International Journal of Progressive Sciences and Technologies (IJPSAT)
ISSN: 2509-0119.

http://ijpsat.ijsht-journals.org
Vol. 5 No. 1 June 2017, pp. 24-42

Design of Novel FFT Based Image Compression Algorithms and Architectures

Anitha T.G
Department of Electronics & Communication,
VTU-RRC, Belagavi,
Karnataka,India.

K. Vijayalakshmi
Department of Medical Electronics,
B.M.S College of Engineering,
Karnataka, India.

***Abstract* — Image Compression is a pivotal step in Digital Image Processing area. Image Fourier Transform is long established algorithm which converts an Image from spatial domain to frequency domain. New Algorithms for FFT, Quantization and their inverses have been developed using Algorithmic State Machine Charts. Subsequently, novel Architectures have been designed using massively parallel and highly pipelined circuits. These Architectures have been coded using Verilog Hardware Description language conforming to RTL coding guidelines, simulated and tested for color images. The quality of the reconstructed images is better than 35 dB. The reconstructed images are indistinguishable from the original images. The target clock rate is 50 MHz and compression achieved is greater than 15.**

Keywords — Compression; FFT; IFFT; Inverse Quantization; Quantization and Verilog RTL Coding.

I. INTRODUCTION

Compression plays an important role in multimedia applications, such as image storage and transmission. The primary goal of image compression is to represent an image with minimum number of bits without sacrificing on the reconstructed image quality. Image compression is a natural technology for handling the increased spatial resolutions of today's sensors and evolving broadcast Television standards. A popular and more efficient compression scheme is known by the generic name, Transform Coding. In Transform coding, a block of image pixels is linearly transformed into another block of coefficients of similar size in which only a few of them will be significant, and therefore, the rest may be discarded by quantization. Although the DFT transform has good energy compaction, due to its complex operations and the great amount of calculations involved, has not long been widely used in the image compression. In contrast to this, FFT has been very popular for image compression in order to reduce the data storage and transmission bandwidth as it is less complex and offers faster processing. The divide-and-conquer approach of the FFT algorithm makes FPGAs an ideal solution because of their unhindered potential for parallelization. It is amenable to efficient FFT computations on the FPGA.

In this paper, an algorithm for Fast Fourier transform data compression and decompression has been presented. Image compression is possible because images, in general, are highly coherent, which means that there is redundant and irrelevant information [1]. Most of the natural images have significant number of coefficients with less magnitude and can be discarded entirely with little image distortion.

Corresponding Author: Anitha T.G

Furthermore, image compression plays a major role in many important and diverse areas; including tele-video conferencing, remote sensing and medical image processing, facsimile transmission and the control of remotely piloted vehicles in military, space and hazardous waste management applications [2]. The raw digital image and video signal usually contain immense amount of information and therefore require a large channel or storage capacity. Despite advances in communication and storage capacity, the implementation cost often put damper on the storage capacity. Generally, the transmission or storage cost increase with increase in bandwidth requirements. To meet the channel or storage capacity need, it is imperative to make use of compression techniques that reduces the data rate while retaining the subjective quality of the decoded image signal. Image compression techniques achieve compression by exploiting statistical redundancies in the data and eliminating or reducing data to which the human eye is less sensitive. In this paper, we have considered compression techniques that are based on modifying the transform of an image. In transform coding, a reversible linear transform (Fourier Transform) is used to map the image into a set of transform coefficients, which are then quantized and coded [3]. Transform is basically a mathematical tool, which allows us to proceed from one domain to another domain to perform task in an easy manner. Transforms do not change the information content present in the signal but give the frequency content information in a signal.

The Transform closely packs the signal information into a small number of coefficients and hence maps from a higher dimensional space to lower dimensional space, facilitating compression by quantization and encoding.

Two dimensional signals have two independent variables and, Images are examples of 2D signals. The Discrete Fourier Transform pair of 2D is given by:

$$F(u,v)=\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} f(x,y)W_N^{-(ux+vy)} \qquad (1)$$

for x = 0, 1, 2….., N-1

for y = 0, 1, 2….., N-1

Inverse Fourier Transform is given by

$$f(x,y)=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} F(u,v)W_N^{(ux+vy)} \qquad (2)$$

for u = 0, 1, 2…., N-1

for v = 0, 1, 2…., N-1

These equations show that the computation complexities of DFT and IDFT increase with the increase of the N value and hence results in high computation time. But real time applications require Fast Fourier Transforms for processing and, there exists lots of algorithms to compute the FFT. The most common algorithm used is Cooley and Tukey algorithm based on recursion. This FFT was discovered in 1965 by Cooley and Tukey, which reduced the number of calculations drastically and paved the way for real time processing of discrete signals which revolutionized the field of Digital Signal Processing [4].

The FFT relies on redundancy in the calculation of the basic DFT. It is one type of recursive algorithm that repeatedly rearranges the problem into two simpler problems of half the size. Hence the basic algorithm operates on signals of length, a power of two [5]. The periodicity property and symmetry property of DFT to divide computation of DFT of length N into successively smaller DFTs.

The FFT reduces the number of complex multiplications to N/2 log2 N from N2 and the number of complex additions to N log2N from (N2-N). Thus there is large reduction in calculation making real time processing a reality. Various FFT algorithms provide various benefits, but there is always tradeoff between the speed and the chip area if implemented as a VLSI chip. In Radix algorithms, the value of ‘N’ is selected such that N=2v, where ‘v’ represents the number of stages of computation. This N-point DFT is decomposed successively such that the smallest DFT will be of size N=2. Hence this type of algorithm is called Radix 2 algorithm. Each decomposition stage doubles the number of separate DFTs, but halves the number of points in DFT. If the computational cost of multiplication is taken into consideration, Radix-2 brings integer twiddle factor on angles 0, 180°, and Radix-4 brings integer twiddle factor on angles 0, 90°, 180°, 270° [6].

In computing an N-point DFT, the decimation process can be repeated log2 =N times [7]. This Paper is organized as follows. In the following Section 2, the basic blocks of the proposed FFT Algorithm for Image Compression are described. It also highlights the actual design flow implemented in the present work. FFT Compression is

explained in Section 3 along with Quantization and Normalization comprised in JPEG algorithm. Section 4 presents the Development of Algorithm State Machine for FFT Quantization & Inverse Quantization of IFFT computation. Detailed Architecture for FFT and its Inverse Processor is unveiled in Section 5. Simulation waveforms and Results are presented in Section 6.The FPGA implementation of the design is allotted in the next section and Conclusions are presented in last section.

II. PROPOSED FFT ALGORITHM FOR IMAGE COMPRESSION

Twiddle Factor is a key component in FFT/IFFT computation and is represented by 'ω'. It is any of the trigonometric constant coefficients that are multiplied by the data in the course of the FFT algorithm. They are the coefficients used to combine results from a previous stage to form inputs to the next stage [8]. The twiddle factor illustrates a rotating vector that rotates in increments based on the number of samples. The same values of ω are repeated for different values of 'n' in computing DFT of a signal. The twiddle factor has redundant values as vector rotates encircle. It has symmetry also as values out of 180 degrees are negative of each other. The butterfly diagram takes the advantage of this symmetry and redundancy to make FFT possible. The butterfly diagram is built on the properties of the twiddle factor for fast and efficient computation of DFT. Basically, the great contribution of Fourier Transformation states that any function can be expressed as the integrals of sine and/or cosine multiplied by a weighting function [9].

The twiddle factor is expressed as

$$W_N = e^{-(j2\pi kn)/N} \tag{3}$$

The exponential term in the above equation can also be written using Euler's formula as

$$e^{-j\theta} = \cos\theta - j\sin\theta \tag{4}$$

Hence the twiddle factor can also be written as

$$e^{-(j2\pi kn)/N} = \cos\frac{(2\pi kn)}{N} - j\sin\frac{(2\pi kn)}{N} \tag{5}$$

The right hand side terms of the equation is expressed as (M x N) matrix for various values of (u, x) and (v, y) as cosine and sine matrices. These matrices and their transposes may be stored as lookup table in ROM and accessed for each block of image in a hardware implementation, especially in an FPGA or an ASIC. The FFT computation consists of complex arithmetic operations and twiddle factor irregularities pulling down the computation speed in hardware realization [10]. Hence the developed algorithm should be tractable for parallelism and pipelining, thus improving the speed of computation. In the proposed algorithm, the twiddle factor irregularities can be overcome by sine and cosine transforms. In this algorithm, the FFT is calculated by adding the sine, cosine transforms and their transposes with the image input signal. The cosine and sine transforms are obtained by varying the values of u and x and their transposes by varying v and y from 0 to 7.

The 2 D FFT algorithms can be obtained by cosine and sine transforms of the signal using the conventional expression:

$$F(u,v) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\{f(x,y)[\cos\frac{(2\pi(ux+vy))}{N} - j\sin\frac{(2\pi(ux+vy))}{N}]\} \tag{6}$$

for x = 0, 1, 2……., N-1
for y = 0, 1, 2……., N-1

Correspondingly 2-D IFFT can be obtained by adding the cosine and sine transforms of the transformed signal.

$$f(x,y) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\{F(u,v)[\cos\frac{(2\pi(ux+vy))}{N} + j\sin\frac{(2\pi(ux+vy))}{N}]\} \tag{7}$$

for u = 0, 1, 2……., N-1
for v = 0, 1, 2……., N-1

The FFT and IFFT algorithms have been developed by the present authors and reported in an earlier paper [11].The same has been made use of in the present work for developing ASM Chart based algorithm and detailed hardware architecture in Sections 4 and 5 respectively.

A. Design Flow for 2D FFT Algorithm

The design flow starts with the MATLAB surrounding. The image is segmented in to 8x8 pixel blocks and the pixel values determined are stored in RAM for 2D FFT computation. The spatial domain image pixels are converted into frequency domain coefficients by applying FFT. These FFT coefficients are compressed by quantization and encoding. To reconstruct the image the compressed data is decoded and IFFT applied. The architecture proposed in this work is based on parallel and pipeline approach with 2D FFT and IFFT computations. The core design is to accept 8

bits of data input for each of the 3 color components and to produce 32 bits of data output for each stage of FFT.

III. FFT DATA COMPRESSION

In order to thoroughly understand the proposed algorithm, a block system level description is presented in Fig. 3. The input is an image of block size 8x8 pixels and it is pre-processed prior to transformation. The aim of Pre-processing is to improve the image data by suppressing the unwanted distortions or to enhance some image features for further processing. The pre- processing involves primitive operations to reduce noise, contrast enhancement and Image sharpening. In pre-processing, the input is down sampled so that the sampling rate reduction can be done. Down sampling is also termed as Decimation and the motivation behind down sampling is to reduce the cost of processing. The computation and memory required to implement DSP system is analogous to sampling rate and hence the lower sampling rate results in cheaper implementation [12].The good nature of Fourier transform is that it has a wide range of applications in the signal encoding, segmentation, reconstruction and other areas. DFT transform has good energy concentration, and owing to inconvenient operations and the great amount of calculations, has not long been widely used in the image compression. In spite of its complex algorithm and time consuming disadvantages, FFT is used for image compression to reduce the data storage.

Compression is done to make optimal use of limited memory space, save time and also help to optimize resources. Compression is used in sending data over communication line to lessen transmission time and also storage to host. The efficiency of a compression method is related directly to the compression ratio which is given by the ratio of the amount of original data to that of the compressed data.

.

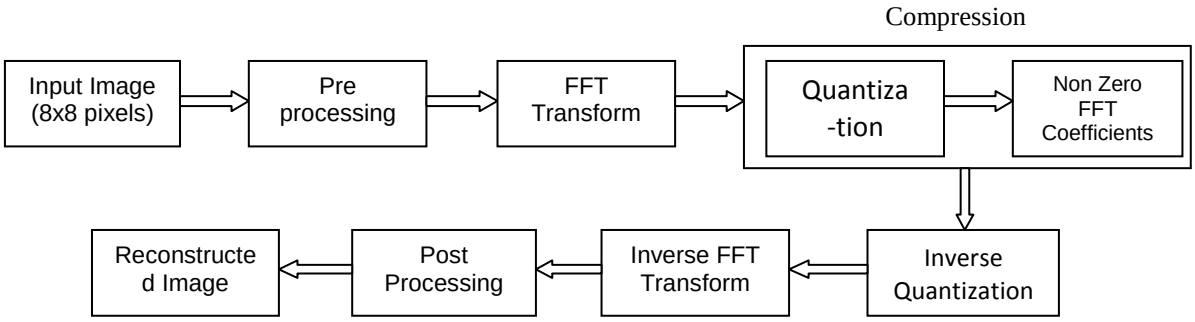

Fig.1 Block Diagram of the Proposed FFTQ-IQIFFT System algorithm

The energy of an image often varies significantly throughout the image, making its compression difficult in spatial domain. However, images have a compact representation in frequency domain gathered around low frequency area making compression more effective and efficient.

A color image consists of three primary colors, namely, red, green and blue and these three components have high correlations. The real time images are stored in RGB space and transmitting RGB color space images is not practical as that contain a lot of redundant values. Their high bandwidth requirement forbids the transmission and hence the images in RGB color space are converted in to other color spaces, for examples:

YUV, YIQ and YCbCr.The Y Cb Cr space is one such space, where Y is the luma, Cb is blue chroma and Cr is red chroma.

TABLE I

16	11	10	16	24	40	51	61
14	16	40	69	56	69	40	16
18	37	68	103	774	103	68	37
49	78	103	120	101	120	103	78
72	95	112	103	99	103	112	95
49	78	103	120	101	120	103	78
18	37	68	103	774	103	68	37
14	16	40	69	56	69	40	16

TABLE II

32	22	20	32	48	80	102	122
28	32	80	138	112	138	80	32
36	74	156	206	1548	206	136	74
78	156	206	240	202	240	206	156
154	190	224	206	198	206	224	190
78	156	206	240	202	240	206	156
36	74	136	206	1548	206	136	74
28	32	80	138	112	138	80	32

Using JPEG compression algorithm, after pre-processing, the image is accessed as sequential 8x8 blocks and 2D FFT is calculated individually for each block.The FFT coefficients are then quantized and many of these quantized values are insignificant that are nearly equal to zero [13]. Quantization makes JPEG algorithm to illustrate Lossy Compression. The quantization matrix values are designed in such a way that the elements near to zero will be converted to zero and other elements are shrunken so that they are close to zero. The quantization matrices used in this algorithm are shown in Table 1 and Table 2.After Quantization, the values are rounded off or normalized to the nearest integer and, the insignificant values can be easily discarded without affecting the quality of the reconstructed image. Hence the compression achieved in the Transform domain can be calculated as shown in Eq. 8.

$$\text{CompressionRatio} = \frac{M * N * 3 * 8}{Number\_of\_Nonzero\_FFT\_coefficients} \quad (8)$$

M: number of rows of the Original Image

N: number of columns of the Original Image

The image is reconstructed by applying Inverse Quantization followed by Inverse FFT to each block of FFTQ coefficients. In spite of loss of quality in the image, the reconstructed image is indistinguishable from the original, thus serving the desired purpose very efficiently. If the quality measure PSNR is above 35 dB, then the original and reconstructed images are indistinguishable.

IV. DEVELOPMENT OF ALGORITHMIC STATE MACHINE FOR FFTQ AND IQIFFT COMPUTATION

A novel Algorithm has been developed using Algorithmic State Machine (ASM) Charts for processing FFT for Image compression and is presented in Fig. 4 to Fig. 6. State 0 initializes all registers and outputs used in the design. State 1 starts reading the input image data from memory. The State 2 selects the 8x8 pixels block of image. State 3 initializes multiplicand and multiplier of finding the cosine and sine transforms. State 4 finds the cosine and sine transforms of the image block. The transpose of cosine and sine of transforms of the same image block is calculated in State 5. The FFT of image is computed in State 6 using the transforms and their transposes of input image. In State 7, the result is written on to fft memory. During State 8, the Quantization and Normalization of the fft image are performed and the State 9 extracts the Non Zero Coefficients of fft image.

The developed ASM chart for IQ IFFT for Image reconstruction is similar and is shown in Fig. 7 and Fig. 8. State 0 initializes all registers and outputs used in the design. State 1 reads the Quantized image data (FFTQ coefficients) from memory. State 2 selects 8x8 pixels block of image. State 3 initializes multiplicand and the multiplier of finding the cosine and sine transforms. State 4 reads the quantization table and performs the inverse quantization of the FFTQ coefficients of the image block by block. State 5 finds the cosine and sine transforms of the inverse quantized image block. State 6 finds the multiplication of image block with the Cosine and Sine Transpose matrices. State 7 performs subtraction of the resultant matrices to find the Inverse Fast Fourier Transform of the image. The State 8 writes the result to the memory. State 9 repeats the same operation for G and B matrices of the image matrix. State 10 concatenates the three color components R, G, B and writes the reconstructed image values to the memory so that the reconstructed image can be displayed.

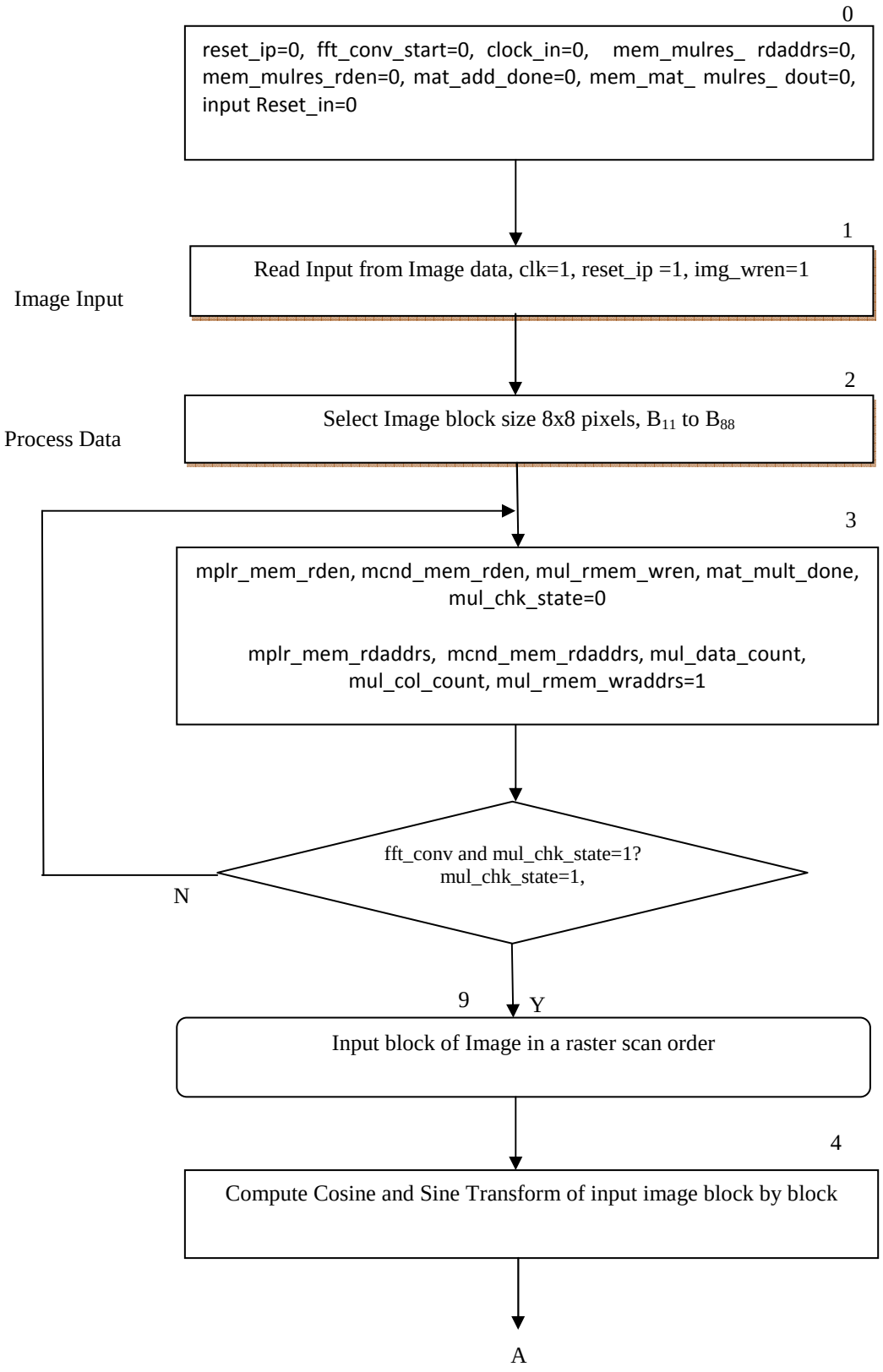

Fig. 2 ASM Chart for FFTQ Computation for Image Compression (Continued)

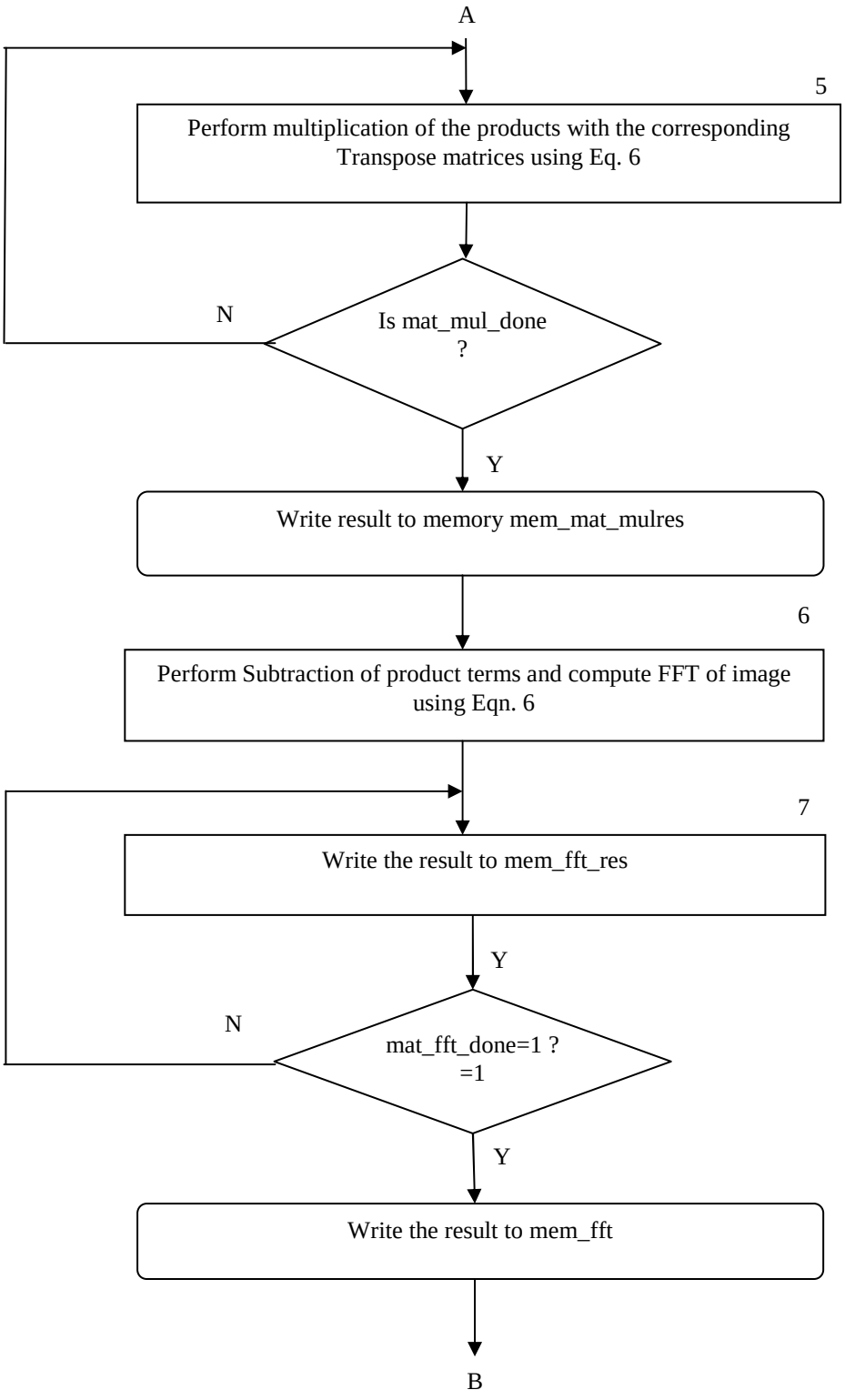

Fig.2 ASM Chart for the Computation of FFTQ for Image Compression (Continued)

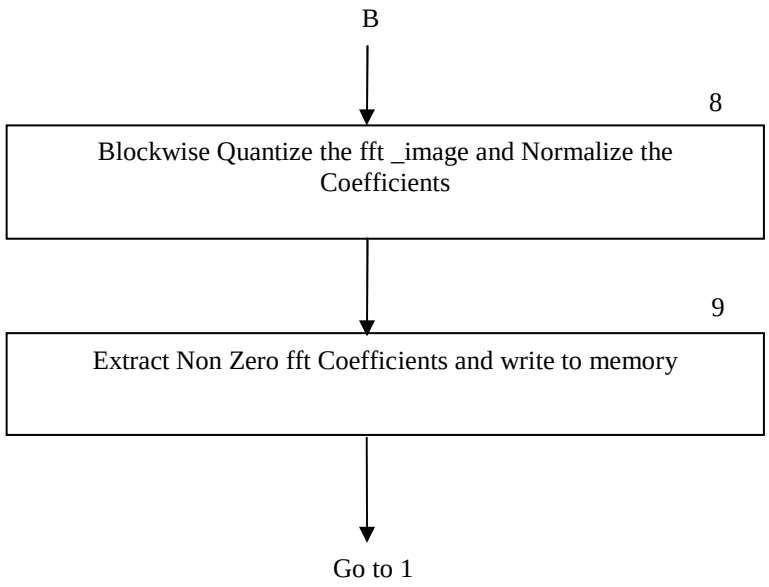

Fig.2 ASM Chart for the Computation of FFTQ for Image Compression

V. ARCHITECTURE OF THE PROPOSED FFT, QUANTIZATION AND THEIR INVERSE PROCESSORS

It may be noted that quantization does not really bring about the compression of an image but only prepares the ground for compression by succeeding in reducing the insignificant Quantized coefficients to zero. The real compression is brought about by the next stage of Encoding using Variable Length or Arithmetic Coder, which work is under progress currently. The Encoder codes only the Non-zero FFTQ Coefficients, paving way for effective compression. The schematic block diagrams of FFTQ and IQ IFFT Processors as realized in the present work are shown in Fig. 10 and Fig. 11 respectively. The FFTQ Processor determines the Fourier Transform of the input image using cosine and sine transform of the image. The input consists of 8*8 blocks formed by converting RGB image to YCbCr (4:2:0) standard format. This format has less number of pixels to be processed when compared to 4:2:2 formats, thus resulting in less processing time and more compression. Hence this format of image is stored in Image memory. The Cosine and Sine memory consists of cosine and sine look up tables used in finding the cosine and sine transform of the image that are determined by multiplying the image block with the cosine matrix and sine matrix respectively using Eq. 6.

The same procedure is repeated to find the cosine and sine transpose of the image of the FFT equation. The cosine and cosine transpose are fed to subtractor that outputs the partial output of FFT transform. Similarly the sine and sine transpose are fed to subtractor to find the remaining part of FFT. Thus, the FFT coefficients are computed, storing them in FFT memory. These FFT coefficients contain both significant and insignificant data and hence the insignificant coefficients have to be removed by Quantization and Normalization so that the memory required in the transform domain can be optimized [14]. The Normalization also contributes towards the compression of the image which is mainly brought about in the Encoder Processor. It converts all negligible values to zero and reduces the magnitude of the lower high frequencies while preserving the magnitude of lowest frequency components.

Thus, Normalization is done by dividing the each frequency component of transformed image by a known value called Quantizer [15]. Furthermore, the compression is accomplished by Variable Length Coding where the bitstream in representing a pixel of an image is minimized.

At the IQ IFFT Processor, the FFTQ Coefficients are inversely quantized using the quantization Tables 1 and 2 presented earlier. Then the Inverse Transform is computed analogously done in computing fft transform except an adder occupies the subtractor and fft coefficients of transformed image are fed as in to the Inverse FFT processor. The Inverse Transform exudes coefficients for the reconstructed image using Eq.7.

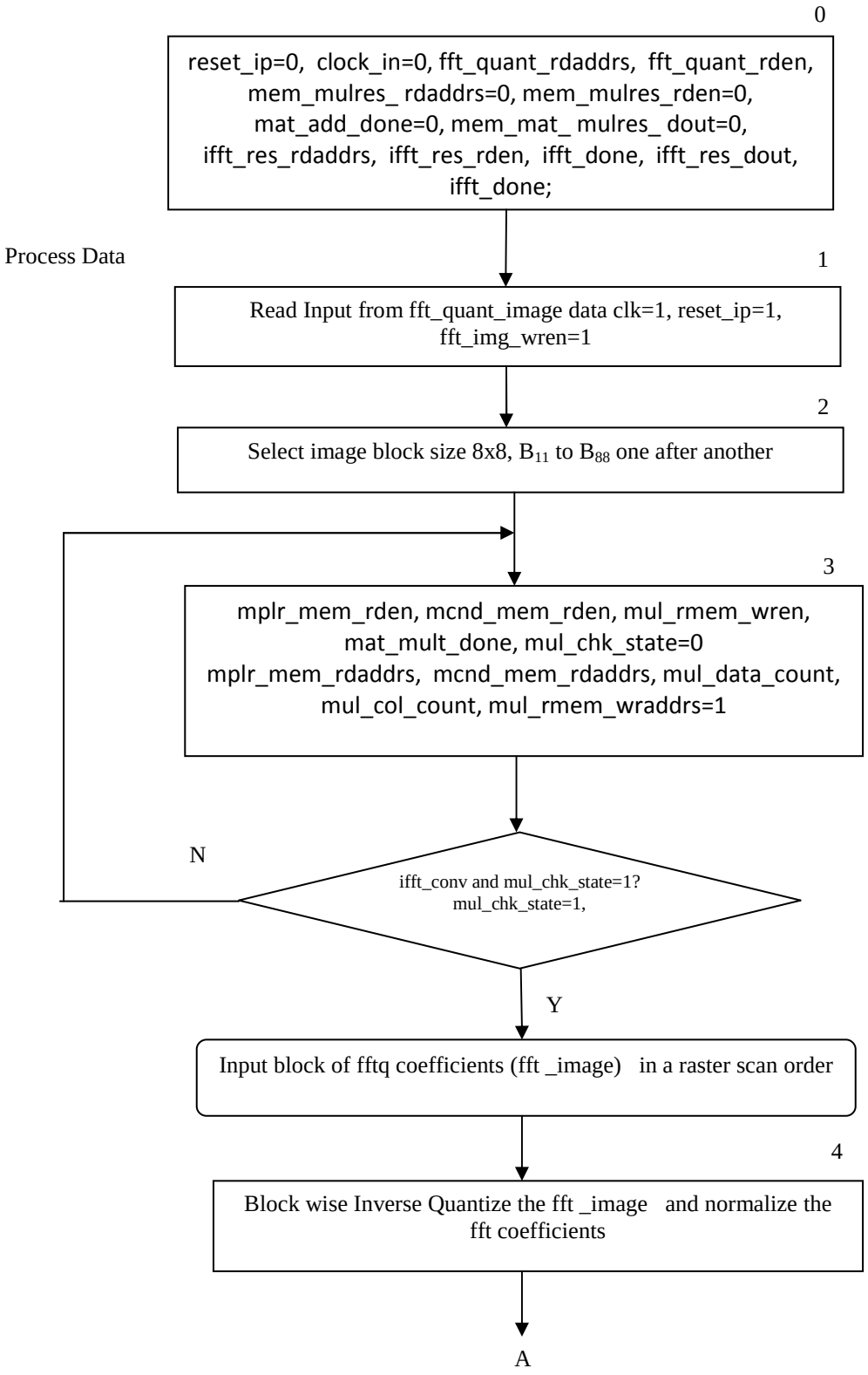

Fig. 3 ASM Chart for IQ IFFT for Image Reconstruction (Continued)

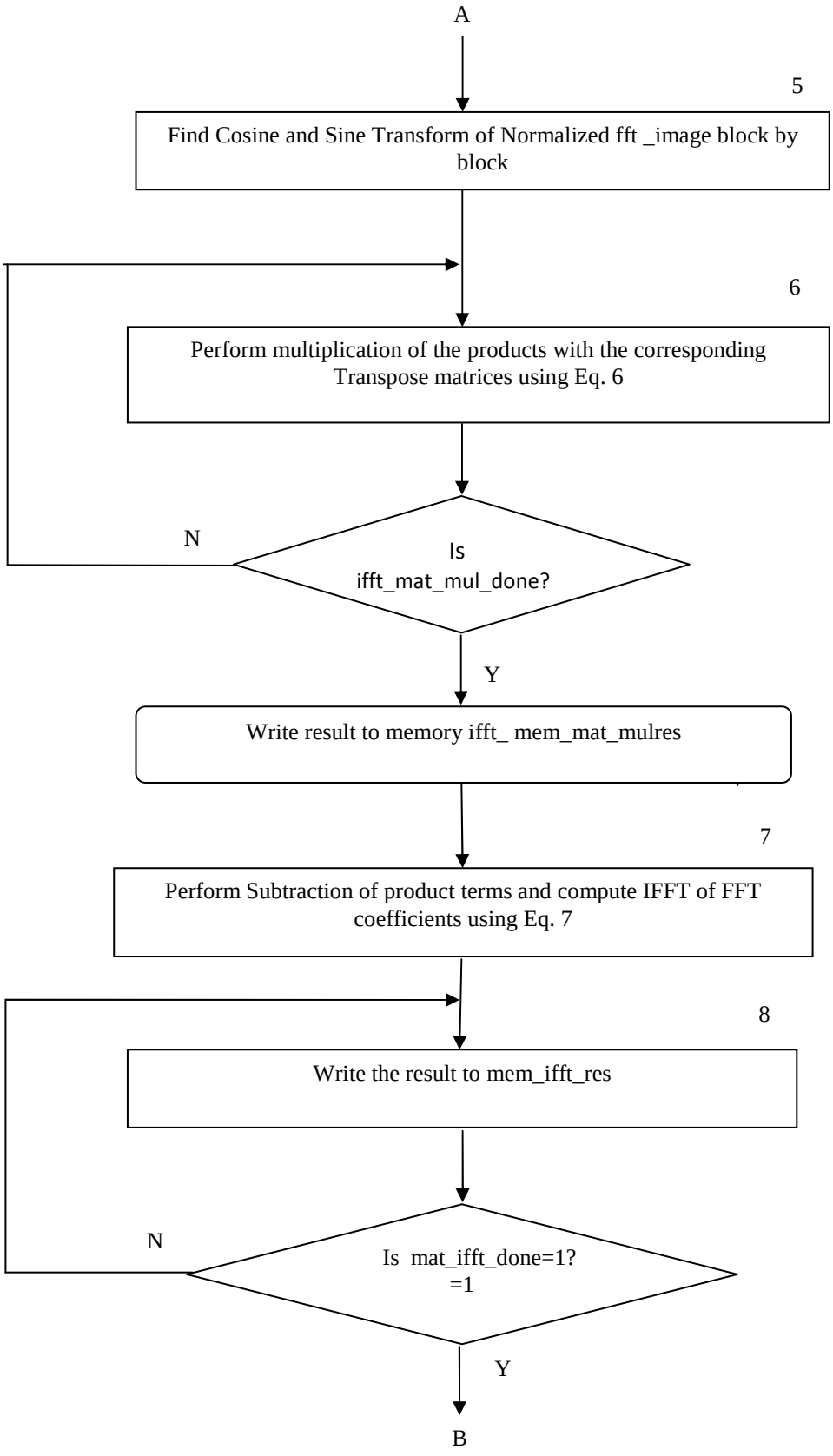

Fig. 3 ASM Chart for IQ IFFT for Image Reconstruction (Continued)

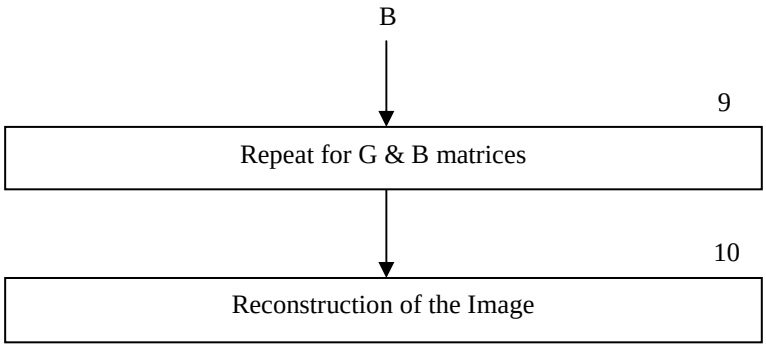

Fig.3 ASM Chart for IQ IFFT for Image Reconstruction

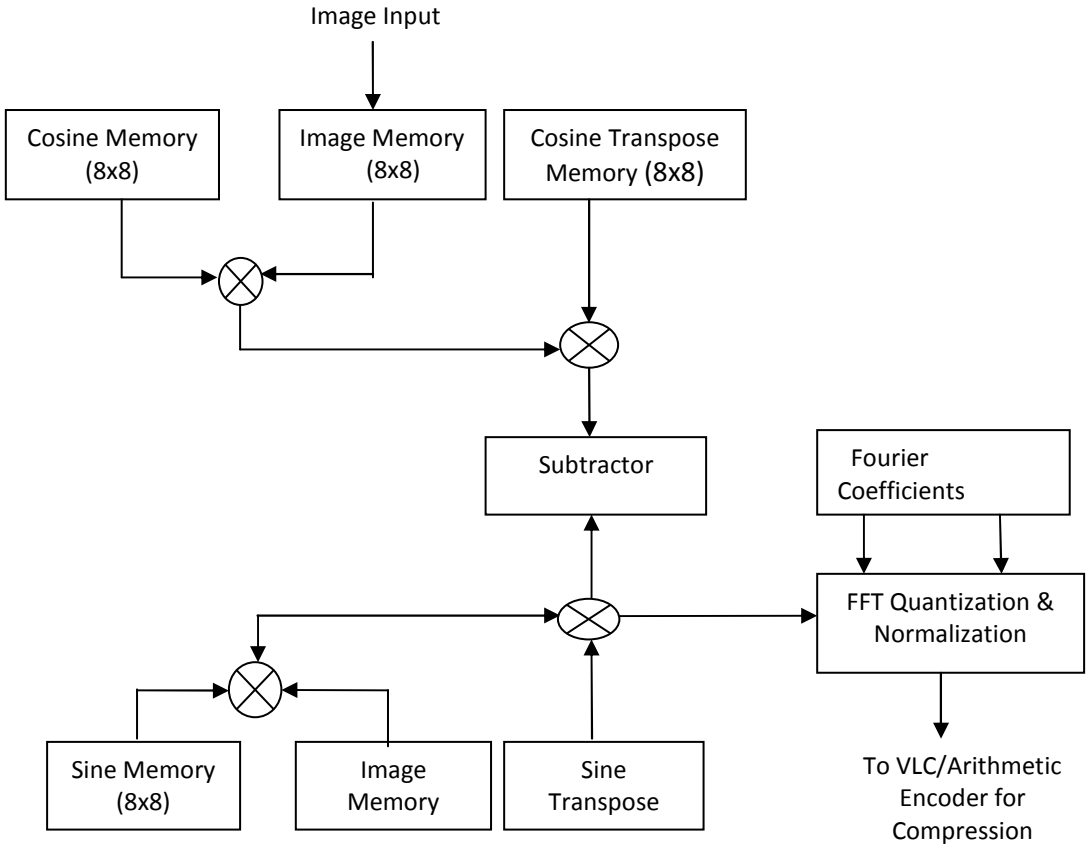

Fig.4 Schematic Architecture of FFTQ Processor

The schematic architecture of the same is shown in Fig. 11.

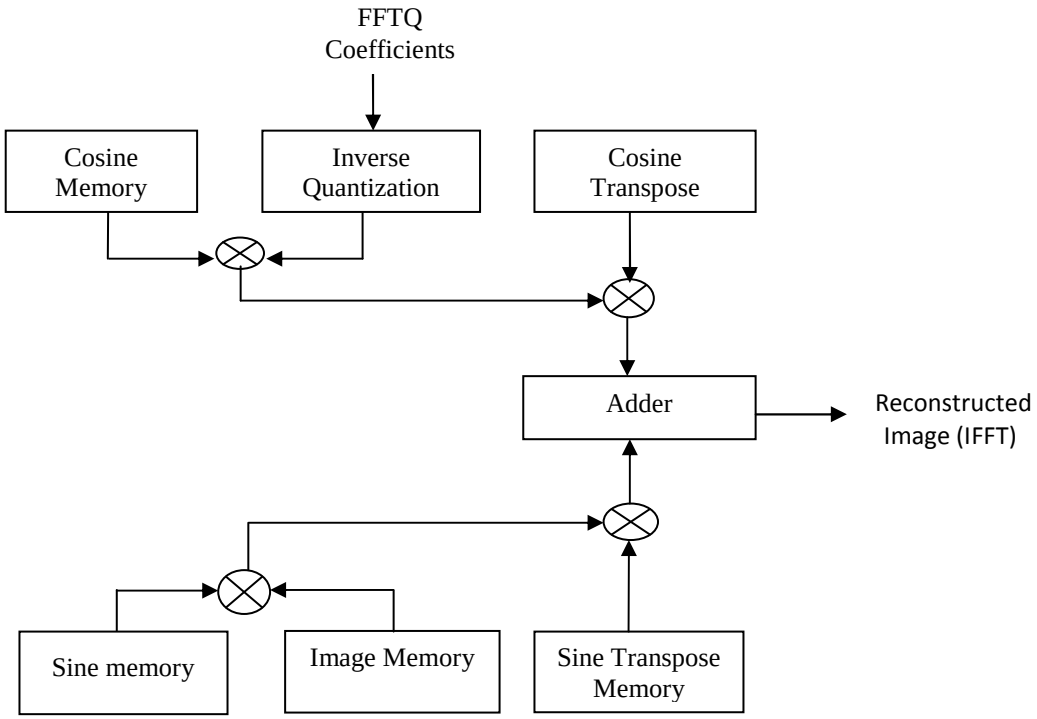

Fig.5 Schematic Architecture of IQ IFFT Processor

VI. SIMULATION RESULTS AND DISCUSSIONS

The proposed Novel Algorithms for FFTQ and its Inverse for Image Compression, whose architecture was presented in the previous section, has been coded and tested in Matlab first in order to ascertain the correct functioning of the algorithm [16]. Subsequently, the complete system has been coded in RTL compliant Verilog so that it may be implemented on Xilinx FPGA. The design has been simulated using ModelSim and synthesized using Xilinx ISE 14.5. The RTL design has been targeted on Xilinx Spartan 6 xc6slx45-3fgg676 FPGA device [17].

Matlab program was written in order to convert the images into text format since Modelsim simulation tool accepts only text inputs. The Verilog design for FFTQ, IQ IFFT Processors were simulated using Modelsim to get the reconstructed picture in "txt" format[18]. The "txt" file was converted back to image format using another Matlab program. This program displays both the original picture as well as the reconstructed picture. The Matlab program also computes the quality of the reconstructed image referred to as PSNR expressed in dB.

The transform process starts when the "fft_conv_start" signal is asserted. The simulation waveforms for inputting the image pixel data for FFTQ and its Inverse are shown in Fig. X, X+1 respectively. The images that have been compressed in this simulation are tabulated along with the size of images. The transforming process commences at 8µs with the start of computation of CxIxCT and SxIxST and is shown in Fig.6 and Fig.7. As shown in the Fig.9, an address counter such as "fft_done" keeps track of the number of blocks processed for an image such as Lena. Owing to pipeline inherent in the design, the actual transformed output pixels ("fft_dout") start issuing out only at 293.7 µs as presented in Fig. 18. The FFT computation ends at 4087 µs and the same is evident in Fig. 9. The quantization of FFT coefficients starts at 560 µs and can be observed in Fig. 10. Their validity is indicated by asserting "fft_quant_done" signal. The signal "fft_quant_addr" counter continues from 1 to 4087 µs, the last value representing the last block of Image processed. The "count_nonzero" counts the number of non zero fft quantized co-efficients present in the "mem-_quant_data" from the next clock cycle, i.e., from 4085 µs onwards. Hereafter the reconstruction of the quantized

image starts by de quantizing at 3253µs and computing the Inverse FFT which ends at 4407 µs as shown in Fig.10. It may be noted that the number of pixels in the picture Lena is 1600 x 1200, which means that every pixel is processed in a little over a clock cycle, latency being 1202 clock cycles per frame for a transformed picture of size 512 x 512 pixels. The complete processing of an image of size 1600 x 1200 pixels takes, therefore, 5 ms (1921202 clock cycles) if clock rate is 50 MHz.

Since for high resolution pictures, 100 MHz clock rate is normally used, we can comfortably claim that the RTL Verilog design is capable of compressing color pictures of size 1600 x 1200 pixels at a real time rate of 30 Frames/second.

The RTL Verilog simulation results were compared with Matlab results described earlier in order to validate the hardware design. Elaborate experiments were conducted on various images and consistently good results have been obtained. The simulation results for compressing an fft transformed images are shown for standard data bases in Fig. 13. In these figures, (a) are the original images and (b) are the compressed and reconstructed image. The FFT and IFFT processor were first coded in Matlab in order to evaluate the quality of the reconstructed image and the compression that can accomplished. Besides Matlab output serves as reference for Verilog output[19]. Subsequently the core modules for the FFT processor were realized in Verilog for FPGA implementation.

The original and reconstructed images obtained using the proposed Verilog Design are shown in Fig. 25 from Fig. 25.a to Fig 23.f. The pictures show that the reconstructed images are indistinguishable from the original images with a PSNR value greater than 35 dB.

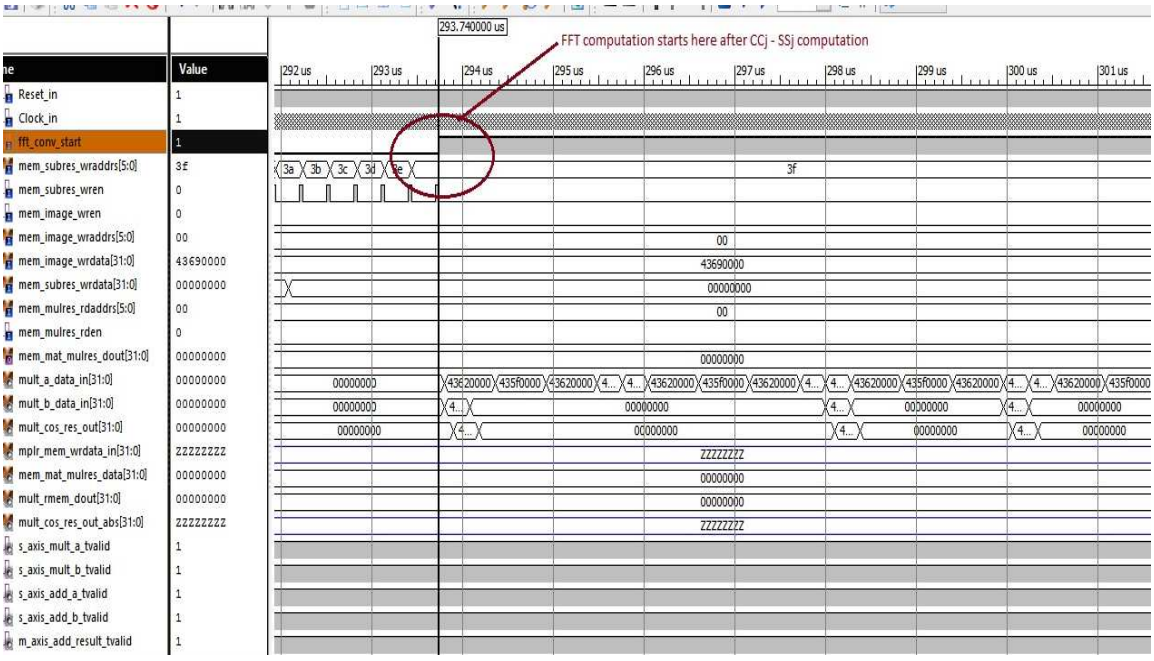

Fig.6 Simulation Result of Start of Conversion for FFT Computation

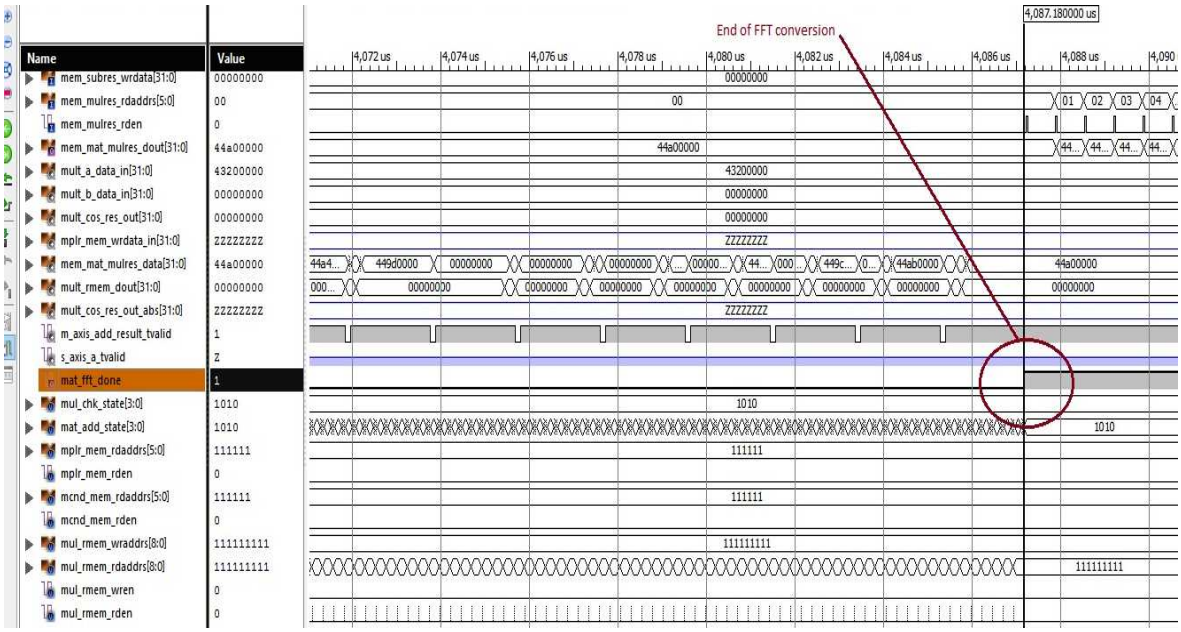

Fig.7 Simulation Result of End of Conversion for FFT Computation

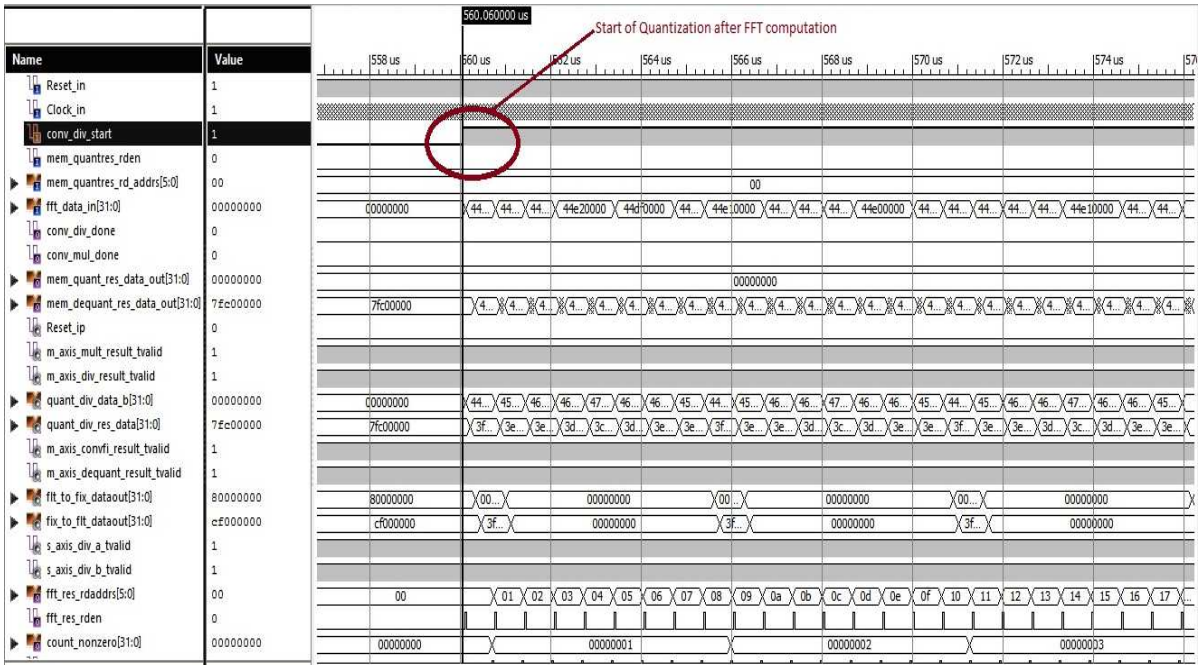

Fig.8 Simulation Result of Start of Conversion for Quantization

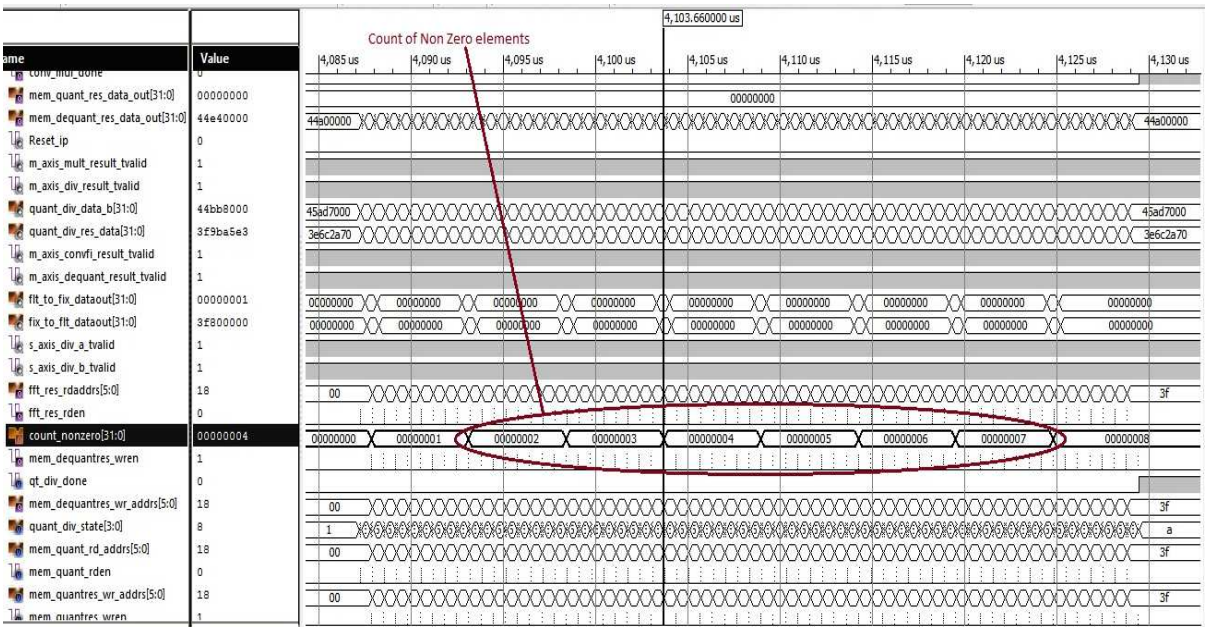

Fig.9 Simulation Result for Non Zero Counting

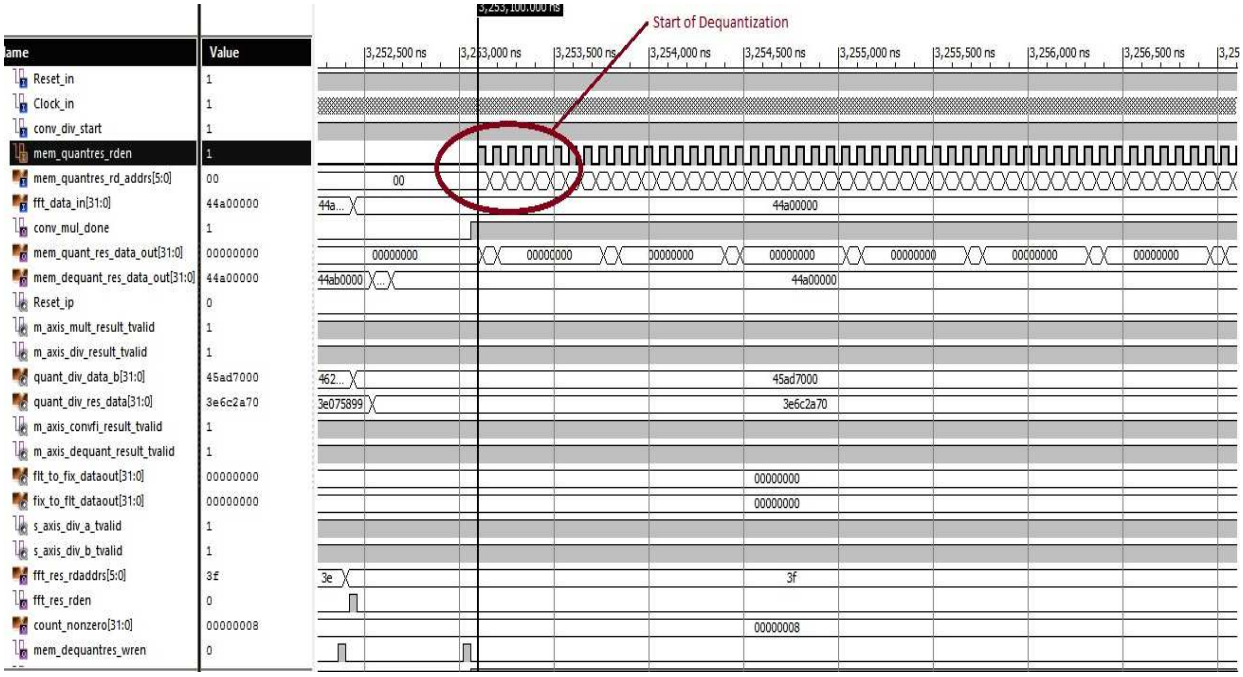

Fig. 10 Simulation Results of Start of Conversion for De quantization

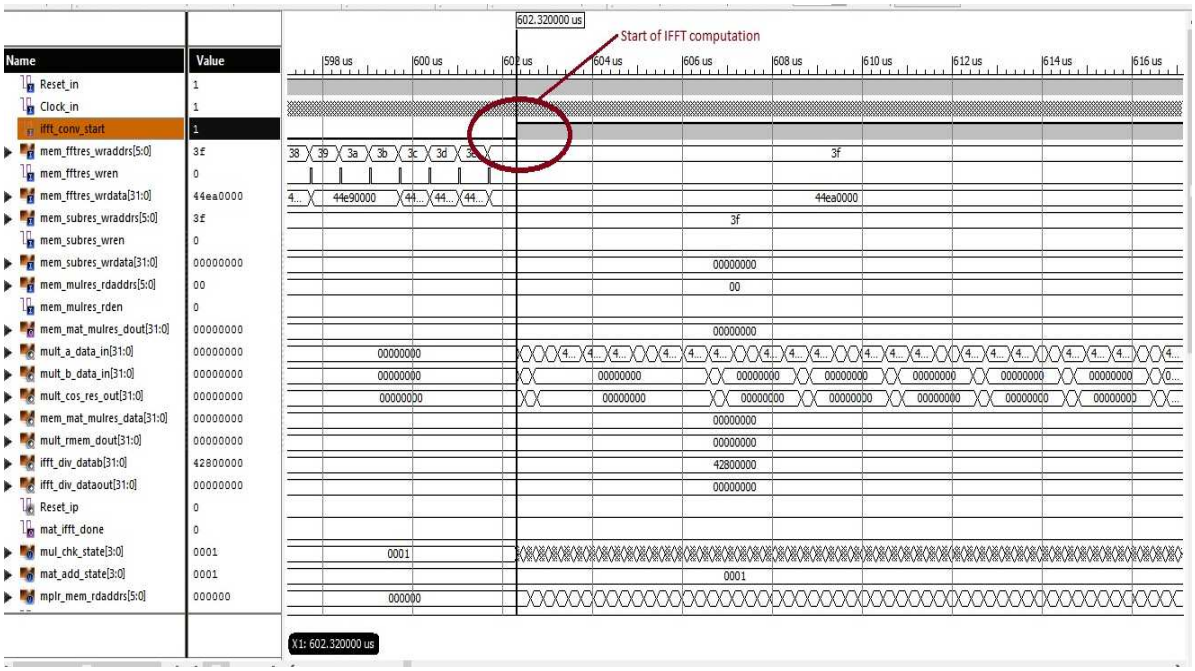

Fig. 11 Simulation Result for Start of Conversion for IFFT Computation

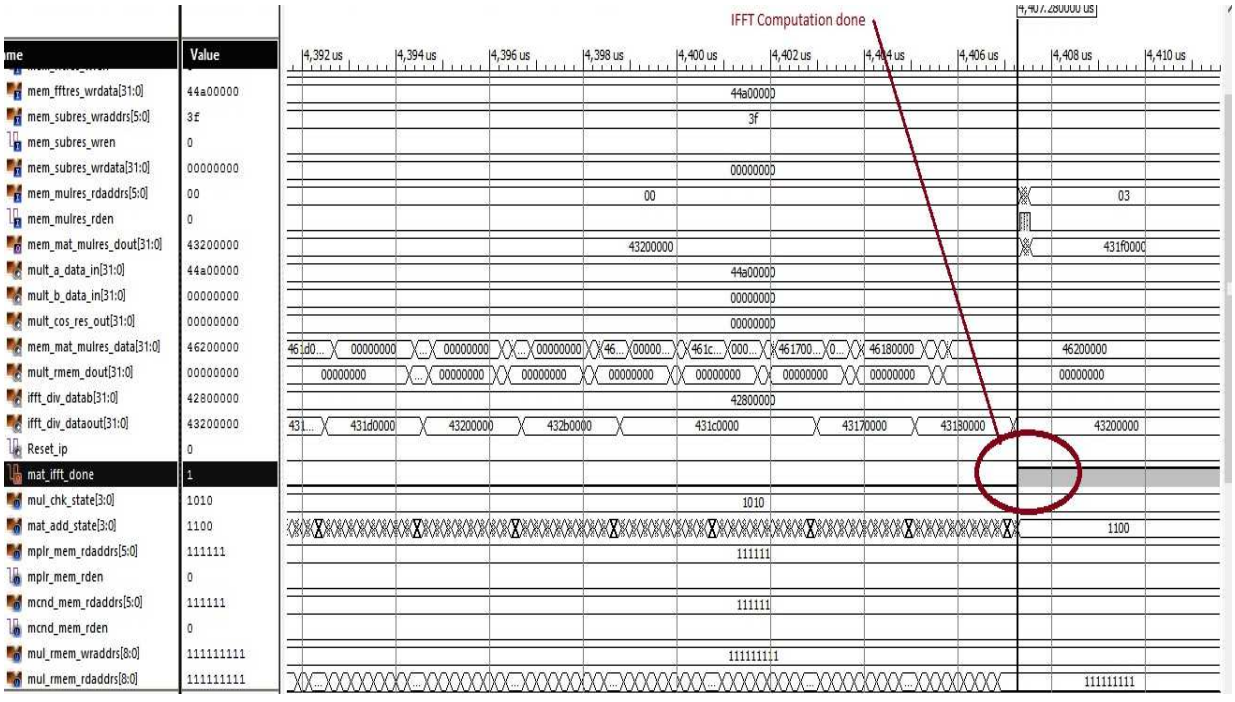

Fig. 12 Simulation result of End of Conversion for IFFT computation

a.Original Image, Lena Picture Size: 512x512 Pixels

b. Reconstructed Lena

c. Original Image, Sunflower Picture Size:530 x 363Pixels

d. Reconstructed Sunflower

e. Original Image: Cleveland. Picture Size: 560x816

f. Reconstructed Image

Fig. 13 Original and Reconstructed Images resulted from the proposed algorithm

TABLE III. The results of proposed algorithm

Sl. No.	Image	Image Size in Pixels	PSNR in dB	Compression Ratio
1.	Lena.tiff	512x512	35.6	15:1
2.	Sunflower.bmp	530x363	35	14
3.	Canada.bmp	403x686	39.5	21.1
4.	Luvre_Paris.bmp	805x501	35.2	22.5
5.	Cleveland.bmp	560x816	34.5	9.1
6.	Niagara Falls.jpg	312x480	37.4	20.4
7.	Pyramid.bmp	296x456	35.3	54.2
8.	Sandstone_Canyon.bmp	328x530	37.1	11.4
9.	Sunset_7.bmp	363x529	36.1	9.5
10.	Volcano.bmp	361x528	35	16.5

VII. CONCLUSION

A VLSI realization for FFTQ and IQ IFFT Processors using new algorithm and architecture has been presented. It uses real number arithmetic operations for the computation of FFT and it's Inverse. Also massive parallel and pipeline circuits have been used for the RTL realization [20].The algorithm was first coded in MATLAB to check the correctness of the concepts and subsequently coded in Verilog conforming to RTL coding guide lines. The architecture can be implemented as an FPGA or as an ASIC.

REFERENCES

[1] S Jayaraman, S Esakkirajan and T Veerakumar ,"Digital Image Processing" by Tata McGraw Hill l Publications.

[2] Rafael C. Gonzalez, Richard E. Woods, "Digital Image Processing " by II Edition Prentice Hall Publications.

[3] S.Uzun and A. Amira A. Bouridane "FPGA Implementations of Fast Fourier Transforms for Real-Time Signal and Image Processing ".

[4] More T. V. and Panat A. R., "Comparative Study of Pipelined Reconfigurable FFT Processor", International Journal of Knowledge Engineering, Volume 3, Issue 1, pp. 107-110, 2012

[5] Gokhan Polat, Sıtkı Ozturk, and Mehmet Yakut, "Design and Implementation of 256 Point Radix-4 100 Gbit/s FFT Algorithm into FPGA for High Speed Applications",
ETRI Journal, Vol. 37, pp. 667-676, Aug. 2015

[6] G A Jullien, "High performance Arithmetic processor for DSP systems," VLSI Signal Processing Technology, Kluwer Academic Publishers, 1994.

[7] Enis Çerri and Marsida Ibro , "FFT Implementation on FPGA using Butterfly Algorithm" International Journal of Engineering Research & Technology, Vol.4-Issue 02, February 2014.

[8] Mohammad Nazmul Haque, Mohammad Shorif Uddin, "Accelerating Fast Fourier Transformation for Image Processing using Graphics Processing Unit" ,Journal of Emerging Trends in Computing and Information Sciences, Volume 2 No.8, August 2011.

[9] Anitha R, Bagyaveereswaran V, "Braun's Multiplier Implementation using FPGA with Bypassing Techniques" International Journal of VLSI design & Communication Systems

[10] Juanli Hu, Jiabin Deng and Juebo Wu "Image Compression Based on Improved FFT Algorithm", Journal of Networks, VOL. 6, NO. 7, JULY 2011

[11] Anitha T G and S. Ramachandran , "Novel Algorithms for 2-D FFT and their Inverses for Image Compression," Signal Processing, Image Processing & Pattern Recognition (ICSIPR), IEEE Publishers, pp 62-65. Feb 2013.

[12] Riya Saini, R.D.Daruwala, "Efficient Implementation of Pipelined Double Precision Floating Point Multiplier" IJERA Vol. 3, Issue 1, January -February 2013, pp.1676-1679

[13] Zou Wen, Qiu Zhongpan and Song Zhijun, "FPGA Implementation of efficient FFT algorithm based on complex sequence" by 2010 IEEE pp 614-617

[14] N. Keshaveni, S. Ramachandran and K. S. Gurumurthy, "Design and FPGA Implementation of Integer Transform and Quantization Processor and Their Inverses for H.264 Video Encoder", International Journal of Computer Science and Communication, pp. 43-50, Vol. 1,No. 1,January- June 2010.

[15] Dr.Thamer M Jamel "Implementation of FFT algorithm using FPGA technique", Jordanian International Electrical & Electronics Engineering Conference JIEEEC" 2006.

[16] Nick Mehta "Design how to design Xilinx 7 series FPGA User Guide Lite",19/4/2011

[17] Tutorial on FPGA Design Flow based on Xilinx ISE WebPack and ISim version 2.0 "Implementation And Analysis Of FPGA-Based Design Of 32-Bit FPAU" in Spring Publishers 2014, pp .80

[18] Chu Chao, Zhang Qin,Xie Yingke1 Han Chengde " Design of a High Performance FFT Processor Based on FPGA" ,2005 IEEE. pp 920-923.

[19][Online] Available: www.xilinx.com/itp/xilinx10/isehelp/ise_c_schematic_overview.html

[20] Dr. S. Ramachandran, "Digital VLSI Systems Design", Published by Springer.