Volume 01, No. 01

International Journal of Scientific Engineering and Technology Research

Jul-Dec 2012, P.P. 60-66

www.semargroups.org

Implementation of Pipelined Architecture Based on the DCT and Quantization For JPEG Image Compression

A.PAVANI[1],C.HEMASUNDARA RAO[2],A.BALAJI NEHRU[3]

[1]*M.Tech Student of* CMRIT, Ranga Reddy(Dt), AP-India,e-mail:pavani.672@gmail.com,

[2]Asst Professor, ECE Dept, CMRIT, Ranga Reddy(Dt), AP-India,

[3]HOD, ECE Dept, CMRIT, Ranga Reddy(Dt), AP-India

Abstract: Two dimensional DCT takes important role in JPEG image compression. Architecture and VHDL design of 2-D DCT, combined with quantization and zig-zag arrangement, is described in this paper. The architecture is used in JPEG image compression. DCT calculation used in this paper is made using scaled DCT. The output of DCT module needs to be multiplied with post-scaler value to get the real DCT coefficients. Post-scaling process is done together with quantization process. 2-D DCT is computed by combining two 1-D DCT that connected by a transpose buffer. This design aimed to be implemented in cheap Spartan-3E XC3S500 FPGA. The 2-D DCT architecture uses **3174** gates, **1145** Slices, **21** I/O pins, and **11** multipliers of one Xilinx Spartan-3E XC3S500E FPGA and reaches an operating frequency of **84.81** MHz. One input block with 8 x 8 elements of 8 bits each is processed in 2470 ns and pipeline latency is 123 clock cycles.

Keywords: *DCT, MCU, I/O pins,Quantization.*

I. INTRODUCTION

Data compression method is different depending on the type of data. For information in the form of images, one of the most popular compression method is JPEG. JPEG stands for Joint Photographic Expert Group. According to Magli [5], widely used in JPEG image included on the internet web pages. Use JPEG create a web page with a picture can be accessed faster than a web page with an image without compression Color image JPEG compression consists of five steps [1]. This is shown in figure 1. The steps are: color space conversion, down sampling, 2-D DCT, quantization and entropy coding. Grayscale image compression uses only last three steps.

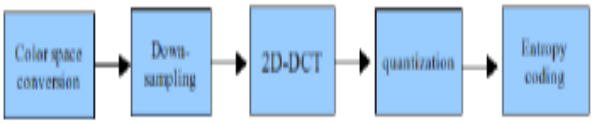


Figure 1 –JPEG compression steps of color images

However, this paper's main interest is only on hardware implementation of 2-D DCT combined with quantization and zig-zag process. To achieve high throughput, this paper uses pipelined architecture, rather than single clock architecture designed by Basri et.al [4].

II. DISCRETE COSINE TRANSFORM(DCT)

Using DCT-2D, pixel values of a spatial image in the region will be transformed into a set of DCT coefficients in the frequency region. 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 [5].

Two dimensional DCT, because of its advantage in image compression, is an interesting research subject that invite many researcher [1] ,[2], [3], [4], [7] and others to participate in. That makes many algorithms of DCT is developed.

A. 1-D Discrete Cosine Transform (DCT)

There are several ways to compute 1-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. The other

Copyright @ 2012 SEMAR GROUPS TECHNICAL SOCIETY. All rights reserved.

ways is computing DCT with multiplier reduction algorithm [1],[2],[4]. Reduced multiplier means reduced complexity. FPGA chip usually has only a few multipliers. In this case, Spartan-3E XCS500E has only 20 multipliers. This paper adopts the work of Agostini [1] that implemented Arai scaled 1-D DCT algorithm [2]. It means the DCT coefficients produced by the algorithm are not the real coefficients. To get the real coefficients, the scaled ones must be multiplied with post-scaler value. Equation (1) is showing scaled 1-D DCT process.

$$\mathbf{y' = C\,x} \qquad (1)$$

Variable **x** is 8 point vector. **C** is Arai's DCT matrix and **y'** is vector of scaled DCT coefficients.

Step 1:	$a_0 = x_0 + x_7$ $a_1 = x_1 + x_6$ $a_2 = x_3 - x_4$ $a_3 = x_1 - x_6$ $a_4 = x_2 + x_5$ $a_5 = x_3 + x_4$ $a_6 = x_2 - x_5$ $a_7 = x_0 - x_7$
Step 2:	$b_0 = a_0 + a_5$ $b_1 = a_1 - a_4$ $b_2 = a_2 + a_6$ $b_3 = a_1 + a_4$ $b_4 = a_0 - a_5$ $b_5 = a_3 + a_7$ $b_6 = a_3 + a_6$ $b_7 = a_7$
Step 3:	$d_0 = b_0 + b_3;$ $d_1 = b_0 - b_3;$ $d_2 = b_2;$ $d_3 = b_1 + b_4;$ $d_4 = b_2 - b_5;$ $d_5 = b_4;$ $d_6 = b_5;$ $d_7 = b_6;$ $d_8 = b_7;$
Step 4:	$e_0 = d_0;$ $e_1 = d_1;$ $e_2 = m_3 * d_2;$ $e_3 = m_1 * d_7;$ $e_4 = m_4 * d_6;$ $e_5 = d_5;$ $e_6 = m_1 * d_3;$ $e_7 = m_2 * d_4;$ $e_8 = d_8;$
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:	$y'_0 = f_0;$ $y'_1 = f_4 + f_7;$ $y'_2 = f_2;$ $y'_3 = f_5 - f_6;$ $y'_4 = f_1;$ $y'_5 = f_5 + f_6;$ $y'_6 = f_3;$ $y'_7 = f_4 - f_7;$

Table 1. 1D – DCT Algorithm from Agostini et.al.[1]

The algorithm produces **y'** vector as scaled DCT coefficients from input vector **x**. The 1D DCT post-scaler in equ. 1, **s** vector is defined in (3).

$$s = \begin{bmatrix} c4 \\ c7/c6 \\ c6/c4 \\ c5/c2 \\ c4 \\ c3/c2 \\ c2/c4 \\ c1/c6 \end{bmatrix} \qquad (3)$$

where cn = cos (nπ/16).

To get the real DCT coefficients, y' must be element by element multiplicated with post-scaling factor. It is shown in (2).

$$\mathbf{y = s\ .*\ y'} \qquad (2)$$

Constant **s** is vector of post-scaling factor. Element by element multiplication is shown with ".*' operator adopted from MATLAB. Output vector **y** is the real DCT coefficients. The DCT matrix **C** will not be discussed in this paper. The complete 1D-DCT algorithm from Agostini el.al. [1] is presented in Table. 1, where:

• m1 = cos(4π/16) • m3 = cos(2π/16) – cos(6π/16)

• m2 = cos(6π/16) • m4 = cos(2π/16)+ cos(6π /16))

B. Two Dimensional Discrete Cosine Transform (2DDCT)

2D-DCT is made from two 1D-DCT. Using DCT matrix, scaled 2D-DCT operation was shown in (4).

$$\mathbf{Y' = [C]\,[X]\,[C]T} \qquad (4)$$

X = 8x8 input matrix

Y' =Scaled 2D-DCT 8x8 output matrix

C = DCT matrix , same as matrix in equ. 1.

To get 2D-DCT real value, Y' must be element by element multiplied by post-scaler as shown in (5). Now, the post-scaler is in matrix form.

$$\mathbf{Y = [S].*[Y']} \qquad (5)$$

with **S** = post scaler matrix and Y = real DCT coefficients. Post scaler matrix **S** is computed in (6).

$$\mathbf{S = s\ sT} \qquad (6)$$

C. Quantization

Quantized output is made by divide each DCT coefficient by quantization value. It is done by doing element by element matrix multiplication between DCT output and quantization matrix as shown in (7) and (8).

$$Yq = [Q] .*[Y] \quad (7)$$

$$Yq = Q .* [S].*[Y'] \quad (8)$$

Since the image used in this paper is only grayscale picture, the quantization matrix applied is only for the luminance. The matrix was modified from [6] and postscaled by matrix S then multiplied with 212 to produce integer number in order to be applied in VHDL. The scaled quantization matrix was shown in (9).

$$Q = \begin{bmatrix} 32 & 34 & 39 & 27 & 21 & 16 & 19 & 30 \\ 31 & 22 & 20 & 17 & 14 & 8 & 11 & 24 \\ 28 & 22 & 19 & 14 & 10 & 9 & 10 & 25 \\ 31 & 18 & 15 & 13 & 9 & 6 & 10 & 25 \\ 28 & 17 & 11 & 8 & 8 & 6 & 9 & 24 \\ 27 & 13 & 9 & 9 & 8 & 8 & 11 & 26 \\ 19 & 11 & 9 & 9 & 9 & 19 & 15 & 34 \\ 26 & 15 & 15 & 16 & 17 & 24 & 33 & 68 \end{bmatrix} \quad (9)$$

D. Zig-zag process

Quantized output is sent sequentially byte-by-byte in zig-zag pattern. Zig-zag operation is done for every 8X8 block. The pattern is shown in figure 2 [6]. 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 2 – Zig-zag pattern for output data

III. IMPLEMENTATION

A. Entire System

Block diagram of entire system to be implemented in FPGA is shown in figure 3. 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],[3],[4]. 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 Xilinx also use 8-bit input [10]. The 8-bit input architecture is also fit to many camera modules like OV9620. The camera module has 8-bit output [9].

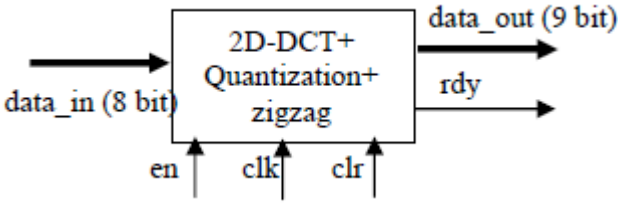


Figure 3 – Block representation of entire system

B. System Architecture

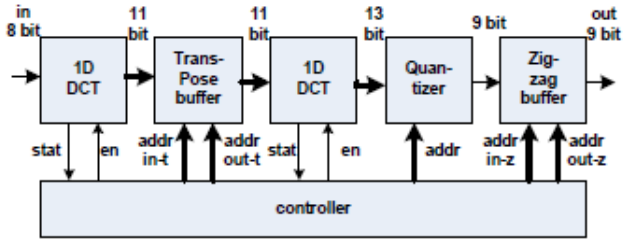


Figure 4 – 2D-DCT combined with zigzag and quantizer architecture

The 2D-DCT architecture, combined with zigzag and quantization used in this paper is shown in Fig. 4. The 2DDCT module construction is modified from [7], that also puts the data sequentially into the module. Thus, the architecture of 2D-DCT was divided into two 1D DCT modules and one transpose buffer. The two 1D DCT modules are similar but the bit widths at each module are different. 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. C. 1D- DCT pipeline process.

Algorithm defined in table 1 from [1] is used to compute 1D-DCT. The algorithm itself has 6 steps. 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 [7]. The input and output process visualization is shown in figure 5. The difference from system [1] is that it computes every single operation in a clock cycle, so every step needs 8 – 9 clock cycles. In this paper, system computes every step in a clock cycle, so DCT computation can be done faster.

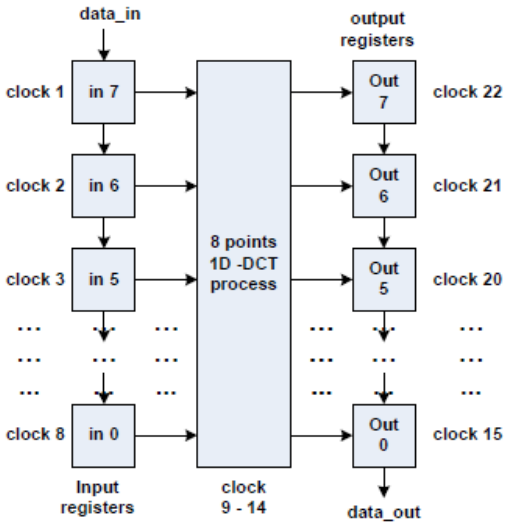


Figure 5 – Data input/output process visualization of 8 points 1D-DCT

1D-DCT computation is done in pipeline process. The pipeline process is made in three stages. Thus, one pipeline stage is done in 8 clock cycles. The timing detail is described in table 2(a-c).

Table 2 describes overall process including input, 2DDCT– quantization – zigzag, and output process. In pipeline mode, it is needed 24 clock cycles to complete overall process of quantized – zigzag 2D DCT.

Table 2a : process from 1st to 8th clock

clock	1	2	3	4	5	6	7	8
Data-n	0	1	2	3	4	5	6	7
stage								
1	x0 = inp	x1 = inp	x2 = inp	x3 = inp	x4 = inp	x5 = inp	x6 = inp	x7 = inp
2								
3								

Table 2b : process from 9th to 16th clock

clock	9	10	11	12	13	14	15	16
Data-n	8	9	10	11	12	13	14	15
stage								
1	x0 = inp	x1 = inp	x2 = inp	x3 = inp	x4 = inp	x5 = inp	x6 = inp	x7 = inp
2	step1	step2	step3	step4	step5	step6	outp=y0	outp=y1
3								

Table 2c : process from 17th to 24th clock

clock	17	18	19	20	21	22	23	24
Data-n	16	17	18	19	20	21	22	23
stage								
1	x0 = inp	x1 = inp	x2 = inp	x3 = inp	x4 = inp	x5 = inp	x6 = inp	x7 = inp
2	step1	step2	step3	step4	step5	step6	outp=y0	outp=y1
3	outp=y2	outp=y3	outp=y4	outp=y5	outp=y6	outp=y7	-	-

The 25th clock process and the following are the same as table 2c. From the table, it can be seen that first 1DDCT output from first data set (8 data) come at 15th clock and last output from first data set come at 22nd clock. Processes in step 1 – 6 are referred from table 1.

D. Transpose Buffer

Transpose buffer is static RAM, designed with two set of data and address bus. It has input and output data address buses. The module block symbol is shown in fig.6.

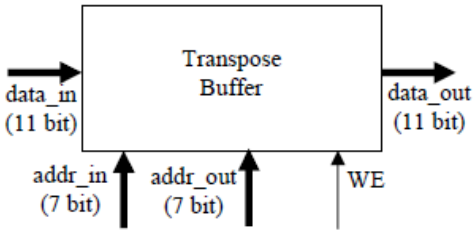
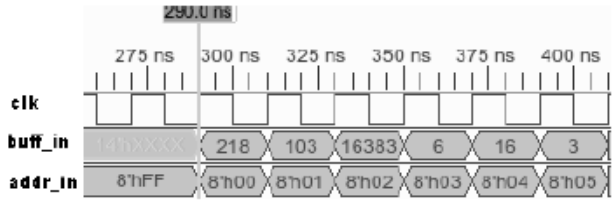
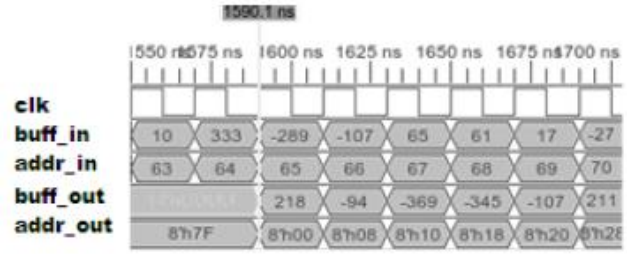


Figure 6 - Transpose buffer primitive symbol

Data input come 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 64th input. It gives time latency between first input and first output. Figure 7a shows the first entry of transpose buffer input and 7b shows generated output sequence was delayed 65 clock cycles from first input.



(a)



(b)

Figure 7 Input (a) and output (b) data of transpose buffer. Output data is generated in 65th sequence, count from first input entrance.

The output of transpose buffer is fed directly to the input of second 1D-DCT unit.

E. Quantizer

Originally, quantization process in JPEG compression is done by divide every 2D-DCT coefficient by quantizing value from quantization table. To apply the operation in VHDL, division is converted into multiplication. The modified quantization table will be post-scaled with some post-scaling table. The post-scaled quantization table, written in matrix form in (9), is the table that will be applied to the VHDL code.

To implement the quantization process, the output of 2D-DCT is multiplied by quantization value. The value is generated by quantization ROM that made for store the quantizing-post scaling value. Block Diagram of the implementation is shown in figure 8.

F. Zigzag Buffer

Like transpose 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 figure 2. Zigzag address is generated by a zigzag ROM. The sequence is stored or preprogrammed in the ROM. When the ROM address bus is accessed by normal address sequence, ROM data bus will emit zigzag value. Figure 8 describe zigzag buffer and ROM construction in the system.

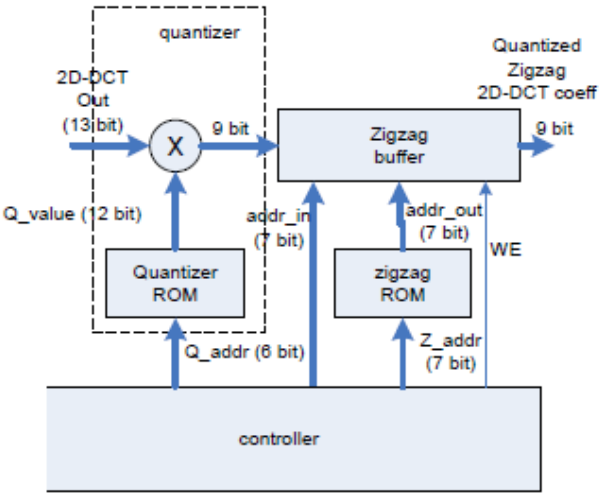


Figure 8 - Zigzag and Quantization Section

G. Implementation Result

The 2-D DCT architecture was described in VHDL. This VHDL was synthesized into an Xilinx Spartan 3E family FPGA [8]. Nevertheless, there is a requirement to use this architecture. The FPGA must have at least 11 multipliers, because the system utilizes only internal multiplier. System is tested with real photographic image. Simulation results give the quantized 2D DCT coefficients. To verify the numerical result, the system is given by data from grayscale picture in figure 9. Data from the picture was converted to VHDL test bench. The result then compared to the result from MATLAB computation.

Figure 9 – Grayscale picture used as test bench

Comparison result between VHDL and MATLAB computation of quantized 2D-DCT is enlisted in table 3. Generally, mean squared error between VHDL and nonrounded MATLAB result is 0.060552 for 64 data. Using figure 9 as sample, there is no error between VHDL and rounded MATLAB result. The result only displayed in

table 3 only for 16 data.

The complete synthesis results to Spartan-3E FPGA are presented in table 4, whose hardware was fit in an XCS500E device. The table 5 presents the comparison between this work (quantized 2D-DCT) and the pure 2DDCT designed in [3]. System designed in [3] uses Virtex FPGA and straightforward multiplication without special algorithm.

Table 3 - Comparison between VHDL and MATLAB computation of quantized 2D-DCT (first 16 data)

no	vhdl	matlab	matlab rounded	no	vhdl	matlab	matlab rounded
0	1	1,25	1	8	3	2,72	3
1	-13	-13	-13	9	1	1,44	1
2	2	1,81	2	10	-1	-0,86	-1
3	-3	-3,14	-3	11	-2	-1,67	-2
4	3	2,54	3	12	-3	-2,78	-3
5	18	18,23	18	13	-3	-3,03	-3
6	8	8,4	8	14	-1	-0,57	-1
7	6	5,6	6	15	1	0,61	1

System designed in [3] uses 64 bits input, so it takes more pins than system proposed in this paper. it also needs 8 multiplications in every single clock to compute 1D-DCT. So, to compute 2D-DCT via transpose buffer, it takes 16 multiplications. Because it doesn't use internal multiplier, it takes more slices to construct multiplier modules.

Table 4 – Device Utilization Using Xilinx Spartan-3E Logic Unit Used Available Utilization

Logic Unit	Used	Available	Utilization
Number of Slices	1145	4656	24%
Number of Slices FFs	1696	9312	18%
Number of 4 input LUTs	1750	9312	18%
Number of Bonded IOBs	21	232	9%
Number of Multipliers 18X18	11	20	55%

Table 5 – Device utilization comparison between this work and 2D-DCT designed in [3]

Logic Unit	This work	Presented in [3]
Number of Slices	1145	7260
Number of Slices FFs	1696	9644
Number of 4 input LUTs	1750	11194
Number of Bonded IOBs	21	101
Number of Multipliers 18X18	11	-

According to synthesis result, maximum time delay produced is 5.895 ns. That constraint yields minimum clock period 11.790 ns. Maximum clock frequency can be used is 84.818 MHz. 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 [3] uses Virtex FPGA that has higher frequency than Spartan, the delay is much smaller than this system and maximum frequency is higher. Maximum frequency in [3] is 308.182 MHz. This system is also faster than 2D-DCT described in [1]. System [1] has minimum period 82.1 ns.

The use of pipeline process gives the system latency. The output exists several clock cycles after the first input. Latency produced in 2D-DCT is 94 clock cycles. Overall system (quantized and zigzag 2D-DCT) has latency 124 clock cycles. As comparison, 2D-DCT designed in [1] has latency 160 clock cycles. The better result reached by system in [3]. It takes 37 clock cycles as system latency to compute 2D-DCT.

IV. CONCLUSION

2D-DCT combined with quantization and zigzag buffer is designed using VHDL. System is tested with real grayscale image. The accuracy of computation is compared to Matlab computation result with similar operation. Comparison yields mean squared error MSE = 0. 060552, computed for 64 data. 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 84.818 MHz.

Sequential – pipeline design gives higher frequency than fully parallel design in [4]. The design takes 1145 slices, 1696 slice FF, and 1750 LUT including the control block that suitable for low cost FPGA like Xilinx XCS500E. The sequential operation of DCT saves much logic utilization in FPGA compared with [3]. Each step of DCT algorithm is executed on each clock cycle. Every step consists of 8-9 operation. It makes this design produces lower latency compared with [1].

V. REFERENCES

[1] 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.

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

[3] D. Trang, N. Bihn, "A High-Accuracy and High-Speed 2-D 8x8 Discrete Cosine Transform Design". Proceedings of ICGCRCICT 2010, vol. 1, 2010, pp. 135-138

[4] 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

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

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

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

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

[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.