An Optimised Distributed Arithmetic Architecture for 8×8 DTT

Ranjan K. Senapati [#1], P.M.K. Prasad[*2], S. Shabnam[#3], Ch. Jagadeesh [#4], K. Srinath[#5]

[#] Dept of Electronics and Communication Engg. K L University
Green Fields, Vaddeswaram, Guntur Dist, AP
*Dept. of ECE, GMR Institute of Technology, Rajam.
Srikakulum dist., AP, 532127, India.
[1] ranjan.senapati@kluniversity.in, [2]mkprasad.p@gmrit.org
[3] syedshabnam39@gmail.com, [4] jagadeesh.chadarajupalli@gmail.com
[5]srinathkommineni24249@gmail.com

***Abstract*—Discrete Tchebichef Transform (DTT) is an orthogonal transform and is used in many applications like image and video compression, feature extraction, artefact analysis, blind integrity verification and pattern recognition. In comparison with DCT, DTT has better image reconstruction quality for certain class of images. Direct implementation of DTT requires large number of multiplications, which are time-consuming and expensive in a simple processor. To perform in real time, these large number of operations can be completely avoided in our proposed architecture. The proposed architecture uses distributed (DA) based technique which offers high speed and small area. The basic architecture consists of one dimensional (1D) row DTT followed by a transpose register array and another 1D column DTT. The 1D DTT structure only requires 15 adders to build a compressed adder matrix and is also ROM free. Compared with DCT architecture, the proposed architecture shows an improvement in speed and reduction in area by 5% on a Xilinx vertex-4 FPGA platform.**

Keyword-Discrete Tchebichef Transform, Discrete Cosine Transform, Distributed Arithmetic, New Distributed Arithmetic, Image Compression, FPGA.

I. INTRODUCTION

Portable electronic systems like mobiles, PDAs, digital cameras run on batteries. In order to achieve maximum performance, the design of such devices should be optimised in terms of area and power. Image transforms such as discrete cosine transform (DCT) is widely used for processing and storage in such devices. For example, DCT is adopted as the core transform in JPEG, MPEGx, and H.26x [1]-[2]. Direct implementation of DCT needs floating point multiplication. Floating point multipliers are the major sources of area and power consumption for such devices. In order to eliminate multipliers, distributed arithmetic (DA) based technique has been emerged. DA was invented over two decades ago and has since been seen widespread applications in areas of VLSI implementation and DSP algorithms [3].

DA has become an efficient tool to implement multiply and accumulate (MAC) unit in a DSP processor. Using ROM based DA, the usage of multipliers in the MAC unit can be efficiently replaced by pre-computing all possible products and storing them in each address of ROM [4]-[10]. However, the size of ROM grows exponentially with the size of the transform and bit width precision. Usage of ROM can be eliminated if the set of inputs are of fixed size. This is done by distributing the coefficients to the input of the unit. The approached is called New Distributed Arithmetic (NEDA) [11]. Thus, NEDA can be used to implement the inner product of vectors in the form of 2's complement numbers using only addition followed by a small number of shifts at the final stage. Recently, NEDA is successfully applied to many transforms such as DCT [12]-[13], DHT [14], FFT [15]-[16].

Discrete Tchebichef transform is a novel orthogonal transform which has similar energy compaction properties like DCT. Recently, DTT is applied to many image and signal processing applications, like compression [17]-[21], feature extraction [22], pattern recognition [23], blind integrity verification [24], and artefact measurement [25]. Some applications requires real time manipulation of images. So many fast algorithms and specific circuits have been developed using DCT [4]-[10]. As DTT outperform DCT for some class of images, it is expected that DTT can be applied for such applications. Recently, Oliveira *et al.* [26] have reported a low complexity approximated DTT transform. The transform can be applied in distributed video coding where video is encoded once and decoded several times. Several ROM free DA implementation of DCT, FFT and DHT is reported in literature [12]-[16]. However, to the best of our knowledge, ROM free DA implementation is not reported for DTT. Out of several approaches, Row-column decomposition method is best adopted for H/W implementation in DCT, FFT and DHT. Similar in kind, we have also implemented the ROM free DA implementation on DTT in this paper.

In this paper, we have proposed a reduced area and reduced power 2D DTT architecture based on row-column decomposition. In this method 1D-DTT is taken on row wise, and then column wise 1D-DTT is performed. A compressed adder/subtractor matrix is adopted. This results a reduced hardware with speed improvement. The input data width is kept at 8-bit and the DTT basis elements are set to 13-bit precision to ensure minimum error during reconstruction.

The rest of the paper is as follows: A brief explanation of NEDA is presented in section 2. In Section 3, DTT transform and formulation of DTT using DA approach is demonstrated. Proposed 2D DTT architecture is presented by using row-column decomposition of 1D DTT in Section 4. Simulation and synthesis results are presented in Section 5. Finally, we conclude the paper with mentioning further improvements.

II. BRIEF EXPLANATION OF NEDA

Earlier works witness that transforms such as DCT, FFT, DHT, DST, etc. can be efficiently implemented using NEDA. This is because, NEDA improves the performance of the system in terms of area, speed and power. The brief mathematical derivation of NEDA can be as follows:

The inner product calculation of two sequences may be represented as

$$Z = \sum_{k=1}^{K} D_k X_k \tag{1}$$

where D_k are the constant coefficients and X_k are the varying inputs.

Equation (1) can be represented in matrix form as

$$Z = \begin{bmatrix} D_1 & D_2 & ..... & D_K \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ . \\ . \\ X_K \end{bmatrix} \tag{2}$$

Considering both D_i and X_i are in 2's complement format, these can be expressed in the form as

$$D_i = -d_i^M 2^M + \sum_{k=N}^{M-1} d_i^k 2^k \tag{3}$$

$d_i = 0$ or 1. d_i^M is the sign bit and d_i^N is the LSB. Substituting (3) in (2), we can write Z as

$$Z = \begin{bmatrix} -2^0 & 2^{-1} & ... & 2^{-(M-1)} \end{bmatrix} \begin{bmatrix} d_1^0 & . & . & . & d_k^0 \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ d_1^{-(M-1)} & . & . & . & d_k^{-(M-1)} \end{bmatrix} \begin{bmatrix} X_1 \\ . \\ . \\ . \\ X_{M-1} \end{bmatrix} \tag{4}$$

The matrix d_i^k is a sparse matrix. The number of rows in d_i^k determines the precision. Equation (4) can be rearrange as

$$Z = \begin{bmatrix} -2^0 & 2^{-1} & ... & 2^{-(M-1)} \end{bmatrix} \begin{bmatrix} W_1 \\ . \\ . \\ . \\ W_{M-1} \end{bmatrix} \tag{5}$$

where, $$\begin{bmatrix} W_0 \\ . \\ . \\ . \\ W_{M-1} \end{bmatrix} = \begin{bmatrix} d_1^0 & . & . & . & d_k^0 \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ d_1^{-(M-1)} & . & . & . & d_k^{-(M-1)} \end{bmatrix} \begin{bmatrix} X_1 \\ . \\ . \\ . \\ X_{M-1} \end{bmatrix}$$

The [W] represents the sum/difference of the input depending on the coefficient values. Multiplication with 2^{-k}, where $k \in Z^+$ in (5) can be realized with shifters. Thus the output Z is realized by shift and add operations.

The output Z is essentially a column matrix consisting of partial products. Therefore, NEDA based architecture designs have less critical path compared to traditional MAC unit without multipliers and memory.

III. DTT TRANSFORM AND FORMULATION USING DISTRIBUTED ARITHMETIC

The 2D DTT (Y_{pq}) of order $p+q$ is defined as [18]

$$Y_{pq} = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} t_p(x) t_q(y) X(x,y) \tag{6}$$

where, $t_p(x)$ and $t_q(y)$ are Tchebichef polynomials of order p and q respectively. $X(x,y)$ is the input 2D signal (e.g., image). $p,q = 0,1,.....,N-1$. The values of Tchebichef polynomials are defined using the recursive relation as

$$t_n(i) = (A_1 i + A_2) t_{n-1}(i) + A_3 t_{n-2}(i) \tag{7}$$

where, $A_1 = \frac{2}{n}\sqrt{\frac{4n^2-1}{N^2-n^2}}$, $A_2 = \frac{1-N}{n}\sqrt{\frac{4n^2-1}{N^2-n^2}}$

and $A_3 = \frac{n-1}{n}\sqrt{\frac{2n+1}{2n-3}\sqrt{\frac{N^2-(n-1)^2}{N^2-n^2}}}$

Using separable properties, we can write (6) as

$$Y_{pq} = \sum_{x=0}^{N-1} t_p(x) \sum_{y=0}^{N-1} t_q(y) X(x,y) \tag{8}$$

Equation (8) can be written as

$$Y_{pq} = \sum_{x=0}^{N-1} t_p(x) g_q(x) \tag{9}$$

where, $g_q(x) = \sum_{y=0}^{N-1} t_q(y) X(x,y)$ is the row wise 1D DTT. It is clear from (9) that 2D DTT can be evaluated by taking row wise 1D DTT and then column wise 1D DTT.

Equation (6) can be also written as

$$Y_{pq} = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} \psi(p,q,x,y) X(x,y) \tag{10}$$

Where $\psi(p,q,x,y) = t_p(x) t_q(y)$ is called the basis matrix, and is defined as follows:

$$\psi = \begin{bmatrix} t_p(0)t_q(0) & t_p(0)t_q(1) & . & . & t_p(0)t_q(N-1) \\ t_p(1)t_q(0) & t_p(1)t_q(1) & . & . & t_p(1)t_q(N-1) \\ . & . & . & . & . \\ . & . & . & . & . \\ t_p(N-1)t_q(0) & t_p(N-1)t_q(1) & . & . & t_p(N-1)t_q(N-1) \end{bmatrix} \tag{11}$$

Expanding the basis matrix defined in (11), we can evaluate the transform kernel of for $N=8$ as

$$\psi = \begin{bmatrix} 0.3536 & 0.3536 & 03536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 \\ -0.5401 & -0.3858 & -0.2315 & -0.0772 & 0.0772 & 0.2315 & 0.3858 & 0.5401 \\ 0.5401 & 0.0772 & -0.2315 & -0.3858 & -0.3858 & -0.2315 & 0.0772 & 0.5401 \\ -0.4308 & 0.3077 & 0.4308 & 0.1846 & -0.1846 & 0.4308 & -0.3077 & 0.4308 \\ 0.2820 & -0.5238 & -0.1209 & 0.3626 & 0.3626 & 0.1209 & -0.5238 & 0.2820 \\ -0.1498 & 0.4922 & -0.3638 & -0.3210 & 0.3210 & 0.3638 & -0.4922 & 0.1498 \\ 0.0615 & -0.3077 & 0.5539 & -0.3077 & -0.3077 & 0.5539 & -0.3077 & 0.0615 \\ -0.0171 & 0.1195 & -0.3585 & 0.5974 & -0.5974 & 0.3585 & -0.1195 & 0.0171 \end{bmatrix}$$

The transform in (10) can be written in matrix form as

$$Y = \psi \; X \; \psi^T \tag{12}$$

ψ^T denotes the transpose operation.

Representing (12) in 1D form, we can write

$$Y = \psi \; X \tag{13}$$

Elaborating (13) we can write with assumption that Y and X are 8 input column vectors

$$\begin{bmatrix} Y(0) \\ Y(1) \\ Y(2) \\ Y(3) \\ Y(4) \\ Y(5) \\ Y(6) \\ Y(7) \end{bmatrix} = \begin{bmatrix} 0.3536 & 0.3536 & 03536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 \\ -0.5401 & -0.3858 & -0.2315 & -0.0772 & 0.0772 & 0.2315 & 0.3858 & 0.5401 \\ 0.5401 & 0.0772 & -0.2315 & -0.3858 & -0.3858 & -0.2315 & 0.0772 & 0.5401 \\ -0.4308 & 0.3077 & 0.4308 & 0.1846 & -0.1846 & 0.4308 & -0.3077 & 0.4308 \\ 0.2820 & -0.5238 & -0.1209 & 0.3626 & 0.3626 & 0.1209 & -0.5238 & 0.2820 \\ -0.1498 & 0.4922 & -0.3638 & -0.3210 & 0.3210 & 0.3638 & -0.4922 & 0.1498 \\ 0.0615 & -0.3077 & 0.5539 & -0.3077 & -0.3077 & 0.5539 & -0.3077 & 0.0615 \\ -0.0171 & 0.1195 & -0.3585 & 0.5974 & -0.5974 & 0.3585 & -0.1195 & 0.0171 \end{bmatrix} \begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \\ X(6) \\ X(7) \end{bmatrix}$$

The value of above basis matrix of 1D DTT can be broken as,

$$\begin{aligned}
Y(0) &= 0.3536\,[X(0) + X(7)] + 0.3536\,[X(1) + X(6)] + 0.3536\,[X(2) + X(5)] + 0.3536\,[X(3) + X(4)] \\
Y(1) &= -0.5401\,[X(0) - X(7)] - 0.3858\,[X(1) - X(6)] - 0.3536\,[X(2) + X(5)] + 0.3536\,[X(3) + X(4)] \\
Y(2) &= -0.5401\,[X(0) - X(7)] - 0.3858\,[X(1) - X(6)] - 0.2315\,[X(2) + X(5)] - 0.3858\,[X(3) + X(4)] \\
Y(3) &= -0.4308\,[X(0) - X(7)] + 0.3077\,[X(1) - X(6)] + 0.4308\,[X(2) - X(5)] + 0.1846\,[X(3) - X(4)] \\
Y(4) &= 0.2820\,[X(0) + X(7)] - 0.5238\,[X(1) + X(6)] - 0.1209\,[X(2) + X(5)] + 0.3626\,[X(3) + X(4)] \\
Y(5) &= -0.1498\,[X(0) - X(7)] + 0.4922\,[X(1) - X(6)] - 0.3638\,[X(2) - X(5)] - 0.3210\,[X(3) - X(4)] \\
Y(6) &= 0.0615\,[X(0) + X(7)] - 0.3077\,[X(1) + X(6)] + 0.5539\,[X(2) + X(5)] - 0.3077\,[X(3) + X(4)] \\
Y(7) &= -0.0171\,[X(0) - X(7)] + 0.1195\,[X(1) - X(6)] - 0.3585\,[X(2) - X(5)] + 0.5974\,[X(3) - X(4)]
\end{aligned} \tag{14}$$

Now, ROM free DA based algorithm can be used to implement the above equation of DTT. Constant coefficients can be written in 2's complement binary fractional form to exploit DA. For example, $Y(1)$ coefficient can be written with 12-bit DA precision according to (14) as:

$$Y(1)=\begin{bmatrix}-2^{0} & 2^{-1} & \cdots & 2^{-12}\end{bmatrix}\begin{bmatrix}1&1&1&1\\0&1&1&1\\1&0&1&1\\1&0&0&1\\1&1&0&0\\0&1&0&1\\1&1&1&1\\0&0&0&0\\1&1&0&0\\1&0&1&0\\0&0&0&0\\1&1&1&1\\1&1&1&1\end{bmatrix}.\begin{bmatrix}(X(0)-X(7))\\(X(1)-X(6))\\(X(2)-X(5))\\(X(3)-X(4))\end{bmatrix} \quad (15)$$

where powers of Y denote the number of times shifting is required after evaluating right part of (15).

From (15), it can be deduced that $Y(1)$ is the sum of the terms $Y^{0}(1), Y^{1}(1)/2, \ldots\ldots\ldots\ldots, Y^{12}(1)/2^{12}$. The computation process of $Y^{0}(1), Y^{1}(1), \ldots\ldots, Y^{12}(1)$ are shown in Fig. 1. Here adders and subtractors are being shared. i.e. addition and subtraction operation are done by a simple ALU, which comprises adder and subtractor simultaneously. Therefore, the usage of hardware can be further reduced.

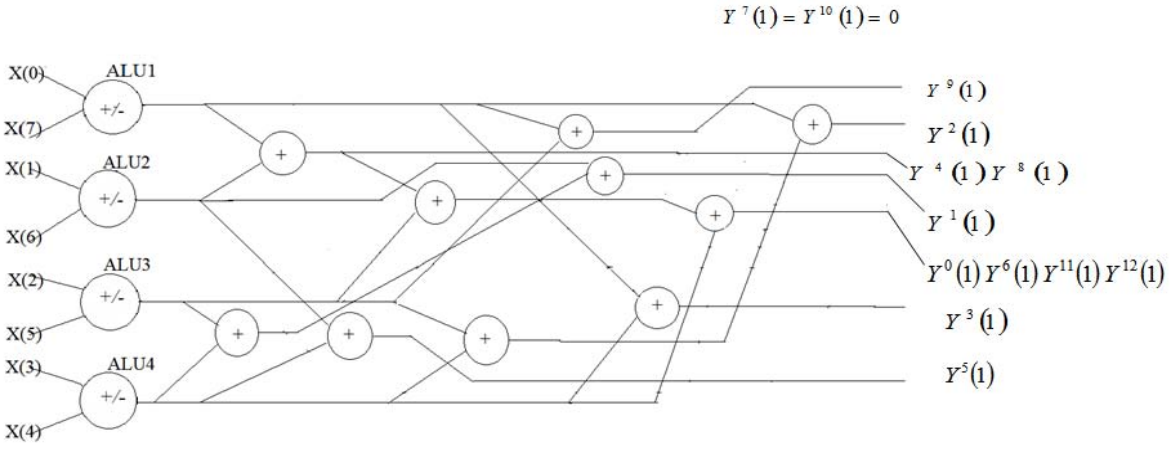


Fig. 1. The adder/subtractor matrix of $Y(1)$

In the next step, $Y^{0}, Y^{1}, \ldots\ldots\ldots Y^{11}, Y^{12}$ are needed to shift and add up for finding the value of $Y(1)$. The structure of compressed shift-adder tree can be shown in Fig. 2. In Fig. 2 Mi where i=0, 1,…..,12 represents the number of right shift of the output of adder/substractor matrix of Fig. 1. For example, M1 represents Y shift right by 1 and M2 represents Y shift shift right by 2. If M1 is 10 bit and M2 is 9 bit after arithmetic shift, then the value of M1+M2=10 bits in compressed shift adder tree. The final value of Y(1) in the compressed adder tree is obtained by required number of shift of M values of Fig. 2.

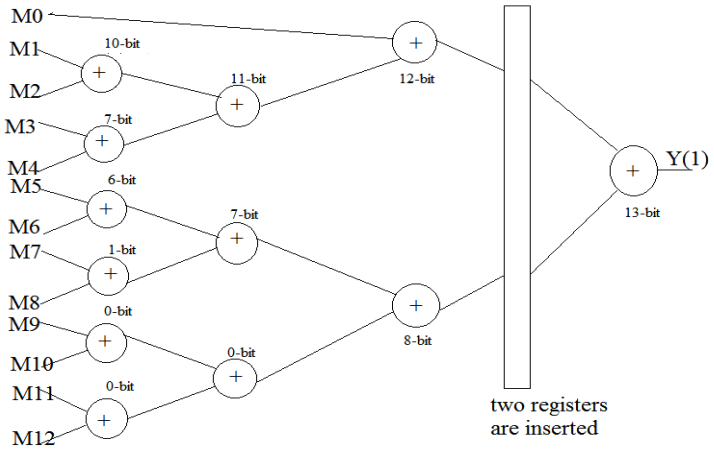


Fig. 2. Compressed Shift-Adder-Tree

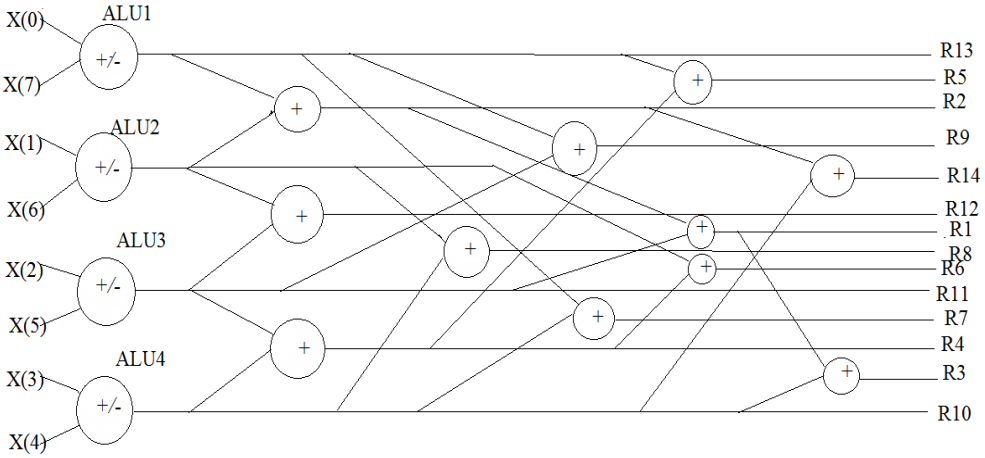


Fig. 3. Adder/Subtractor matrix of all data (Y(0),Y(1),Y(2),Y(3), Y(4), Y(5), Y(6), Y(7))

TABLE 1. Functions of each ALU for different coefficients

	Y(0)	**Y(1)**	**Y(2)**	**Y(3)**	**Y(4)**	**Y(5)**	**Y(6)**	**Y(7)**
ALU1	+	-	+	-	+	-	+	-
ALU2	+	-	+	-	+	-	+	-
ALU3	+	-	+	-	+	-	+	-
ALU4	+	-	+	-	+	-	+	-
$\mathbf{Y^0}$	0	R3	R4	R13	R12	R5	R8	R9
$\mathbf{Y^1}$	0	R6	R5	R13	R11	R5	R6	R5
$\mathbf{Y^2}$	R3	R5	R11	R12	R3	R2	0	R13
$\mathbf{Y^3}$	0	R7	0	R4	R12	R6	R8	R9
$\mathbf{Y^4}$	R3	R2	R8	R13	R8	R2	R8	R14
$\mathbf{Y^5}$	R3	R8	R7	R6	R14	R14	R9	R14
$\mathbf{Y^6}$	0	R3	R4	R6	R8	R8	R9	R12
$\mathbf{Y^7}$	R3	0	R2	R6	0	R12	R13	R2
$\mathbf{Y^8}$	0	R2	R8	R7	R12	R7	R3	R13
$\mathbf{Y^9}$	R3	R9	R12	R2	R8	R5	R9	R14
$\mathbf{Y^{10}}$	0	0	R2	R6	R8	R4	R11	R10
$\mathbf{Y^{11}}$	0	R3	R4	R13	R2	R13	R14	R4
$\mathbf{Y^{12}}$	0	R3	R4	R13	R7	R4	R14	R1

All the 1D DTT coefficients can be calculated by using adder/subtractor structure in Fig. 3. Table 1 describe the functionality of Fig. 3. The '+' and '-' sign in the rows indicate the required function (addition or subtraction) to be performed by the ALUs. Then, each DTT coefficient Y(i) can be obtained by

summing R_i values in the same column. Shifting must be done before summing the column values and number of bit to be shifted is decided by the power of Y shown in that row. For example, the coefficient $Y(1)$ is calculated as follows:

$$Y(1)=(R2/2^4)+(R2/2^8)+(R3/2^0)+(R3/2^6)+(R3/2^{11})+(R3/2^{12})+(R5/2^2)+(R6/2^1)+(R7/2^3)+(R8/2^5)+(R9/2^9) \quad (16)$$

The R_i values obtained from Fig. 3 and Table 1 are

$$R2=[X(0)-X(7)]+[X(1)-X(6)]$$
$$R3=[X(0)-X(7)]+[X(1)-X(6)]+[X(2)-X(5)]+[X(3)-X(4)]$$
$$R5=[X(0)-X(7)]+[X(2)-X(5)]+[X(3)-X(4)]$$
$$R6=[X(1)-X(6)]+[X(2)-X(5)]+[X(3)-X(4)]$$
$$R7=[X(0)-X(7)]+[X(3)-X(4)]$$
$$R8=[X(1)-X(6)]+[X(3)-X(4)]$$
$$R9=[X(0)-X(7)]+[X(2)-X(5)]$$

Division operations of power of 2 in (16) is performed by shifting the corresponding R_i values right. Since the binary contents become smaller after shifting i.e., bit width decreases, adders can be made smaller accordingly in order to reduce the area. For example, if R_1 and R_2 sizes are of 11-bits each, then adder bit width required to add them should be of size 11-bits. Since R_5 is shifted by 2-bits and R_7 is shifted by 3-bits right, it can be realized by using 9-bits adder as shown in Fig. 4.

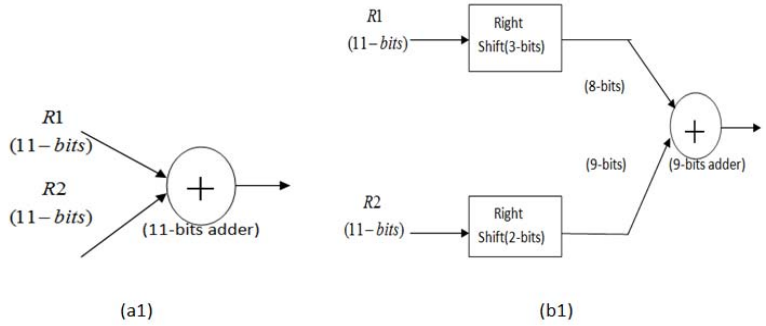
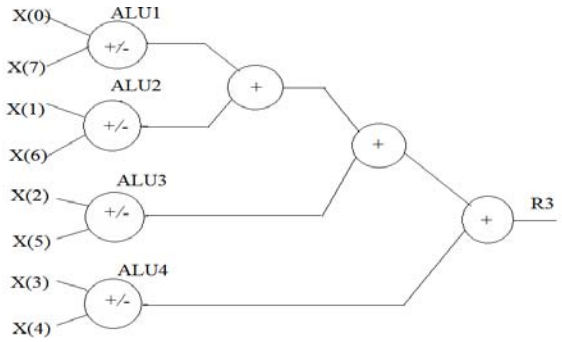


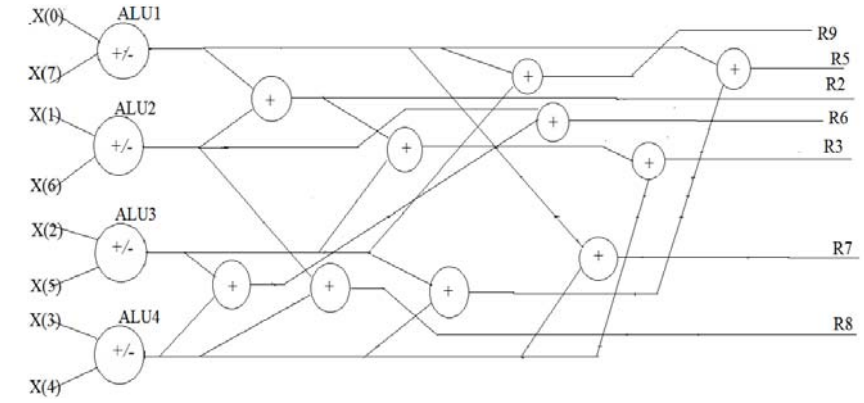
Fig.4. Adder bit width reduction in ROM free DA to save area and power (a1) without shift and (b1) with right shift

A. Realization of Y(0), Y(1), …., Y(7) of 8×1 DTT

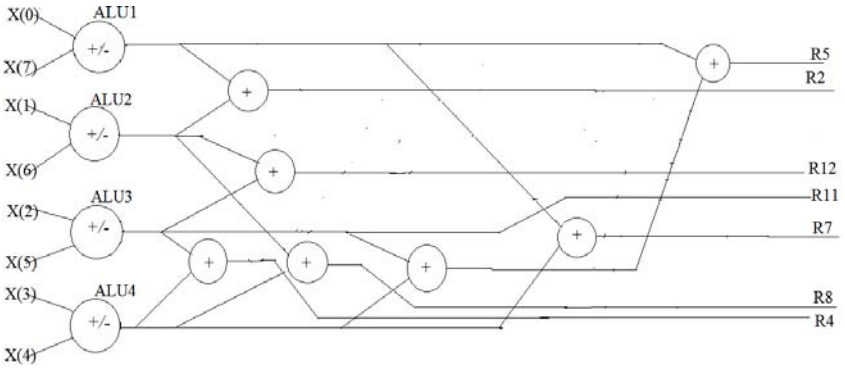
The hardware realization of 1D DTT outputs are presented in Fig. 5. The first step of realization is computed in figures from a-h below. In the next step R1, R2, ...,R14 are needed to shift and add up for the value of Y(0), Y(1).....,Y(7). The shift operation is implemented by wirings, which costs little delay and hardware resources.



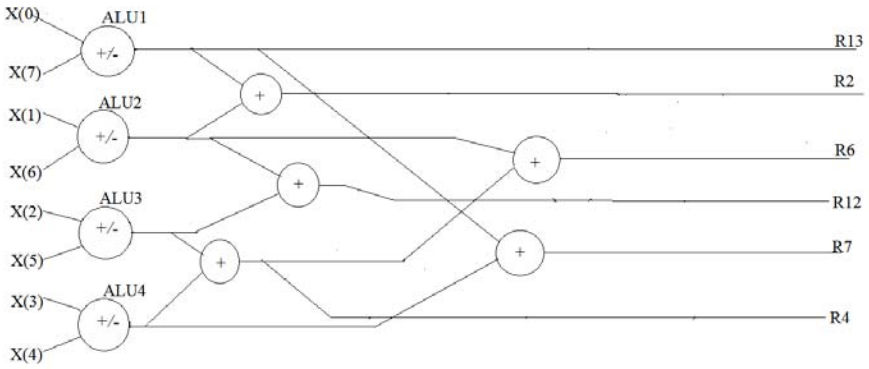
(a)


X(0)
X(7)
X(1)
X(6)
X(2)
X(5)
X(3)
X(4)
ALU1
ALU2
ALU3
ALU4
+/-
R9
R5
R2
R6
R3
R7
R8

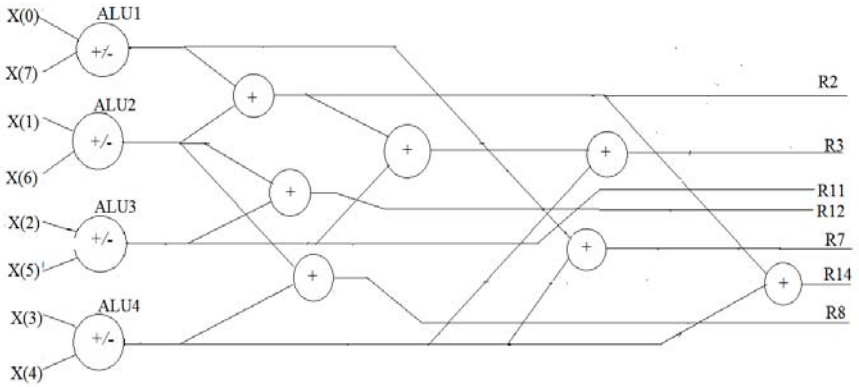
(b)


X(0)
X(7)
X(1)
X(6)
X(2)
X(5)
X(3)
X(4)
ALU1
ALU2
ALU3
ALU4
+/-
R5
R2
R12
R11
R7
R8
R4

(c)


X(0)
X(7)
X(1)
X(6)
X(2)
X(5)
X(3)
X(4)
ALU1
ALU2
ALU3
ALU4
+/-
R13
R2
R6
R12
R7
R4

(d)


X(0)
X(7)
X(1)
X(6)
X(2)
X(5)
X(3)
X(4)
ALU1
ALU2
ALU3
ALU4
+/-
R2
R3
R11
R12
R7
R14
R8

(e)

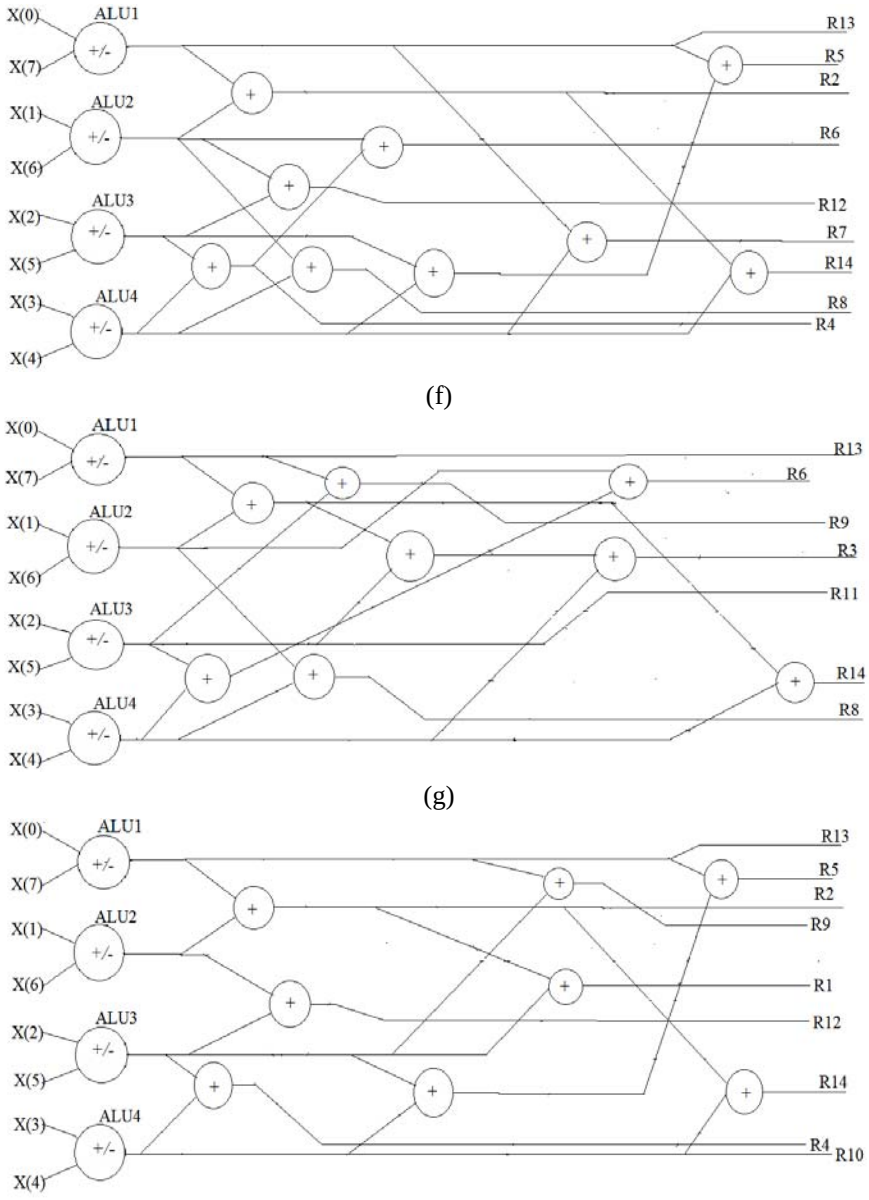


Fig. 5. Butterfly structures for computing 8×1 DTT for (a)Y(0), (b)Y(1), (c)Y(2), (d)Y(3), (e)Y(4), (f)Y(5), (g)Y(6), (h)Y(7).

IV. IMPLEMENTATION OF 2-D 8×8 DTT

2D DTT is a separable transform. Therefore, an 8×8 DTT can be implemented by row-column decomposition techniques, i.e., taking the 8×1 DTT of each row of the input data matrix. In Fig. 6 the input vectors $X_0 - X_7$ are applied to the 1D DTT. The transformed outputs are latched into the buffer one row at a time. The buffer needs 8 clock pulses to latch all the 8 rows of transformed coefficients. In order to perform column wise transformation, the transformed coefficients of each row need to be pass through a transposition matrix. The transposition matrix generally performs a row to column transformation which can be simply a wiring operation. The output from the transposition matrix need to be pushed in to the register file. This is done by pushing each column of the transformed coefficients per clock pulse (parallel load). In the same clock, the column transposition matrix converts the column of the transformed values to row and input the values to second 1D DTT module. It is seen that a total 8+8=16 clock pulses are required to get the first rows of transformed coefficients, i.e., Y0-Y7. Additional 1+1=2 clock pulses for initial clear and wait state between row-to-column transpositions. Therefore, a total of 18+8=26 clock pulses are required to do a complete 8×8 conversion.

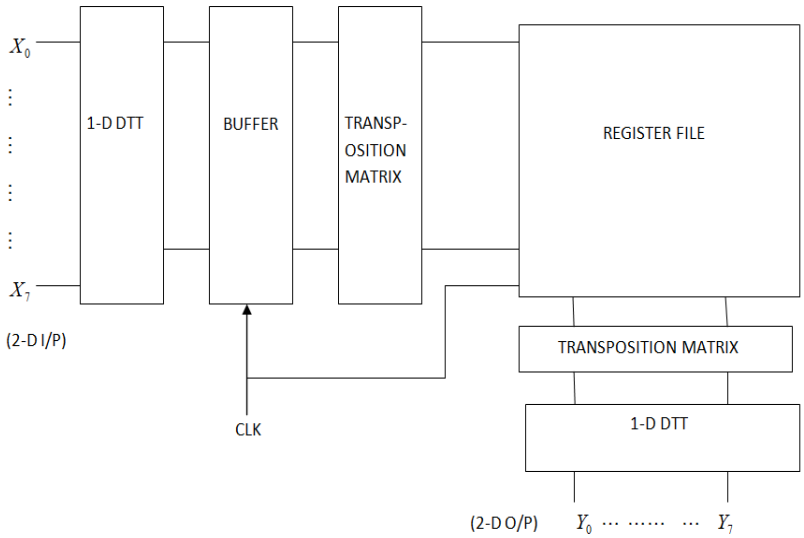


Fig. 6. Diagram of 2D 8×8 DTT core processor architecture using 1D DTT module

V. RESULTS AND DISCUSSIONS

Table 2 shows a comparison of adder cost savings between different methods of implementation. It can be observed that the proposed DTT implementation using NEDA shows 94% saving of the cost metric. In 2D DTT, the adder matrix consists of 4 ALUs and 11 adders while 2D DCT implementation consists 9 ALUs and 6 adders. Each ALU consists of two components (adder and subtractor). This implies that DCT has 15 adders and 9 subtractors, whereas in DTT there are 15 adders and 4 subtractors. Hence, there is a 24% adder cost saving in the proposed DTT.

A. Device and Power Utilization Summary

According to the results shown below, we find that 2-D DTT saves 5% more area than 2-D DCT. Area saving is estimated in terms of device utilization parameter from Table 3. The power and max. frequency comparison is shown in Table 4.

B. 2D DTT Simulation Results

Applying an arbitrary input X_{in} in (17), the Simulation results using ModelSim and Matlab is shown in Fig. 7 (a) and (b) respectively.

TABLE 2. Comparison of adder cost savings

Scheme	Adder Matrix	Adder bit-width	Saving
Direct DCT	308	2496	-
NEDA [11]	35	1800	88%
DCT in [12]	9ALU+6	850	92%
Proposed DTT	4ALU+11	840	94%

TABLE 3. Device utilization for 2D 8×8 DCT and 8×8 DTT on the Xilinx vertex-4 FPGA

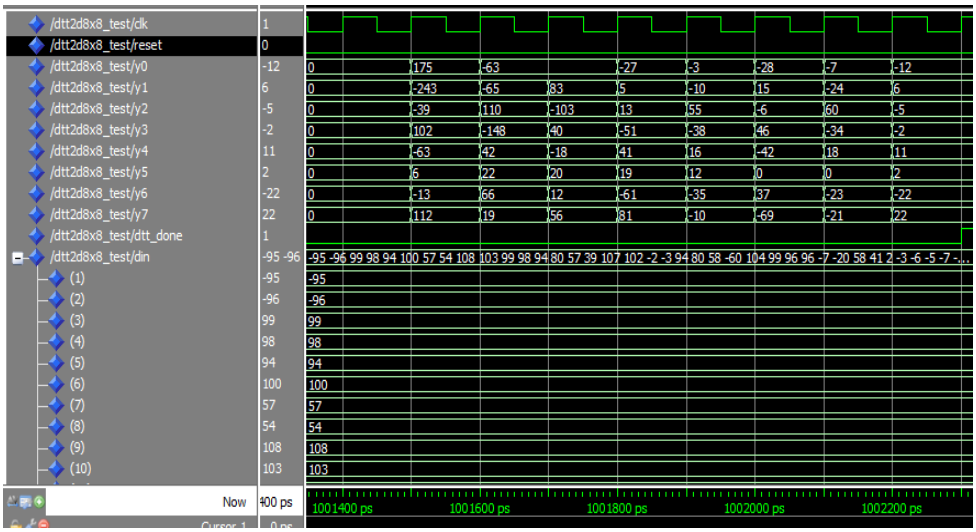
FPGA Resources	2D DCT	2D DTT	Available	Utilization For DCT	Utilization For DTT
No of Slices	2612	2596	5472	47%	47%
No of Slice Flip Flops	875	819	10944	7%	7%
No of 4-Input LUTs	4764	4165	10944	43%	38%
No of IOB Flip Flops	115	115	240	47%	47%

TABLE 4. Power and Frequency calculation on Xilinx XC4VFX12 FPGA

Parameter	**DCT**	**DTT**
Power (mW)	Total: 197; Static: 194; Dynamic: 23	Total: 186; Static: 167; Dynamic: 19
Max. Frequency (MHz)	87.7	111.38

$$X_{in} = \begin{bmatrix} -95 & -96 & 99 & 98 & 94 & 100 & 57 & 54 \\ 108 & 103 & 99 & 98 & 94 & 80 & 57 & 39 \\ 107 & 102 & -2 & -3 & 94 & 80 & 58 & -60 \\ 104 & 99 & 96 & 96 & -7 & -20 & 58 & 41 \\ 2 & -3 & -6 & -5 & -7 & -19 & -41 & -58 \\ -1 & -5 & -8 & -6 & -8 & -19 & 60 & 42 \\ -3 & -7 & -9 & -7 & -8 & -19 & -40 & -57 \\ -4 & -7 & -10 & -8 & -8 & -19 & -40 & -57 \end{bmatrix} \quad (17)$$

It can be noted that the error difference between the simulate output and actual output is +/- 4%. The actual 2D output results are shown in matrix. The error can be further reduced by increasing the precision of the DA length sequence.

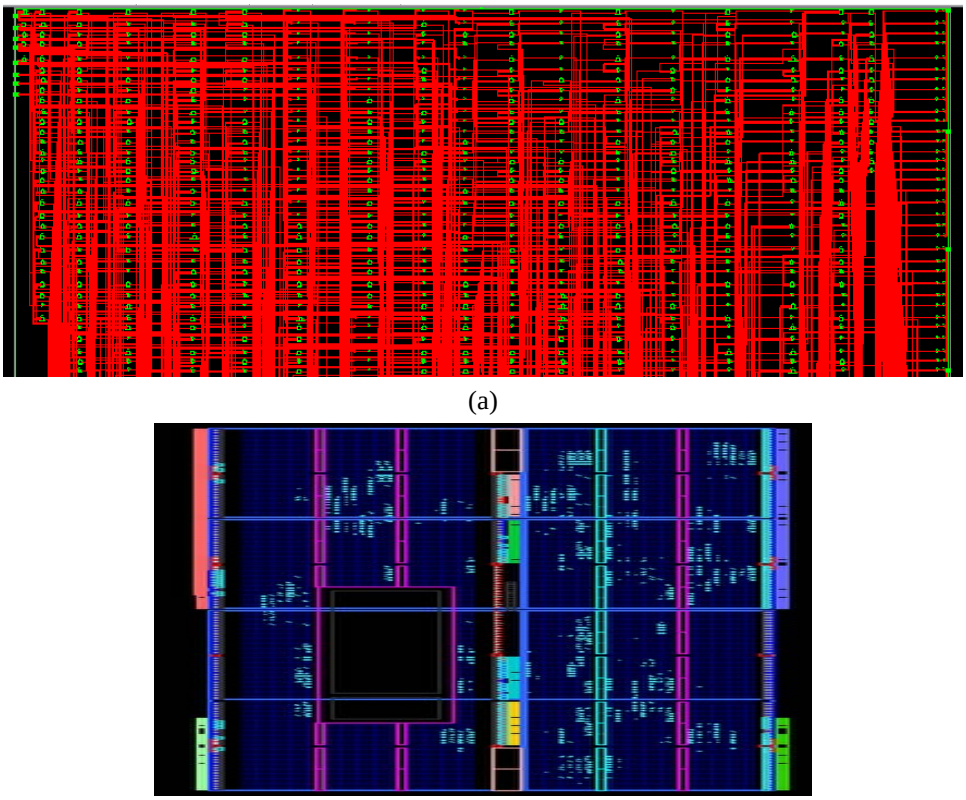


(a)

$$Y = \begin{bmatrix} 181 & -63 & -63 & -24 & -3 & -26 & -7 & -9 \\ -244 & -61 & 73 & 7 & -18 & 17 & -30 & 9 \\ -36 & 104 & -106 & 11 & 53 & -9 & 55 & -8 \\ 108 & -144 & 41 & -49 & -38 & 50 & -33 & 3 \\ -58 & 37 & -20 & 40 & 14 & -43 & 15 & 9 \\ 10 & 22 & 18 & 19 & 11 & -1 & 1 & 0 \\ -6 & 60 & 12 & -69 & -33 & 28 & -20 & -28 \\ 119 & 20 & 58 & 79 & -5 & -72 & -14 & 20 \end{bmatrix}$$

(b)

Fig. 7 Simulation Results of 2D DTT (a) using ModelSim, (b) Matlab



(a)

(b)

Fig. 8 (a) Partial synthesis results (b) Floor plan design of 8x8 DTT on Xilinx-4 FPGA device.

Fig. 8 (a) shows a partial synthesis results of 1D DTT. As the design is large, it is difficult to accommodate the entire synthesis result in one page. The floor plan design is also shown in Fig. 8 (b).

VI. CONCLUSION

In this paper we have presented an area efficient architecture for the computation of 2-D DTT. The architecture is ROM free and implemented in Xilinx vertex-4 FPGA. A detail comparison with ROM free 2-D DCT is carried out. An area reduction of 5% and power improvement of 6% is achieved in proposed architecture. We have selected row-column decomposition technique because of its computational advantages. The proposed DTT shows 20% adder cost savings over DCT by Chungan *et al*. Simulation results of 2D DTT in ModelSim shows the transform error falls within ±4% of the actual result. This is because of finite precision selected for the floating point basis values during DA calculation. The future direction is to develop novel integer Tchebichef transform (ITT) for extremely low power applications.

REFERENCES

[1] W.B.Pennebaker, J.L.Mitchell "JEPG still image compression standard," Chapman Hall, New York, 1993.

[2] I.E. Richardson, "H.264 and MPEG-4 video compression," John Wiley and Sons, 2003.

[3] S.A.White, "Applications of distributed arithmetic to digital signal processing: A tutorial review," IEEE ASSP Mag., Vol. 6, No. 3, pp. 4-19, Jul. 1989.

[4] M.T. Sun, L. Wu, and M.L.Liou, "A concurrent architecture for VLSI implementation of discrete cosine transform," IEEE Trans, Circuits Syst., Vol. CAS-34, no. 8, Aug. 1987, pp. 992-994.

[5] M. T. Sun, T.C. Chen, and A.M.Gouulieb, "VLSI implementation of a 16×16 discrete cosine transform," IEEE Trans. Circuits Syst., Vol. CAS-36, No. 4, Apr. 1989, pp. 610-617.

[6] S. Uramoto, Y. Inoue, A. Takabatake, J. Takeda, Y. Yamashita, H. Terane, and M. Yoshimoto, "A 100 MHz 2-D discrete cosine transform core processor", J. Solid-State Circuits, Vol. 27, Apr. 1992, pp. 492-498.

[7] Y. –H. Chan and W.-C.Siu, "On the realization of discrete cosine transform using distributed arithmetic," IEEE Trans, Circuits Syst. I: Fundam. Theory Appl., Vol. 39, no. 9, Sept. 1992, pp. 705-711.

[8] H.C. Karthanasis, "A low ROM distributed arithmetic implementation of the forward/inverse DCT/DST using rotations," IEEE Trans. Consumer Electron., Vol. 41, No. 2, May 1995, pp. 263-272.

[9] V. Srinivasan and K.J.R. Liu, "VLSI design of high-speed time recursive 2-D DCT/IDCT processor for video applications," IEEE Trans, Circuits Syst. Video Technol., Vol. 6, No. 1, Feb. 1996, pp. 87-96.

[10] D. Slawecki and W. Li, "DCT/IDCT processor design for high data rate image code," IEEE Trans, Circuits Syst. Video Technol., Vol. 2, No. 2, Jun. 1992, pp. 135-144.

[11] A. M. Shames, A. Chidanandan, W. Pan and M. A. Bayoumi, "NEDA: A low-power high-performance DCT Architecture," IEEE Trans. on signal process., Vol. 54, No. 3, Mar. 2006, pp. 955-963.

[12] P. Chungan, C A O Xixin, Y U Dunshan and Z Xing, "A 250 MHz optimized distributed architecture of 2D 8×8 DCT", 7th Int. Conf. on ASICs, Oct. 2007, pp. 189-192.

[13] A.Mankar, N Prasad, and A D Das, "FPGA Implementation of Retimed Low Power and High Throughput DCT Core Using NEDA,"SCES 2013, 12-14 April 2013, Allahabad, India.
[14] V. K. Sharma, R. Agrawal and K.K. Mahapatra, "2-D Separable discrete Hartley transform for efficient FPGA resource," ICCCT, 2010.
[15] A. Das, A. Mankar, N. Prasad, K. K. Mahapatra, A. S. Swain, "Efficient VLSI architectures of split radix FFT using NEDA," IJSCE, Vol. 3, No. 1, Mar. 2013, pp. 264-271.
[16] A.Mankar, A D Das, and N Prasad, "FPGA Implementation of 16-Point Radix-4 Complex FFT Core Using NEDA,"SCES 2013, 12-14 April 2013, Allahabad, India.
[17] G.K.Wallace, The JPEG still picture compression standard, Communications of the ACM, Vol. 34, No. 4, 1991, pp. 30-44.
[18] K. Nakagaki and R. Mukundan, "A fast 4×4 forward discrete Tchebichef transform algorithm," IEEE Signal Proc. Letts., Vol. 14, No. 10, Oct. 2010, pp. 684-687.
[19] R. K. Senapati, U. C. Pati and K. K. Mahapatra, "Reduced memory, low complexity embedded image coding algorithm using hierarchical listless DTT," IET Image Proc., Vol. 8, No. 4, Apr. 2014, pp. 1-26.
[20] R. K. Senapati, U. C. Pati and K. K. Mahapatra, "A fast zigzag prune 4×4 DTT algorithm for image compression," WSEAS Trans. on signal proc., Vol. 7, No. 1, 2011, pp. 34-43.
[21] S. Ishwar, P. K. Meher and M.N.S. Swamy, "A fast 4×4 algorithm and its application to image/video compression," ISCAS Seattle, USA, 2008, pp. 260-263.
[22] R. Mukundan, "Image analysis by Tchebichef moments," IEEE Trans. on image proc., Vol. 10, No. 9, 2001, pp. 1357-1364.
[23] Bin Xiao, Jian-Feng Ma, Jiang-Tao Cui, "Invariant pattern recognition using radial Tchebichef moments," Chinese conference on pattern recognition, China, 21-23 oct., 2010.
[24] Hui Huang, G. Coatrieux, Huazheng Shu, Limin Luo, Christian Roux, "Blind integrity verification of medical images," IEEE Trans. on information technology in Biomedicine, Vol. 16, No. 6, Nov. 2012, pp. 1122-1126.
[25] Leida Li, H. Zhu, G. Yang, J. Qian, "Referenceless measure of blocking artifacts by Tchebichef kernel analysis," IEEE signal proc. Letts., Vol. 21, No. 1, Jan 2014, pp. 122-125.
[26] P.A.M.Oliveira, R.J.Cintra, F.M.Bayer, S.Kulasekera and A.Madanayake, "A discrete Tchebichef transform approximation for image and video coding," IEEE signal proc. lett., Vol. 22, No. 8, Aug. 2015, pp. 1137-1141.

AUTHOR PROFILE

Ranjan Kumar Senapati received his M. Tech degree from National Institute of Technology, Warangal, India and Ph. D. degree from National Institute Technology, Rourkela, India in 2004 and 2014 respectively. At present he is working as a Professor in the department of Electronics and Communication Engineering, K L University, Vijayawada, Andhra Pradesh. He has published more than 20 research papers in international as well as national journals and conference proceedings. His research interests include development and VLSI implementation of image and video coding algorithms, microprocessor and microcontroller-based system designing and signal processing. He is a life member of various professional societies such as IETE and ISTE.

PMK Prasad received M.E. in Systems and Signal Processing from University college of Engineering, Osmania University, Hyderabad. Now, he is pursuing Ph.D. from Andhra University, Visakhapatnam. He has nearly twenty years of experience in teaching. Currently working as Associate professor in Dept. of ECE, GMR Institute of Technology, Rajam, India. His research interests include Signal processing, communications and Image processing.