International Journal of Electronics and Communication Engineering.
ISSN 0974-2166 Volume 4, Number 4 (2011), pp. 425-442
© International Research Publication House
http://www.irphouse.com

A Novel VLSI Architecture for Digital Image Compression using Discrete Cosine Transform and Quantization

Dr. Anubhuti Khare[1], Manish Saxena[2] and Shweta Tiwari*[3]

[1]*Reader, Department of Electronics and Communication,*
University Institute of Technology,
Rajeev Gandhi Technical University, Bhopal, India
E-mail: anubhutikhare@gmail.com
[2]*Head of Electronics and Communication Department,*
Bansal Institute of Science and Technology Bhopal, India
E-mail: manish.saxena2008@gmail.com
[3]*Student, M.Tech. (Digital Communication),*
Bansal Institute of Science and Technology Bhopal, India
E-mail: shweta.tiwari86@gmail.com

Abstract

For multimedia and medical image transmission applications the data compression is an essential technique [1]. Many data compression techniques have been proposed in the past [2] But there is a scope to further improve such proposals. This paper in that direction proposes a new technique to improve the data compression technique. We have designed a new Discrete Cosine Transform and Quantization (DCTQ) architecture for image compression, in this work Discrete Cosine Transform is a type of image transform which can be applied on images to achieve compression of image data. The quantization of the DCT data coefficients is then performed to achieve compression. This technique makes use of lesser number of multipliers. The JPEG method is used for both color and black-and-white images, but the focus of this work will be on compression of the blackand- white images. The functionality of the DCTQ is verified in MATLAB.The design is also implemented onVertex2p to verify its capability on silicon. Results shows the area saving in terms of no of gate counts with an excellent efficiency.

Keywords: DCT, Image Compression, IDCT.

Introduction
With the increasing in multimedia applications the demand for digital information has increased dramatically. Medical and satellite images are good examples for static images. Digital images have become attractive from the point of storage and transmission. They produce an enormous amount of digital data. Reduction in the size of this image data for both transmission and storage is very important [5]. Image compression is a mapping technique of images from a higher dimensional space to a lower dimensional space. The basic goal of image compression is to represent an image with minimum number of bits of an acceptable image quality. There are several image compression techniques available today.

These fall into two general categories namely lossless and lossy image compression. The JPEG process is a widely used form of lossy image compression.

The Discrete Cosine Transform has been shown to be near optimal for a large class of images in energy concentration. The DCT is a type of Image Transform which expresses a Sequence of finitely many data points in terms of a sum of cosine functions oscillating at different frequencies. The DCT Works on the basis of decomposing the images into segments [6] and obtaining the corresponding frequency components. During the Quantization process the pixels with less important frequencies are discarded, hence the use of the term called lossy compression.

The DCT Equation
The DCT equation (Eq1.0) represents the two dimensional implementation of DCT on a block of image data represented by the pixel values, This equation represents the corresponding DCT co-efficient of the i, jth entry of an image.

$$D(i,j) = \frac{1}{\sqrt{2N}} C(i)C(j) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} p(x,y) \cos\left[\frac{(2x+1)i\pi}{2N}\right] \cos\left[\frac{(2y+1)j\pi}{2N}\right]$$

$$C(u) = \begin{cases} \frac{1}{\sqrt{2}} & if\, u = 0 \\ 1 & if\, u > 0 \end{cases} \qquad (1.0)$$

P(x, y) is the x, yth element of an input image represented by the matrix. N is the size of the block on which DCT is done. The equation calculates one entry of the transformed image (i,jth) of the transformed image from the pixel values of the original image matrix. For the standard 8X8block that JPEG compression uses,N equals 8 and x and y range from 0 to7 hence D(i,j) would be reduce as in Equation (2.0).

$$D(i,j) = \frac{1}{4} C(i)C(j) \sum_{x=0}^{7} \sum_{y=0}^{7} p(x,y) \cos\left[\frac{(2x+1)i\pi}{16}\right] \cos\left[\frac{(2y+1)j\pi}{16}\right] \qquad (2.0)$$

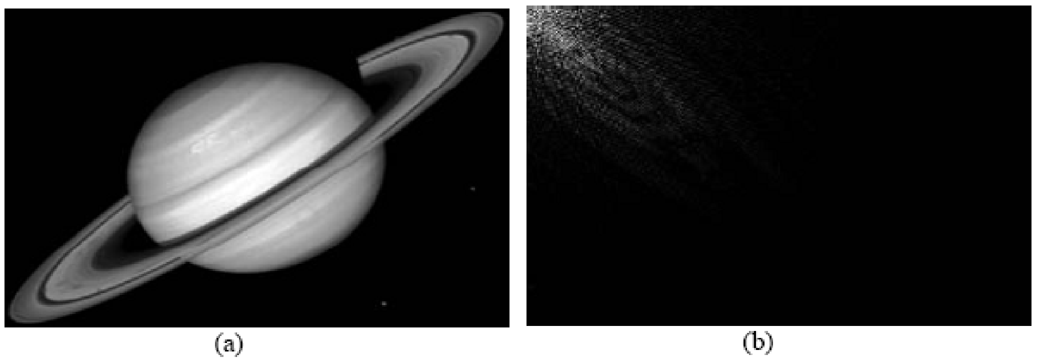


Figure 1: The Image of the planet (a) before (b) after applying DCT.

The Fig.1(a) shows the original image of the planet and Fig.1 (b) shows the image after applying the DCT it shows the low frequency components are concentrated only at the top left corner side of the image, from these pixels only it is possible to reconstruct the image during reconstruction this shows DCT can provides the excellent energy compaction .

Design Flow

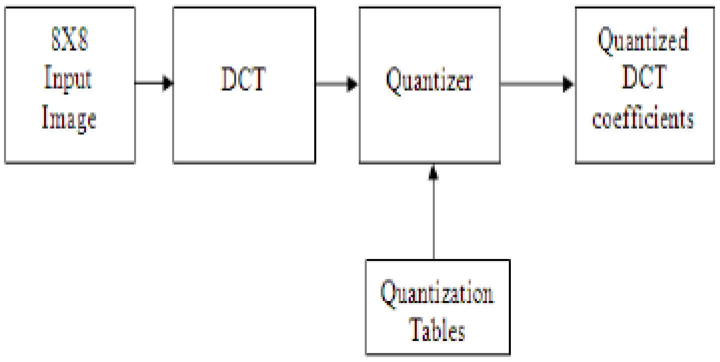


Figure 2: Different stages of DCTQ process.

The Fig.2 shows the different stages of the DCTQ process.

The following steps are followed to find the DCTQ.

1. The original image is broken into 8x8 blocks of pixels.
2. Working from left to right, top to bottom, the DCT is applied to each block of pixel.
3. Each DCT block is compressed through quantization process using Quantization matrix.

4. The compressed amount of image data is stored drastically in reduced amount of space.
5. To reconstruct the image through decompression the Inverse Discrete Cosine Transform (IDCT) is used. The entire process of the steps undertaken is summarized in the form flow charts shown in Fig.3. and Fig.4.

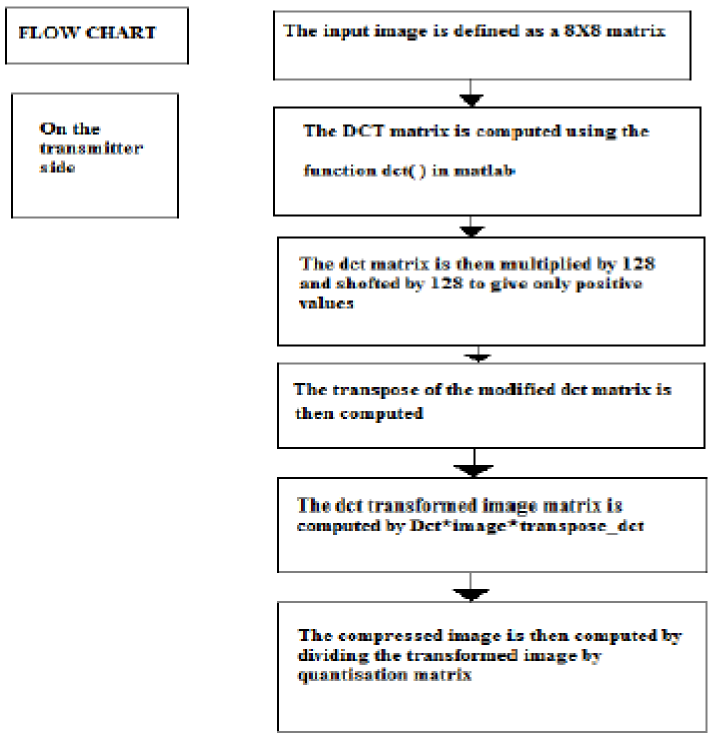

FLOW CHART
On the transmitter side
The input image is defined as a 8X8 matrix
The DCT matrix is computed using the function dct( ) in matlab
The dct matrix is then multiplied by 128 and shofted by 128 to give only positive values
The transpose of the modified dct matrix is then computed
The dct transformed image matrix is computed by Dct*image*transpose_dct
The compressed image is then computed by dividing the transformed image by quantisation matrix

Figure.3: Flow chart of Transmitter.

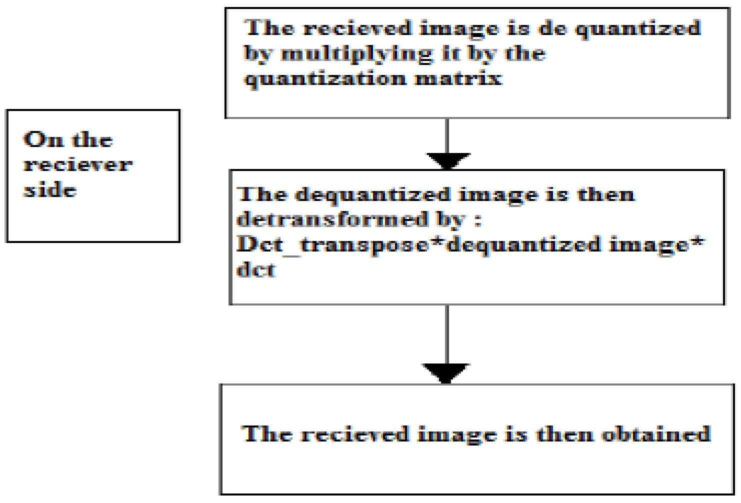

The recieved image is de quantized by multiplying it by the quantization matrix
On the reciever side
The dequantized image is then detransformed by : Dct_transpose*dequantized image* dct
The recieved image is then obtained

Figure 4: Flow chart of Receiver.

Level Shifting

We notice that the DCT values are between -1 and +1. But the values that device can operate on is in the range of positive integers only. Hence we multiply the DCT coefficients by128 and level shift by 128 as shown in Fig.5.

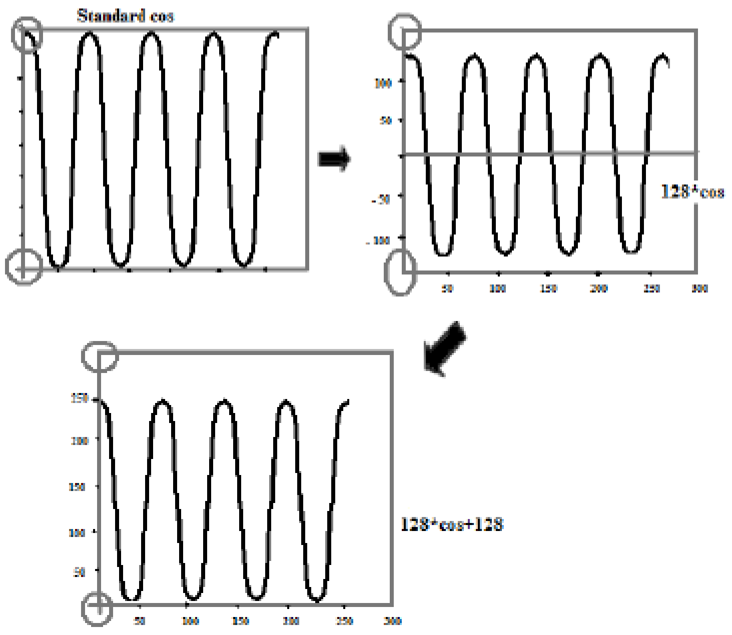


Figure 5: Level shifter.

DCT transformed image = (cpos-128) * Image_ matrix/128*cpos-128/128 Where cpos and ctpos are level shifted DCT coefficients. The entire process undertaken can thus be summarized in the form of compressed and final image shown in the Fig.6.

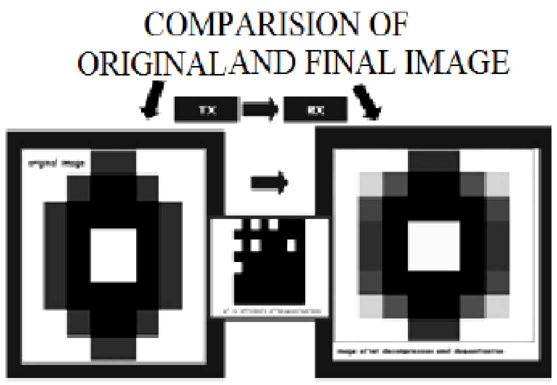


Figure 6: Comparison of original and final Image.

Proposed DCTQ Architecture

The architecture required for implementation of DCTQ is designed as follows. The design should contain a “main” Module that calls other modules for multiplication purposes. Since the computation involves two stages of multiplication we need to store the intermediate product after the first stage of multiplication. This is then pipelined to the second stage of multiplication. Thus we also require a dual RAM stage for the storage of the products computed. The computation also involves the compression part where the transformed matrix is divided by the quantization matrix. Thus a module exclusively for the purpose of quantization is also designed. The control and enable signals required for operation of the designed architecture is provided by the DCTQ controller block. All these processes are in turn provided with the inputs from the test bench. The proposed architecture is diagrammatically shown in the Fig 7.

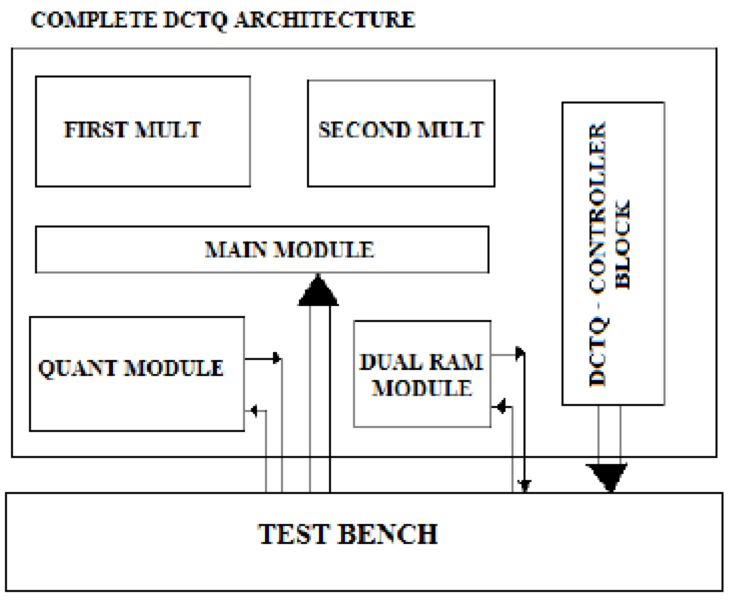


Figure 7: DCTQ Architecture.

The designed modules are:

1. MAIN MODULE
2. FIRST MULT
3. SECOND MULT
4. DCTQ_CONTROLER MODULE
5. DUAL RAM
6. TESTBENCH

The main module is the DCTQ architecture shown in the Fig.7. The functionality of the modules is explained in detail in the following pages.

First Multiplier

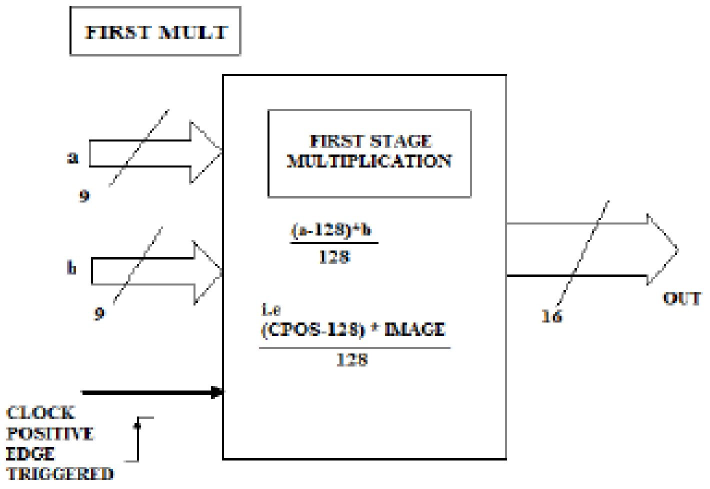

FIRST MULT
FIRST STAGE MULTIPLICATION
(a-128)*b / 128
i.e (CPOS-128) * IMAGE / 128
a
9
b
9
16
OUT
CLOCK POSITIVE EDGE TRIGGERED

Figure 8: First stage Multiplier.

The first stage multiplications are computed using first mult modules shown in Fig.8. At every positive edge of clock pulse, the inputs viz. a, b which are 9-bit long are pipelined and product is computed using the formula: out = (a-128)*b/128

Where,‘a’ is image pixel value. ‘b’ is level shifted DCT coefficient. “out” is the computed product.

Second Multiplier

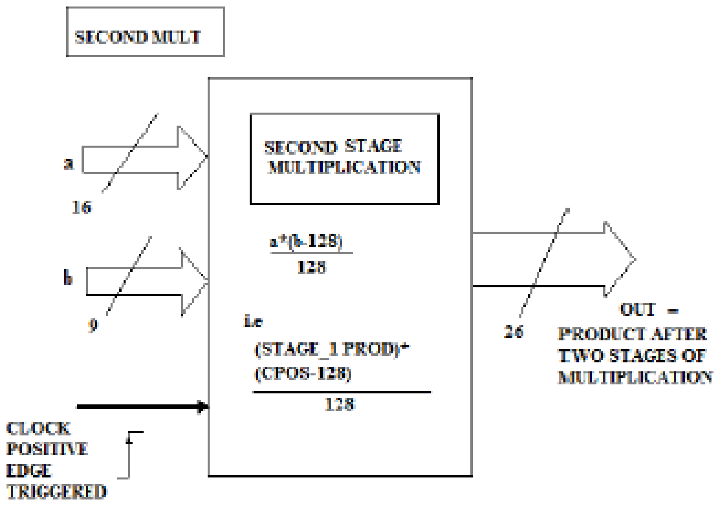

SECOND MULT
SECOND STAGE MULTIPLICATION
a*(b-128) / 128
i.e (STAGE_1 PROD)* (CPOS-128) / 128
a
16
b
9
26
OUT = PRODUCT AFTER TWO STAGES OF MULTIPLICATION
CLOCK POSITIVE EDGE TRIGGERED

Figure 9: Second stage Multiplier.

The second stage product is computed by using second mult as shown in Fig.9. At every positive clock 16-bit partial product ‘a’computed in stage one is fed as input along with DCT-transpose matrix element- ‘b’. Product after two stages of multiplication is thus computed using the formula: out= a*(b-128) /128

Where,‘a’ is first stage product. ‘b’ is level shifted DCT transpose coefficient. “out” is the computed product.

Main Module

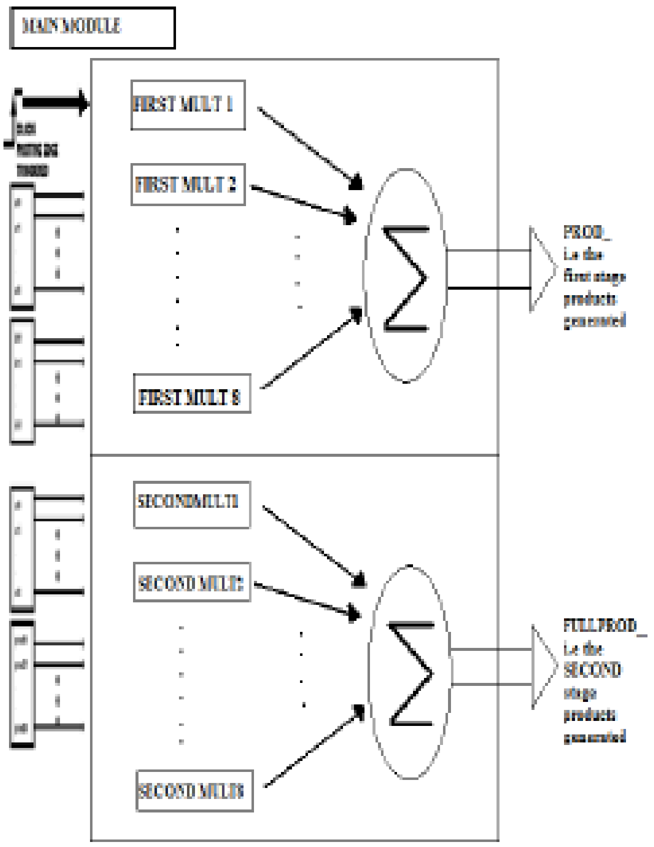

MAIN MODULE
FIRST MULT 1
FIRST MULT 2
FIRST MULT 8
PROD_ i.e the first stage products generated
SECONDMULT1
SECOND MULT2
SECOND MULT8
FULLPROD_ i.e the SECOND stage products generated

Figure 10: Main Module.

There are two stages in the main module as shown in the Fig.10.At every positive edge of the clock pulse eight different first multiplier modules are instantiated. The eight products generated are then added to compute the first stage product which is stored in dual RAM. We also notice that the second multiplier do not come into picture until all the first stage products are available. This ensures that we use only “eight different multipliers” at any given point of time. Similarly the second stage products are computed by instantiating eight second multipliers at the positive edge of the clock pulse. The second stage products are then added and stored in dual RAM stage2.

DCTQ Controller

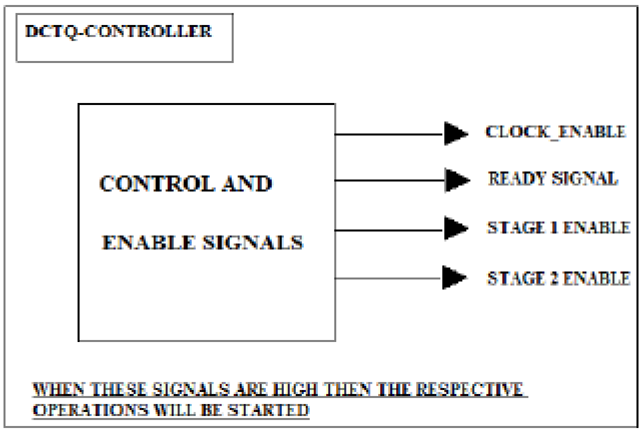
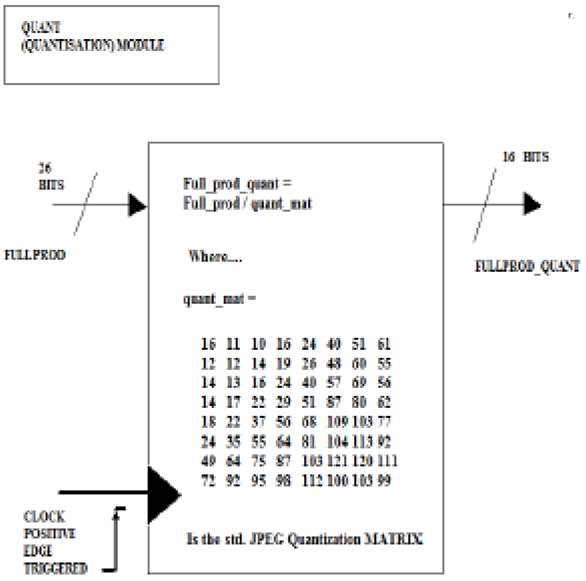

DCTQ-CONTROLLER
CONTROL AND ENABLE SIGNALS
CLOCK_ENABLE
READY SIGNAL
STAGE 1 ENABLE
STAGE 2 ENABLE
WHEN THESE SIGNALS ARE HIGH THEN THE RESPECTIVE OPERATIONS WILL BE STARTED

Figure 11: DCTQ controller.

DCTQ controller shown in the Fig.11 is used to generate various controls and enable signals. When Clock_ enable and ready signals are high clock generation and clock counter starts operation. When stage 1 enable becomes high, first stage module starts computing and first stage products are generated. Similarly when stage2 enable signal becomes high, second multiplier modules come into picture and second stage products start appearing.

Quantization Module



Full_prod_quant =
Full_prod / quant_mat

Where....

quant_mat =

16	11	10	16	24	40	51	61
12	12	14	19	26	48	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Is the std. JPEG Quantization MATRIX

Figure 12: Quantization Module.

To perform the Quantization we use JPEG standard quantization matrix as shown in Fig.12.At every positive edge of clock pulse, full product is divided by qunt_matrix. This division is performed as element by element division.Full product is 26 bits where as full prod_quant is 16 bits only resulting from division.This Quantized output is called the compressed output.

Dual RAM Module

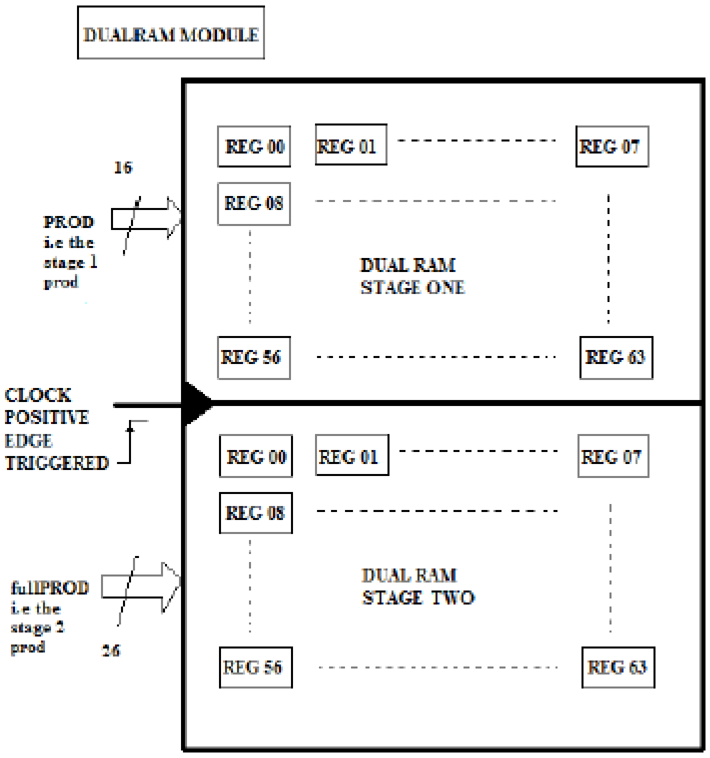


Figure 13: Dual RAM Module.

Dual RAM is used to store intermediate first stage summed products and fullprod values in a pipelined manner. The Dual RAM module is shown in the Fig13. After computing first partial products they are stored in the dual ram stage one at every positive edge of the clock pulse. After the storage of the 64 first stage products, we obtain the second stage products (fullprod) which are again stored in a pipelined manner at every positive edge of the clock pulse in “dual rams tage two”.

Test Bench

The test bench shown in Fig14 provides the input values to the main module, quant module, and dual RAM module. The clock that is used for the functioning of the entire circuit is generated in the test bench and this coordinates all the operations taking place. The cpos and ctpos matrices which are the level-shifted DCT and transposed DCT matrices respectively are also initialized in the test bench. The image under consideration is also defined with its corresponding pixel values as an 8 x8 block of image. The first stage products which are generated in stage one is temporarily stored in prod11 to prod88 registers.

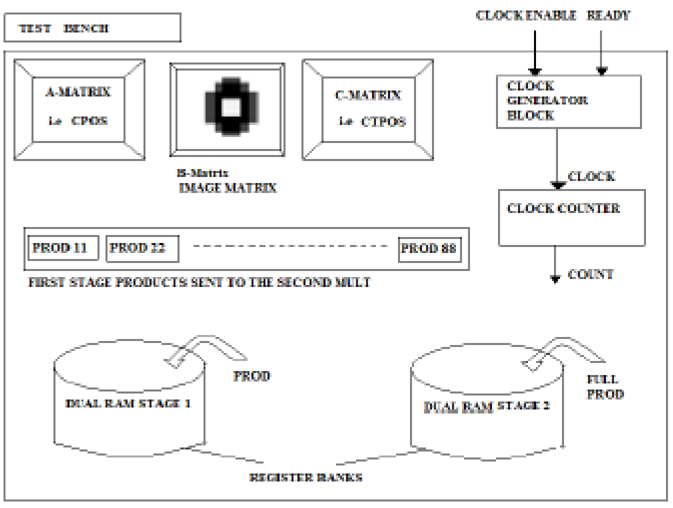
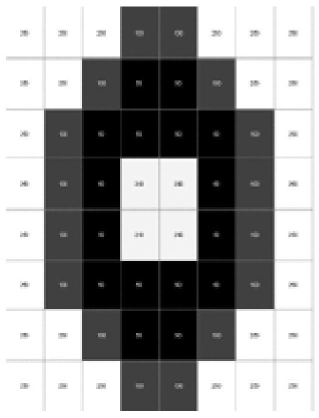


Figure.14: Test Bench.

Results and Discussion

The original image which has to be compressed is read using Matlab and generate the Image Matrix. Find the DCT coefficients on the obtained Image then perform the Quantization using standard JPEG matrix using Matlab itself. The proposed Architecture is implemented with each modules using VERILOG HDL, the functional simulation is done using MODELSIM simulator tool and finally synthesis is done using Xilinx synthesis tool by targeting the design on to the Vertex 2pro device.

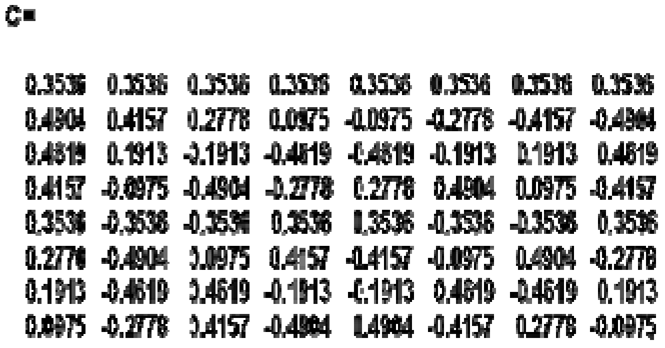


```
image =

  250  250250  100  100250  250250
  250  250  100  50   50  100  250  250
  250  100  50   505050 100  250
  250  100  50   240  24050   100  250
  250  100  50   240  24050   100  250
  250  100  50   505050   100  250
  250  250 100  50   50  100  250  250
  250  250250  100  100   250  250250
```

Figure 15: Original Image and Image matrix.

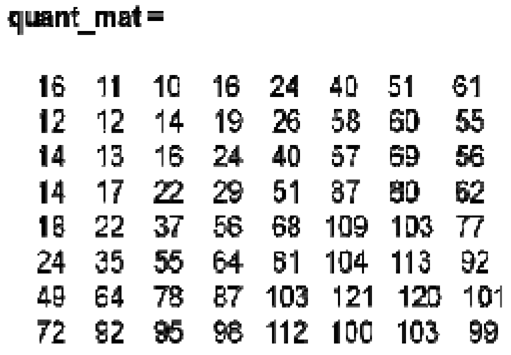
The Fig 15shows the image generated in Matlab along with the pixel values of the Image along with the image matrix. After transforming to frequency domain using DCT, the result is shown in the DCT matrix C.



```
C=

0.3536   0.3536   0.3536   0.3536   0.3536   0.3536   0.3536   0.3536
0.4904   0.4157   0.2778   0.0975  -0.0975  -0.2778  -0.4157  -0.4904
0.4619   0.1913  -0.1913  -0.4619  -0.4619  -0.1913   0.1913   0.4619
0.4157  -0.0975  -0.4904  -0.2778   0.2778   0.4904   0.0975  -0.4157
0.3536  -0.3536  -0.3536   0.3536   0.3536  -0.3536  -0.3536   0.3536
0.2778  -0.4904   0.0975   0.4157  -0.4157  -0.0975   0.4904  -0.2778
0.1913  -0.4619   0.4619  -0.1913  -0.1913   0.4619  -0.4619   0.1913
0.0975  -0.2778   0.4157  -0.4904   0.4904  -0.4157   0.2778  -0.0975
```

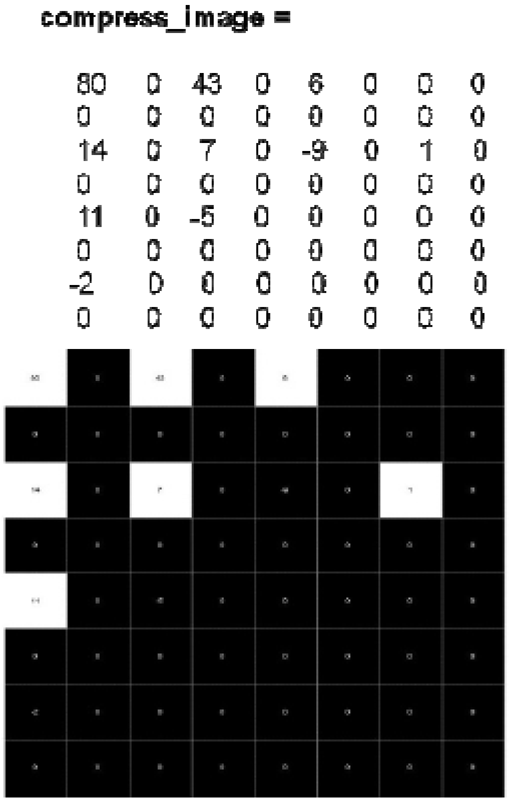
The standard JPEG Quantization matrix



```
quant_mat =

 16   11   10   16   24   40   51   61
 12   12   14   19   26   58   60   55
 14   13   16   24   40   57   69   56
 14   17   22   29   51   87   80   62
 18   22   37   56   68  109  103   77
 24   35   55   64   81  104  113   92
 49   64   78   87  103  121  120  101
 72   92   95   98  112  100  103   99
```

The image after compression with the compressed image shown in Fig.16, this figure shows that only low frequency components are considered and ignoring the high frequency components by making zeros with respect to high frequency components



```
compress_image =

    80    0   43    0    6    0    0    0
     0    0    0    0    0    0    0    0
    14    0    7    0   -9    0    1    0
     0    0    0    0    0    0    0    0
    11    0   -5    0    0    0    0    0
     0    0    0    0    0    0    0    0
    -2    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0
```

Figure 16: Compressed Image.

The image obtained after the IDCT along with the IDCT matrix is shown in the Fig.17

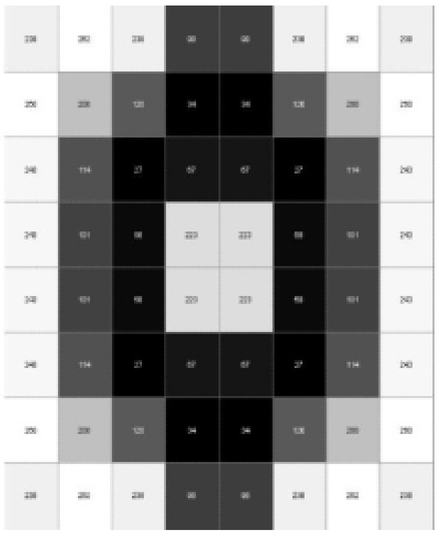


Figure 17: Reconstructed Image and Image Matrix.

```
Idct_Image =

  239  263  239   99   99   239  263  239
  271  210  131   45   45   131  210  271
  236  107   20   60   60   20   107  236
  246  104   61   226  226  61   104  246
  246  104   61   226  226  61   104  246
  236  107   20   60   60   20   107  236
  271  210  131   45   45   131  210  271
  239  263  239   99   99   239  263  239
```

The summary of the entire DCT process by comparing the VERILOG result obtained from the proposed architecture and the result obtained from the Matlab results are shown in Fig.18 and the illustrations are in the waveform diagrams.

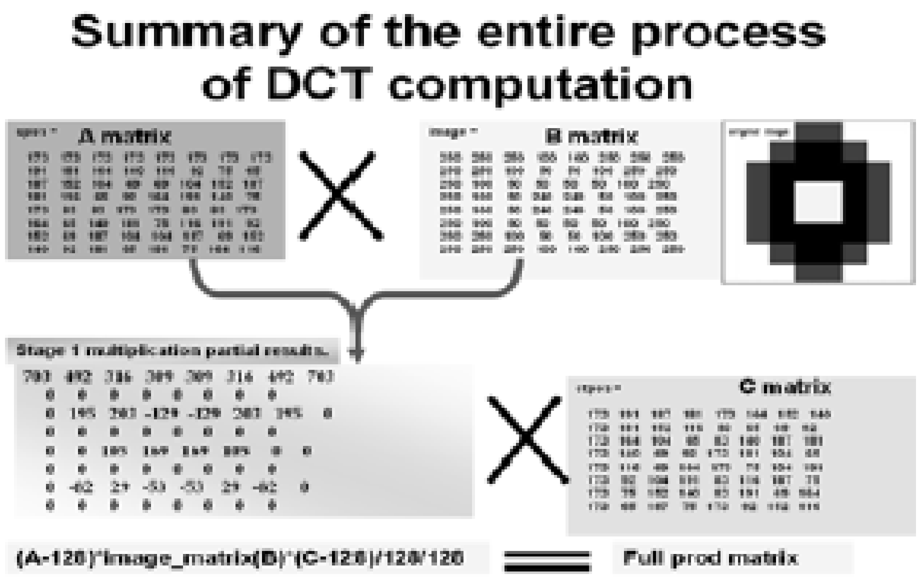
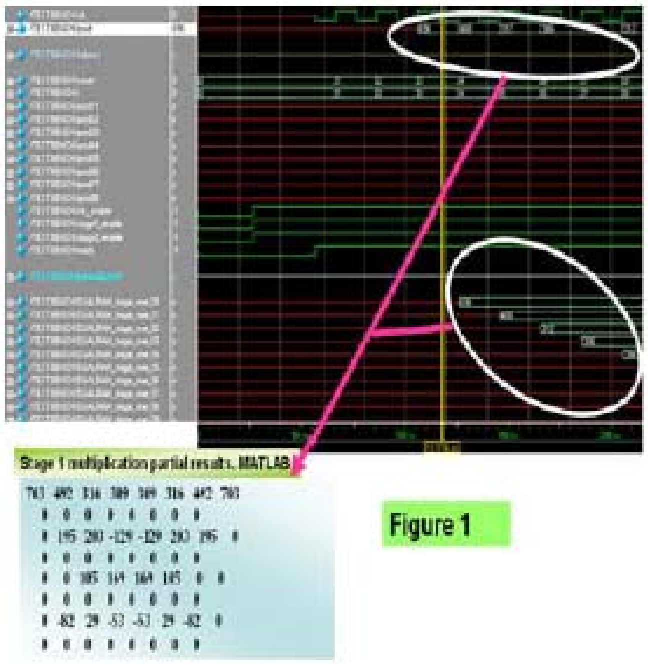

Summary of the entire process of DCT computation
A matrix
B matrix
Stage 1 multiplication partial results.
C matrix
(A-128)*image_matrix(B)*(C-128)/128/128
Full prod matrix

Figure 18: Summary of the results.

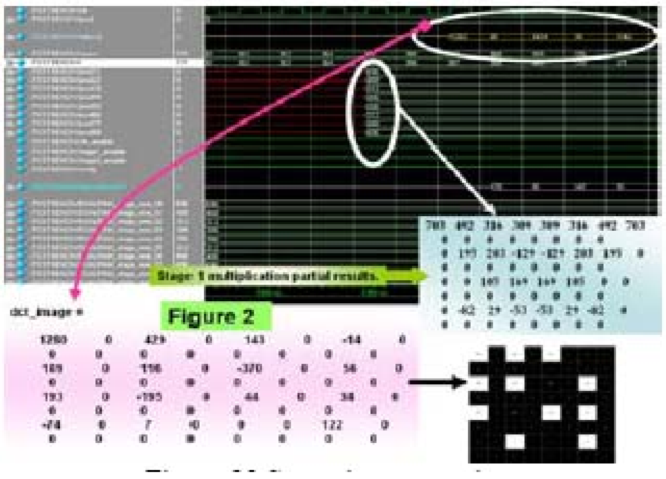
In the above diagram the matrices under consideration viz. A-cpos, B-image, C-ctpos are shown along with the first stage intermediate products generated using Matlab. The proposed DCTQ architecture is verified by modeling it in using Verilog. The results obtained after simulation are now compared with the results obtained in Matlab in the next section.



```
703 492 316 309 309 316 492 703
  0   0   0   0   0   0   0   0
  0 195 203 -129 -129 203 195  0
  0   0   0   0   0   0   0   0
  0   0 105 169 169 105   0   0
  0   0   0   0   0   0   0   0
  0 -82  29 -53 -53  29 -82   0
  0   0   0   0   0   0   0   0
```

Figure 19: First stage product.

The waveform Fig. 19 illustrates the comparison between the results obtained in Matlab and the waveforms obtained by Modelsim of the first stage multiplier. We observe that the product stored in every positive edge of clock pulse gets stored in the dual RAM stage 1 in a pipelined manner. The stage 1 matrix results obtained in both the cases agree to a large extent thus justifying our design



```
703 492 316 309 309 316 492 703
  0   0   0   0   0   0   0   0
  0 195 203 -129 -129 203 195  0
  0   0   0   0   0   0   0   0
  0   0 105 169 169 105   0   0
  0   0   0   0   0   0   0   0
  0 -82  29 -53 -53  29 -82   0
  0   0   0   0   0   0   0   0
```

```
dct_image =

  1280     0   429     0   143     0   -14     0
     0     0     0     0     0     0     0     0
   189     0   116     0  -370     0    56     0
     0     0     0     0     0     0     0     0
   193     0  -195     0    44     0    38     0
     0     0     0     0     0     0     0     0
   -74     0     7     0     0     0   122     0
     0     0     0     0     0     0     0     0
```

Figure 20: Second stage product.

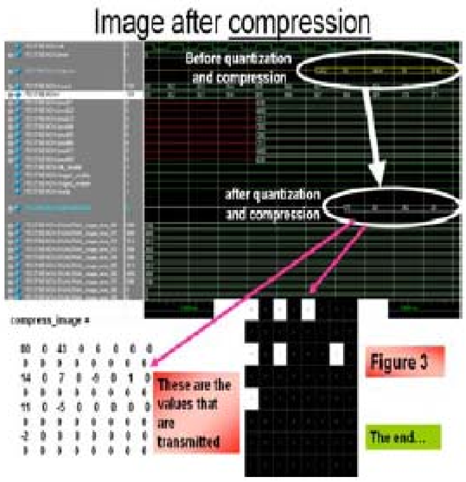


Image after compression
Before quantization and compression
after quantization and compression
compress_image =
These are the values that are transmitted
Figure 3
The end...

Figure 21: Compressed Image and Matrix.

The wave forms shown in Fig 20 illustrate the agreement in the results obtained in the stage 2 multiplier of our design and the transformed image generated by Matlab and modelsim. We also observe that the first stage products are added in a pipelined manner to get the second stage products which are subsequently stored in dual RAM stage 2.The image corresponding to the matrix obtained in Matlab is also shown.

This wave form Fig.21illustrates the process of "COMPRESSION BY QUANTIZATION". It also shows the general agreement between the results generated in Matlab and the simulated results in Modelsim. The compressed image with their pixel values is also shown.

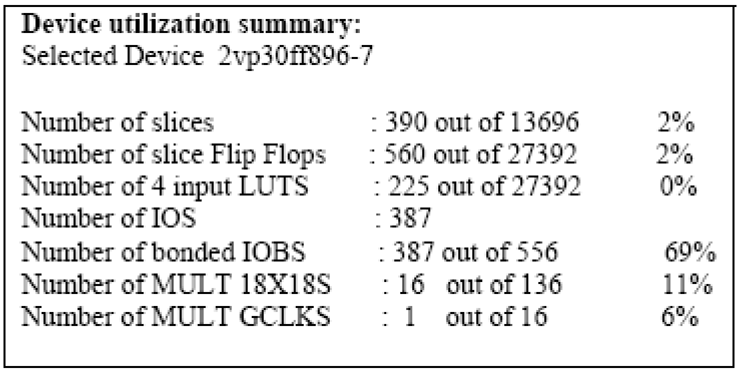
Synthesis results

The synthesis results of main module are as follows:

Macro Statistics		
# Multipliers	:	16
16x9-bit multiplier	:	8
9x9-bit multiplier	:	8
# Adders/Sub tractors	:	30
16-bit adder	:	7
26-bit adder	:	7
9-bit subtract or	:	16
# Registers	:	48
16-bit register	:	16
26-bit register	:	8
9-bit register	:	24

Figure 22: Synthesis results.

Table 1



Device utilization summary:
Selected Device 2vp30ff896-7

Number of slices	: 390 out of 13696	2%
Number of slice Flip Flops	: 560 out of 27392	2%
Number of 4 input LUTS	: 225 out of 27392	0%
Number of IOS	: 387	
Number of bonded IOBS	: 387 out of 556	69%
Number of MULT 18X18S	: 16 out of 136	11%
Number of MULT GCLKS	: 1 out of 16	6%

The synthesis results of the Fig 22illustrate the number of multipliers required in the first stage (i.e.9 x 9 multiplier) is 8.Also the number of multipliers required in the second stage (i.e.16 x 9 multiplier) is 8. Thus at any positive edge of clock pulse, the no. of multipliers utilized is 8 thereby reducing the hardware. The Table.1 shows the Device utilization summary of the Design.

Conclusion

The desired objective of design and implementation of DCTQ architecture using verilog HDL and the corresponding functional verification using MATLAB has been achieved with Excellent Energy compaction. All the second stage products are available by the 132nd clock pulse. We make use of two stage of multipliers, thus at every positive edge of the clock pulse only 8 multipliers are used in a parallel manner. Thus the hardware required is also reduced. The desired objective of data compression is achieved by quantization. Use of reparability property of DCT ensures that the row and column computations are evaluated in two separate steps. Thus the number of multiplications in every positive edge of clock pulse is reduced when compared to the worst case scenarios. Hence we can achieve less area and low power.

Acknowledgment

Mrs. Anubhuti Khare one of the authors is indebted to Director UIT RGPV Bhopal for giving permission for sending the paper to the journal. Manish Saxena is also thankful to the Chairmen, Bansal Instituteof Science & Technology Bhopal for giving permission to send the paper for publication. Last but not least, I would also like to thanks our HOD and colleagues for supporting us.

References

[1] C.HemasundaraRao and M. Madhavi “A Novel VLSI Architecture of Hybrid Image Compression Model based on Reversible Blockade Transform “International journal of Electronics circuits and systems January 2009.

[2] Ken Cabeen and Peter Gent, College of the Redwoods. “Image compression and Discrete cosine transform”.

[3] Discrete cosine transform (DCT): theory and its applications.” A paper by Syed Ali Khayam, Dept. of Electrical and Computer Engineering, Michigan state University.

[4] S.Ramachandran “Digital VLSI systems design” published by Springer 2007.

[5] B Heyne and J Goetz “A low power and high Quality implementation of the Discreate Cosine Transform”Ady Radio Sci .5.305-111, 2007

[6] Jongsun Park · Kaushik Roy “A Low Complexity Reconfigurable DCT Architecture to Trade off Image Quality for Power Consumption”, Received: 2 April 2007 / Revised: 16 January 2008 / Accepted: 30 April 2008 / Published online: 3 June 2008.

[7] S. Ramachandran And S. Srinivasan Department of Electrical Engineering “A Novel, Automatic Quality Control Scheme for Real Time Image Transmission”, India VLSI Design, 2002 Vol. 14 (4), pp. 329–335.

[8] Sungwook Yu and Earl E. Swartzlander Jr., Fellow, IEEE, “DCT Implementation with Distributed Arithmetic”, IEEE Transactions on Computers, VOL. 50, NO. 9, September 2001.

[9] B.Heyne and J.Gotze “A Low Power and high-quality implementation of Discreate Cosine Transform” Adv. Radio Sci,5,305-311 2007.

[10] Sunan Tugsinavisut, YoupyoHong, Daewook” Efficient Asynchronous Bundled Data Pipelines for DCT Matrix- Vector Multiplication”IEEE Transactions on VLSI Systems vol.13, no 4April 2004.