Image compression on FPGA using DCT

Conference Paper · August 2009

DOI: 10.1109/ACTEA.2009.5227881 · Source: IEEE Xplore

3 authors, including:

Abdallah Kassem
Notre Dame University
36 PUBLICATIONS **110** CITATIONS
SEE PROFILE

Mo Hamad
The New School
34 PUBLICATIONS **142** CITATIONS
SEE PROFILE

Image Compression on FPGA using DCT

A. Kassem, *Member, IEEE,* M. Hamad , *Member, IEEE,* and E. Haidamous

Abstract— One of the major building blocks in an image data compression system is the discrete cosine transform or DCT; which can be achieved using specialized algorithms. This paper presents a method to implement the DCT compression technique using the Lee algorithm. Rapid prototyping based on FPGA platform of the Spartan-3E family is used to validate the operation of the described DCT system. This system offers significant advantages: portability, rapid time to market and real time, continuing parametric change in the DCT transform.

I. Introduction

A large number of image data compression techniques are available, each one being adapted to a specific type of application, such as: compact disc, videoconference, videophone and multimedia systems. In all of these applications the transmission line bandwidth will determine the compression standard to be used [1]. Among these, there is a compression technique based on a frequency transform called a Discrete Cosine Transform (DCT). This transform contains unique characteristics which allow for the creation of an efficient image compression; image and video compressors and decompressors are implemented in both software and hardware. However, hardware implementations are especially important for the realization of highly parallel algorithms and can achieve much higher throughput than software solutions. The DCT transform was greatly enhanced by its implementation in VLSI circuits, which are becoming increasingly faster. VLSI circuits are now capable of executing a DCT transform in real time.

In this paper, we present the methodology to implement the DCT. The proposed system could be useful in many other applications that require image data compression. We describe in section II the general description of the codec image video system that required the data compression. The DCT hardware design implementations are subjects of section III. Section IV contains the implementation process of the DCT using FPGA and its experimental results. Finally, conclusion is given in section V.

II. Codec Image Video

Figure 1 shows the subsystem of CoDec (Compressor/Decompressor) video system [2, 3]. It consists of a compressor and a decompressor. The compressor is made up of an image pre-processor, a discrete cosine transform (DCT), a quantizer Vector (QV) and a Variable Length Coding (VLC). The compressed image is then transmitted via a channel line or wirelessly to the decompressor. The decompressor consists of an Inverse Variable Length Coding (IVLC), an inverse discrete cosine transform (IDCT), an inverse quantizer (IQV), and a post-processor as shown in Figure 1.

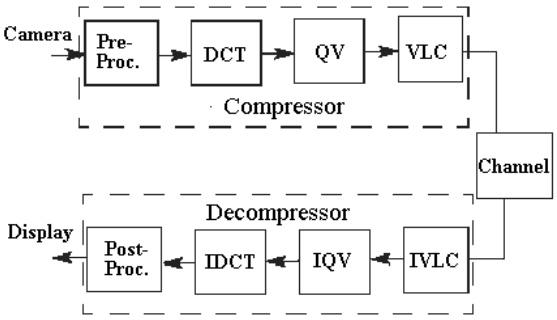

Fig. 1. CoDec image video system.

Discrete Cosine Transform (DCT) block receives an NxN matrix image, which is divided into smaller image blocks (4x4, 8x8, 16x16, ...) where each block is transformed from the spatial domain to the frequency domain. DCT decomposes signal into spatial frequency components called DCT coefficients [2]. The lower frequency DCT coefficients appear toward the first line/first column of the DCT matrix, and the higher frequency coefficients are in the last line/last column of the DCT matrix. The quantization is used to discard insignificant data without introducing any artifacts to the image. After quantization, the majority of the DCT coefficients are equal to zero [4, 5]. A run-length coding (RLC) and variable length coding (VLC) are used to retrieve code words and their lengths from predefined lookup tables. The decompressor block is used to reconstruct the compressed image using the inverse process.

III. DCT Hardware Design

A. Theory of DCT for Hardware Implementation

Equation 1, shows the 1-D Discrete Cosine Transform (DCT) [6]:

$$F(u) = \frac{2}{N} C(u) \sum_{x=0}^{N-1} f(x) Cos\left(\frac{\pi(2x+1)u}{2N}\right), \quad (1)$$

where $F(u)$: coefficient value in the transform domain,
$f(x)$: coefficient value in the pixel domain,
x: spatial coordinate in the pixel domain,
u: coordinate in the transform domain,

Manuscript received in April 30, 2009.

A. Kassem and M. Hamad are with the Electrical and Computer, Communication Engineering Department, Notre Dame University, P.O.BOX 72, Zouk Mikael, Lebanon, e-mail: mhamad or akassem@ndu.edu.lb.

$C(u) = 1/\sqrt{2}$ for u=0, otherwise 1.

The DCT is a frequency transform which is equivalent to the real part of the discrete Fourier transform (DFT). Equation (2) shows the forward transformation for the generation of the two dimensional discrete cosine transform 2D-DCT, of the original NxN image block:

$$F(u,v) = \frac{4}{N^2} C(u)C(v) \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i,j) \; Cos(\frac{\pi(2i+1)u}{2N}) Cos(\frac{\pi(2j+1)v}{2N}), \quad (2)$$

where $F(u,v)$: coefficient values in the transform domain,
$f(i,j)$: coefficient values in the pixel domain,
i,j: spatial coordinates in the pixel domain,
u,v: coordinates in the transform domain,
$C(u) = 1/\sqrt{2}$ for u=0, otherwise 1.
$C(v) = 1/\sqrt{2}$ for v=0, otherwise 1.

The separability property of the DCT, has an advantage that $F(u, v)$ can be computed in two successive steps, 1-D operations on rows and columns of an image block and then calculate the 2D-DCT as shown in figure 2. Using this property equation (2), becomes as the following equation (3):

$$F(u,v) = \frac{2}{N} C(u) \sum_{i=0}^{N-1} \left\{ \frac{2}{N} C(v) \sum_{j=0}^{N-1} f(i,j) Cos(\frac{\pi(2j+1)v}{2N}) \right\} Cos(\frac{\pi(2i+1)u}{2N}), \quad (3)$$

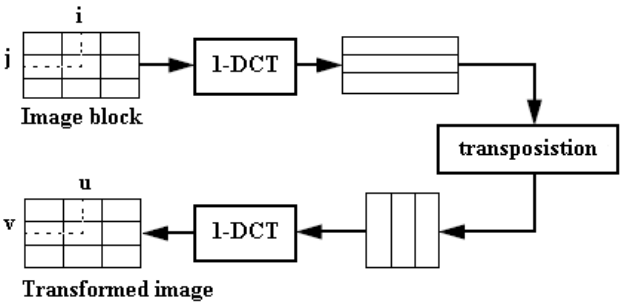

Fig. 2. Computation of 2-D DCT using separability property

To minimize the computation of the 1D-DCT in equation (1), a LEE graph will be used [7]. The LEE graph can be produced by rewriting the equation (1) as follows:

$$F_c(m) = \frac{2}{N} \sum_{n=0}^{N-1} \dot{F}(n) C_{2N}^{(2n+1)m}, \; m = 0, 1, \ldots, N-1, \quad (4)$$

where $C_k^i = \cos(i\pi/k)$;

and $\dot{F}(n) = C(n)F(n)$

This graph, shown in figure 3, for $N=8$ requires

$\frac{3N}{2}\log_2(N) - N + 1$ Real additions,

and

$\frac{N}{2}\log_2(N)$ Real multiplications.

B. DCT Hardware Implementation

The block diagram of the hardware implementation of the 2D-DCT using LEE graph is shown in figure 4. The operative part is used to calculate the 1D-DCT on the rows, and the 1D-DCT on the columns, alternatively, to obtain the 2D-DCT for N=8.

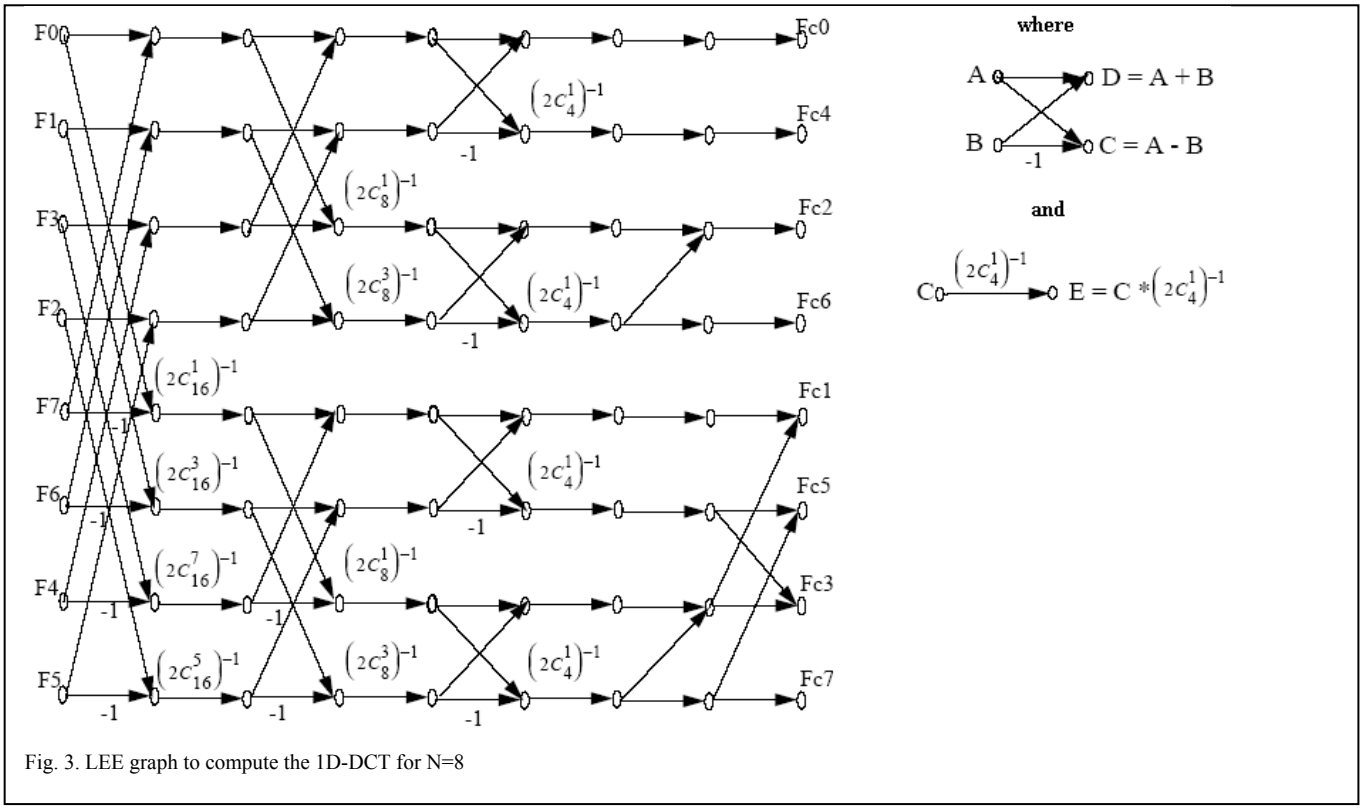

Fig. 3. LEE graph to compute the 1D-DCT for N=8

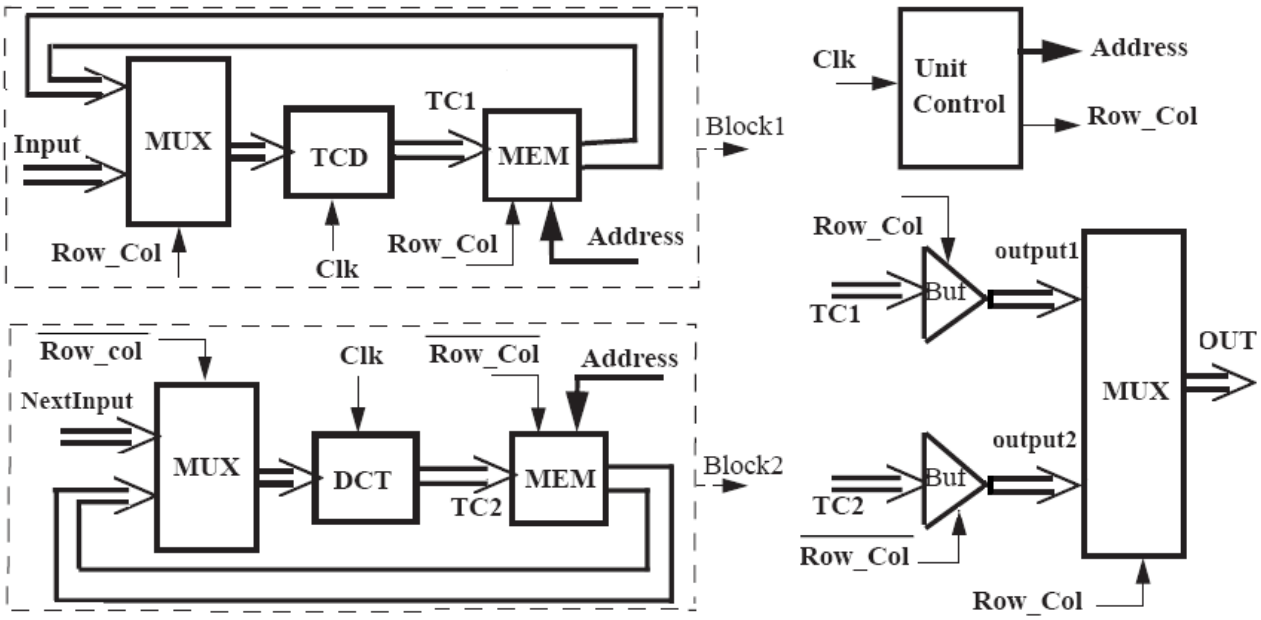

Fig. 4. Block diagram of the hardware implementation of the 2D-DCT using LEE graph

The 1D-DCT algorithm is applied firstly on the data sequences for the rows and the results are stored in the memory. Afterwards the algorithm is applied on the columns obtaining the final results of the 1D-DCT. During operation of the 1D-DCT on the columns, the next data sequences enter into the system, thus creating a pipeline architecture [8]. Each output transform sequence is rounded off by using an 11 bits adder and comparator subsystems.

IV. RESULTS

2D Discrete Cosine Transform can be implemented onto an FPGA through system generator using hardware models, which can be used as a building block for various image processing systems. System Generator works with standard Simulink models including "Gateway In" and "Gateway Out" defines the boundary of the FPGA.

The input image is obtained by MATLAB and transformed into a matrix representation. This image is then decomposed into (8x8) block images. Figure 5 shows an 8x8 block image obtained from a 256x256 gray scale image.

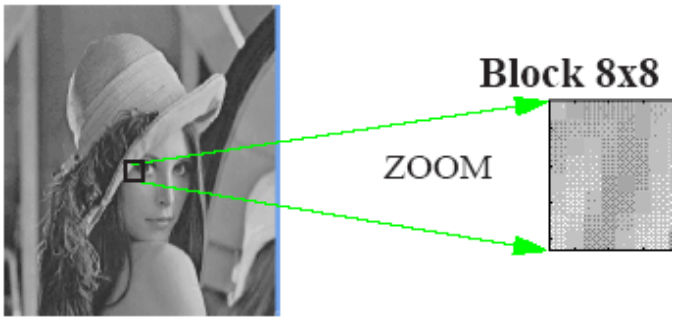

Fig.5. Image test divided into 8x8 blocks used for DCT

And the corresponding matrix representation, obtained by MATLAB, of the 8x8 block is shown by matrix 1.

Matrix 1. Pixel level of 8x8 block using MALAB.

$$\begin{bmatrix} 106 & 109 & 97 & 101 & 104 & 109 & 91 & 57 \\ 83 & 77 & 89 & 71 & 70 & 67 & 62 & 54 \\ 66 & 60 & 62 & 61 & 59 & 57 & 58 & 59 \\ 58 & 62 & 62 & 61 & 55 & 53 & 55 & 58 \\ 59 & 62 & 57 & 63 & 59 & 56 & 81 & 76 \\ 72 & 94 & 60 & 85 & 58 & 47 & 64 & 73 \\ 78 & 115 & 77 & 96 & 93 & 73 & 94 & 80 \\ 87 & 109 & 115 & 67 & 85 & 93 & 82 & 73 \end{bmatrix}$$

The 2D-DCT matrix of this block calculated by MATLAB is represented by matrix 2.

Matrix 2. 2D-DCT result using MATLAB.

597.079	35.59	-1.172	-5.127	-15.25	-4.32	-19.81	-1.6
-17.287	8.613	-14.64	20.567	6.254	14.25	17.41	2.37
99.4702	29.92	-18.89	18.946	-24.64	-9.7	6.752	6.47
30.9677	0.635	-1.171	2.7918	10.074	5.313	-24.98	-15.4
18.5024	-6.76	-4.47	11.6	-15.75	0.051	14.03	7.99
19.6004	0.68	-10.38	6.8812	5.0155	-3.58	-17.08	-10.4
-5.4927	6.033	3.503	1.6981	1.2775	0.91	5.144	-6.26
13.2943	-20.1	0.54	8.287	0.1484	3.332	-7.513	3.68

After downloading the bit mapping file of the design (2D-DCT) onto the FPGA board type SPARTAN 3-E STARTER mounted on the laptop through the USB port. The obtained matrix by the hardware is presented by the matrix 3 using a frequency of 50 MHz. The hardware implementation of the 2D-DCT occupies only 14% of the total number of slices and 10% of the total LUT. However, inherent limitations in the interface to the FPGA limited the overall performance of the design.

The Maximum percentage error is found to be 7.86%≈8% on the grayscale pixel range [0,255]. This error doesn't appear when motion images in process; this due of the fact that the Human Visual System (HVS) is less sensitive to errors in high frequency coefficients than it is to lower frequency coefficients, the higher frequency components can be more finely quantized, as done by the quantization matrix.

Matrix 3. 2D-DCT result using LEE graph implemented in Hardware.

597.1	35.6	-1.2	-5.1	-15.3	-4.3	-19.8	-1.6
-17.3	8.6	-14.6	20.6	6.3	14.3	17.4	2.4
99.5	29.9	-18.9	18.9	-24.6	-9.7	6.8	6.5
31.0	0.6	-1.2	2.8	10.1	5.3	-25.0	-15.4
18.5	-6.8	-4.5	11.6	-15.8	0.1	14.0	8.0
19.6	0.7	-10.4	6.9	5.0	-3.6	-17.1	-10.4
-5.5	6.0	3.5	1.7	1.3	0.9	5.1	-6.3
13.3	-20.1	0.5	8.3	0.2	3.3	-7.5	3.7

The 2D-DCT architecture shows a good performance when it is applied to 32x32 sub-images where each sub-image is composed by 8x8 pixels as shown in table 1. The timing diagram of this implementation is depicted by figure 6, where

Ni: Number of images;
Ns: Number of sub-images;
Nl: Number of Lines of the sub-image;
Nc: Number of Columns of the sub-image;
Tc: T-cycle = 1 clock cycle;
Tci: Time to initialize the 1st sub-image = Nl * Tc;
Ts: Time to initialize the next sub-image = Nc * Tc;
Tsi: Time required to compute one sub-image;
Tti: Total time required to compute one image= Ns * Tsi;

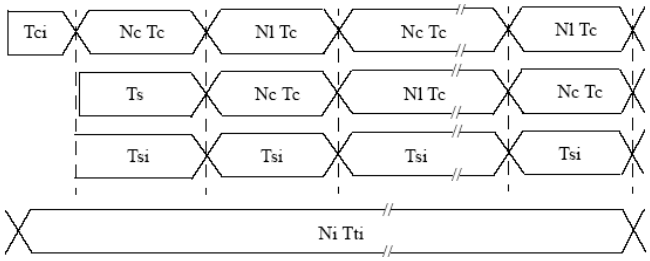

Fig.6. Timing diagram of the implemented 2D-DCT

Block 1 and 2 of figure 4 work in pipeline, when they are loaded by the sub-image pixels. In addition the initialization time to load the 1st sub-image can be omitted.

Table 1: Performance of the 2D-DCT implemented in hardware.

Ni	Ns	Nl*Tc (us)	Nc*Tc (us)	Tsi (us)	Tti (ms)
1	1024	0.16	0.16	0.16	0.16
12	12288	0.16	0.16	0.16	1.97
25	25600	0.16	0.16	0.16	4.1
30	30720	0.16	0.16	0.16	4.92

As shown in table 1, about 0.2% (5 ms) is used by the 2D-DCT, which means that 98.8% is left to the other blocks such as QV, LVC and the decompressor block to produce a real time motion image (30 image/second).

V. Conclusion

We have presented the implementation of the 2D-DCT algorithm using LEE graph with combined pipeline architecture. This implementation was realized with a Xilinx XC3S500E Spartan-3E Starter FPGA, clocked at 50 MHz. The use of a reprogrammable device permits the continuing parametric changes of the DCT in real time. The Xilinx System Generator, embedded in MATLAB Simulink was used to program the model and test in the FPGA board using the hardware co-simulation feature tools. Finally, the error percentages about 8 % of the grayscale pixel were very small between the images before the 2D-DCT implementation and after the hardware implementation.

Acknowledgment

The authors would like to acknowledge the financial support of the use of CAD tools from the Xilinx Corporation.

References

[1] A. K. Jain, "Fundamental of Digital Image Processing", Prentice-Hall, 1st Ed.,1989.
[2] I. E. G. Richardson, "Video Codec Design: Developing Image and Video Compression Systems", Wiley & Sons, 1st Ed., 2002.
[3] A. Puri, "Video Coding using the MPEG-1 Compression Standard", Society for Information Display Digest of Technical papers, pp. 123-126, 1992.
[4] L. V. Agostini, I. S. Silva, S. Bampi, "Pipelined fast 2D DCT architecture for JPEG image Compression", The 14th Symposium on Integrated Circuits and Systems Design, pp. 226-231, Sept, 2001.
[5] A. B. Watson, "DCT Quantization Matrices Visually Optimized for Individual Images", Proceedings of SPIE, pp. 202-2216, 1993
[6] N. Ahamed, T. Natarjian, and K. R. Rao, "*Discrete Cosine Transform*", IEEE Trans. on Comp., Vol. C-23, pp. 90-93, Jan. 1974.
[7] B.G. LEE, "*A new algorithm to compute the discrete cosine transform*", IEEE Trans. on Acc., Speech, and Signal Process., pp. 1243-1245 vol. ASSP-32, no. 6, Dec.1984.
[8] A. Kassem et al., "Simulation and Implementation of DCT for Image Processing Applications", Second LAAS International Conference on Computer Simulation, 1997.