Pipelined Architecture of 2D-DCT, Quantization and Zigzag Process for JPEG Image Compression using Verilog

Sunil W. Bhise

Abstract: **This paper presents the architecture and verilog design of a Two Dimensional Discrete Cosine Transform (2D-DCT) with Quantization and zigzag arrangement. This architecture is used as the core and path in JPEG image compression hardware. The 2D- DCT calculation is made using the 2D- DCT separability property, such that the whole architecture is divided into two 1D-DCT calculations by using a transpose buffer. Architecture for Quantization and zigzag process is also described in this paper. The quantization process is done using division operation. This design aimed to be implemented in Spartan-3E XC3S500E FPGA. The 2D-DCT architecture uses 2115 Slices and 11 multipliers of one Xilinx Spartan-3EXC3S500E FPGA reaches an operating frequency of 56.27 MHz One input block with 8 x 8 elements of 8bits each is processed in 4.603 ns and pipeline latency is 124 clock cycles.**

Keywords: **JPEG, discrete cosine transform (DCT), quantization, zigzag, FPGA**

1. INTRODUCTION

One of the most popular lossy compression methods is JPEG. JPEG stands for Joint Photographic Expert Group. According to Magli [6], widely used in JPEG image included on the internet web pages. The picture using JPEG can be accessed faster than the image without compression. The JPEG compression can be divided into five main steps , as shown in Fig.1 color space conversion, down-sampling, 2-D DCT, quantization and entropy coding. The first two operations are used only for color images. For gray scale image we use only last three steps. In this present paper we concentrated on hardware architecture of 2D-DCT, quantization and zigzag arrangement. To achieve high throughput, this paper uses pipelined architecture, rather than single clock architecture designed by Basri et.al [5].

Figure 1 –JPEG compression steps of color images

Department of E&TC, VDF school of Engineering & Technology, Latur, Maharashtra, India.
sunilbhise@aol.in

2 . DISCRETE COSINE TRANSFORM(DCT)

A. *Color Space Converter*

The process of the JPEG starts with color space conversion. This process is not applicable to gray-scale image, where there is only one luminance component for gray scale image. Color image data in computers is usually represented in RGB (Red-Green-Blue) format. Each color component uses 8 bits to store, thus, a full color pixel would require 24 bits. From the fact that human eyes are more sensitive to intensity change rather than color change, the JPEG algorithm exploits this by converting the RGB format to another color space called YCbCr. Y is luminance component, Cb and Cr are chrominance components. After converting the color space, the encoder stores the luminance Y in more detail than the other two chrominance components.The Y component represents the brightness of a pixel, the Cb and Cr components represent the chrominance (split into blue and red components). This is the same color space as used by digital color television as well as digital video including video DVDs, and is similar to the way color is represented in analog PAL video and MAC but not by analog NTSC, which uses the YIQ color space. The YCbCr color space conversion allows greater compression without a significant effect on perceptual image quality (or greater perceptual image quality for the same compression). The compression is more efficient as the brightness information, which is more important to the eventual perceptual quality of the image, is confined to a single channel, more closely representing the human visual system.

The RGB image is converted to YCbCr by using the following equations

$$Y = 0.299R + 0.587G + 0.114B$$
$$Cb = 0.564B - 0.564\,Y$$
$$Cr = 0.713R - 0.713\,Y$$

B. *Down Sampling*

Due to the densities of color- and brightness sensitive receptors in the human eye, humans can see considerably more fine detail in the brightness of an image (the Y component) than in the color of an image(the Cb and Cr components). Using this knowledge, encoders can be designed to compress images more efficiently. The

transformation into the YCbCr color model enables the next step, which is to reduce the spatial resolution of the Cb and Cr components (called "down sampling" or "chroma sub sampling"). The ratios at which the down sampling can be done on JPEG are 4:4:4 (no down sampling), 4:2:2 (reduce by factor of 2 in horizontal direction), and most commonly 4:2:0(reduce by factor of 2 in horizontal and vertical directions). For the rest of the compression process, Y, Cb and Cr are processed separately and in a very similar manner.

Down sampling the chroma components saves 33% or 50% of the space taken by the image without drastically affecting perceptual image quality.

C. Block Splitting

After sub sampling, each channel must be split into 8×8 blocks (of pixels). If the data for a channel does not represent an integer number of blocks then the encoder must fill the remaining area of the incomplete blocks with some form of dummy data: Filling the edge pixels with a fixed color (typically black) creates dark artifacts along the visible part of the border. Repeating the edge pixels is a common but non optimal technique that avoids the visible border, but it still creates artifacts with the colorimetric of the filled cells. A better strategy is to fill pixels using colors that preserve the DCT coefficients of the visible pixels, at least for the low frequency ones (for example filling with the average color of the visible part will preserve the first DC coefficient, but best fitting the next two AC coefficients will produce much better results with less visible 8×8 cell edges along the border).

D. Discrete Cosine Transform (DCT)

The discrete cosine transforms (DCT) is a technique for converting a signal into elementary frequency components. It is widely used in image compression. Before compression, image data in memory is divided into several blocks MCU (minimum code units). Each block consists of 8x8 pixels. Compression operations including DCT-2D in it will be done on each block [2]. Two dimensional DCT, because of its advantage in image compression, is an interesting research subject that makes many algorithms of DCT is developed.

E. 2-D Discrete Cosine Transform (DCT)

There are several ways to compute 2-D DCT. It can be computed with straightforward computation just multiply input vector by raw DCT coefficients without any algorithm [3]. This method is fast but need large logic utilization, especially multiplier. This method is fully pipelined in this paper. FPGA chip usually has only a few multipliers. In this case, Spartan-3E XCS500E has only 11 multipliers. This paper adopts the work of [3] The Discrete Cosine Transform is an orthogonal transform consisting of a set of basis vectors that are sampled cosine functions. The 2-D DCT of a data matrix is defined as equation (1)

$$Z = C \times C^{T}$$

Where X is the data matrix, C is the matrix of DCT Coefficients, and Ct is the Transpose of C.

$$Z = \begin{bmatrix} c_{00} & c_{01} & \dots & c_{07} \\ c_{10} & c_{11} & \dots & c_{17} \\ \vdots & \vdots & \vdots & \vdots \\ c_{70} & c_{71} & \dots & c_{77} \end{bmatrix} \begin{bmatrix} x_{00} & x_{01} & \dots & x_{07} \\ x_{10} & x_{11} & \dots & x_{17} \\ \vdots & \vdots & \vdots & \vdots \\ x_{70} & x_{71} & \dots & x_{77} \end{bmatrix} \begin{bmatrix} c_{00} & c_{10} & \dots & c_{70} \\ c_{01} & c_{11} & \dots & c_{71} \\ \vdots & \vdots & \vdots & \vdots \\ c_{07} & c_{17} & \dots & c_{77} \end{bmatrix}$$

Where, for an *N* x *N* data matrix.

The complete multiplication algorithm is presented in Table. 1, where:

in1 = cos(4d16) m3 = cos(2d16) - cos(6d16)
m2 = cos(6d16) m4 = cos(2dI6)+ cos(6dl6))

Step 1		
b0 = a0 + a7	*b1 = a1 + a6*	*b2 = a2 - a4*
b3 = a1 - a6	*b4 = a2 + a5*	*b5 = a3 + a4*
b6 = a2 - a5	*b7 = a0 - a7*	
Step 2		
c0 = b0 + b5	*c1 = b1 - b4*	*c2 = b2 + b6*
c3 = b1 + b4	*c4 = b0 - b5*	*c5 = b3 + b7*
c6 = b3 + b6	*c7 = b7*	
Step 3		
d0 = c0 + c3	*d1 = c0 - c3*	*d2 = c2*
d3 = c1 + c4	*d4 = c2 - c5*	*d5 = c4*
d6 = c5	*d7 = c6*	*d8 = c7*
Step 4		
e0 = d0	*e1 = d1*	*e2 = m3 x d2*
e3 = m1 x d7	*e4 = m4 x d6*	*e5 = d5*
e6 = m1 x d3	*e7 = m2 x d4*	*e8 = d8*
Step 5		
f0 = e0	*f1 = e1*	*f2 = e5 + e6*
f3 = e5 - e6	*f4 = e3 + e8*	*f5 = e8 - e3*
f6 = e2 + e7	*f7 = e4 + e7*	
Step 6		
S0 = f0	*S1 = f4 + f7*	*S2 = f2*
S3 = f5 - f6	*S4 = f1*	*S5 = f5 + f6*
S6 = f3	*S7 = f4 - f7*	

Table 1 1-D DCT Agostini algorithm

The 2-D DCT (8 x 8 DCT) is implemented by the row-column decomposition technique. We first compute the 1-D DCT (8 x 1 DCT) of each column of the input data matrix X to yield Xt C .after appropriate rounding or truncation, the transpose of the resulting matrix, Ct X, is stored in an transpose buffer. We then compute another 1-D DCT (8 x 1 DCT) of each row of Ct X to yield the desired 2-D DCT as defined in equation (1). A block diagram of the design is shown in Fig 2.

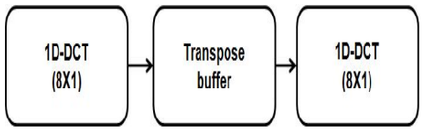

Figure 2: 2D-DCT Architecture

F. Quantization

Our 8x8 block of DCT coefficients is now ready for compression by quantization. A remarkable and highly useful feature of the JPEG process is that in this step, varying levels of image compression and quality are obtainable through selection of specific quantization matrices. This enables the user to decide on quality levels ranging from 1 to 100, where 1 gives the poorest image quality and highest compression, while 100 gives the best quality and lowest compression. As a result, the quality/compression ratio can be tailored to suit different needs. Subjective experiments involving the human visual system have resulted in the JPEG standard quantization matrix. With a quality level of 50, this matrix renders both high compression and excellent decompressed image quality. From [1] the Quantization matrix is obtained.

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

Figure 3. Quantization Matrix

G. Zigzag Reordering Buffer

Each block of data that is output by the quantization module needs to be reordered in a zigzag. This reordering is achieved using an 8 x 8 array of register pairs organized in a fashion similar to the transpose buffer. Quantized output is sent sequentially byte-by-byte in zigzag pattern. Zigzag operation is done for every 8X8 block. The pattern is shown in figure 2 [1]. Numbers listed in the figure are the address of 64 data that is arranged in a zigzag pattern.

```
 0  1  8 16  9  2  3 10
17 24 32 25 18 11  4  5
12 19 26 33 40 48 41 34
27 20 13  6  7 14 21 28
35 42 49 56 57 50 43 36
29 22 15 23 30 37 44 51
58 59 52 45 38 31 39 46
53 60 61 54 47 55 62 63
```

Figure.4 Zigzag Pattern

3. FPGA IMPLEMENTATION

3.1. System Architecture

The entire system architecture to be implemented in FPGA is shown in figure 5. Input data is inserted into the system every 8 bit sequentially. Actually, many DCT designs insert the input to the DCT in parallel. For example is 8 x 8 bit [1], [4],[8]. This is ideal for DCT computing because it only consumes a clock cycle to insert data to 1D-DCT unit. With sequential manner, it takes 8 clock cycles to insert a set of data (8 points) to the DCT unit. The sequential architecture is chosen to save I/O port in FPGA chip. Some 2D-DCT intellectual property designs from Xilin*x* also use 8-bit input [11]. The 8-bit input architecture is also fit to many camera modules.

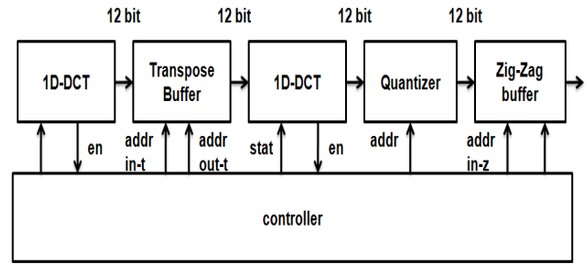

Figure.5 System Architecture

The 2D-DCT architecture, combined with zigzag and quantization used in this paper is shown in Fig. 5. The 2DDCT module construction is modified from [1] that also put the data sequentially into the module. Thus, the architecture of 2D-DCT was divided into two 1D DCT modules and one transpose buffer. The same 1D DCT module is used twice. The transpose buffer operates like a temporal barrier between the first and the second 1D DCT. It made from static RAM with two sets of data and address bus. One for read process and the other for write.

3.2.1D-DCT Pipeline Process

Since the DCT input/output has 8 points and data has to be entered and released in sequential manner, it takes 8 clock cycles for each input and output process. Totally, 8 points 1D-DCT computation needs 22 clock cycles. Design for data input and output in this paper is inspired by design from [1]. The input and output process visualization is shown in figure 6. In this paper, system computes every step in a clock cycle, so DCT computation can be done faster.

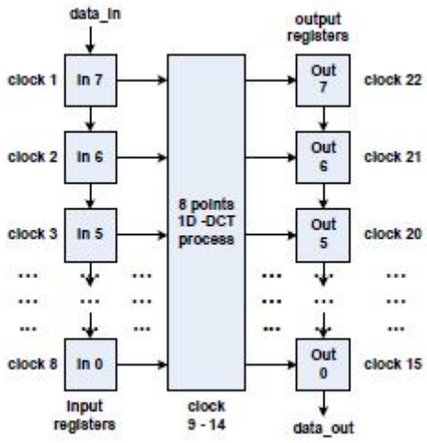

Figure.6 Data input/output process visualization of clock cycles for 8 points 1D-DCT

3.3. Transpose Buffer

Transpose buffer is static RAM, designed with two set of data and address bus. It has input and output data and address buses, the structural construction of transpose buffer is shown in fig.7. The input to the transpose buffer comes from output of first 1D-DCT. Address in, out, and WE (write enable) are generated from controller module. Input address is generated in normal sequence (0,1,2,3,4,5,6, …, 63) but output address is generated in transposed sequence (0,8,16,24,32,40,48,56,1,9,17,..,55, 63). Output process begins after the entry of all 64 inputs. It gives the time latency between first input and first output. The output of transpose buffer is fed directly to the input of second 1D-DCT unit and the chain of sequences continues in same order.

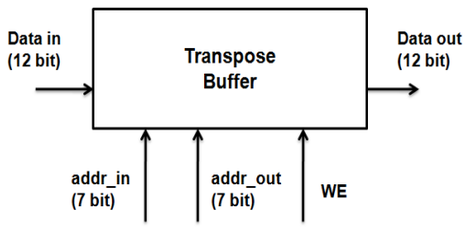

Figure.7 Transpose Buffer Block Diagram

3.4. Quantizer

The Quantization process in JPEG image compression is done by dividing each and every 2DDCT coefficient by quantizing values from quantization table shown in fig 3. This quantizer module consists of ROM and divider. These quantizing values are first stored in ROM. The divider carries out division in a pipelined manner. The first DCT coefficient coming out from 2DDCT module is divided by the first value from the quantization table (which was already stored in ROM), and second DCT coefficient is divided by second value from the table, like wise total 64 coefficients are divided by the values in quantization table. In this quantization process also we used pipeline architecture. Block Diagram of the implementation is shown in fig 8.

3.5. Zigzag Buffer

Zigzag buffer is made from static RAM. Its construction is like transpose buffer. It has two sets of data – address bus. Input address bus is accessed by normal sequence, but output address is given some zigzag sequence described in fig 4. Zigzag address is generated by a zigzag RAM. The sequence is stored in the RAM. When the RAM address bus is accessed by normal address sequence, RAM data bus will emit zigzag value. Figure 8 describe zigzag buffer and RAM construction in the system.

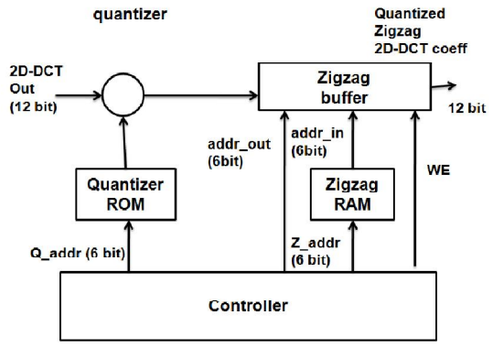

Figure.8 Quantization & Zigzag Architecture

4. SIMULATION RESULTS

The 2-D DCT, Quantization and Zigzag architecture was described in verilog. This VERILOG was synthesized into a Xilinx Spartan 3E family FPGA [7]. System is tested with gray scale image. Simulation of VERILOG values are compared with MATLAB values. The complete synthesis results to Spartan-3E FPGA are presented in table 2, whose hardware was fit in an XCS500E device.

Logic Utilization	Used	Available	Utilization
Number of Slices	2115	4656	45%
Number of Slice Flip Flops	1908	9312	20%
Number of 4 input LUTs	3134	9312	33%
Number of bonded IOBs	103	66	156%
Number of MULT18X18SIOs	11	20	55%
Number of GCLKs	1	24	4%

Table 2.device utilization using Xilinx spartran-3E for total architecture proposed in this paper.

Number	*Verilog result*	*Matlab result*
0	18	18
1	4	4
2	1	-3
3	-2	0
4	-1	-1
5	-4	1
6	0	0
7	-1	0

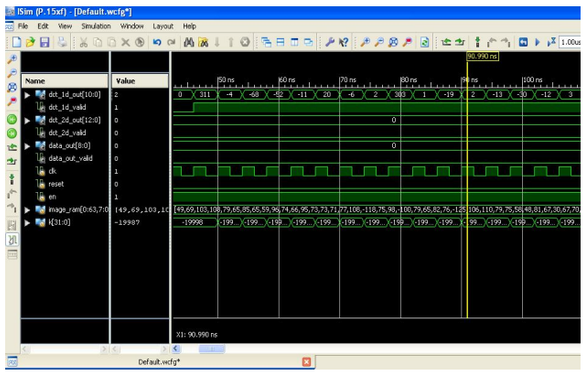

Figure 9 : Output only after 1-D DCT

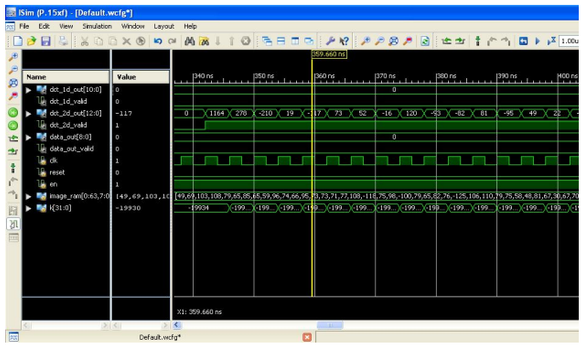

Figure 10 : Output only after 2-D DCT

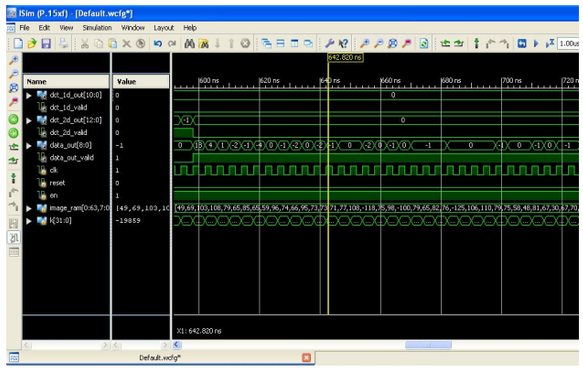

Figure 11 : Output only after quantization and zigzag

5. CONCLUSION

2D-DCT combined with quantization and zigzag buffer is designed using Verilog system is tested with real grayscale image. The accuracy of computation is compared to Matlab computation result with similar operation. Pipeline process causes latency in the system .Latency produced from this system is 124 clock cycles .Maximum frequency can be achieved by this system is 56.27 Mhz. Sequential pipeline design gives higher frequency than fully parallel design in. The design takes 2115 slices ,1908 slice filp flops and 3134 LUTs including the control block that suitable for low cost FPGA like Xilinx XCS500E.The sequential operation of DCT saves much logic utilization in FPGA. Each step of DCT algorithm is executed on each block cycle.

6. REFERENCES

1. Kusuma E.D. ;Widodo T.S.,"FPGA Implementation of Pipelined 2D-DCT and Quantization Architecture for JPEG Image Compression'. IEEE 2010

2. L. Agostini, S. Bampi, "Pipelined Fast 2-D DCT Architecture for JPEG Image Compression" Proceedings of the 14th Annual Symposium on Integrated Circuits and Systems Design, Pirenopolis, Brazil. IEEE Computer Society 2001. pp 226-231.

3. Y. Arai, T. Agui, M. Nakajima. "A Fast DCT-SQ Scheme for Images". Transactions of IEICE, vol. E71, nÂ. 11, 1988, pp.1095-1097.

4. D. Trang, N. Bihn, "A High-Accuracy and High-Speed 2-D 8x8Discrete Cosine Transform Design". Proceedings of ICGCRCICT2010, vol. 1, 2010, pp. 135-138

5. I. Basri, B. Sutopo, "Implementasi 1D-DCT Algoritma Feig-Winograd di FPGA Spartan-3E (Indonesian)". Proceedings of CITEE 2009, vol. 1, 2009, pp. 198-203

6. E. Magli, “The JPEG Family of Coding Standard,” Part of “Document and Image Compression”, New York: Taylor and Francis, 2004.

7. Wallace, G. K. ,"The JPEG Still Picture Compression Standard", Communications of the ACM, Vol. 34, Issue 4, pp.30-44. 1991.

8. Sun, M., Ting C., and Albert M., ‘‘VLSI Implementation of a 16X 16 Discrete Cosine Transform’’, IEEE Transactions on Circuitsand Systems, Vol. 36, No. 4, April 1989.

9. Xilinx, Inc., “Spartan-3E FPGA Family : Data Sheet ”, Xilinx Corporation, 2009.

10. Omnivision, Inc., “OV9620/9120 Camera Chip Data Sheet ”,Xilinx Corporation, 2002.

11. Xilinx, Inc., “2D Discrete Cosine Transform (DCT) V2.0 ”,Logicore Product Specification, Xilinx Corporation, 2002.

12. “Home site of the JPEG and JBIG committees”<http://www.jpeg.org/> (21/04/01)

13. W. Chen, C. Smith, **S.** Fralick. “A Fast Computational Algorithm for the Discrete Cosine Transform”. *IEEE Transactions on Communications*, v. COM-25, n. 9, p. 1004-1009, 1977.

14. E. Feig, S. Winograd. “Fast Algorithms for the Discrete Cosine Transform”. *IEEE Transactions on Signal Processing*,v.40, n. 9, p. 2174-2193, 1992.

15. M. Kovac, N. Ranganathan. “JAGUAR: A Fully PipelineVLSI Architecture for JPEG Image CompressionStandard”.*Proceedings of the IEEE*, vol. 83, no. 2, 1995, pp. 247-258.