

Implementation of Image Compression algorithm on FPGA

S.A.Gore¹, S.N.Kore²

¹PG Student, Department of Electronics Engineering, Walchand College of Engineering, Sangli, Maharashtra,

²Associate Professor, Department of Electronics Engineering, Walchand College of Engineering, Sangli,

Abstract –The discrete cosine transform is a fast transform. It is a widely used and robust method for image compression. It has excellent compaction for highly correlated data. DCT has fixed basis images DCT gives good compaction between information packing ability and computational complexity. With the growth of technology and entrance into the Digital Age, the world has found itself amid a vast amount of information. Dealing with such enormous information can often present difficulties. As such there is a need for development of optimized image/video compression algorithms and their efficient implementation in hardware. This paper presents a novel architecture for obtaining DCTQ coefficients suitable for Virtex-E FPGA Implementation. The design is highly parallel and pipelined so as to exploit the massive parallelism of FPGA and occupies considerably less LUTs (5,418/29,504) with a very high processing speed (171.185MHz).

Key Words: DCT, DCTQ, FPGA, Image Processing, Compression.

I.INTRODUCTION

Transform coding is integral part of image/video processing applications. Transform coding is based on the concept that pixels in an image have some amount of correlation with the neighboring pixels. In the similar way, a high correlation exists in the adjacent frames of a video/image. By using this correlation information the value of the pixel can be predicated from its neighboring pixels. So transformation maps correlated data of spatial image into uncorrelated transformed coefficients. Transformation is the lossless process and inverse transformation of transformed image gives original image. Discrete Cosine Transform (DCT) is widely used for transform coding in most image processing applications. The large amount of data in the digital image is a big problem for transmission & storage of images. Different compression schemes have been developed to transmit/store the image & video with fewer amounts of data. Because of its symmetry, energy compaction and simplicity DCT algorithm is more effective for image compression. A DCT based image processing system gives lossy compression of an image because of the use of quantizer. After transformation most of the image information is packed in a few coefficients. So quantizer discards the coefficients with small amplitudes. This is done by dividing the DCT coefficients by the quantization matrix. The direct computation of 2D DCT using formula requires huge arithmetic operations. This leads to large hardware requirement & time consumption.

There are several methods for computation of 2D-DCT among which the row-column decomposition is best adapted to a hardware implementation. The DCT is also implemented using Distributed arithmetic which is used in VLSI implementations of DSP architectures.

The Distributed arithmetic method is used to implement a 2-D DCT processor for the synchronous design in a Xilinx Vertex IV FPGA device [1]. The method is highly modular and utilizes a highly parallel structure to achieve high-speed performance.

In [2], a low-power and high speed 2-D DCT image compression was implemented. It was implemented on a FPGA platform using simplified multipliers based on Canonical Signed Digit (CSD) encoding. The proposed architectures were implemented on a Spartan3E device and operated at 110 MHz.

An algebraic integer based on 8x8 2-D DCT architecture for digital video processing was proposed in [3]. The proposed architecture could operate up to 121 MHz and consume 10318 (out of 15360) slices, 2731 (out of 7680) SLICEMs, 11570 (out of 30720) slice flip-flops and 15252 (out of 30720) 4- input LUTs.

An 8x8 2D-DCT processor using the row-column decomposition method, based on the residue number system (RNS) is reported [4]. In this method a single multiplication stage for each signal path was used to implement a fast cosine transform algorithm by the 2D-DCT processor. But it requires large multipliers hence more hardware which degrades system performance.

The following paper proposes and implements a modified way of row column decomposition for DCTQ computation.

The proposed architecture occupies very less area coupled with a high processing speed and also less latency. The remainder of this paper is organized as follows: Section II presents DCT algorithm, the proposed architecture is presented in Section III and then the synthesis results and concluding remarks are given in Sections IV and V respectively.

II. ALGORITHM

The 2-D DCT is defined by the following relation:

$$C(u, v) = \alpha(u)\alpha(v) \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i, j) \Phi(x, u) \Phi(x, v) \quad \dots (1)$$

Where,

$$\Phi(a, b) = \cos[\pi(2a+1)b/2N]$$

$$\alpha(u) = \begin{cases} 1/\sqrt{N} & \dots u=0 \\ \sqrt{2/N} & \dots u \neq 0 \end{cases}$$

$$\alpha(v) = \begin{cases} 1/\sqrt{N} & \dots v=0 \\ \sqrt{2/N} & \dots v \neq 0 \end{cases}$$

For $u, v = 0, 1, 2, 3, \dots, N-1$

and 'N' is the number of samples for which the DCT is to be calculated.

The focus in this paper is on 8x8 DCTQ, i.e., N = 8. Therefore equation (1) can be rewritten as a product of three square matrices, each of order 8, as follows:

$$DCT = C * I * C^T$$

The architectures proposed in [5], [6] continue their work using this relation and compute the DCT as a multiplication of three matrices, i.e. one set of multipliers are used to calculate the matrix product of C*I and another set of multipliers would multiply this product with C^T

But, in this paper, we propose a new algorithm for computation of DCT based on the following formula

$$DCT = \sum_{i=0}^7 \left\{ \sum_{j=0}^7 C[v] * I[i][j] \right\} * C[u][i]$$

where, C and I are the matrices used to represent the set of cosine terms and the image block respectively; and i, j are the row, column indices of matrix I. whereas u, v are the same for matrix C. The order of the matrices is 8x8, so the range of i, j, u, v is 0 to 7.

This algorithm can be implemented in the four basic stages as illustrated below:

The indices u, v and i, j are first set to zero each.

Stage I: A set of 8 two-input multipliers is used in this

stage to multiply the terms: - $C[v][j] * I[i][j]$

The multipliers give some crude products at their outputs which are enumerated as:-

$C_p[j] = C[v][j] * I[i][j]$ with the index j assuming value from 0 to 7, one for each multiplier.

Stage II: The crude products obtained from the first stage are added in this stage to realize the summing equation

$$\sum_{k=0}^n C_p[j]$$

Stage III: This stage again demands the use of multipliers, which is satisfied by a single two-input multiplier. The two inputs of this multiplier would be the sum output of the previous stage and the term $C[u][i]$ respectively. The index i is successively incremented from 0 to 7, so as to obtain 8 such products.

Stage IV: The final stage in the computation of a DCT Coefficient is an adder again which would obviously sum up the above mentioned eight products to give the first DCT coefficient.

To obtain the remaining coefficients, the indices u and v are incremented in such a fashion that the processor will provide the DCT coefficients in a row-wise manner. Furthermore these coefficients need to be compressed and so these coefficients are quantized or divided by certain terms. But, as division is tedious to realize, the DCT coefficients are multiplied with the inverse of the quantization values. Such inverse values are stored in the system memory.

III. ARCHITECTURE

The proposed design doesn't conform to the conventional Row-Column technique for matrix multiplication. Instead, the same algorithm employing $C * I * C^T$ is implemented using a different technique. The block diagram

representation of the same is as shown in the Fig. 1. The complete image frame is divided into sub-frames or blocks of 8 pixels by 8 pixels; each pixel of 8 bits. A Deckerd RAM is used to store these blocks in a simultaneous read-write fashion. When the first deck is being processed upon, the second is being written into. Thus reducing the idle time and boosting the speed. The image coefficients are multiplied with the corresponding cosine terms to obtain some crude products; which are then added together. This is in accordance with the first two stages of the algorithm as mentioned in the preceding section. This sum is then multiplied with another set of cosine terms and stored in the stack along with seven such partial results. When the stack is full, its contents are added to give a 12 bit DCT

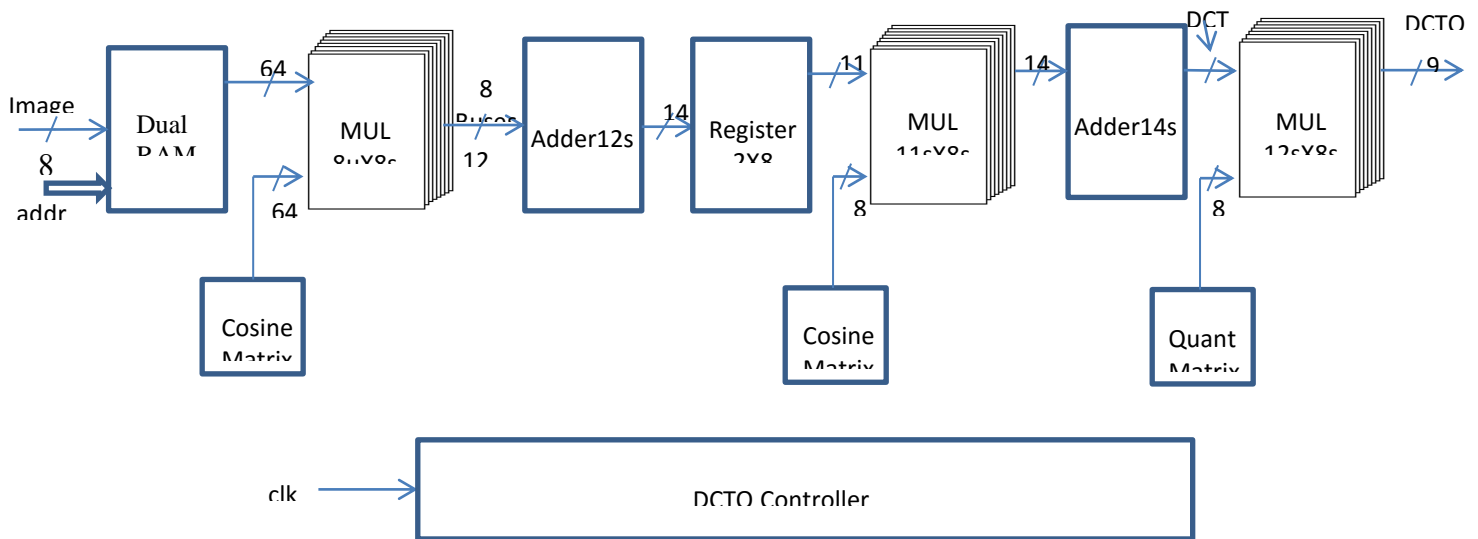


Fig.1. Architecture of DCTQ Processor

Coefficient. However, it is also needed to quantize these coefficients, which is accomplished by the final multiplier. It takes the system 45 clock cycles to compute first DCTQ coefficient; but since the architecture employs pipelining, next DCTQ values are successively obtained.

Each of the blocks shown in the Fig. 1 can be classified into the following units

A. Memories

The architecture employs memories of two different types namely Data memory, Program memory. The data memory or simply RAM is used to store the pixel information of the image. As the name Deckerd RAM suggests, the Data memory unit is sub-divided into 2 RAMs; so that when one RAM is being written into, data can be read from the other. Also, when the first RAM is fully written into, the controller switches writing to the second RAM. The second unit under memories is the Program memory or the ROM. It stores the cosine terms and the quantization coefficients. This data is invariant as far as the architecture for a 8x8 2-D DCT is concerned. Hence, a ROM is employed.

B. Multipliers

In [7], a review of various multiplier algorithms viz. Wallace, Vedic, Toom-Cook, Karatsuba and Booth has been conducted. It can be concluded from the same that Toom- Cook algorithm gives the highest speed but Vedic and Wallace consume the least area on board amongst their other counterparts. However the increase in speed provided by Toom-Cook over that of Vedic or Wallace is barely 1% but the area consumption is increased by approximately 10%. Thus considering the area-speed trade-off, the authors chose to implement the design with Vedic multipliers. The design incorporates three multiplier units. Each unit has different bus widths as per the architecture demands, and so varies the emphasis being given to the sign bit.

C. Adders

The Carry Save and Ripple Carry adder utilizes the least amount of area. Also, Carry Skip and Carry Save adder provide the maximum frequency operation. Thus, usage of Carry Save adder in the DCTQ architecture was finalized.

D. Stack/Register

The products obtained from the second stage of multipliers are stored in a temporary storage called the Stack. This provisional memory block stores these values for eight clock cycles and sends a burst of data to the succeeding stage of adder. These are the 8 products obtained in the Stage III of Section II.

E. DCTQ Controller

The complete process is coordinated throughout by the DCTQ Control Block. The block is so designed that it meets all the timing and triggering sequences. Any mismatch in the timings or false triggering can lead to catastrophic results. The addressing of the memory blocks is also accomplished in this block. The entire task is accomplished using various internal signals.

The number of multipliers has been dramatically reduced in this architecture along with the added features of being pipelined, using Vedic algorithm for multiplication and Carry Save Adders for addition; thus, increasing the speed of operation and reducing the area occupied by an appreciable amount. The same is designed and tested on the "Cameraman" image. The verified results of the simulations of the same are provided in the following sections.

IV. RESULTS AND DISCUSSIONS

Xilinx ISE Design Suite 9.2 was used for the design and implementation of the proposed design on VIRTEX-E FPGA device. The results procured by the same, exhibited the following utilization: 4,493/14,772 as slice registers. The device utilization summary is shown in Table 1. The results further indicated a maximum path delay of 5.842ns which yields a maximum operating frequency of 171.185MHz.

Figure 2 shows the simulation results of DCTQ processor. The first signal "clk" is the clock input to the DCTQ processor. The signal "d0" is the output of the deked ram which gives out 8 pixels of data to 8 vedic 8x8 unsigned multipliers respectively. The other input of the multiplier comes from the Cosine matrix saved in the ROM. The multipliers are 8 stage pipelined, thus the 7 waveforms "result [15:0]" which are the output of multipliers are available 8 clock cycles after the application of input. The 16 bit output is later truncated to 12 bits to preserve area. These 8 values are later added in the 12 bit signed carry save adder whose 15 bit output arrives 5 clock cycles after the previous multiplier outputs as "sum1[14:0]" due to 5 stage pipelining. The output is again truncated to 11 bits and fed to the 11x8 vedic signed multiplier whose 19 bit output is enabled after 8 clock cycles as "result [18:0]" in the waveform. The other 8 bit input of the multiplier comes from the Cosine matrix saved in the ROM. The 19 bit outputs are truncated to 14 bits and then stored in the stack till the time the other 7 outputs arrive to be fed into the 14 bit carry save adder. Hence the stack adds 8 clock cycles of delay to the final output. The 8 outputs of the stack are "qr0[10:0]-qr7[10:0]". The output of the 14 bit carry save adder "result [16:0]" is valid 6 clock cycles after the stack output is activated. Finally the output of the adder truncated to 12 bits is fed to the 12x8 vedic multiplier whose other input comes from the quantization matrix which gives out 8 bits. The multiplier adds another 8 clock cycles delay due to 8 stage pipelining to the output and thus a total of 45(2+8+5+8+8+6+8) clock cycles are required to get the first coefficient of the DCTQ processor. Thereafter a DCTQ coefficient is available every eighth clock cycle. At various stages, the outputs are truncated to preserve area with minimum change in output. The number of bits to be truncated has been obtained after extensive MATLAB simulations.

The DCTQ and IQIDCT algorithms are implemented on the FPGA cores. The images are fed in after some preprocessing in MATLAB. Image of "cameraman" (256x256 pixels) as shown in Fig. 2(a) and Fig. 2(a) respectively were operated using the explained technique. The reconstructed image as shown in Fig. 2(b) and Fig. 2(b) showed a PSNR of 20.9 dB, root mean square error of 0.0901 and compression ratio of 1.969 upon comparison with the original images. Thus the images before DCTQ processing and after reconstruction from the IQIDCT processor look quite alike. The simulation of DCTQ processor is shown in fig.3.

Resources	DCTQ
Number of Slices occupied as Flip	4315/29504
Number of 4 input LUTs	5418/29504
Speed (MHz)	171.185

Table 1. Device utilization summary

V. CONCLUSION

This paper presents a new efficient architecture for computation of the DCT. The proposed architecture based on row/column decomposition implements a different matrix multiplication. This method not only increases the speed but also requires less hardware than conventional architectures. The proposed DCT architecture can be implemented in an application specific integrated circuit for use in multimedia services. The processing of DCTQ coefficients is a problem of matrix multiplication and hence the usage of several efficient matrix multiplication techniques can effectively enhance the performance of the DCTQ computation.

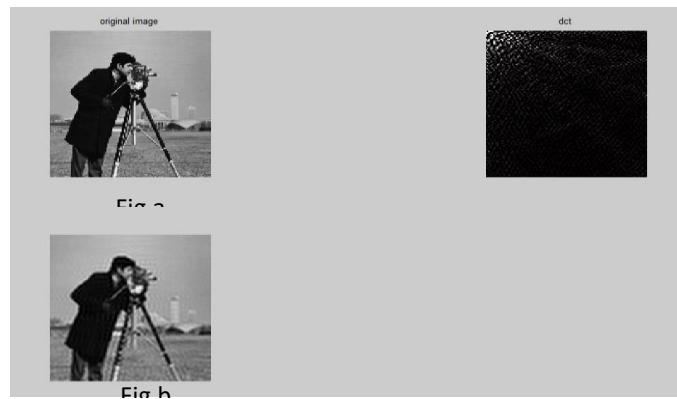


Fig.2.(a)Image before compression,(b)Image after compression.

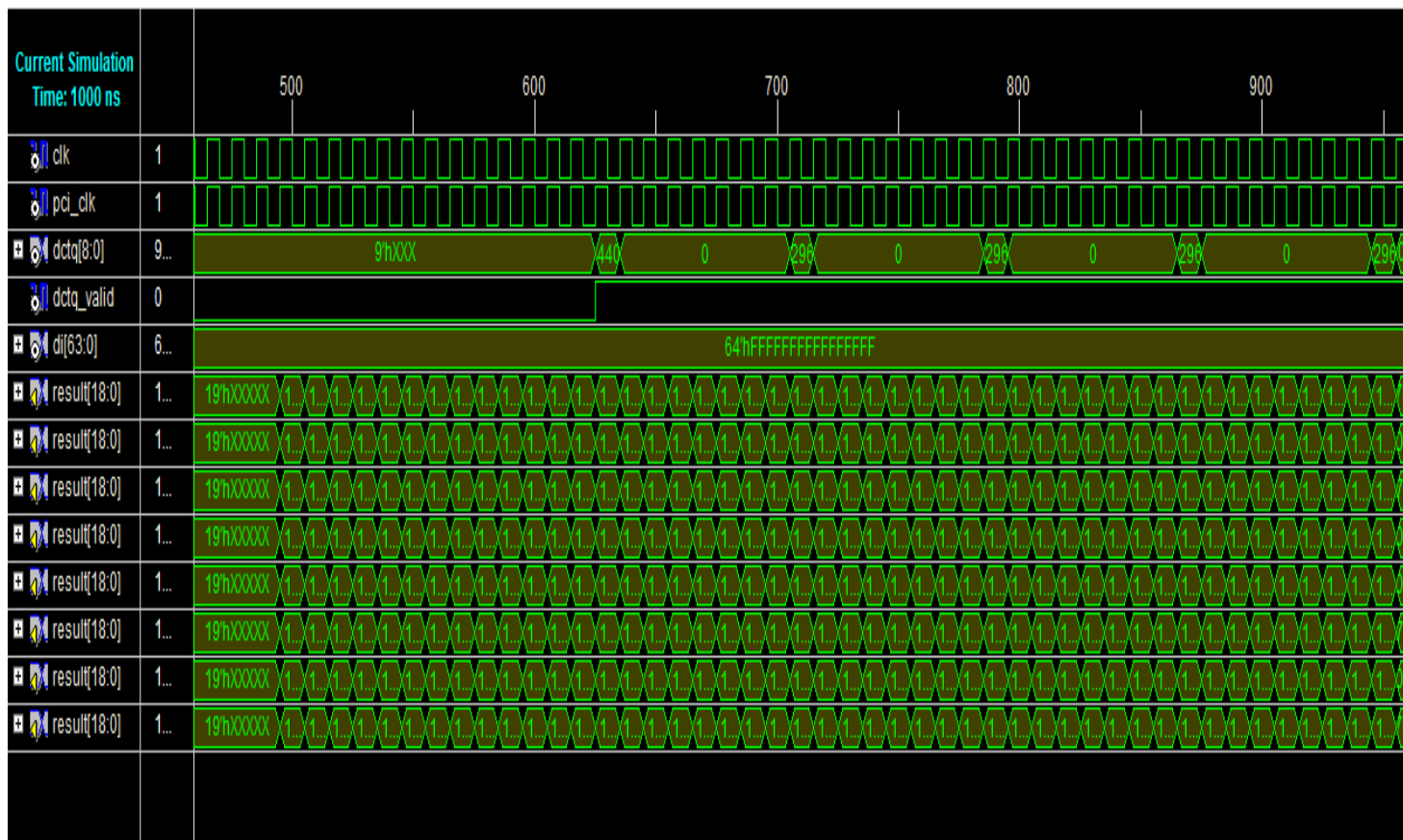


Fig 3.Simulation result of DCTQ Processor

REFERENCES

- [1] R. E. Atani, M. Baboli, S. Mirzakuchaki, S. E. Atani, and B. Zamanlooy, "Design and implementation of a 118 MHz 2D DCT processor," in IEEE International Symposium on Industrial Electronics, 2008, pp. 1076-1081
- [2] M. Jridi and A. Alfalou, "A low-power, high-speed DCT architecture for image compression: Principle and implementation," in VLSI System on Chip Conference (VLSI-SoC), 2010 18th IEEE/IFIP, 2010, pp. 304-309.
- [3] H. L. P. A. Madanayake, R. J. Cintra, D. Onen, V. S. Dimitrov, and L.T. Bruton, "Algebraic Integer based 8×8 2-D DCT Architecture for Digital Video Processing," in 2011 IEEE International Symposium on Circuits and Systems (ISCAS), 2011, pp. 1247-1250.
- [4] Fernandez, P.G., A. Garcia, J. Ramirez and A. Lioris, 2004, "Fast RNS- Based 2D-DCT Computation on Field-Programmable Devices," in the IEEE Workshop on Signal Processing Systems, pp: 365-373.
- [5] Dixit, H.V., Jeyakumar, A., Kasat, P.S., Warty, C., "VLSI design of fast DCTQ-IQIDCT processor for real time image compression," 2013 Tenth International Conference on Wireless and Optical Communications Networks (WOCN), vol., no., pp.1,5, 26-28 July 2013.
- [6] S. Ramachandran, S. Srinivasan, and R. Chen, "Epld-based architecture of real time 2d-discrete cosine transform and quantization for image compression," in Circuits and Systems(ISCAS '99), Proceedings of the 1999 IEEE International Symposium on, vol. 3, 1999, pp. 375-378.
- [7] P. Kasat, D. Bilaye, H. V. Dixit, R. Balwaik, and A. Jeyakumar, "Multiplication algorithms for vlsi-a review," International Journal on Computer Science and Engineering (IJCSE), vol. 4, no. 11, pp. 1761-1765, Nov 2012.