4 General block diagram of the JPEG 2000 Encoder [11]

Advantages OF JPEG 2000

- JPEG 2000 offers high image quality than JPEG.
- In the JPEG 2000 compression the compressor can choose image quality, maximum resolution and losses.
- JPEG 2000 can provide both lossless and lossy compression in the same compression engine.

Disadvantages of DWT

- The cost of computing DWT as compared to DCT is much higher. The complexity of calculating DWT depends upon the length of wavelet filter.
- Larger DWT basis functions or wavelet filters produces blurring and ringing noise near edge regions in images or video frames.

CHAPTER 3

ARCHITECTURE

3.1 Outline of JPEG

The basic model of JPEG is shown below

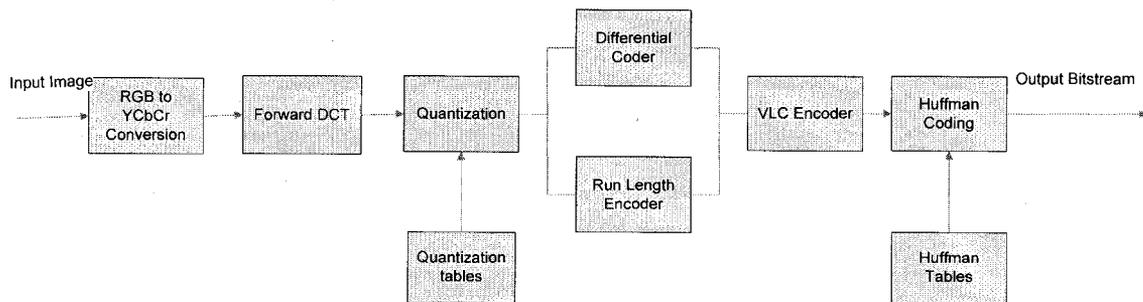


Figure 3.1 JPEG Baseline Encoder

The Joint Photographic Expert Group proposed the JPEG compression standards [6]. The encoder model transforms the input image into suitable form for further processing. After that entropy encoder compresses the output form encoder.

Different modes of JPEG are

- Lossless Coding
- Sequential Coding
- Progressive Coding
- Hierarchical Coding.

In Lossless Coding the image can be reconstructed after decoding. In this process we use methods like differential coding, Huffman coding, Arithmetic coding.

In Sequential Coding image blocks are scanned sequentially from top to bottom and left to right. Baseline Coding is example of sequential coding. In Progressive Coding image blocks are processed sequentially, but coding is completed in multiple scans. The first scan yields the full image but without full details which are provided in successive scans. In Hierarchical Coding each image component is encoded as a sequence of frames. The first frame is usually a low resolution of original image and subsequent frames are differential frames between original and reference reconstructed image [7].

Based on these modes there are four distinct processes for jpeg image compression.

- Baseline process,
- Extended DCT-based process,
- Lossless process,
- Hierarchical process.

Both baseline and extended DCT processes uses DCT in the encoding process, but in the entropy coding Baseline uses Huffman encoding and extended DCT uses Huffman or arithmetic coding. Lossless process uses predictive or sequential methods for encoding and Huffman or arithmetic for entropy process. Hierarchical process uses either DCT or lossless process for encoding and same entropy encoding methods as other process.

In this thesis we are going to discuss about implementation of baseline JPEG image compression.

Baseline JPEG mode is the most widely used jpeg image compression. Baseline mode is simple and is based on sequential mode i.e. Image is scanned from left to right and top

to bottom. Image is divided into non overlapping blocks of 8X8 each of 8 bit, DCT process, Quantization and entropy encoding steps are performed on that.

The JPEG Baseline can be divided mainly into five parts: those are color space conversion, down sampling, 2-D DCT, quantization and entropy encoding. The color space conversion converts the image form RGB color to YCbCr (luminance component Y and two chrominance components Cb and Cr). Luminance components component contains gray image and chrominance components contain color information. The down sampling reduces the sampling rate of color information (Cb,Cr). 2-D DCT transform image information from spatial domain to frequency domain. By quantization operation high frequency components are eliminated and low frequency components are represented by less number of bits. JPEG uses predefined quantization tables for eliminating the high frequency components. The selection of quantization tables is critical since it affects both compression efficiency and image quality. After quantization, the DCT coefficients are arranged in zigzag order to get low frequency components at the top and high frequency components at the bottom. It maps 8X8 block to 1X64 values. Finally entropy coding is applied. It uses differential coding for the DC components and Run Length Encoding for AC Components. The location of (0, 0) of each block i contains DC Coefficient represented as DC_i . Since the adjacent blocks are likely have similar average energy levels so we can send only the difference of current and previous DC coefficients which is know as Differential Pulse Code Modulation (DPCM). The 1X64 vectors have lot of zeros, it is represented by [run length, count, and value] pair by Run Length encoding to reduce the number of bits to represent data. In Run Length encoding only non-zero values will be sent with counting the number of zeros preceding it. After

that Variable length coding (VLC) and Huffman coding is applied to represent data with less number bits [6]. In the VLC coder the amplitude is represented with its significant bit as most significant bit. For each pair of run length codes there is a variable length Huffman code which will be used by the Huffman encoder to perform the compression. The Huffman codes are stored in tables. In the JPEG image compression process down sampling and quantization are irreversible, but the losses can be controlled depending the necessity of image quality [8].

3.2 Architectures of jpeg

3.2.1 Color Conversion

The color space conversion is the first operation in a JPEG compressor if the input images are in RGB color space. Although the JPEG algorithm is unaffected by color, since it processes each color independently, but change in color space improves compression ratio significantly. This is due to defect in Human Visual System (HVS) that is less particular for some of the characteristics of the image and also RGB is not efficient in dealing of real world images. In RGB all the three components need equal band width to generate colors and highly correlated. RGB images are not very best for processing of the image too. For example if want to change intensity of pixel we should call all the three colors and process the colors. If we have any access to intensity of colors directly the processing will be faster. The appropriate representation of colors for JPEG compression is YCbCr Where Y is Luminance component and Cb and Cr are two Chrominance components. Luminance component contains image information (Gray

scale) and Chrominance component contains color information. Component Cb contains information relative to blue color and Cr component contains information relative to red color. The range of YCbCr is 16-235 for 8 bit representation.

The below calculation are used in converting RGB to YCbCr.

$$Y_{i,j} = 0.299R_{i,j} + 0.587G_{i,j} + 0.114B_{i,j}$$

$$Cb_{i,j} = -0.169R_{i,j} - 0.331G_{i,j} + 0.5B_{i,j}$$

$$Cr_{i,j} = 0.5R_{i,j} - 0.419G_{i,j} - 0.081B_{i,j}$$

The source image is portioned into non-overlapping 2-d blocks of 8x8, which are scanned sequentially from left to right and top to bottom. The nominal range of Luminance component is 0 to 1 and Chrominance component's nominal range is -0.5 to 0.5. To make Chrominance components range equal to Luminance, 128 is added to the Cb and Cr components [17].

This color conversion architecture is based on simplified models provided by Xilinx Corporation which uses only four multipliers [17]. The architecture has latency of six clock cycles and operates at frequency of 285 MHz.

3.2.2 Discrete Cosine Transform (DCT)

Discrete Cosine Transform (DCT) is a lossy compression scheme where an $N \times N$ image block is transformed from the spatial domain to the DCT domain. DCT decomposes the signals into frequency domain which are called DCT coefficients. The lower frequency DCT coefficients appear towards upper left corner and higher frequency DCT coefficients are in the right-hand corner of DCT matrix. The Human visual System is less sensitive to high frequency components so we can quantize high frequency components by quantization.

For implementing of DCT we use vector processing using four parallel multipliers. The output Y of 8 X 8 DCT for input X is given by $Y = C \cdot X \cdot C^t$, where C is the matrix with the cosine basis functions, and C^t is the transpose coefficients [18]. Using row column decomposition Y can be computed by 1-D DCT transforms as

$$Y=C \cdot Z \text{ where } Z= X \cdot C^t. \quad (2.6)$$

The mathematical equation for DCT is given as

$$XC_{PQ} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} XN_{mn} \cdot \frac{C(p)C(q)}{4} \cdot \cos \frac{\pi(2m+1)p}{2M} \cdot \cos \frac{\pi(2n+1)q}{2N} \quad (2.7)$$

First 1-DCT is performed for rows and then for columns. The 1-D DCT is calculated by separating equation-1 into rows and column parts.

Where C and C^t are calculated as

$$C = \begin{pmatrix} 23170 & 23170 & 23170 & 23170 & 23170 & 23170 & 23170 & 23170 \\ 32148 & 27246 & 18205 & 6393 & -6393 & -18205 & -27246 & -32138 \\ 30724 & 12540 & -12540 & -30274 & -30274 & 12540 & 12540 & 30274 \\ 27246 & -6393 & -32138 & -18205 & 18205 & 32138 & 6393 & -27246 \\ 23170 & -23170 & -23170 & 23170 & 23170 & -23170 & -23170 & 23170 \\ 18205 & -32138 & 6393 & 27246 & -27246 & -6393 & 32138 & -18205 \\ 12540 & -30274 & 30274 & -12540 & -12540 & 30724 & -30724 & 12540 \\ 6393 & -18205 & 27246 & -32138 & 32138 & -27246 & 18205 & -6393 \end{pmatrix}$$

$$C^t = \begin{pmatrix} 23170 & 32138 & 30274 & 27246 & 23170 & 18205 & 12540 & 6393 \\ 23170 & 27246 & 12540 & -6393 & -23170 & -32138 & -30274 & -18205 \\ 23170 & 18205 & -12540 & -32138 & -23170 & 6393 & 30274 & 27246 \\ 23170 & 6393 & -30274 & -18205 & 23170 & 27246 & -12540 & -32138 \\ 23170 & -18205 & -12540 & 32138 & -23170 & -6393 & 30274 & -27246 \\ 23170 & -27246 & 12540 & 6393 & -23170 & 32138 & -30274 & 18205 \\ 23170 & -32138 & 30274 & -27246 & 23170 & -18205 & 12540 & -6393 \end{pmatrix}$$

The intermediate value $Z = X \cdot C^t$ can be calculated as follows:

Where

$$X = \begin{bmatrix} x_{00} & x_{01} & x_{02} & x_{03} & x_{04} & x_{05} & x_{06} & x_{07} \\ x_{10} & x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & x_{16} & x_{17} \\ x_{20} & x_{21} & x_{22} & x_{23} & x_{24} & x_{25} & x_{26} & x_{27} \\ x_{30} & x_{31} & x_{32} & x_{33} & x_{34} & x_{35} & x_{36} & x_{37} \\ x_{40} & x_{41} & x_{42} & x_{43} & x_{44} & x_{45} & x_{46} & x_{47} \\ x_{50} & x_{51} & x_{52} & x_{53} & x_{54} & x_{55} & x_{56} & x_{57} \\ x_{60} & x_{61} & x_{62} & x_{63} & x_{64} & x_{65} & x_{66} & x_{67} \\ x_{70} & x_{71} & x_{72} & x_{73} & x_{74} & x_{75} & x_{76} & x_{77} \end{bmatrix}$$

$$Z_{(0,0)} = 23170(x_{00} + x_{01} + x_{02} + x_{03} + x_{04} + x_{05} + x_{06} + x_{07})$$

$$\begin{aligned} Z_{(0,1)} &= 32138x_{00} + 27246x_{01} + 18205x_{02} + 6393x_{03} - 6393x_{04} - 18205x_{05} - 27246x_{06} - 32138x_{07} \\ &= 32138(x_{00} - x_{07}) + 27246(x_{01} - x_{06}) + 18205(x_{02} - x_{05}) + 6393(x_{03} - x_{04}) \end{aligned}$$

$$Z_{(0,2)} = 30274(x_{00} - x_{07}) + 12540(x_{01} - x_{06}) - 12540(x_{02} + x_{05}) - 30274(x_{03} + x_{04})$$

$$Z_{(0,3)} = 27246(x_{00} - x_{07}) - 6393(x_{01} - x_{06}) - 32138(x_{02} - x_{05}) - 18205(x_{03} - x_{04})$$

$$Z_{(0,4)} = 23170(x_{00} - x_{07}) - 23170(x_{01} - x_{06}) - 23170(x_{02} + x_{05}) + 23170(x_{03} + x_{04})$$

$$Z_{(0,5)} = 18205(x_{00} - x_{07}) - 32138(x_{01} - x_{06}) + 6393(x_{02} - x_{05}) + 27246(x_{03} - x_{04})$$

$$Z_{(0,6)} = 12540(x_{00} - x_{07}) - 30274(x_{01} - x_{06}) + 30274(x_{02} + x_{05}) - 12540(x_{03} + x_{04})$$

$$Z_{(0,7)} = 6393(x_{00} - x_{07}) - 18205(x_{01} - x_{06}) - 27246(x_{02} - x_{05}) - 32138(x_{03} - x_{04})$$

Or

$$Z_{(k,0)} = 23170(x_{k0} + x_{k1} + x_{k2} + x_{k3} + x_{k4} + x_{k5} + x_{k6} + x_{k7})$$

$$Z_{(k,1)} = 32138x_{k0} + 27246x_{k1} + 18205x_{k2} + 6393x_{k3} - 6393x_{k4} - 18205x_{k5} - 27246x_{k6} - 32138x_{k7}$$

$$= 32138(x_{k0} - x_{k7}) + 27246(x_{k1} - x_{k6}) + 18205(x_{k2} - x_{k5}) + 6393(x_{k3} - x_{k4})$$

$$Z_{(k,2)} = 30274(x_{k0} - x_{k7}) + 12540(x_{k1} - x_{k6}) - 12540(x_{k2} + x_{k5}) - 30274(x_{k3} + x_{k4})$$

$$Z_{(k,3)} = 27246(x_{k0} - x_{k7}) - 6393(x_{k1} - x_{k6}) - 32138(x_{k2} - x_{k5}) - 18205(x_{k3} - x_{k4})$$

$$Z_{(k,4)} = 23170(x_{k0} - x_{k7}) - 23170(x_{k1} - x_{k6}) - 23170(x_{k2} + x_{k5}) + 23170(x_{k3} + x_{k4})$$

$$Z_{(k,5)} = 18205(x_{k0} - x_{k7}) - 32138(x_{k1} - x_{k6}) + 6393(x_{k2} - x_{k5}) + 27246(x_{k3} - x_{k4})$$

$$Z_{(k,6)} = 12540(x_{k0} - x_{k7}) - 30274(x_{k1} - x_{k6}) + 30274(x_{k2} + x_{k5}) - 12540(x_{k3} + x_{k4})$$

$$Z_{(k,7)} = 6393(x_{k0} - x_{k7}) - 18205(x_{k1} - x_{k6}) - 27246(x_{k2} - x_{k5}) - 32138(x_{k3} - x_{k4})$$

Where $k=0,2,\dots,7$

Then 2-d DCT function is calculated from $Y = CZ$. Where Z is 1-D DCT matrix for input X and C is matrix of cosine coefficients.

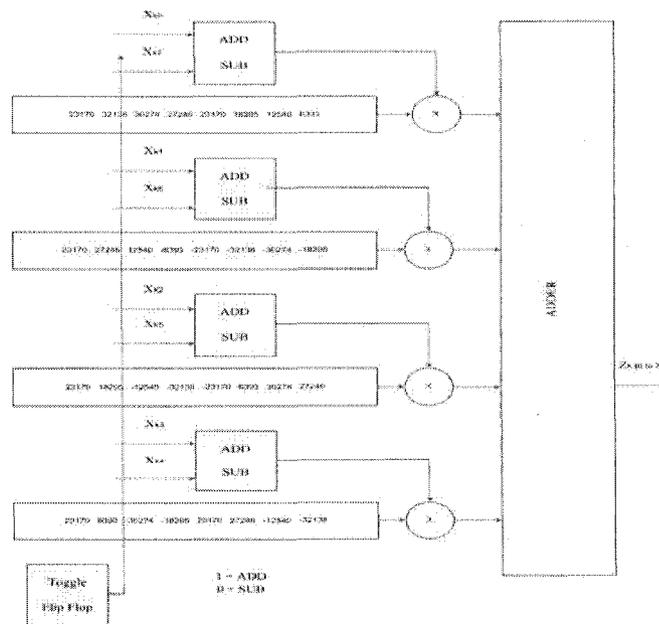


Figure 3.2 1-D DCT Implementation [18]

The above block diagram is used for implementation of 1-d DCT. First 1-D DCT values are calculated and stored in a RAM and second 1-D DCT is done on the values stored in the RAM. 8X8 inputs are loaded into adder/subtractor whose outputs are fed to the multiplier. The multiplier takes constant coefficients from the ROM and feed into the second input of the multiplier. The multiplier outputs are given to adder which will perform additions and gives 1-D DCT Coefficients which will be stored in a transpose buffer (RAM). The toggle flip flop controls the addition and subtraction operations.

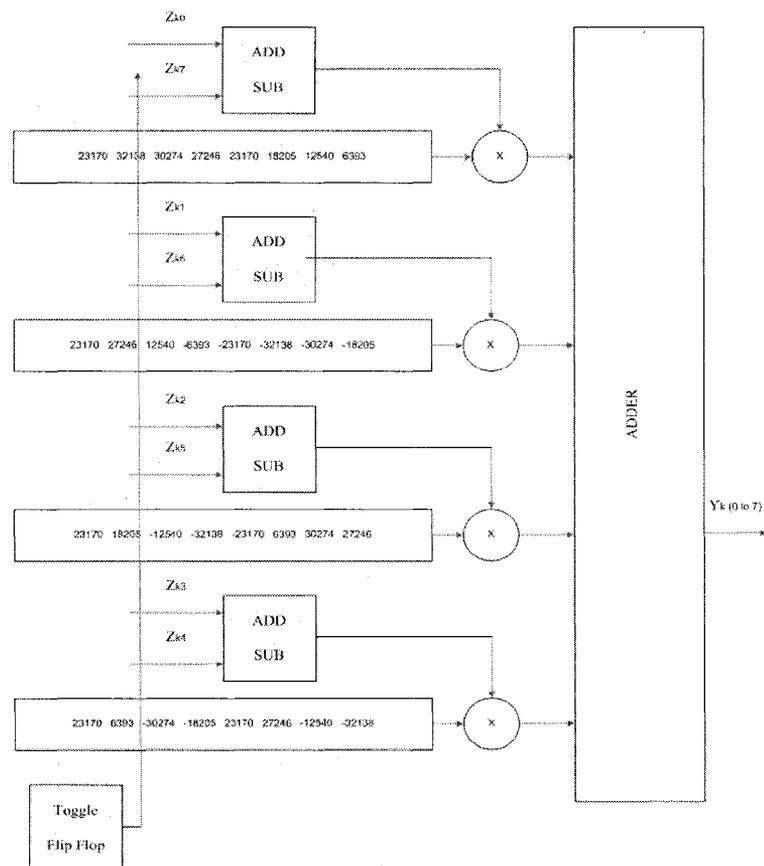


Figure 3.3 2-D DCT Implementation [18]

The values stored in the transpose buffer are read column by column and fed as input to second DCT. The output of DCT is 2-D DCT coefficients which are used as inputs to quantizer for further processing.

3.2.3 Quantization

The quantization process reduces number of bits used to represent the DCT coefficients. Since Human eye is less sensitive to high frequency components than low frequency components so quantization factors are high for high frequency components than low frequency components.

The quantization operation is an integer division of the 2-D DCT coefficients by pre-defined values. These pre-defined values are stored in tables called quantization tables. There are two quantization tables for baseline JPEG standards one for Luminance (Y) and other for Chrominance components (Cb and Cr). The optimum values of the components in quantization tables are dependent on the application, but the JPEG standard suggests typical tables that have a good efficiency for any application [9]. By quantization we can eliminate coefficients which are less perceptible to human eye.

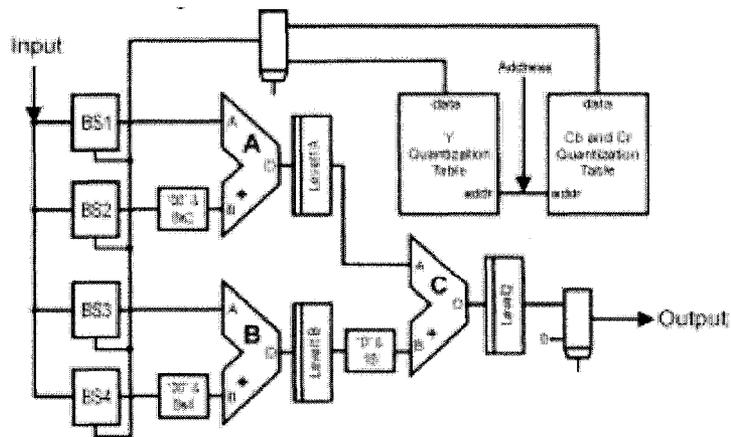


Figure 3.4 Quantization Architecture [9]

The quantization architecture designed as shown in fig. 3.5. The quantization architecture uses two ROMs for storing the quantization tables for Luminance and Chrominance components. For the multiplier we use barrel shifters controlled by quantization values stored in the ROMs. By using the barrel shifters for the multiplication we can reduce the number of clock cycles required for multiplication. For each array element of 8X8 blocks, there is a specific constant to be used from the quantization table for the division operation [9].

The quantization tables used for the JPEG compressions are presented in these are tables proposed by standard JPEG-92. The quantization tables used for compression and reconstruction are exactly same. Scaling factor is used to get the desired compression levels. The scaling factors after 2-d DCT are multiplied with quantization values and multiplied values are stored in ROM [9].

The Quantization operation is given by

$$C_{qij} = \text{round} \left(C_{ij} \times \frac{1}{Q_{ij} \times Fe_{ij}} \right) \quad 0 \leq i, j \leq 7$$

Where

- C_{qij} quantization coefficient
- C_{ij} Coefficient of 2-d DCT
- Q_{ij} quantization constant
- Fe_{ij} Scaling factor

The quantization values (Q_{ij}) and scaling factors (Fe_{ij}) are as given below:

$$Q_{ij} = \begin{pmatrix} 11 & 10 & 16 & 124 & 140 & 151 & 161 & 145 \\ 12 & 12 & 14 & 19 & 126 & 158 & 160 & 155 \\ 14 & 13 & 16 & 24 & 140 & 157 & 169 & 156 \\ 14 & 17 & 22 & 29 & 151 & 187 & 180 & 162 \\ 18 & 22 & 37 & 56 & 168 & 109 & 103 & 162 \\ 24 & 35 & 55 & 64 & 181 & 104 & 113 & 192 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 199 \end{pmatrix}$$

$$QC_{ij} = \begin{pmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{pmatrix}$$

$$Fe_{ij} = \begin{pmatrix} 8, 10 & 11,10 & 10, 45 & 9, 41 & 8, 00 & 6, 29 & 4, 33 & 2, 21 \\ 11, 10 & 15, 39 & 14, 50 & 13, 05 & 11, 10 & 8, 72 & 6, 01 & 3, 07 \\ 10, 45 & 14, 50 & 13, 66 & 12, 29 & 10, 45 & 8, 21 & 5, 66 & 2, 88 \\ 9, 41 & 13, 05 & 12, 29 & 11, 06 & 9, 41 & 7, 39 & 5, 09 & 2, 60 \\ 8, 00 & 11, 10 & 10, 45 & 9, 41 & 8, 00 & 6,29 & 4, 33 & 2, 21 \\ 6, 29 & 8, 72 & 8, 21 & 7, 39 & 6, 29 & 4, 94 & 3, 40 & 1, 73 \\ 4, 33 & 6, 01 & 5, 66 & 5, 09 & 4, 33 & 3, 40 & 2, 34 & 1, 20 \\ 2, 21 & 3, 07 & 2, 88 & 2, 60 & 2, 21 & 1, 73 & 1, 20 & 0, 61 \end{pmatrix}$$

The basic operation of quantizer is multiplying C_{ij} with $1/(Q_{ij} * Fe_{ij})$. In the quantization architecture it uses ROM memory for storing the controls of the four dislocated ones indicating the displacements that must be carried by each barrel shifter, instead of storing quantization matrix. Each barrel shifter uses three bit control for the displacement i.e. total of 12 bits. BS1 uses three most significant bits where as BS4 uses three least significant bits. The input to quantizer is 15 bit and output is 10 bits that means it reduces periodically the number of bits that represent the data. Quantization operation is carried in the pipeline of three stages. In the first clock cycle quantizer takes the input and corresponding calculates displacement and addition of shifted inputs form BS1, BS2,

BS3, BS4 are carried in A, B adders and in the next clock cycle adder C adds the result of adder A and B. In the next clock cycle we will get the output

Table 3.1 Barrel Shifter Controls

Control	Barrel Shifter			
	1	2	3	4
000	6	zero	zero	zero
001	7	9	10	11
010	8	10	11	12
011	9	11	12	13
100	10	12	13	14
101	11	13	14	15
110	X	14	15	15
111	X	15	15	15

In the quantizer architecture the control word are stored in column by column. So when we are reading the inputs for quantizer that is outputs of DCT we should read column by column.

3.2.4 Zig Zag Scanning

Quantized DCT coefficients will have zeros in the high frequency region of image block i.e. right bottom corner of the block. For getting more number of zeros at the same place, we scan the image block in such a way that all the zeros will accumulate at the end. Zig Zag scanning is very useful for further processing of the image. It maps 2-D 8X8 image to 1X64 1-D coefficients.

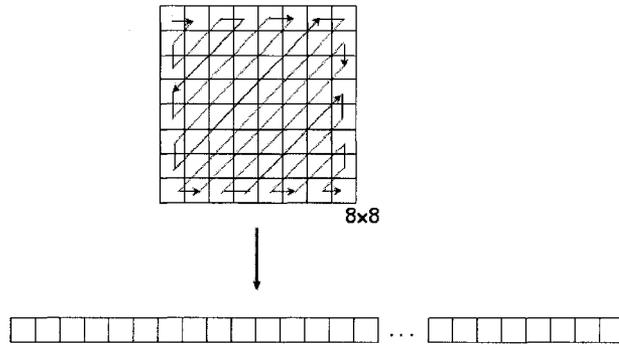


Figure 3.5 Zig Zag Scanning

3.2.5 Entropy Encoder

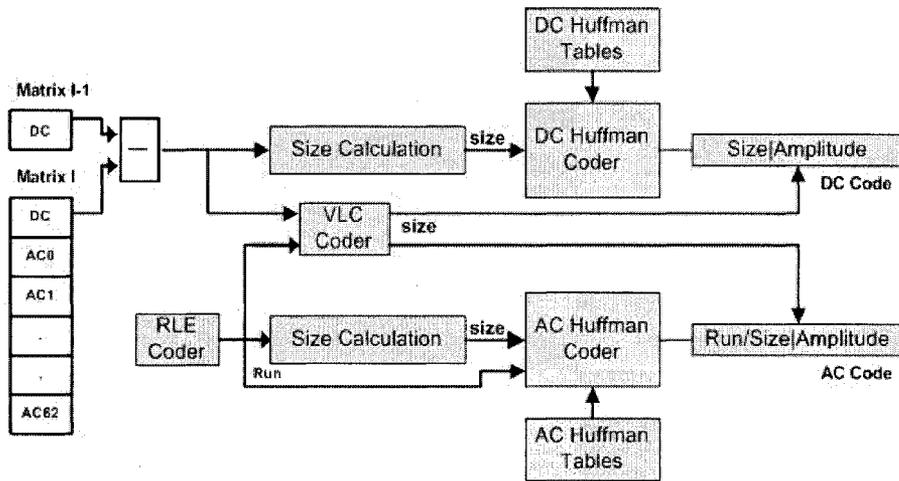


Figure 3.6 Entropy Encoder [7]

The last stage of JPEG Compression is entropy encoding. This block improves overall compression efficiency by performing lossless coding on the quantized DCT coefficients. The entropy encoder receives 10 bits input after quantization and gives output of 32 bit JPEG in an asynchronous way. The entrance of the architecture of the entropy encoder is synchronous and exit is asynchronous. This is mainly due to two

reasons, one is the output of the Run Length Encoder (RLE) is asynchronous and it is propagated throughout the architecture and other is due to different lengths of Huffman codes.

After quantization the resulting matrix will have large number of zeros which are read in Zig Zag manner to increase the sequence of zeros. Entropy encoder uses differential coding, Run Length Encoding (RLE), Variable Length Coding (VLC) and Huffman encoding to make the reduction in the number of bits used to represent the image after JPEG Compression [6,9,10]. Color and gray scale image follow same step for entropy coding, but the differential coder and Huffman coder are different for color and gray scale images. In entropy coding DC and AC components are processed separately. The component in the position $\langle 0, 0 \rangle$ (first line first column) of the 8×8 matrix is called DC Component and remaining 63 components are AC Components. The first operation is Differential coder for DC components and Run Length Coder (RLE) for the AC components.

The DC Components of successive 8×8 windows in an image are highly correlated. So by differential encoding we will only take the difference between actual DC Component and previous DC Component of the previous matrix. The differential code is coded by VLC coder and it is also used to calculate number of significant bits that are generated by VLC coder. This is done by Size calculator. So, by VLC encoder we will get the amplitude and by Size calculator we will get the sizes which are given as inputs to Huffman coder. Huffman coder uses Huffman tables which are stored in ROMs (one for Luminance and one for Chrominance DC Components) to get the outputs. The values

generated by Huffman coder and VLC coder are concatenated to get the JPEG DC code [7].

For Processing AC components, first step is counting the number of zeros before non zero coefficient which is will be done by Run Length Encoder (RLE). The RLE encoder compresses an input stream by representing consecutive zeros by their run-length. The Run Length Encoder counts number of zeros until the last zero is present or it reaches maximum zero count. So the output of Run Length Encoder is [Run Length and Amplitude]. The non-zero values are passed through VLC encoder to get the amplitude and also it is given to Size Calculator to calculate the size. Both Run and Size are concatenated and are Huffman Coded. Huffman encoder takes codes from predefined Huffman tables which are stored in FPGA internal ROMs. VLC coder amplitude, Coefficient Size, Huffman Code and Huffman Size are given to Pre assembler which will concatenate the VLC amplitude and Huffman Size to give variable length code which is applied to Assembler for further processing. The number of significant bits in the amplitude of Preassembler is given by addition of Coefficient Size and Huffman Size. In the assembler stage amplitude is assembled into 32 bit words output Compressed JPEG bit stream.

The pipelined architecture of entropy encoder is given as

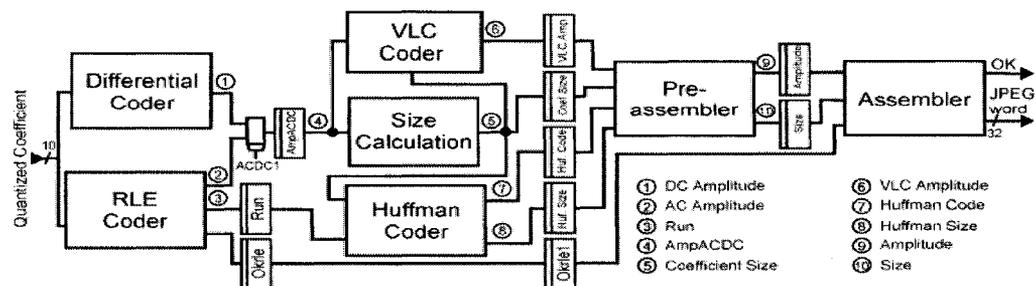


Figure 3.7 Pipelined Architecture for Entropy coder [20]

In the pipelined architecture intermediate registers are used for the synchronization of different operations.

3.2.5.1 Differential Coder

Differential coding is the first operation in the entropy encoding. It is used only for DC components. Differential coder performs simple subtraction between the current matrix DC component and previous matrix DC component of the same color elements. The result is called Amplitude DC.

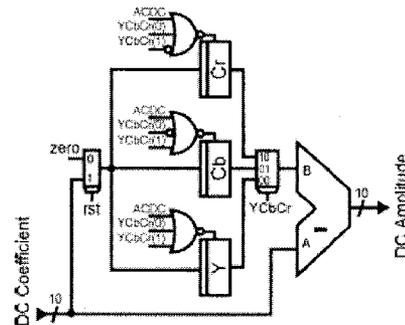


Figure 3.8 Differential Coder [20]

The Differential coder for color images is presented in above Fig 3.8. It consists of three 10 bit registers for storing the previous DC Coefficients of each color matrix (Y,Cb,Cr) and one 10 bit adder for performing the subtraction operation. The writing of the register is done when the ACDC signal is active high that indicates the matrix value is DC. By using YCbCr signal we can select which register we need to write. We will use multiplexer signal YCbCr to select exact adder input form Luminance or Chrominance registers. When rst signal is low the DC coefficient is writes the input into the register according to the signal ACDC and YCbCr and the same input is given to Adder. The 10

The AmpACDC and Run, Okrle registers presented in architecture are same registers which are used in the global Entropy coder architecture for pipelining. The output of RLE coder is asynchronous where as input is synchronous. While Run length Encoder is counting number of zeros, Okrle signal goes low and there won't be any valid output. When the non-zero input occurs RLE coder stops counting the zeros and it updates the outputs with new pair of Run and AC amplitude. RLE architecture has a flag (Okrle) to indicate when new valid outputs are available. There are two restrictions in RLE coder that are imposed by JPEG standards [20].

The first restriction is maximum value of Run should be 15 then zero counter has 4 bits. When there are more than 16 zeros in the sequence, the zero counter will be restarted and the output is sent has 15/0 Run/Amplitude pair, which indicates there are 15 continuous zeros followed by zero. Counter used in the RLE coder controls this restriction. When it reaches 15 zeros followed by 0 it will automatically give outputs 15 for Run and zero for AmpACDC. The second restriction is when the input is sent which is last input and it is Zero than Last signal will come into picture and it controls the outputs by sending this bit as the last bit. If the value is zero than it will reset the output register forming the pair 0/0. In the normal operation when these two restrictions doesn't occur the counter operation counts the number zeros and when it is counting, Okrle signal goes low indicating output is not ready. When the non-zero input occurs Okrle signal goes high indicating valid output and ACC signal gives number of zeros to Run register and amplitude is sent to AmpACDC register [9].

Differential coder and RLE coder must operate in perfect synchronism, so that they can be used by other components in the Entropy coder at the same pipeline stages. The

DC and AC amplitudes generated by Differential and RLE coder pass through the multiplexer controlled by ACDC signal to get the correct output to be used in the rest of the architecture.

3.2.5.3 Size Calculator

DC and AC amplitudes are applied to Size calculator that indicates number of significant bits of the AmpACDC value. The size calculation is done by looking at the tables proposed by JPEG standards [6].

The size calculation table is given as

Table 3.3 Size Calculation table

Value	Size
0	0(0000)
-1,1	1(0001)
-3,-2,2,3	2(0010)
-7...-4,4...7	3(0011)
-15....-8,8....15	4(0100)
-31....-16,16...31	5(0101)
-63...-32,32...63	6(0110)
-127...-64,64....127	7(0111)
-255...-128,128...255	8(1000)
-511...-256,256....511	9(1001)
-1023..512,512.....1023	10(1010)

From the above table we can generate coefficient size of 4 bits which is used to control the VLC coder architecture and it is also given as input to the Huffman coder and pre assembler architectures. Amplitude form Differential coder or Run Length Encoder is given as input to Size Calculator to find out the Coefficient Size.

3.4.5.4 Variable Length Coder:

Variable Length Coder (VLC) is used to identify which bits among the 10 bits AmpACDC are significant with the objective to discard the not significant bits, including the sign bit. The negative number must be represented in one's compliment to be VLC coded. The entrance of the VLC coder has a controller to discard sign bit. The signal interpretation is also inverted: a number that starts with zero is negative and a number that starts with one is positive.

The number of shifts to left to each Coefficient Size value is given as.

Table 3.4 VLC Architecture Shifts

Coefficient Size	Number of shifts to left
0	10
1	9
2	8
3	7
4	6
5	5
6	4
7	3
8	2
9	1

VLC encoder uses Barrel shifter controlled by Coefficient Size which is calculated from size calculator. This barrel shifter shifts the AmpACDC value to the left to put the

first significant bit as most significant bit of the word discarding sign bit. The calculated amplitude is called VLC Amplitude. The output of the VLC coder is 9 bit which is not really variable length. The assembler in the next stage will discard the not significant bits and generates Variable length Codes.

3.2.5.5 Huffman Encoder:

The Coefficient size (to DC Coefficients) and concatenation of Coefficient Size and Run (to AC coefficients) are Huffman coded. The architecture proposed below uses static Huffman tables proposed by JPEG 92 standards. In the Huffman coding, the compression is achieved by assigning short code words to input symbols of high probability and long code words to low probability input symbols. For a given source-probability distribution Huffman coder gives optimum symbols to represent the data [7]. The use of standard tables simplifies hardware but decreases the compression rate [7]. Huffman Coder architecture designed for color images is given below.

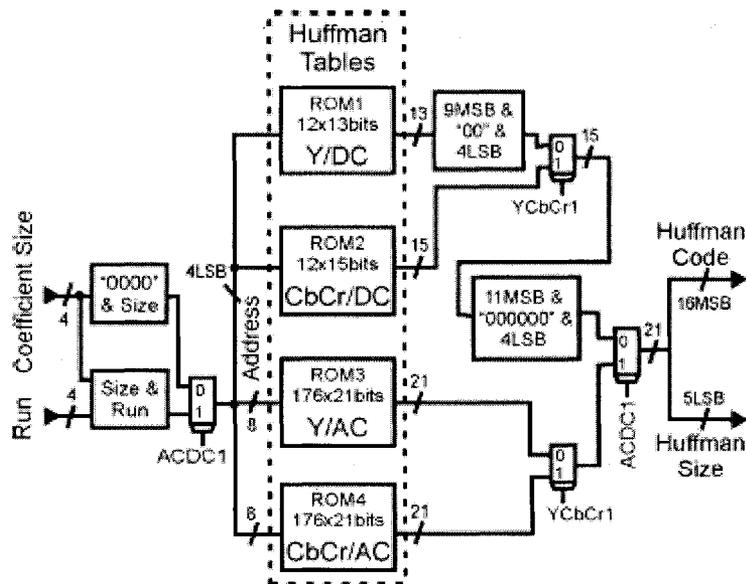


Figure 3.10 Huffman Coder Architecture [20]

The architecture presented above uses four ROM memories for storing the Huffman tables used to code color images: one for DC Luminance, one for AC Luminance, One for DC Chrominance and one for AC Luminance components. The Huffman tables store Huffman code and Huffman size. The size of Huffman codes can be calculated by Size calculator but we know the sizes of Huffman codes so we can directly store the Huffman size into static tables which eliminates delay. So the output of ROM memories is Huffman code followed by Huffman Size. The values to be Huffman coded are used like address to these memories. The number of words and bit width used to represent Huffman codes were optimized. The DC tables use 12 memory positions with 4 address bits (Size). AC tables uses 176 memory positions with 8 bit address bits (Run & Size). DC Luminance table 9 bits to Huffman codes and 4 bits for Huffman size. DC Chrominance table uses 11 bits for Huffman codes and 4 bits for Huffman tables. AC Luminance and Chrominance components uses 16 bits for Huffman Code and 5 bit Huffman Size. Two multiplexer are used to which of the four memories should be connected to the output. In the two multiplexers YCbCr signal is used as a controller to process either Luminance(Y) or Chrominance (CbCr). ACDC signal is used to get the DC or AC component as the output. The Huffman code and Huffman Size are applied to Pre-assembler for further processing.

3.4.5.6 Preassembler

The Pre-assembler architecture receives four inputs generated form the previously explained blocks: VLC Amplitude from VLC coder, Coefficient Size from Size Calculator and Huffman Size, Huffman code from Huffman Coder and generates two outputs Amplitude and Size which will be used in Assembler architecture.

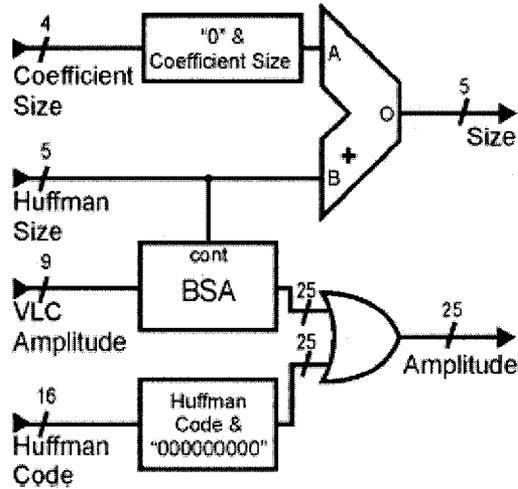


Figure 3.11 Preassembler Architecture [20]

VLC Amplitude bits are shifted to right by Barrel Shifter (BSA) that is controlled by Huffman Size. These shifted codes are assembled with Huffman Code by ‘OR’ logic operation. The Huffman code is concatenated with zeros in the right which are used as a mask in performing the ‘OR’ logic operation with VLC amplitude. The ‘OR’ logic operation preserves only significant bits which makes the code variable length. Assembly of the Huffman code and VLC code generates Pre-assembler output Amplitude of 28 bits. The addition of Huffman Size and Coefficient Size gives the number of significant bits in the Amplitude output which represented by Size.

3.4.5.7 Assembler

The final assembling of words in JPEG is carried through the Assembler architecture considering only significant bits of the Amplitude input. The Size input generated from the Pre-assembler indicates how many bits are significant among the 25 Amplitude bits [7].

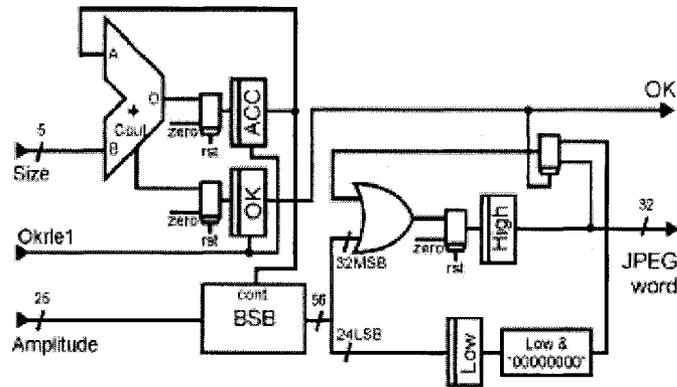


Figure 3.12 Assembler Architecture [20]

The Assembler architecture consists of one Barrel Shifter (BSB) controlled by accumulation of Size values and an 'OR' logic operation to assemble the significant bits of different inputs. The assembly of the words is controlled by an adder which accumulates different sizes of input Amplitudes and stores into the register ACC.

The Assembler uses two 32 bit registers to assemble jpeg words. The High registers store the 32 most significant bits from the Barrel shifter (BSB) output. When it records the 32 bits it will send the word as output JPEG word and OK register sends the output is valid. The Low register is used to store Overflow bits when the generated values from the Barrel shifter (BSB) as more than 32 significant bits. This overflow bits are again sent to High register when new jpeg word is ready to assemble. The maximum size of Amplitude input is 25 bits and the biggest displacement possible by Barrel shifter is 31 bits, so the output of Barrel shifter should be 56 bits. Of these 56 bits, 32 bits most significant bits are used in the 'OR' logic operation whose result will be stored in High register and the remaining 26 bits are stored in the Low register. 'OK' register indicates new valid data is ready. OK signal is also act as control signal for the multiplexer which decides input for the 'OR' logic operation. If the new jpeg word is ready it takes the values from the Low

register otherwise it allows High register as input to the jpeg word. The assembler operation is enabled only when the RLE coder generates valid outputs. This is controlled by Okrele1 signal. Okrle1 signal is generated when the valid output is present at the RLE coder with one delay clock cycle. In this way we can eliminate the effect asynchronous results given by RLE coder [20].

CHAPTER-4

RESULTS AND DISCUSSIONS

4.1 Simulation Waveforms

4.1.1 Color Conversion

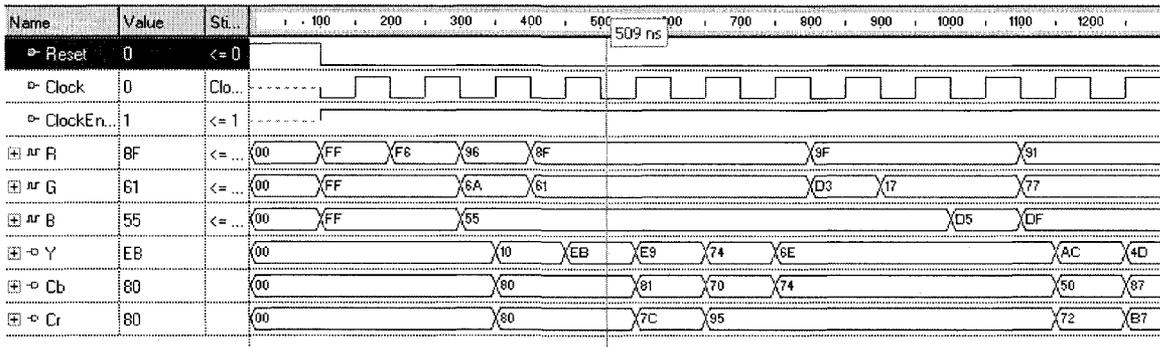


Fig 4.1 Simulation of Color Conversion

The color conversion module inputs are R, G, B, clk, clken and outputs are Y, Cb, Cr
 For the inputs R = FF, G = FF, B = FF Outputs are Y = EB, Cb = 80, Cr = 80.
 Latency of color conversion module is five clock cycles.

4.1.2 Discrete Cosine Transform (DCT)

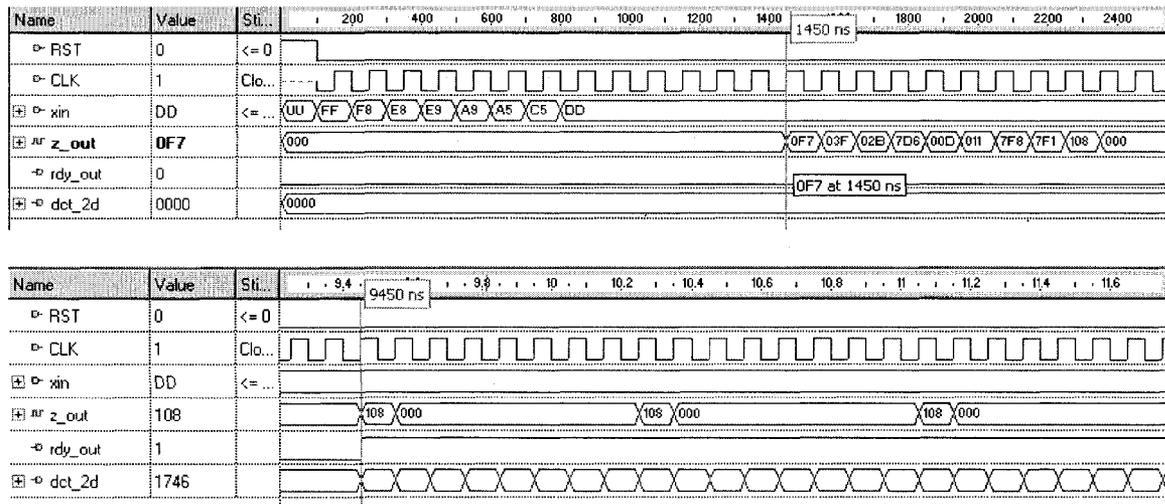


Fig 4.2 Simulation of DCT

DCT input matrix is

$$X_{in} = \begin{pmatrix} FF & F8 & E8 & E9 & A9 & A5 & C5 & DD \\ DD & DD \\ DD & DD \\ DD & DD \\ DD & DD \\ DD & DD \\ DD & DD \\ DD & DD \end{pmatrix}$$

2-D DCT coefficients are obtained by calculating 1-D DCT on rows first and then on the columns.

First 1-D DCT Output we get on the 14th clock cycle and these outputs are stored in a transpose buffer. Second DCT is applied on the outputs of the transpose buffer.

We get 2-D DCT coefficients starts from 95th clock cycle DC coefficient is 1746 and after that for every clock cycle we get other AC coefficients.

4.1.3 Quantizer

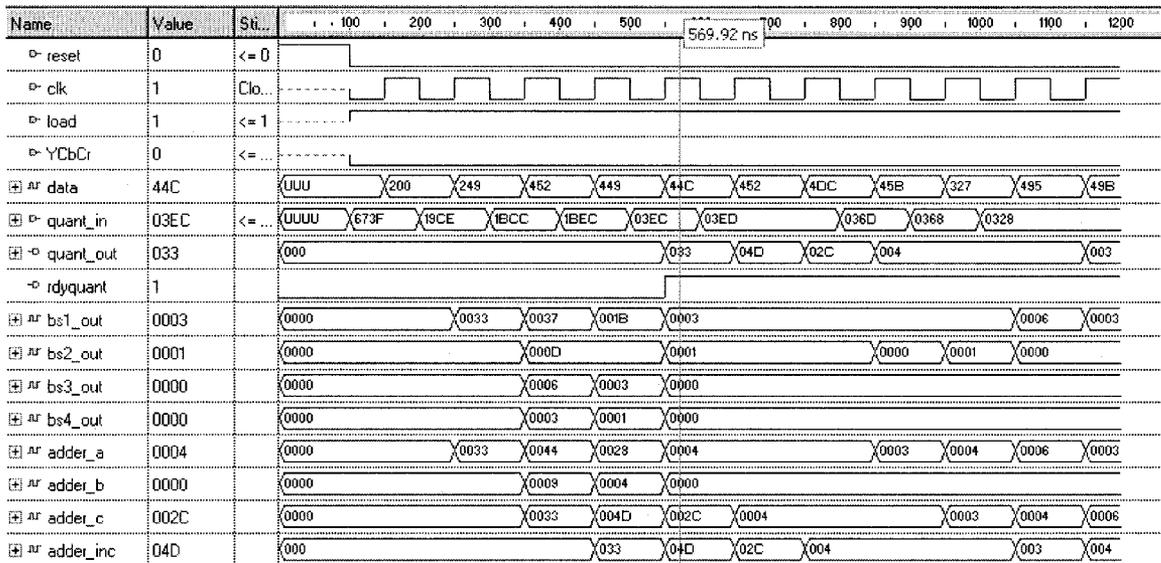


Fig 4.3 Simulation of Quantizer

Quantizer divides input DCT coefficients with predefined number to represent data with less number of bits.

Quantizer inputs are YCbCr, load(rdy signal form DCT), Quantin and Outputs are Quant_out and rdyquant.

YCbCr selects the predefined value form luminance or Chrominance component ROMs.

For the inputs quant_in = 673F (110011100111111), YCbCr = 0 (Luminance)

The first predefined value data = 200(100000000000).

Output is quant_out = 033(0000110011).

For the inputs quant_in = 673F (0001100111001110), YCbCr = 0 (Luminance)

The first predefined value data = 249(001001001001).

Output is quant_out = 033(0001001101)

4.1.4 Zig Zag Scanning

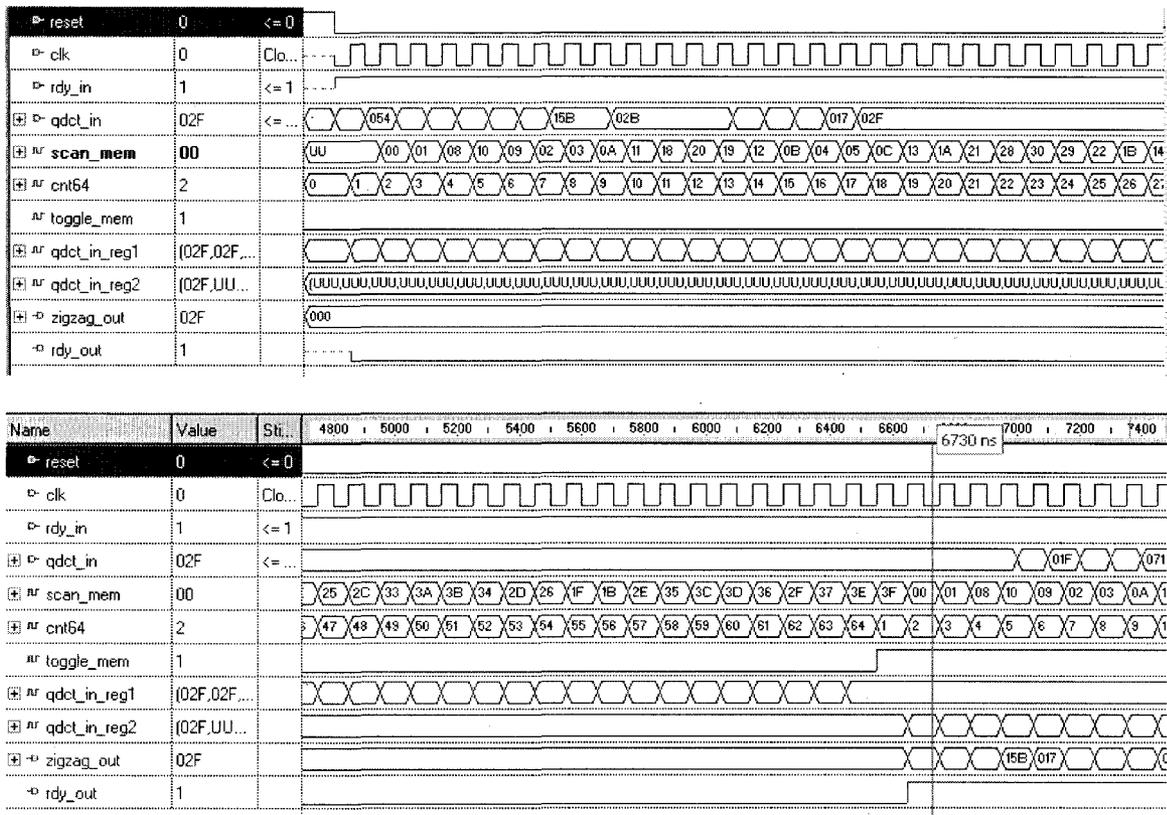


Fig 4.4 Simulation of Zig Zag Scanner

In the Zig Zag Scanning inputs are clk, rdy_in, qdot_in and Outputs are zigzag_out, rdy_out.

When quantization output is ready qdot_in goes high and Zig Zag Scanner starts loading quantized DCT coefficients into ROM1 and it continues for 64 clock cycle. After 64 clock cycle Zig Zag scanner gives output in the Zig Zag manner by reading inputs from ROM1 using scan_mem.

For input addresses 00,01,02,03 ... so on. Zig Zag Output addresses are 00, 01, 08,...so on.

4.1.5 Differential coder

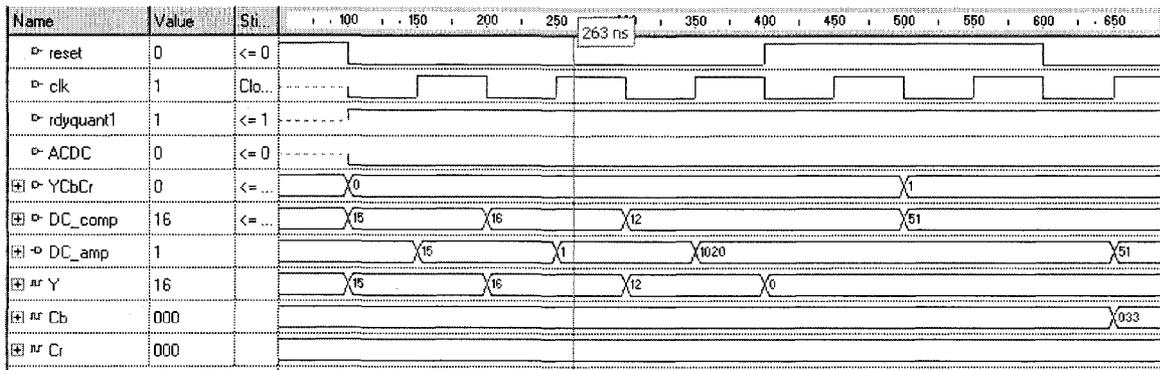


Fig 4.5 Simulation of Differential Coder

The Differential coder is used for taking the difference of current and previous DC components.

Here Inputs are clk, reset, YCbCr, ACDC, rdyquant1 and DC_comp are inputs.

Dc_amp is the output.

When rdy_quant1 is high and ACDC is low Differential coder starts processing.

If YCbCr is '00' the output is Y and "01" the output is Cb and "10" output is Cr.

Latency of Differential coder is zero.

4.1.6 Run Length Encoder

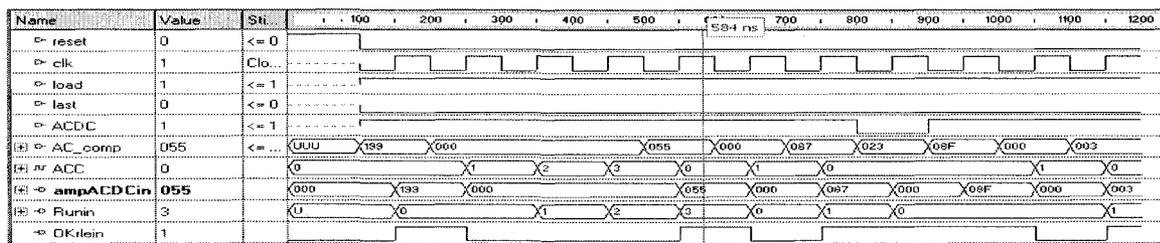


Fig 4.6 Simulation of Run Length Encoder

RLE coder counts number of zeros preceding non-zero input.

RLE coder activates for when ACDC signal is high.

Here the input sequence is AC_comp = 199, 0,0,0,55,0,87,23, 8F, 0, 3.

Output (Runin, ampACDCin) is [(0,199), (3, 55), (1, 87), (0, 23), (0, 8F), (0, 3)].

OKrlein is goes high when non-zero output occurs. When OKrlein is high the outputs are valid.

The latency of RLE coder depends on the occurrence of non-zero input.

4.1.7 Size Calculator

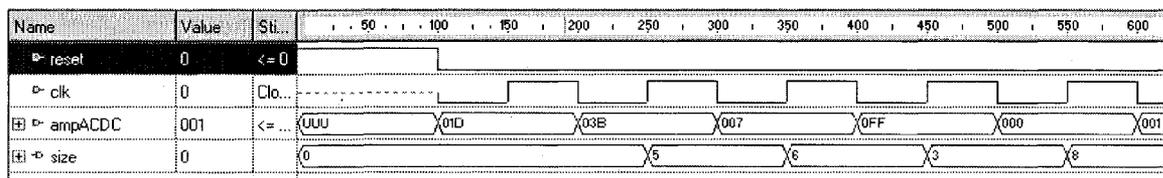


Fig 4.7 Simulation of Size Calculator

The Size Calculator calculates size of the input amplitude i.e. number of significant bits in the signal.

Input is ampACDC from Differential coder or RLE coder.

For input 01D (0000011101) Output size = 5

03B (0000111011) Output size = 6

007 (0000000111) Output size = 3

OFF (0011111111) Output size = 8

000 (0000000000) Output size = 0

The Latency of Size Calculator is one clock cycle.

4.1.8 VLC Coder

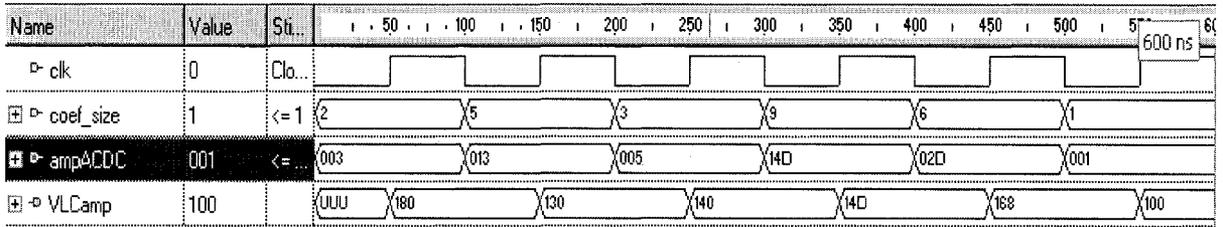


Fig 4.8 Simulation of VLC Coder

VLC coder identifies the significant bits and shifts the significant bits to most significant bits. Inputs to the VLC coder are coef_size and ampACDC. Coef_size inputs comes from Size calculator process which gives the number of significant bits. ampACDC input is form Differential coder or RLE coder.

If input is (10, 0000000011) output is 1100000000.

(101, 0000010011) output is 1001100000.

(1001, 101001101) output is 1010011010.

Latency of VLC coder is zero.

4.1.9 Huffman Coder

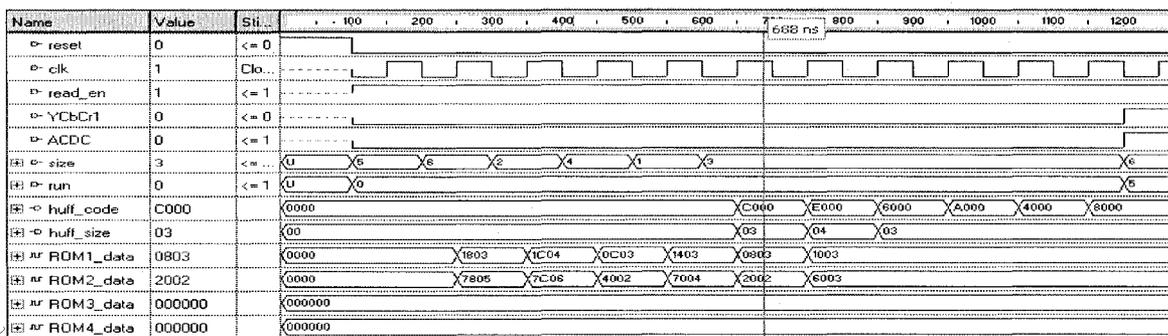


Fig 4.9 Simulation of Huffman Coder for DC Components

For DC components

Huffman coder inputs are size (Size calculator), run (from RLE coder) and outputs are huff_code(Huffman code), huff_size (Huffman size).

For the DC components (ACDC=0) Output is from ROM1 or Rom2.

YCbCr signal selects the Output should be form Rom1 or Rom2.

For the inputs size= 5(0101) run =0 (0000) ACDC=0, YCbCr=0.

Outputs are Huffman code = C000 (1100000000000000) and Huffman size= 03 (00011).

For the inputs size= 6(0111) run =0 (0000) ACDC=0, YCbCr=0.

Outputs are Huffman code = E000 (1110000000000000) and Huffman size= 03 (00100).

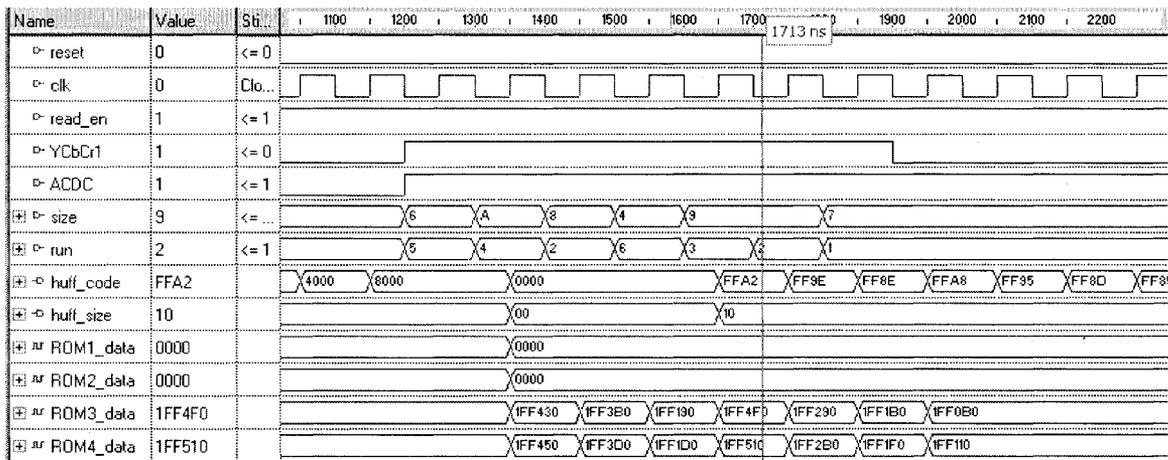


Fig 4.10 Simulation of Huffman Coder for AC Components

For AC components (ACDC=1)

For AC components Outputs are taken form ROM3 or ROM4.

YCbCr signal selects the Output should be form Rom3 or Rom4

For inputs size= 6(0110), run = 5 (0101), ACDC = 1, YCbCr = 1.

Output is taken from ROM4.

Outputs are Huffman code = FFA2 (1111111110100010) and Huffman size = 10(10000).

For inputs size= A(1010), run = 4 (0100), ACDC = 1, YCbCr = 1.

Outputs are Huffman code = FF9E (1111111110011110) and Huffman size = 10(10000).

Latency Huffman coder is six clock cycles.

4.1.10 Preassembler

Name	Value	St...	20	40	60	80	100	120	140	160	180	200	220	240	260	300 ns
coef_size	2	<=	5	4	2											
VLCamp	180	<=	150	180	180											
huff_size	07	<...	06	02	07											
huff_code	3780	<=	FC00	3000	3780											
ar t1	02		05	04	02											
ar t2	0030000		0054000	0580000	0030000											
ar t3	06F0000		F80000	0600000	06F0000											
bs_in	C0000000		A8000000	B0000000	C0000000											
bs_out	01800000		02A00000	2C000000	01800000											
size	09		0B	06	09											
amplitude	06F0000		1FD4000	0780000	06F0000											

Fig 4.11 Simulation of Preassembler

Pre assembler concatenates the significant bits of Huffman code and VLC amplitude.

Inputs for Preassembler are coef_size (Size calculator), VLCamp (VLC coder), huff_size, huff_code(Huffman coder).

For the inputs coef_size = 5 (0101), VLCamp = 150 (101010000), huff_size = 6 (00110), huff_code = FC00 (11111100000000000)

Outputs are size = b (01011) amplitude = 1FD4 (11111110101000000000000000000000).

4.1.11 Assembler

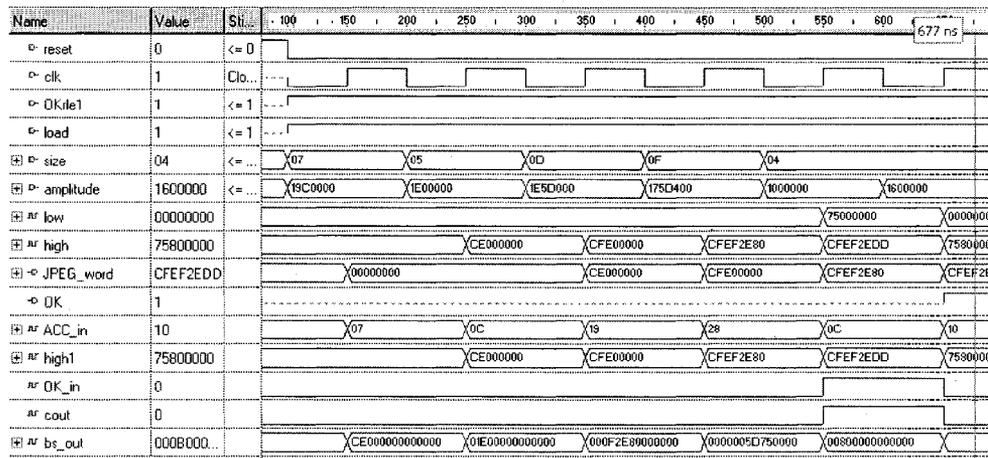


Fig 4.12 Simulation of Assembler

Assembler assembles the amplitude inputs to 32 bit words and sends output as JPEG bit stream.

Inputs to the assembler are size and amplitude (from Preassembler) and Okrle1 (from RLE coder) and Output is JPEG_word and OK.

For the inputs (amplitude, size) = (07, 19C0000), (05, 1E00000), (0D, 1E6D000), (0F, 175D400), (04, 1000000), (04, 1600000)

The Output JPEG word is CFEF2ED0.

Acc_in accumulates all the sizes and when the Acc_in is 32 bits are more than that OK signal goes high and JPEG word is the valid output.

4.1.12 JPEG Encoder

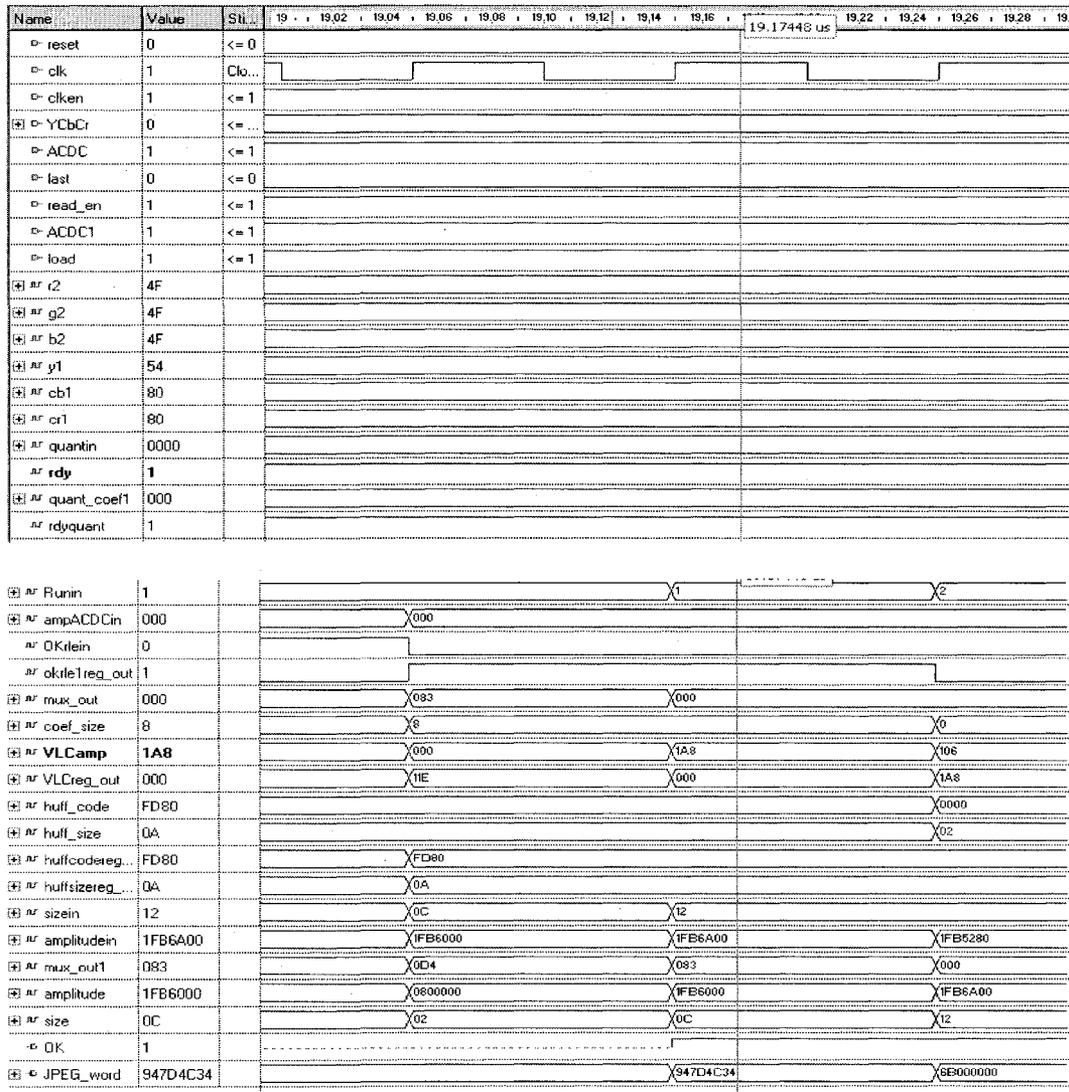


Fig 4.13 Simulation of JPEG Encoder

Encoder module reads inputs from a text file and these inputs are applied to Color Conversion module. At the end of 6th clock cycle Color conversion module gives Output which is applied to DCT module. After performing 2-D DCT operation, we get the Output at the end of 100th clock cycle which is applied to Quantizer module. Quantized

DCT coefficients are stored in a ROM memory of Zig Zag Coder. After storing all 64 coefficients Zig Zag scanner sends Output in Zig Zag manner (172th clock cycle) and these outputs are applied to Differential Coder, Run Length Coder, size calculation, VLC Coder, Huffman Coder, Preassembler and Assembler modules. The JPEG word is obtained at the end of 192th clock cycle.

4.2 Synthesis Report

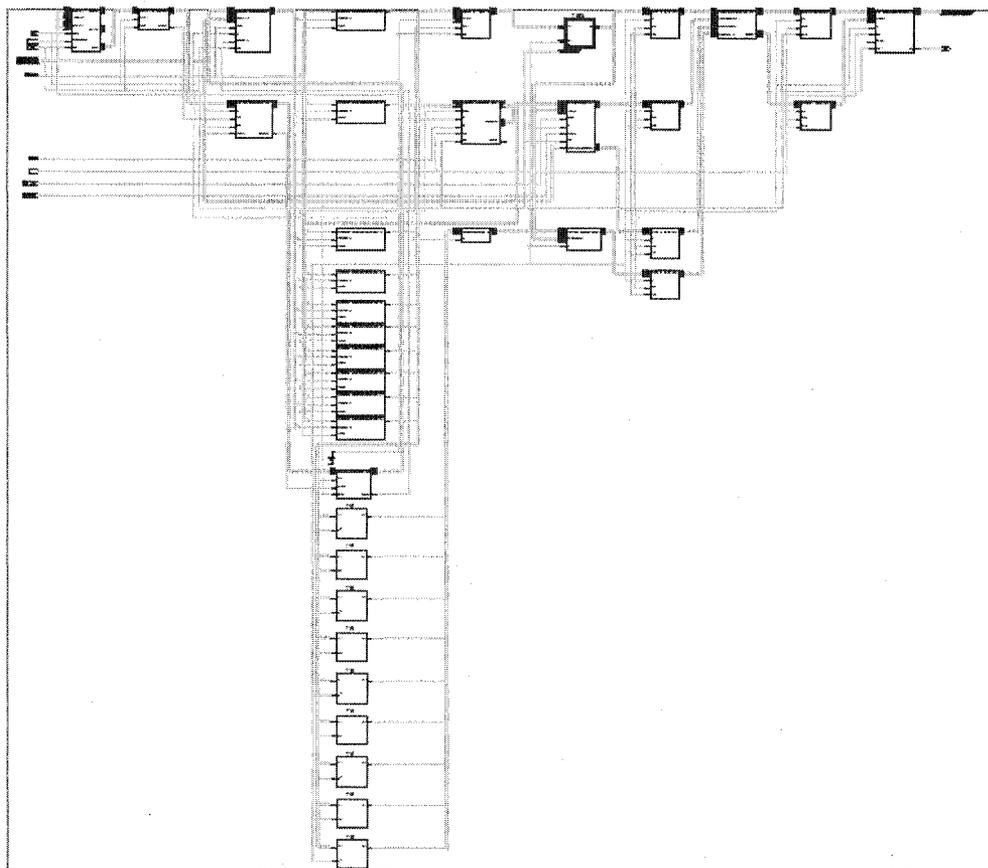


Fig 4.14 RTL Schematic of JPEG Encoder

Design Summary

Synthesis Report for Xilinx Virtex 4 FPGA

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Total Number Slice Registers	2,692	10,944	24%	
Number used as Flip Flops	2,668			
Number used as Latches	24			
Number of 4 input LUTs	2,592	10,944	23%	
Logic Distribution				
Number of occupied Slices	2,346	5,472	42%	
Number of Slices containing only related logic	2,346	2,346	100%	
Number of Slices containing unrelated logic	0	2,346	0%	
Total Number of 4 input LUTs	2,708	10,944	24%	
Number used as logic	2,592			
Number used as a route-thru	67			
Number used as Shift registers	49			
Number of bonded IOBs	43	320	13%	
Number of BUFG/BUFGCTRLs	2	32	6%	
Number used as BUFGs	2			
Number used as BUFGCTRLs	0			
Number of FIFO16/RAMB16s	3	36	8%	
Number used as FIFO16s	0			
Number used as RAMB16s	3			

Fig 4.15 Design Summary of JPEG for Vertex-4 FPGA

Timing Report

Speed Grade: -11

Minimum period: 7.445ns (Maximum Frequency: 134.318MHz)

Minimum input arrival time before clock: 7.703ns

Maximum output required time after clock: 4.221ns

CHAPTER 5

CONCLUSIONS

This thesis presents pipelined implementation of 'Baseline JPEG image compression architecture' on color images. The optimized architectures for the modules such as DCT, quantization, differential coder, Run length encoding, Huffman encoding were explained and coded in VHDL (Hardware description language) using Active HDL simulator. From simulation results it has been observed that the architecture has a minimum latency of 187 clock cycles for an image of 8X8 pixels.

The VHDL code is synthesized using Xilinx ise 9.1 simulator. The architecture requires 2768 of logic blocks and frequency is 134.318 MHZ for Xilinx Virtex-4 FPGA.

BIBLIOGRAPHY

- [1] “Home site of the JPEG and JBIG committees”, www.JPEG.org.
- [2] “Lapped Transform via Time-Domain Pre- and Post-Filtering” by Trac D. Tran, Member, IEEE, Jie Liang, Student Member, IEEE, and Chengjie Tu, Student Member, IEEE.
- [3] <http://pagesperso-orange.fr/polyvalens/clemens/wavelets/wavelets.html#section1>
- [4] <http://www.thepolygoners.com/tutorials/dwavelet/DWTTut.html>
- [5] JPEG2000 Image Coding System Theory and Applications, by N Athanassios. Skodras Touradj Ebrahimi School of Science and Technology - Computer Science Ecole Polytechnique Federale de Lausanne – EPFL.
- [6] The International Telegraph and Telephone Consultative Committee (CCITT), “Information Technology – Digital Compression and Coding of Continuous-Tone Still Images – Requirements and Guidelines”. Rec. T.81, 1992.
- [7] “Image and video compression standards – Second Edition, Kluwer Academic Publishers, USA, 1999 by vasudev bhaskaran, Konstantinos Konstantinides .
- [8] J. Miano. Compressed Image File Formats – JPEG, PNG, GIF, XBM, BMP, Addison Wesley Longman Inc, USA, 1999.
- [9] L. Agostini, S. Bampi, “Integrated Digital Architecture for JPEG Image Compression,” European Conference on Circuit Theory and Design, Vol. III, pp. 181-184, 2001.

- [10] "JPEG Still Image Data Compression Standard", by W. Pennebaker and J. Mitchell. Van Nostrand Reinhold, USA, 1992.
- [11] "Packet Analyzer for JPEG2000 Code streams and its VHDL model" by Masayuki Kurosamt, Akemi IKEDAS, Khairul Munadi and Hiioshi Kiyatt, Department of Electrical Eng., Tokyo Metropolitan Univ., Japan
- [12] <http://mtg.upf.edu/~xserra/cursos/TDP/referencies/Park-DWT.pdf>
- [13] Kumar, C.S., "Comments on 'Subband coding of images'," Acoustics, Speech and Signal Processing, IEEE Transactions on , vol.36, no.7, pp.1089-1090, Jul 1988 [14]
- "Verilog HDL: A Guide to Digital Design and Synthesis", by Samir Palnitkar, SunSoft Press, Prentice Hall.
- [15] "Design of Architectures for JPEG Image Compression (portuguese). Master Dissertation by L. Agostini, Federal University of Rio Grande do Sul. Informatics Institute. Pos-Graduation in Computer Science Program, Porto Alegre, Brazil-
- [16] "JAGAR: A Fully Pipeline VLSI Architecture for JPEG Image Compression Standard", by M. Kovac and N. Ranganathan, Proceedings of the IEEE, vol. 83, 1995, pp. 247-258.
- [17] http://www.xilinx.com/support/documentation/application_notes/xapp930.pdf.
- [18] http://www.xilinx.com/support/documentation/application_notes/xapp610.pdf.
- [19] Lei, S.-M.; Sun, M.-T., "An entropy coding system for digital HDTV applications," Circuits and Systems for Video Technology, IEEE Transactions on , vol.1, no.1, pp.147-155, March 1991.

[20] Agostini, L.V.; Silva, I.S.; Bampi, S., "Pipelined entropy coders for JPEG compression," *Integrated Circuits and Systems Design, 2002. Proceedings. 15th Symposium on*, vol., no., pp. 203-208, 2002

VITA
Graduate College
University of Nevada, Las Vegas

Arun kumar reddy Toomu

Local Address:
4248 Grove Circle, apt 3
Las Vegas, NV-89119

Degree:
Bachelor of Technology in Electrical and Computer Engineering, 2006
JNT University, Hyderabad, India

Thesis title: Pipelined Implementation of JPEG Image Compression using VHDL

Thesis Examination Committee:
Chairperson, Dr. Henry Selvaraj, Ph.D.
Committee member, Dr. Emma Regentova, Ph.D.
Committee member, Dr. Yahia Baghzouz, Ph.D.
Graduate College Faculty Representative, Dr. Laxmi gewali, Ph.D.