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

*Vijaya Prakash.A.M, **K.S.Gurumurthy

* Research scholar,Dr.MGR University, Chennai, Faculty, Department of ECE,BIT, Bangalore-560004, India

** Professor, Department of ECE, UVCE,Bangalore-560001, India,

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 black-and-white images. The functionality of the DCTQ is verified in MATLAB. The design is also implemented on Vertex2p to verify its capability on silicon. Results shows the area saving in terms of no of gate counts with an excellent efficiency.

Key words:

DCT, Image Compression, IDCT

1.0 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 most important frequency components are retained and they are used to retrieve the image through the decompression process. The loss of information in the reconstructed images can be controlled during the compression stage.

1.1 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 ith, jth entry of an image.

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

$$C(n) = \begin{cases} \frac{1}{\sqrt{N}} & \text{if } n = 0 \\ 1 & \text{if } n > 0 \end{cases}$$

(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 to 7 hence D(i,j) would be reduce as in Equation (2.0).

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

(2.0)
Figure 1 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 provide excellent energy compaction.

2. Design Flow.

![Flow Chart of Transmitter](image)

![Flow Chart of Receiver](image)

2.1 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 by 128 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.

3. 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.

The designed modules are:
1. MAIN MODULE
2. FIRST MULT
3. SECOND MULT
4. DCTQ CONTROLLER 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.

3.1 First 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:
\[
\text{out} = \frac{(a - 128) \times b}{128}
\]
Where, 'a' is image pixel value. 'b' is level shifted DCT coefficient. “out” is the computed product.

### 3.2 Second Multiplier

![Second stage Multiplier](image)

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:
\[
\text{out} = \frac{a \times (b - 128)}{128}
\]
Where, 'a' is first stage product. 'b' is level shifted DCT transpose coefficient. “out” is the computed product.

### 3.3 Main Module

![Main Module](image)

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.

### 3.4 DCTQ Controller

![DCTQ controller](image)

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.

### 3.5 Quantization Module

![Quantization Module](image)
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 quant_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.

### 3.6 Dual RAM Module

![Dual RAM Module](image)

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 Fig.13. 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”.

### 3.7 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 cp0s 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 x 8 block of image. The first stage products which are generated in stage one is temporarily stored in prod11 to prod88 registers. Clock counter value is incremented at every positive edge of the clock pulse. The products generated in the 2 stages are also sent to the register banks of dual RAM for storage purposes.

### 4. 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.

![Original Image and Image Matrix](image)
The Fig 15 shows 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.

\[
C = \\
\begin{bmatrix}
0.3536 & 0.4594 & 0.0191 & 0.2778 & 0.0975 & 0.2778 & 0.4904 & 0.2778 & 0.0975 \\
0.4594 & 0.4594 & 0.0191 & 0.2778 & 0.0975 & 0.2778 & 0.4904 & 0.2778 & 0.0975 \\
0.0191 & 0.0191 & -0.4619 & -0.1913 & -0.1913 & -0.1913 & 0.4619 & 0.1913 & 0.1913 \\
0.2778 & 0.2778 & -0.4619 & -0.1913 & -0.1913 & -0.1913 & 0.4619 & 0.1913 & 0.1913 \\
0.0975 & 0.0975 & 0.0191 & 0.2778 & 0.0975 & 0.2778 & 0.4904 & 0.2778 & 0.0975 \\
0.2778 & 0.2778 & 0.0191 & 0.2778 & 0.0975 & 0.2778 & 0.4904 & 0.2778 & 0.0975 \\
0.0975 & 0.0975 & 0.0191 & 0.2778 & 0.0975 & 0.2778 & 0.4904 & 0.2778 & 0.0975 \\
0.2778 & 0.2778 & 0.0191 & 0.2778 & 0.0975 & 0.2778 & 0.4904 & 0.2778 & 0.0975 \\
0.0975 & 0.0975 & 0.0191 & 0.2778 & 0.0975 & 0.2778 & 0.4904 & 0.2778 & 0.0975 \\
0.2778 & 0.2778 & 0.0191 & 0.2778 & 0.0975 & 0.2778 & 0.4904 & 0.2778 & 0.0975 \\
\end{bmatrix}
\]

The standard JPEG Quantization matrix

\[
\text{quant_mat} = \\
\begin{bmatrix}
16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\
14 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\
14 & 17 & 22 & 29 & 51 & 67 & 69 & 56 \\
18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\
24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\
49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\
72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \\
\end{bmatrix}
\]

The 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

\[
\text{compress_image} = \\
\begin{bmatrix}
80 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
11 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

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

\[
\text{idct_image} = \\
\begin{bmatrix}
239 & 263 & 239 & 99 & 99 & 239 & 263 & 239 \\
271 & 210 & 131 & 45 & 45 & 131 & 210 & 271 \\
236 & 107 & 20 & 60 & 60 & 20 & 107 & 236 \\
246 & 104 & 61 & 226 & 226 & 61 & 104 & 246 \\
246 & 104 & 61 & 226 & 226 & 61 & 104 & 246 \\
236 & 107 & 20 & 60 & 60 & 20 & 107 & 236 \\
271 & 210 & 131 & 45 & 45 & 131 & 210 & 271 \\
239 & 263 & 239 & 99 & 99 & 239 & 263 & 239 \\
\end{bmatrix}
\]

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.

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.
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.

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. 21 illustrates 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.

4.1 Synthesis results:

The synthesis results of main module are as follows:

<table>
<thead>
<tr>
<th>Macro Statistics</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td># Multipliers     : 16</td>
<td></td>
</tr>
<tr>
<td>16x9-bit multiplier : 8</td>
<td></td>
</tr>
<tr>
<td>9x9-bit multiplier : 8</td>
<td></td>
</tr>
<tr>
<td># Adders/Subtractors : 30</td>
<td></td>
</tr>
<tr>
<td>16-bit adder : 7</td>
<td></td>
</tr>
<tr>
<td>26-bit adder : 7</td>
<td></td>
</tr>
<tr>
<td>9-bit subtract or : 16</td>
<td></td>
</tr>
<tr>
<td># Registers : 48</td>
<td></td>
</tr>
<tr>
<td>16-bit register : 16</td>
<td></td>
</tr>
<tr>
<td>26-bit register : 8</td>
<td></td>
</tr>
<tr>
<td>9-bit register : 24</td>
<td></td>
</tr>
</tbody>
</table>

The synthesis results of the Fig 22 illustrate 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.

5. Conclusion.

The desired objective of design and implementation of DCTQ architecture using verilog HDL and the corresponding

![Image](image_url)
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.

References
[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.