International Journal of Engineering and Technical Research (IJETR)
ISSN: 2321-0869, Volume-2, Issue-6, June 2014

VLSI Architecture for DCT Based On High Quality DA

Urbi Sharma, Tarun Verma, Rita Jain

***Abstract*— Discrete Cosine Transform (DCT) is the major building blocks in an image and video compression system, which can be achieved using various specialized algorithms. It is also being used in various standardized coding schemes. Such as JPEG, MPEG-2, and various others.The computation involved while performing DCT through direct approach requires a large number of multiplications which are time-consuming and eventually results in adding delay while performing arithmetic operations. It can completely be avoided using Distributed Arithmetic (DA) approach and the proposed architecture is implemented on same grounds. However shifting and addition used in the proposed method is based on utilizing fixed point arithmetic, and thus providing more precise results as compared to existing architectures. The operation when implemented on Vertex-5 FPGA results in more precise results with the delay of 17 ns and only 2% of available LUTs.**

***Index Terms*— DCT, DA,FPGA.**

I. INTRODUCTION

Discrete Cosine Transform (DCT) is very effective and most popular transform technique used in image compression among numerous available approaches due to its energy compaction and low complexity. For several standards like JPEG, MPEG-2, MPEG-4, H.261 etc., DCT has become an integral part for compression. There are a number of "fast" algorithms developed for discrete transform computation [1],[2],[3], however the continuous demand of high speed, high throughput and small latency architectures always put tremendous pressure on VLSI designers.DCT is the core of image compression algorithm and is calculated on comparatively smaller square matrices. DCT block receives an NxN matrix image, which is divided into smaller image blocks (4x4, 8x8, 16x16, etc) where each block is transformed from the spatial domain to the frequency domain [4].

A large number of adders along with a large number of multipliers are required for direct implementation of DCT. Distributed arithmetic (DA) is a good solution to implement multiplication without multiplier (as multiplier consumes more power) and implementing DCT using DA offers several advantages in terms of area and speed. DA is an efficient method

For calculation of inner product when one of the input vectors is fixed [1].DA provides best solution for computing multiply and accumulate (MAC) function. It uses pre computed look-up tables and accumulators instead of multipliers for calculating inner products and has been

widely used in many DSP applications such as DFT, DCT, convolution, digital filters etc [9]. As DCT is based on solving sum of product (SOP) or MAC, DA can reduce multiplication to simply shift and add instead of multiplying.

In this paper the periodicity and symmetry of DCT has been exploited for optimizing the performance and thereby reducing the computational redundancy. Using DA based on NEDA (New Distributed Arithmetic) architecture [3], an efficient architecture for implementing more precise DCT computation architecture using fixed point calculation for shifting and adding operation has been proposed. Moreover the integer and decimal parts of multiplication operation has been done separately for getting more precise results.

Rest of the paper is organized as follows. Section II and Section III deals with the realization of DCT using DA and the proposed work and result on 1-D DCT respectively. Section IV concludes the work.

II. DCT USING DISTRIBUTED ARITHMETIC

DCT transforms the information from the time or space domains to the frequency domain. DCT helps in expressing a finite number of data points as the sum of cosine functions that oscillate at different frequencies [6]. A 8x8 point 2D DCT can be expressed by Eq .1.

$$Y(k,m) = \frac{2C_kC_m}{N}\sum_{i=0}^{7}\sum_{j=0}^{7} x(i,j)\cos\left[\frac{(2i+1)k\pi}{2N}\right]\cos\frac{[(2j)+1m\pi]}{2N} \quad \text{.....(1)}$$

Here:

Y(k, m) and x(i,j) represents the transformed output and two dimensional input sequence respectively. Also k,m,i,j = 0,1,…..,7,and C_kC_m are defined as:

$$C_k, C_m = \begin{cases} \frac{1}{\sqrt{2}} & for k, m = 0 \\ 1 & for k, m \neq 0 \end{cases}$$

Eq. (1) can be expressed as the matrix vector representation.

$$[Y] = [C_{8x8}].[x] \quad \text{.....(2)}$$

Here C_{8X8} is the DCT coefficient matrix.

Manuscript received June 20, 2014.

The 2-D basis functions can be generated by multiplying the horizontally oriented 1-D basis functions with vertically oriented set of the same functions [6]. From the above vector representation, the 2-D DCT can be decomposed in two 1-D DCT vectors by using transpose in between the two same 1-D DCT modules. The architecture basically consist of one 1-D row DCT, transpose operation and one more 1-D column DCT. Based on this Eq. 2 can further be expressed using a matrix transpose and two 1D DCTs as Eq. 3.

$$[Y] = [C].[X].[C^T]$$

….(3)

After row column decomposition, the 8 point 1-D DCT is applied to each row of the input matrix and each (8x8) block of "semi transformed" values is transposed and has a further 1-D DCT applied to it [7]. The 8 point 1-D DCT in matrix representation [7] is shown by Eq. 4.

$$\begin{bmatrix} Y(0)\\ Y(1)\\ Y(2)\\ Y(3)\\ Y(4)\\ Y(5)\\ Y(6)\\ Y(7) \end{bmatrix} = \frac{1}{2}\begin{bmatrix} C_4 & C_4 & C_4 & C_4 & C_4 & C_4 & C_4 & C_4\\ C_1 & C_3 & C_5 & C_7 & -C_7 & -C_5 & -C_3 & -C_1\\ C_2 & C_6 & -C_6 & -C_2 & -C_2 & -C_6 & C_6 & C_2\\ C_3 & -C_7 & -C_1 & -C_5 & -C_5 & C_1 & C_7 & -C_3\\ C_4 & -C_4 & -C_4 & C_4 & C_4 & -C_4 & -C_4 & C_4\\ C_5 & -C_1 & C_7 & C_3 & -C_3 & -C_7 & C_1 & -C_5\\ C_6 & -C_2 & C_2 & -C_6 & -C_6 & C_2 & -C_2 & C_6\\ C_7 & -C_5 & C_3 & -C_1 & C_1 & -C_3 & C_5 & -C_7 \end{bmatrix} \begin{bmatrix} x(0)\\ x(1)\\ x(2)\\ x(3)\\ x(4)\\ x(5)\\ x(6)\\ x(7) \end{bmatrix}$$

…..(4)

From above matrix it can be concluded, $C_X = \cos\left(\frac{x\pi}{16}\right)$. Direct implementation of Eq. 4 requires 64 multiplication and 56 additions and such a solution is not hardware efficient as multiplication operation requires more power and silicon area. Therefore by applying periodicity and symmetry property in DCT coefficient matrix, Eq.4 can be rewritten [7] as Eq.5.

$$\begin{bmatrix} Y(0)\\ Y(2)\\ Y(4)\\ Y(6)\\ Y(1)\\ Y(3)\\ Y(5)\\ Y(7) \end{bmatrix} = \frac{1}{2}\begin{bmatrix} C_4 & C_4 & C_4 & C_4 & 0 & 0 & 0 & 0\\ C_2 & C_6 & -C_6 & -C_2 & 0 & 0 & 0 & 0\\ C_4 & -C_4 & -C_4 & C_4 & 0 & 0 & 0 & 0\\ C_6 & -C_2 & C_2 & -C_6 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & C_1 & C_3 & C_5 & C_7\\ 0 & 0 & 0 & 0 & C_3 & -C_7 & -C_1 & -C_5\\ 0 & 0 & 0 & 0 & C_5 & -C_1 & C_7 & C_3\\ 0 & 0 & 0 & 0 & C_7 & -C_5 & C_3 & -C_1 \end{bmatrix} \begin{bmatrix} x(0)+x(7)\\ x(1)+x(6)\\ x(2)+x(5)\\ x(3)+x(4)\\ x(0)-x(7)\\ x(1)-x(6)\\ x(2)-x(5)\\ x(3)-x(4) \end{bmatrix}$$

…..(5)

By exploiting the coefficient matrix in above discussed form, multiplications can be reduced to 32, at the expense of further 8 additions/subtractions[7].

$$Y(0) = [x(0) + x(1) + x(2) + x(3) + x(4) + x(5) + x(6) + x(7)].C_4$$

$$Y(1) = [x(0) - x(7)]C_1 + [x(1) - x(6)]C_3 + [x(2) - x(5)]C_5 + [x(3) - x(4)]C_7$$

$$Y(2) = [x(0) - x(3) - x(4) + x(7)]C_2 + [x(1) - x(2) - x(5) + x(6)]C_6$$

$$Y(3) = [x(0) - x(7)]C_3 - [x(1) - x(6)]C_7 - [x(2) - x(5)]C_1 - [x(3) - x(4)]C_5$$

$$Y(4) = [x(0) + x(7) - x(1) - x(6) - x(2) - x(5) + x(3) + x(4)]C_4$$

$$Y(5) = [x(0) - x(7)]C_5 - [x(1) - x(6)]C_1 + [x(2) - x(5)]C_7 + [x(3) - x(4)]C_3$$

$$Y(6) = [x(0) + x(7) - x(1) - x(6)]C_6 + [x(2) + x(5) - x(1) - x(6)]C_2$$

$$Y(5) = [x(0) - x(7)]C_7 - [x(1) - x(6)]C_5 + [x(2) - x(5)]C_3 - [x(3) - x(4)]C_1$$

…..(6)

On application of DA to Eq. 6, these 32 multiplication operation can be replaced by addition and shifting operation.

DA is a very common method to implement multiplication without multiplier. Traditional DA differs from NEDA primarily in two ways:

1. In DA, input words are distributed into bits, making DA mechanism a bit-serial design and
2. ROM is introduced to store a lookup table obtained from pre-computing results for all possible combinations of input bit patterns.

Consider following example of sum of products for NEDA:

$$Y = \sum_{k=1}^{L} A_k X_k = [A_1 A_2 \cdots A_L].\begin{bmatrix} X_1\\ X_2\\ \vdots\\ X_L \end{bmatrix} \quad \ldots\ldots(7)$$

Where A_k is constant, X_k is input data. A_k can be represented as:

$$A_k = -A_k^M 2^M + \sum_{i=N}^{M-1} A_k^i 2^i$$

$$= [2^N\ 2^{N+1} \cdots 2^M]\begin{bmatrix} A_1^N & A_2^N & \cdots & A_L^N\\ A_1^{N+1} & A_2^{N+1} & \cdots & A_L^{N+1}\\ \vdots & \vdots & \ddots & \vdots\\ -A_1^M & -A_2^M & \cdots & -A_L^M \end{bmatrix}\begin{bmatrix} X_1\\ X_2\\ \vdots\\ X_L \end{bmatrix}$$

$$= [2^N\ 2^{N+1} \cdots 2^M]\begin{bmatrix} Y^N\\ Y^{N+1}\\ \vdots\\ -Y^M \end{bmatrix} \quad \ldots\ldots(8)$$

From above matrix one can conclude that as the multiplication matrix consists of only '0' and '1', so the computation simply reduces to two operations: addition and shifting.

The constants $C_1, C_2, C_3, C_5, C_6, C_7$ can be assumed to have following values as discussed before (in section II as $(C_X = \cos\left(\frac{x\pi}{16}\right))$

$$C_1 = \frac{1}{2}\cos\left(\frac{\pi}{16}\right), \quad C_3 = \frac{1}{2}\cos\left(\frac{3\pi}{16}\right),$$
$$C_5 = \frac{1}{2}\cos\left(\frac{5\pi}{16}\right), \quad C_7 = \frac{1}{2}\cos\left(\frac{3\pi}{16}\right),$$
$$C_2 = \frac{1}{2}\cos\left(\frac{\pi}{16}\right), \quad C_6 = \frac{1}{2}\cos\left(\frac{3\pi}{8}\right),$$
$$C_4 = \frac{1}{2}\cos\left(\frac{4\pi}{16}\right). \quad \ldots\ldots(9)$$

Y(0),Y(1),Y(2)…….Y(7) as discussed in section (II) can be calculated by exploiting symmetry in DCT and DA. It can be understood by taking some examples (Y1 from Eq.6 is repeated here):

$$Y(1) = [x(0) - x(7)]C_1 + [x(1) - x(6)]C_3 + [x(2) - x(5)]C_5 + [x(3) - x(4)]C_7$$

$\ldots\ldots(10)$

Eq.10 can again be represented in vector matrix form as shown below where values of C_1, C_3, C_5, $C_7$are taken from Eq. 9

$$Y(1) = \left[\left(\frac{1}{2}\cos\left(\frac{\pi}{16}\right)\right)\left(\frac{1}{2}\cos\left(\frac{3\pi}{16}\right)\right)\left(\frac{1}{2}\cos\left(\frac{5\pi}{16}\right)\right)\left(\frac{1}{2}\cos\left(\frac{7\pi}{16}\right)\right)\right]\begin{bmatrix}x(0)-x(7)\\x(1)-x(6)\\x(2)-x(5)\\x(3)-x(4)\end{bmatrix}$$

……..(11)

The value of $\left(\frac{1}{2}\cos\left(\frac{\pi}{16}\right)\right)$ = 0.49039 and its binary representation is:

$$\frac{1}{2}\cos\left(\frac{\pi}{16}\right) = (0.49039)_{10} = (.011111011000)_2$$
…..(12)

The above Eq. can also be represented as:

$$\frac{1}{2}\cos\left(\frac{\pi}{16}\right) = [-2^0 \quad 2^{-1} \quad \dots \quad 2^{-11} \quad 2^{-12}]\begin{bmatrix}0\\0\\1\\1\\1\\1\\0\\1\\1\\0\\0\\0\end{bmatrix}$$

So, the Eq. for Y(1) in vector form can be represented as shown below:

$$Y(1) = [-2^0 \quad 2^{-1} \quad \dots \quad 2^{-11} \quad 2^{-12}]\begin{bmatrix}0&0&0&0\\0&0&0&0\\1&1&1&0\\1&1&0&0\\1&0&0&1\\1&1&0&1\\1&0&1&0\\0&1&1&0\\1&0&1&0\\1&0&0&1\\0&1&0&1\\0&1&0&1\\0&0&1&1\end{bmatrix}\begin{bmatrix}x(0)-x(7)\\x(1)-x(6)\\x(2)-x(5)\\x(3)-x(4)\end{bmatrix}$$

$$= [-2^0 \quad 2^{-1} \quad \dots \quad 2^{-11} \quad 2^{-12}]\begin{bmatrix}Y^0(1)\\Y^1(1)\\\dots\\Y^{11}(1)\\Y^{12}(1)\end{bmatrix}$$

…..(13)
Here,

$$Y^0(1) = Y^1(1) = 0$$
$$Y^2(1) = [x(0)-x(7)] + [x(1)-x(6)] + [x(2)-x(5)]$$
……..
$$Y^{11}(1) = [x(1)-x(6)] + [x(3)-x(4)]$$
$$Y^{12}(1) = [x(2)-x(5)] + [x(3)-x(4)]$$
……(14)

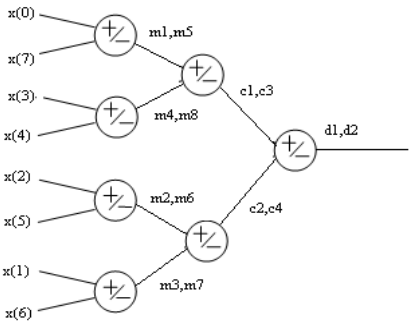

Fig. 1 Adder matrix for intermediate calculations

Fig (1) represents the adder matrix for the intermediate values required for Y(0),Y(1),Y(2),Y(3),Y(4),Y(5),Y(6),Y(7). Where,

$$m1 = x0 + x7 \qquad m2 = x1 + x6$$
$$m3 = x2 + x5 \qquad m4 = x3 + x4$$
$$m5 = x0 - x7 \qquad m6 = x1 - x6$$
$$m7 = x2 - x5 \qquad m8 = x3 - x4$$
$$c1 = m1 + m4 \qquad c2 = m2 + m3$$
$$c3 = m1 - m4 \qquad c4 = m2 - m3$$
$$d1 = c1 + c2 \qquad d2 = c1 - c2$$
…..(15)

In a similar way, 1-D DCT vector coefficient Y5 can be expressed as

$$Y(1) = \left[\left(\frac{1}{2}\cos\left(\frac{5\pi}{16}\right)\right)\left(-\frac{1}{2}\cos\left(\frac{\pi}{16}\right)\right)\left(\frac{1}{2}\cos\left(\frac{7\pi}{16}\right)\right)\left(\frac{1}{2}\cos\left(\frac{3\pi}{16}\right)\right)\right]\begin{bmatrix}x(0)-x(7)\\x(1)-x(6)\\x(2)-x(5)\\x(3)-x(4)\end{bmatrix}$$

$$Y(5) = [-2^0 \quad 2^{-1} \quad \dots \quad 2^{-11} \quad 2^{-12}]\begin{bmatrix}0&1&0&0\\0&1&0&0\\1&0&0&1\\0&0&0&1\\0&0&1&0\\0&0&1&1\\1&0&0&0\\1&1&0&1\\1&0&0&0\\0&1&1&0\\0&0&1&1\\0&0&1&1\\1&0&1&0\end{bmatrix}\begin{bmatrix}x(0)-x(7)\\x(1)-x(6)\\x(2)-x(5)\\x(3)-x(4)\end{bmatrix}$$

….. (16)

III. PROPOSED METHOD FOR SHIFTING AND ADDING

In the proposed method, the shifting and adding operation are performed separately by fixed point calculation for integer and decimal part.The input pixel values being 8 bits with MSB representing sign bit, the adder matrix (as can be seen from Fig. 1) of the maximum bit length for the matrix addition will be of 12 bits (m1+c1+c3+ sign bit).so every individual vector ($Y^0(1), Y^1(2), \dots\dots\dots\dots, Y^{11}(1), Y^{12}(1)$ is of 13 bits with MSB representing the sign bit before decimal.

As evident from the Eq.(16) first value is not shifted as it is multiplied by -2^0 and twelve zero will be padded in LSB

to represent no fraction part. Next combination will be multiplied by 2^{-1} so it will be shifted 1 bit left (for shifting we are padding MSB with the MSB bit of $Y^1(1)$which will be either 0 or 1 depending upon positive or negative result), and LSB will be padded with eleven zeros as one of the LSB of $Y^1(1)$ will get shifted to the fraction part. Similarly, next vector is to be multiplied with2^{-2} ,2 MSB bits of next vector will be padded
with the MSB bit of $Y^2(1)$, and ten zeros at the LSB will be padded as discussed above. Each vector is thus shifted by one bit and added to the previous one, so the last value will totally be shifted to the fraction part. The above stated thing can be understood by following assignment example:

Integer values Decimal values

Z={

$$Y^0(1)[11:0], \quad 000000000000\} +$$
$$\{Y^1(1)[11], Y^1(1)[11:1], \quad Y^1(1)[1], 00000000000\} +$$
$$\{Y^2(1)[11], Y^2(1)[11], Y^2(1)[11:2], \quad Y^1(1)[2:0], 0000000000\} +$$
$$\vdots$$
$$\vdots$$
$$\{Y^{12}(1)[11], Y^{12}(1)[11], \ldots\ldots\ldots\ldots \quad Y^{12}(11:0)\}$$

By doing such a shifting and adding operation, more precise result are obtained and the Table I presents the comparison results with existing method.

TableI. Comparison of proposed method results with the previous existing architecture.

Input pixel variables	Input pixel values	Calculated 1-D DCT coefficients through MATLAB	Calculated values from[8]	Calculated values from [5]	Calculated values from proposed method
x(0)	60	151.3209	149	149	151.3046
x(1)	40	-32.4895	-36	-32	-33.5244
x(2)	25	33.1588	31	31	33.1611
x(3)	55	-1.7108	-5	-2	-2.2888
x(4)	40	17.6777	16	16	17.6757
x(5)	42	18.5074	14	18	18.5046
x(6)	82	-16.0309	-18	-18	-17.9736
x(7)	84	-5.0975	-11	-9	-5.9050

The design has been simulated in Xilinx 13.1 and has been implemented using vertex -5 (XC5VLX110T) FPGA. The results obtained are more precise from the previous reported architectures. The total delay involved in the proposed module is 17.903ns and the device utilization is only 2% (Slice/LUTs used are 1635 out of 69120 (2%)).

IV. CONCLUSION

The proposed design is a ROM free multiplication technique through adders and shifter. DA is mathematically proved for 1-D DCT module and explanation for different input vectors has been given in this paper. By analyzing equations for 1-D DCTY(0),Y(1),Y(2),Y(3),Y(4),Y(5),Y(6),Y(7), it is concludedthat many adder/ subtractor can be shared in order to minimize the hardware by using compressed adder/subtractor matrix which can be designed according to the equations discussed in the paper. The delay of the proposed work is 17 ns and the overall design uses only 2% of the resources available. With proposed design efficient hardware utilization can be achieved and results in saving of chip area.

REFERENCES

[1] S. A. White, "Applications of distributed arithmetic to digital signal processing: a tutorial review," *IEEE ASSP Magazine*, vol.6, no.3, pp.4-19, 1989.

[2] M.T. Sun, T.C. Chen, A.M. Gottlieb, ''VLSI Implementation of a 16x16 Discrete Cosine Transform," *IEEE Transactions on Circuits and Systems*, vol.36, no. 4, pp. 610 – 617, 1989.

[3] A.Shams, W.Pan,A.Chidanandan and M. A. Bayoumi, "A Low Power High Performance Distributed DCT Architecture,"*Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI'02)*, pp.21-27, 2002.

[4] M. Hamad, E. Haidamous "Image Compression on FPGA using DCT,*" International conference on Advances in Computational Tools for Engineering Application*, pp.320-323, 2009

[5] V. K. Sharma, K. K. Mahapatra and U. C. Pati, "An Efficient Distributed Arithmetic based VLSI Architecture for DCT", *International conference on Devices and Communications*, pp. 1-5, 2011.

[6] S. Sanjeevannanavar, NagamaniA.N "Efficient Design and FPGA Implementation of JPEG Encoder using Verilog HDL", *International conference on Nanoscience and Technology*, pp. 584-588, 2011.

[7] D.W. Trainor, J.P. Heron, R.F. Woods, "Implementation of The 2D DCT Using A Xilinx XC6264 FPGA",*IEEE workshop on Signal Processing Systems, Design and Implementation,pp. 541-550, 1997*

[8] P.Chungan, C.Xixin, Y. Dunshan and Z. Xing, "A 250MHzoptimized distributed architecture of2D 8x8 DCT," *International Conference on ASIC*, pp. 189 – 192, 2007.

[9] S.Saravanan, V. Bhaskar "A High Performance Parallel Distributed Arithmetic DCT Architecture for H.264 Video Compression," *ISSN European Journal of Scientific Research*, vol.42, pp.558-564 ,2010.