PIPELINED ARCHITECTURE OF 2D-DCT, QUANTIZATION AND ZIGZAG PROCESS FOR JPEG IMAGE COMPRESSION USING VHDL

T.Pradeepthi[1] and Addanki Purna Ramesh[2]

Department of ECE, Sri Vasavi Engg College, Tadepalligudem, West Godavari (dt), Andhra Pradesh, India

purnarameshaddanki@gmail.com,pradeepthi.t@gamil.com

ABSTRACT

*This paper presents the architecture and VHDL 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 XC3S500 FPGA. The 2D- DCT architecture uses **1891** Slices, **51**I/O pins, and **8** multipliers of one Xilinx Spartan-3E XC3S500E FPGA reaches an operating frequency of **101.35** MHz One input block with 8 x 8 elements of 8 bits each is processed in 6604 ns and pipeline latency is 140 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 [3], 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 [6], 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 [4].

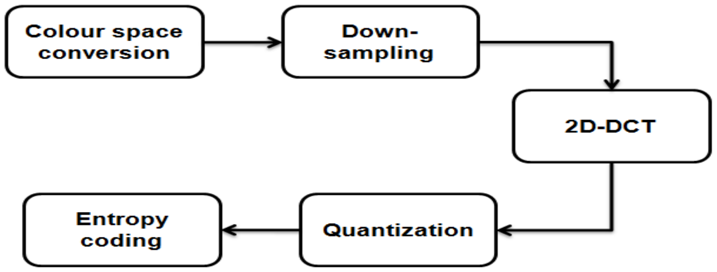


Fig.1 JPEG Compression Steps for Colour Images

1.1. 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

$$Y= 0.299R+ 0.587G +0.114B$$

$$Cb=0.564B- 0.564\ Y$$

$$Cr=0.713R- 0.713\ Y$$

1.2. 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.

1.3. 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 nonoptimal 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).

1.4. 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 [3]. Two dimensional DCT, because of its advantage in image compression, is an interesting research subject that invite many researcher [1],[2], [4], [6] and others to participate in. That makes many algorithms of DCT is developed.

1.4.1. 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 [1]. 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 20 multipliers. This paper adopts the work of [2] 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)

$$Z = CXC^t \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

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

An expanded form of (1) is as followed:

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

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

$$c_{k,l} = \sqrt{\frac{2}{N}} \cos\left[\frac{(2k-1)(l-1)\pi}{2N}\right]. \qquad \ldots(2)$$

For $k = 1, 2\ldots N$, $l = 2, 3\ldots N$, and $c_{k,\,l} = N^{-1/2}$ for $l = 1$.

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 X^tC. after appropriate rounding or truncation, the transpose of the resulting matrix, C^tX, is stored in an transpose buffer. We then compute another 1-D DCT (8 x 1 DCT) of each row of C^tX to yield the desired 2-D DCT as defined in equation (1). A block diagram of the design is shown in Fig 2.

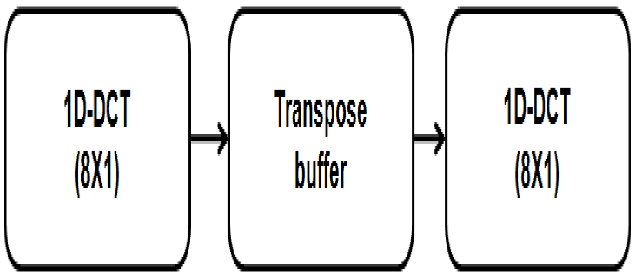


Fig 2: 2D-DCT Architecture

1.5. 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 [5] the Quantization matrix is obtained.

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

Fig 3. Quantization Matrix

1.6. 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 [5]. 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

Fig.4 Zigzag Pattern

2. FPGA IMPLEMENTATION

2.1. 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],[6]. 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 [10]. The 8-bit input architecture is also fit to many camera modules.

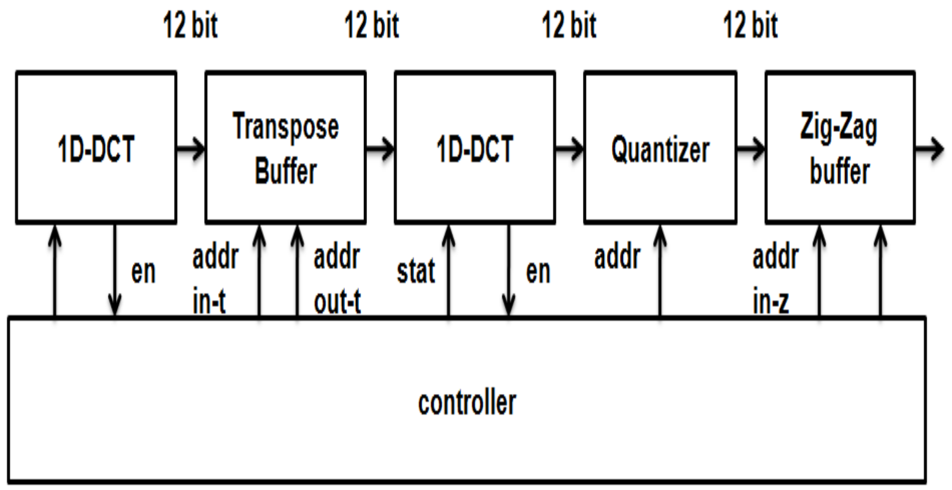


Fig.5 System Architecture

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 [2] 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.

2.2.1D-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 [2]. 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.

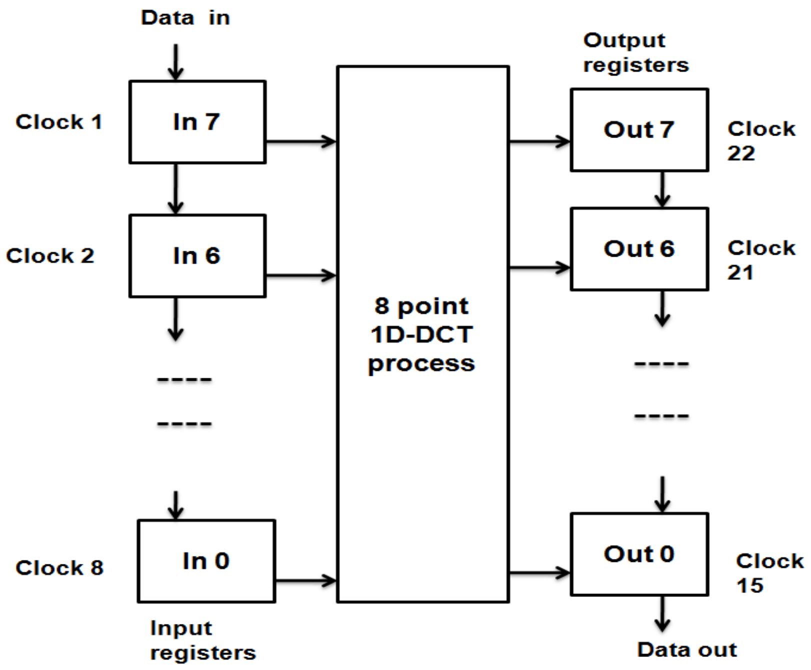
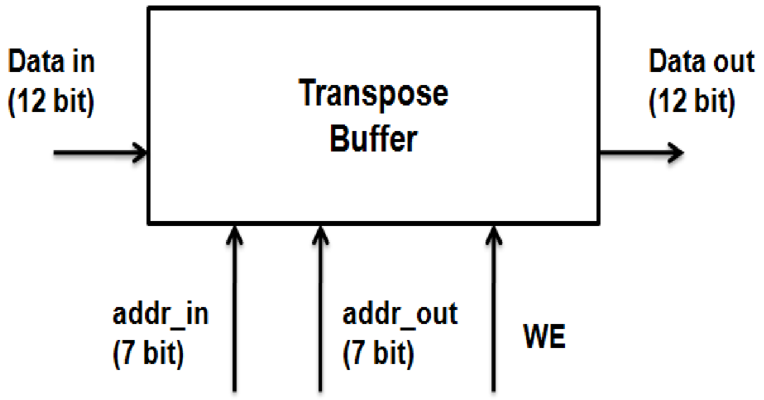


Fig.6 Data input/output process visualization of clock cycles for 8 points 1D-DCT

2.3. 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.



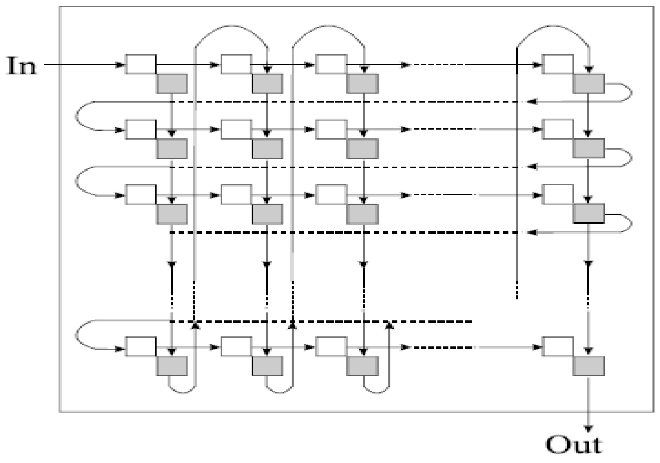


Fig.7 Transpose Buffer Block Diagram

2.4. Quantizer

The Quantization process in JPEG image compression is done by dividing each and every 2D-DCT 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 2D-DCT 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.

2.5. 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.

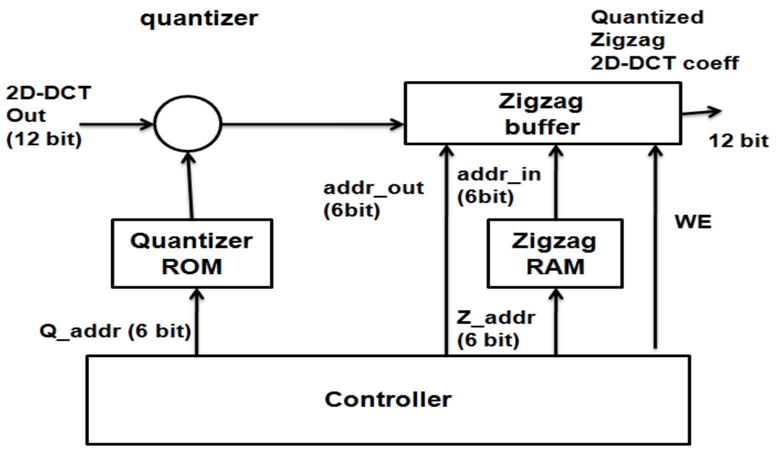


Fig.8 Quantization & Zigzag Architecture

3. SIMULATION RESULTS

The 2-D DCT, Quantization and Zigzag architecture was described in VHDL. This VHDL was synthesized into a Xilinx Spartan 3E family FPGA [7]. System is tested with gray scale image. Simulation of VHDL values are compared with MATLAB values. The complete synthesis results to Spartan-3E FPGA are presented in table 1, whose hardware was fit in an XCS500E device. The table 2 presents the comparison between [1] and the present work in this paper.

Table 1.device utilization using Xilinx spartran-3E for total architecture proposed in this paper.

Logic Units	Used	Available	Utilization
Number of slices	1891	4656	40%
Number of slices FFs	2450	9312	26%
Number of 4 input LUTs	1671	9312	17%
Number of Bonded IOBs	51	231	21%
Number of multipliers 18x18	8	20	40%

Table 2.present paper's 2D-DCT result is compared with work of 2D-DCT design of [1].

Logic Units	This paper (only 2D-DCT)	Presented[1]
Number of slices	1235	7260
Number of slices FFs	1551	9644
Number of 4 input LUTs	1239	11194
Number of Bonded IOBs	23	101
Number of multipliers 18x18	8	-

Table 3. Present paper's 2D-DCT (with Quantization & zigzag process) result is compared with result with [8].

Logic units	This paper(2D-DCT,quantization,zigzag)	Presented [8]
Number of 4 input LUTs	1671	5276
Number of slices	1891	3070
Clock freq(MHz)	101.35	31.1

According to synthesis result, maximum time delay produced is 9.004 ns. That constraint yields minimum clock period 9.866 ns. Maximum clock frequency can be used is 101.355MHz. Maximum delay synthesized is much smaller than delay produced in [4]. 1D-DCT designed in [4] yields maximum time delay 76.03 ns.System [4] uses fully parallel processing without clock to compute 8 points 1D-DCT. That system is used as comparison reference because it uses same FPGA with this system. Since the system in paper [1] uses Vertex FPGA that has higher frequency than Spartan, the delay is much smaller than this system and maximum frequency is higher. Maximum frequency in [1] is 308.182 MHz this system is also faster than 2D-DCT described in [6]. System [6] has minimum period 82.1 ns.The uses of pipeline process gives the system latency. When results are compared with [8] the slices, LUT's are decreased and clock frequency is increased to 101MHz. The output exists several clock cycles after the first input. Latency produced in 2D-DCT is 94 clock cycles, quantizer output at 118 clock cycles. Overall system (quantized and zigzag 2D-DCT) has latency 140 clock cycles. As comparison, 2D-DCT designed in [6] has latency 160 clock cycles. The better result reached by system in [1]. It takes 37 clock cycles as system latency to compute 2D-DCT.

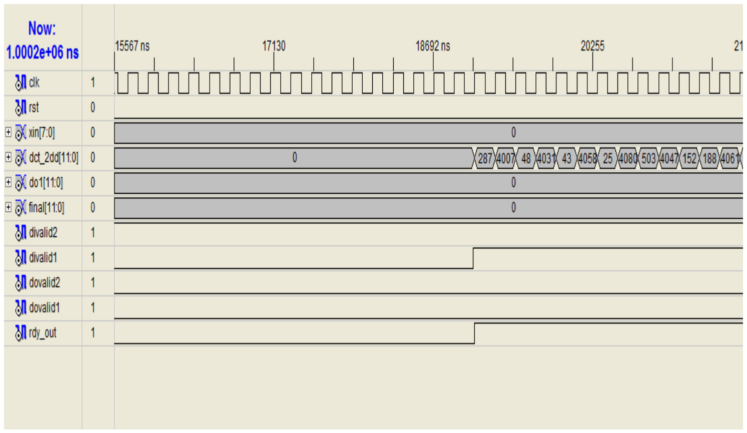


Fig 9 Output for only 2D-DCT

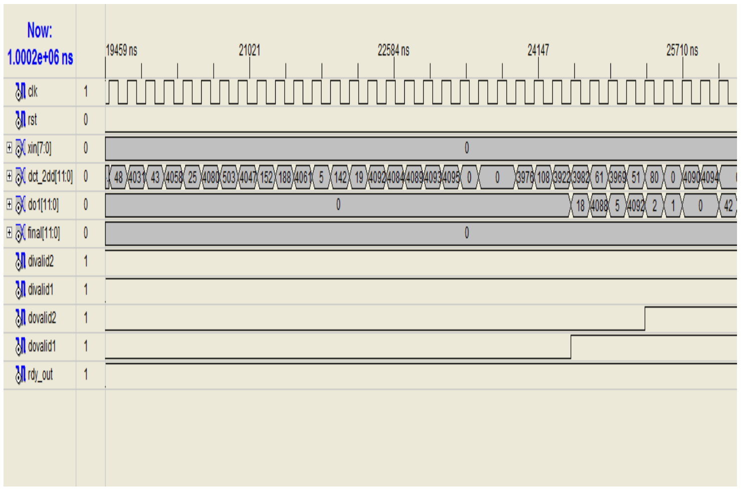


Fig 10 Output after for Quantization

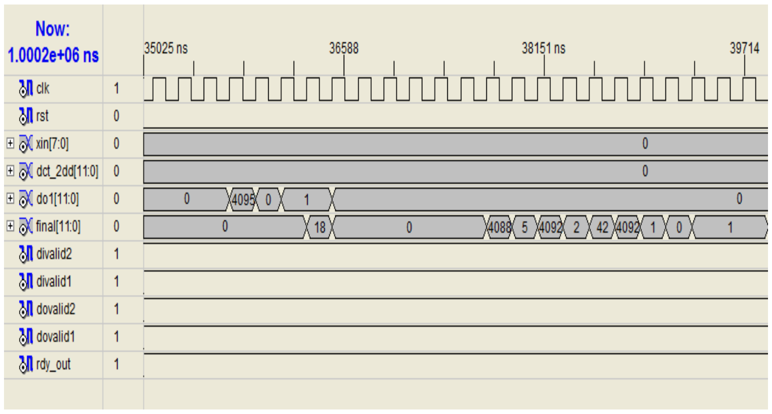
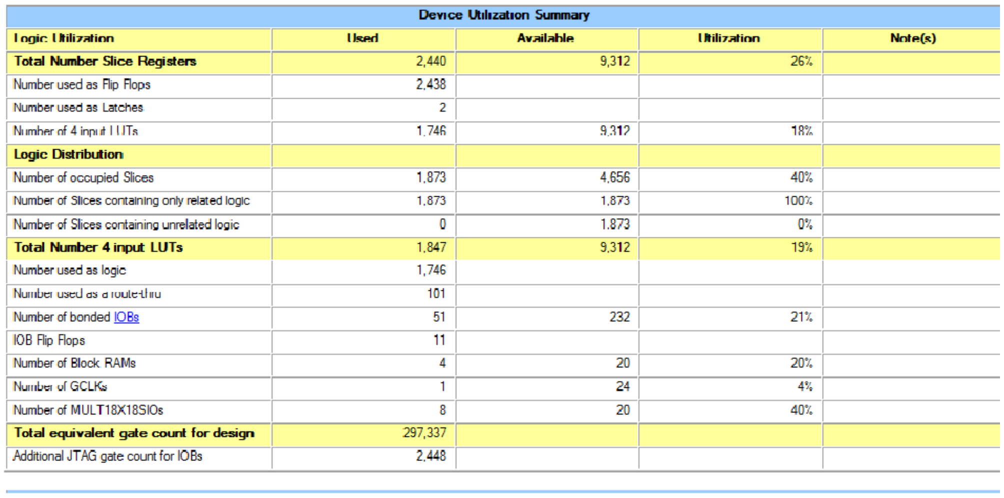


Fig 11 Output after zigzag



Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Total Number Slice Registers	2,440	9,312	26%	
Number used as Flip Flops	2,438			
Number used as Latches	2			
Number of 4 input LUTs	1,746	9,312	18%	
Logic Distribution				
Number of occupied Slices	1,873	4,656	40%	
Number of Slices containing only related logic	1,873	1,873	100%	
Number of Slices containing unrelated logic	0	1,873	0%	
Total Number 4 input LUTs	1,847	9,312	19%	
Number used as logic	1,746			
Number used as a route-thru	101			
Number of bonded IOBs	51	232	21%	
IOB Flip Flops	11			
Number of Block RAMs	4	20	20%	
Number of GCLKs	1	24	4%	
Number of MULT18X18SIOs	8	20	40%	
Total equivalent gate count for design	297,337			
Additional JTAG gate count for IOBs	2,448			

Fig 12 Device Utilization Summery

4. Conclusion

The jpeg image compression is designed in VHDL and is tested with gray scale image. The accuracy of computation is compared to Matlab computation result with similar operation. This Comparison yields Mean Square Error value MSE = 0.060552, which is computed for 64 bits of data in pipelined process causes latency in the system. The latency produced from this system is 140 clock cycles. Maximum frequency can be achieved by this system is 101.35 MHz Sequential pipeline design gives higher frequency than fully parallel design in [4]. The design takes 1891 slices, 2450 slice FF, 1671 LUT's and 8 multipliers such that the area is also reduced when compared to previous work. It is suitable for implementing on FPGA like Xilinx XCS500E. The sequential operation of DCT saves logic utilization in FPGA compared with [1] to a much larger extent. Each step of DCT algorithm is executed on each clock cycle. Every step consists of 8-9 operation. Thus this method is fast and had very less complexity.

5. References

[1] Trang T.T. Do, Binh P. Nguyen "A High-Accuracy and High-Speed 2-D 8x8 Discrete Cosine Transform Design". Proceedings of ICGCRCICT 2010, vol. 1, 2010, pp. 135-138.

[2] Sun, M., Ting C., and Albert M., ''VLSI Implementation of a 16 X 16 Discrete Cosine Transform'', IEEETransactions on Circuits and Systems, Vol. 36, No. 4, April 1989.

[3] E. Magli, "The JPEG Family of Coding Standard," Part of "Document and Image Compression", New York: Taylor and Francis, 2004.

[4] I. Basri, B. Sutopo, "Implementation 1D-DCT Algoritma Feig- Wino grad di FPGA Spartan-3E (Indonesian)". Proceedings of CITEE 2009, vol. 1, 2009, pp. 198-203

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

[6] 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.

[7] Xilinx, Inc., "Spartan-3E FPGA Family : Data Sheet ", Xilinx Corporation, 2009.

[8] Vijay Kumar Sharma, Umesh C. Pati, and K. K. Mahapatra "A Simple VLSI Architecture for Computation of 2-D DCT, Quantization and Zig-zag ordering for JPEG".

[9] Omnivision, Inc., "OV9620/9120 Camera Chip Data Sheet ", Xilinx Corporation, 2002.

[10] Xilinx, Inc., "2D Discrete Cosine Transform (DCT) V2.0 ", Logicore Product Specification, Xilinx Corporation, 2002.

[11] A. Shams, A. Chidanandan, W. Pan, and M. Bayoumi, "NEDA: A low power high throughput DCT architecture", IEEE Transactions on Signal Processing, vol.54(3), Mar. 2006.

[12] Peng Chungan, Cao Xixin, Yu Dunshan, Zhang Xing,"A 250MHz optimized distributed architecture of 2D 8x8 DCT",7th International Conference on ASIC, pp. 189 – 192, Oct. 2007.

[13] B.G. Lee," A new algoritm to compute the discrete cosine transform" —IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-32, pp. 1243-1245, Dect.1984.

[14] H.S Hou, "A fast recursive algorithms for computing the discrete cosine transform", —IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-35, pp. 1455-1461, Oct.1987.

[15] N.I Cho and S.U.Lee, "DCT algorithms for VLSI parallel implementation",—IEEE Trans. Acoust., Speech, Signal Processing, vol 38. pp. 121-127, Jan.1990.

[16] Nam Ik Cho, Sang Uk Lee , "Fast Algorithm and Implementation of 2-D Discrete Cosine Transform", —IEEE Transaction on Circuits and Systems‖, Vol.38,No.3, March 1991.

[17] N. Ahmed, T.Natarajan, and K.R. Rao, "Discrete Cosine Transform", IEEE Trans. Commun., vol, COM-23, pp. 90-93, Jan. 1974

[18] S. Ramachandran, S. Srinivasan and R. Chen, "EPLD-based Architecture of Real Time 2D-Discrete Cosine Transform and Quantization for Image Compression", IEEE International Symposium on Circuits and Systems (ISCAS '99), Orlando, Florida, May–June 1999.

[19] Hassan EL-Banna, Alaa A. EL-Fattah "An Efficient Implementation of the 1D DCT using FPGA Technology", Proceedings of the 11th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS'04),2004.

[20] Y. Arai, T. Agui, and M. Nakajima, "A fast DCT-SQ scheme for images", Trans IEICE, Vol. E71, No. 11, pp 1095-1097, 1998.

[21] Long- Wen Chang and Ching-Yang Wang and Shiuh-Ming Lee "Designing JPEG Quantization Tables Based On Human Visual System,"Proceedings. 1999 International Conference on Image Processing, ICIP 99, pp.376 - 380 vol.2, Oct.1999.

[22] Douglas J. Smith, "HDL Chip Design – A practical guide for designing, synthesizing, and simulating ASICs and FPGAs using VHDL or Verilog", Doone Publications.