Volume 6, Issue 4, April 2016 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Novel Three-Dimensional DCT Architecture

[1]P. B. Mohapatra, [2]D. Patnaik, [3]R. Choudhury, [4]S. S. Nayak
[1] Department of Physics, GPS, Gunupur, Odisha, India
[2] Department of Mathematics, GIET, Gunupur, Scholar CUTM, Paralakhemundi, Odisha,India
[3] Department of Physics, MM College, Berhampur, Scholar UTM, Paralakhemundi, Odisha,India
[4] CUTM, Paralakhemundi, Odisha, India

Abstract*— In this paper, a systolic mess architecture for the implementation of three-dimensional discrete cosine transform (3-D DCT) is presented. The computation of the 3-D DCT is carried out using the row-column-frame (RCF) approach, where 2-D DCT is computed first followed by the final 1-D DCT. Two systolic architectures, one using complex arithmetic and the other using real arithmetic are proposed for the computation of 2-D DCT. It is interesting to note that the proposed architectures do not require any hardware / time for transposition of the intermediate results. The proposed architecture for 3-D DCT provides high throughput of computation due to fully pipelined processing and massive parallelism employed. The proposed architecture may be used for computation of either the forward or inverse 3-D DCT. The number of multiplications can be reduced by exploiting the symmetries of the cosine function. The architectures are relatively easy to implement and highly suitable for VLSI implementation.

Keywords*— 3-D DCT, row-column-frame approach

I. INTRODUCTION

DISCRETE COSINE transform (DCT) [1] has emerged as the most popular substitute of the Karhunen-Loeve transform (KLT) in several speech and image signal processing applications [2]. Many fast algorithms and architectures have been proposed for implementation of 1-D and 2-D DCT [3, 4]. There has been rapid growth in the 3-D applications based on the 3-D DCT. So fast algorithms and efficient architectures for implementation of 3-D DCT have been developed [5, 6].

A new 3-D vector-radix decimation-in-frequency (VR DIF) algorithm for implementation of the 3-D DCT-II has been developed in [5]. It has a regular butterfly structure. Its hardware implementation is difficult.

3-D DCT is usually computed either using the row-column-frame (RCF) approach [7, 8] or through mapping it to 1-D and using other transforms. The RCF approach for the computation of 3-D DCT requires three stages of 1-D DCT computation and two stages to perform matrix and volume transpositions. Most of the 3-D DCT architectures proposed follow one of the two methodologies reported in [7] and [8]. The architecture of [7] uses three 1-D DCTs which accept the input data in a serial fashion. The outputs of 2-D DCT are fed into an $N^2 \times N$ memory, which is shuffled to allow the correct reading for the final N-Point 1-D DCT.

The transpose operation of the $N^2 \times N$ memory required prior to the third 1-D DCT cannot be performed in the conventional manner i.e., row-column transpose. This is due to the fact that this matrix is not square and that each element of the N-point data fed to the final 1-D DCT are collected every N^2 cycles. In [7], this memory is divided into N distinct $N \times N$ memories and a switching network to enable a fast and simple read / write system. The throughput rate of the architecture is very less. The architecture of [8] uses N 2-D DCT modules, one for each of the $N \times N$ block and a final 1-D DCT architecture. Although the architecture is fast, it requires two types of 1-D DCT architectures. The architecture of [9] uses two architectures proposed in [7] and [8]. It uses distributed arithmetic for computation of 1-D DCT. In [6] a fully parallel 3-D DCT / IDCT architecture without the RAM based matrix transposition is reported. The area-time complexity of the proposed architecture is less compared to existing architectures.

In this paper, two pairs of different linear systolic arrays for computing N-point DCT have been used. The linear arrays of each pair are complementary to each other in a sense that the output of one linear array may be fed as the input for the other linear array. This feature of the linear arrays has been utilized for designing a systolic architecture for computing 3-D DCT. It is interesting to note that the proposed structure for 3-D DCT does not require any hardware / time for the transposition of the intermediate results. The desired transposition is achieved by orthogonal alignment of the linear arrays. The proposed structure provides high throughput of computation due to fully pipelined processing, the massive parallelism employed in the architecture.

The rest of the paper is organized as follows. The 3-D DCT algorithm and two recursive algorithms for computation of 1-D DCT are presented in section II. The systolic architecture for implementation of 3-D DCT is discussed in section III. Hardware and throughput considerations of the architecture are presented in section IV. The conclusion is given in section V.

II. 3-D DCT ALGORITHM

The 3-D DCT, $X(k_1, k_2, k_3)$ of a 3-D spatial data sequence $\{x(n_1, n_2, n_3), n_1, n_2, n_3 = 0,1,2,\ldots,N\text{-}1\}$ is defined as

$$X(k_1,k_2,k_3) = \frac{8}{N^3}\varepsilon_{k1}\varepsilon_{k2}\varepsilon_{k3}\sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N-1}\sum_{n_3=0}^{N-1} x(n_1,n_2,n_3)\cos[\frac{\pi}{2N}(2n_1+1)k_1]\cos[\frac{\pi}{2N}(2n_2+1)k_2]\cos[\frac{\pi}{2N}(2n_3+1)k_3]$$

for $k_1 = k_2 = k_3 = 0, 1, 2, \ldots, N\text{-}1$. (1)

where $\varepsilon_{ki} = \begin{cases} \frac{1}{\sqrt{2}} \text{ for } k_i = 0 \\ 1 \text{ otherwise.} \end{cases}$ for i=1, 2 and 3.

The forward and inverse transforms are merely mappings from the spatial domain to the transform domain and vice versa. The 3-D DCT is a separable transform and as such, the row-column – frame decomposition can be used to evaluate equation 1. The scale factor $\frac{8}{N^3}\varepsilon_{k1}\varepsilon_{k2}\varepsilon_{k3}$is neglected.

The row transform can be expressed as

$$V(k_1,n_2,n_3) = \sum_{n_1=0}^{N-1} x(n_1,n_2,n_3)\cos[\frac{\pi}{2N}(2n_1+1)k_1] \quad (2)$$

for k_1, n_2, and $n_3 = 0, 1,2, \ldots, N\text{-}1$.

The column transform can be expressed as

$$W(k_1,k_2,n_3) = \sum_{n_2=0}^{N-1} V(k_1,n_2,n_3)\cos[\frac{\pi}{2N}(2n_2+1)k_2] \quad (3)$$

for k_1, k_2, and $n_3 = 0, 1, 2, \ldots, N\text{-}1$.

The frame transform can be expressed as

$$X(k_1,k_2,k_3) = \sum_{n_3=0}^{N-1} W(k_1,k_2,n_3)\cos[\frac{\pi}{2N}(2n_3+1)k_3] \quad (4)$$

for k_1, k_2, and $k_3 = 0, 1, 2, \ldots, N\text{-}1$.

In order to compute N × N × N– point DCT (where N is even), N row transforms, N column transforms, and N frame transforms are needed to be performed.

A. Recursive algorithm for 1-D DCT using complex arithmetic

T The DCT of a sequence $\{x(n), n = 0, 1, 2, \ldots, N\text{-}1\}$ may be given by

$$X(k) = \sum_{n=0}^{N-1} x(n)\cos[\frac{\pi(2n+1)k}{2N}] \quad (5)$$

for $k = 0, 1, 2, \ldots, N\text{-}1$.

Equation 5 may otherwise be expressed as

$$X(k) = \mathrm{Re}\left[\sum_{n=0}^{N-1} x(n)e^{\frac{j\pi(2n+1)k}{2N}}\right] \quad (6)$$

Equation 6 may be simplified to yield

$$X(k) = \mathrm{Re}[\beta_k\, X'(k)] \quad (7)$$

where

$$X'(k) = \sum_{n=0}^{N-1} x(n)e^{\frac{j\pi nk}{N}} \quad (8)$$

and $\beta_k = e^{\frac{j\pi k}{2N}}$ (9)

Using Horner's rule for polynomial estimation, equation 8 may be expressed by the recurrence relation

$X'(k) = (\ldots+(x(N\text{-}1)\alpha_k + x(N\text{-}2))\alpha_k + \ldots + x(1))\alpha_k + x(0)$ (10)

where $\alpha_k = e^{\frac{j\pi k}{N}}$ (11)

$X'(k)$ for $k = 0, 1, 2, \ldots, N\text{-}1$, given by equation 10 can be calculated by $(N\text{-}1)$ recursions in the processing elements (PEs), where each recursion consists of a complex addition followed by a complex multiplication. The desired DCT components may then be computed by multiplying each $X'(k)$ with β_k according to equation 7.

B. Proposed linear systolic arrays for computing the 1-D DCT

In this section two different linear systolic arrays for computing N– point DCT using recurrence relation (10) are proposed.

The structure of linear array -I is shown in Fig.1a. It consists of N locally connected identical PEs. Function of the $(k+1)$th PE is described in Fig.1b . The $(k+1)$th PE computes $X'(k)$ after $(N\text{-}1)$ identical recursions in $(N\text{-}1)$ successive time-steps and stores it in its accumulator. During the next time-step, the accumulator content of the $(k+1)$th PE is multiplied with β_k to yield the DCT component $X(k)$, and the accumulator content is then replaced by the first input element of the succeeding sequence, if any.

The structure of linear array-II is shown in Fig.2a.It consists of *N* locally connected identical PEs as in the case of linear array-I. The function of each PE is shown in Fig.2b. The first recursion for computing $X'(k)$ according to equation (10) is implemented in the first PE. The successive (*N*-2) recursions are implemented in the succeeding (*N*-2) PEs. The last PE adds *x*(0) with the computed output from its proceeding PE and multiplies with β_k to yield the *k*th DCT component. *N* numbers of output come out alternately through the terminals 1 and 2.

C. Recursive algorithm for 1-D DCT using real arithmetic

The DCT given by equation (5) can be computed as

$$X(k) = x(0)\cos\left(\frac{\pi k}{2N}\right) + \cos\left(\frac{3\pi k}{2N}\right)V_1 - \cos\left(\frac{\pi k}{2N}\right)V_2 \quad (12)$$

where

$$V_m = x(m) + 2\cos\left(\frac{\pi k}{N}\right)V_{m+1} - V_{m+2} \quad (13)$$

for $m = 1, 2, \ldots, N-1$

and $V_m = 0$ for $m \geq N$.

D. Proposed linear systolic arrays for computing the 1-D DCT

The structure of linear array-III is shown in Fig.3a. It consists of *N* locally connected PEs of which the first (*N*-1) PEs are identical. The recurrence relation given by (13) is implemented in the first (*N*-1) PEs. The terminal cell computes the DCT components. Function of each of the first (*N*-1) PEs is shown in Fig.3b. Function of the last PE is shown in Fig.3c. Function of the terminal cell is shown in Fig.3d and Fig.3e.One point of the input data is fed to each PE in the reverse order i.e., *i*th element is fed to the (*N*+1-*i*)th PE. The first output is obtained after 2*N*+1 time-steps and the rest (*N*-1) output are obtained in subsequent (*N*-1) time-steps. However, successive sets of *N*-point DCT are obtained in every *N* time-steps. The terminal cell of linear array-III has one output port (Fig.3d) but the terminal cell of linear array-IV has two output ports (Fig.3e). *N* number of output comes out alternately through the ports 1 and 2 of the terminal cell of linear array-IV.

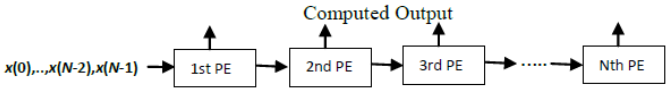


Fig.1a: Linear array-I for computing N-point DCT

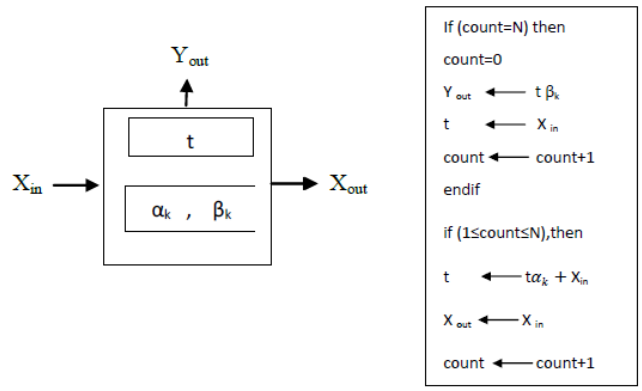


Fig.1b: Function of the (k+1)th PE of linear array-1

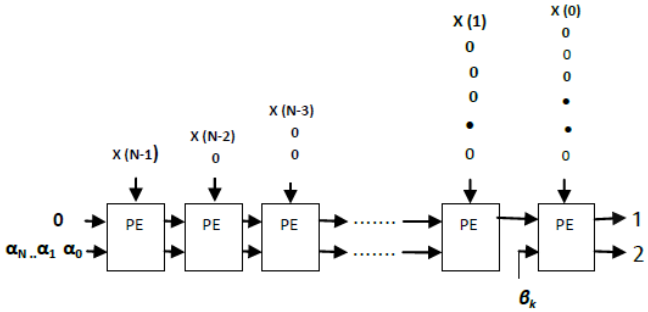


Fig.2a: Linear array-II for computing N-point DCT

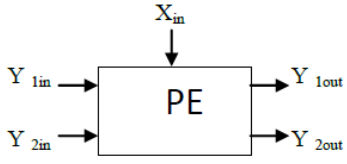


Fig.2b: Function of each PE of linear array-II.

For continuous processing $Y_{1out} \leftarrow (Y_{1in} + X_{in})Y_{2in}$. $Y_{2out} \leftarrow Y_{2in}$

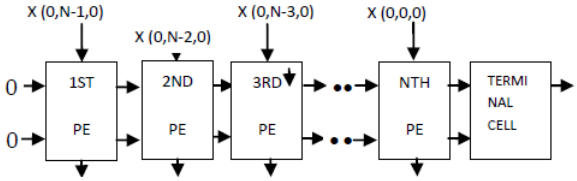
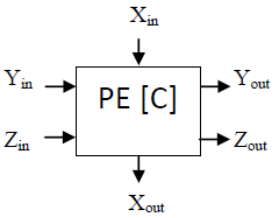


Fig.3a: Linear array-III and –IV



$Y_{out} = X_{in} + C.Y_{in} - Z_{in}$

$Z_{out} = Y_{in}$

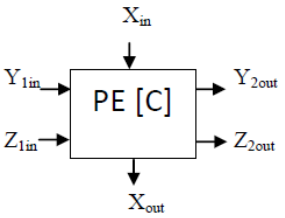
$X_{out} = X_{in}$

X_{in} = Input Sample

C=2 cos $(\frac{\pi k_1}{N})$ for PEs of linear array-III of the first block.

C=2 cos $(\frac{\pi k_2}{N})$ for PEs of linear array-IV of the second block.

Fig.3b: Function of each of the identical PEs.



$V_{1out} = (X_{in} - V_{2in}).C$

$X_{out} = X_{in}$

$V_{2out} = V_{1in}$

C= cos $(\frac{\pi k_1}{2N})$ for the last PEs of the linear arrays of the first block.

C= cos $(\frac{\pi k_2}{2N})$ for the last PEs of the linear arrays of the second block.

Fig.3c: Function of the last PE.

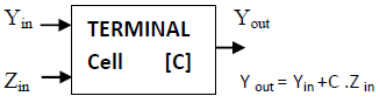
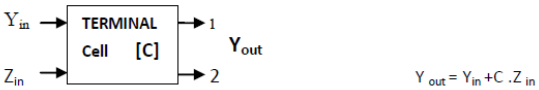


Fig.3d: Function of the terminal cell of the linear arrays of the first block



C = cos($\frac{3\pi k_1}{2N}$) for terminal cell of each linear array of the first block.C = cos($\frac{3\pi k_2}{2N}$) for terminal cell of each linear array of the second block.

Fig.3e: Function of the terminal cell of the linear arrays of the second block.

III. SYSTOLIC ARCHITECTURE FOR IMPLEMENTATION OF 3-D DCT

The proposed systolic architecture for implementation of 3-D DCT is shown in Fig.4a. It consists of two stages. The first stage is for computation of 2-D DCT and the second stage is for computation of the last 1-D DCT.

A. First stage (Using complex arithmetic)

It consists of two layers of systolic arrays placed one over the other. The lower layer consists of *N* number of linear array-I and the upper layer consists of *N* number of linear array-II. Each linear array-I consists of *N* number of locally connected identical PEs (Fig.1b) while each linear array-II consists of *N* number of locally connected identical PEs (Fig.2b). The lower layer makes the first stage of computation to provide the intermediate result [$V(k_1, n_2, n_3)$] to the upper layer. The upper layer makes the second stage of computation to yield the desired 2-D DCT components [$W(k_1, k_2, n_3)$] which come out of the last PE in the sets of *N* outputs alternately through ports 1 and 2. The elements of *n*th column of the input matrix are fed to the (*N*-*n*)th linear array-I in reverse order, staggered by one time-step with respect to the input of (*N*-*n*-1)th linear array-I. The linear arrays of the upper layer are placed orthogonally with respect to the linear arrays of the lower layer, such that each PE of the upper layer is placed over a PE of the lower layer. Each PE of the layer, therefore, receives its desired input in proper sequence of time and processes the rows of intermediate results to yield the desired transform components.

B. First stage (Using real arithmetic)

The proposed systolic architecture for implementation of 2-D DCT, shown in Fig.3f, consists of two blocks of identical linear arrays, shown in Fig.3a, for the two distinct stages of computation given in equations 3 and 2. The processing in the arrays of the two blocks is made to take place in mutually orthogonal directions. Both blocks consist of N number of

linear arrays. Each linear array consists of N number of locally connected PEs. At the end of each linear array there is a terminal cell shown in Fig.3d and Fig.3e. All the PEs of each linear array of both blocks are identical except the last PE. Function of each of the identical PEs of the linear arrays is shown in Fig.3b. The linear arrays of the first block make the first stage of computation to provide the intermediate result [V (k_1, n_2, n_3)] to the linear arrays of the second block. The linear arrays of the second block make the second stage of computation to yield the desired 2-D DCT components. The elements of the ith column of the 2-D input data are fed to the (N+1-i)th PE of the first linear array of the first block. Each PE of the linear arrays of the second block receives its desired input in proper sequence of time.

C. *Second stage*

The second stage of the proposed systolic architecture for computation of last 1-D DCT consists of *N*-number of linear array-V. Each linear array-V consists of *N* number of locally connected identical PEs shown in Fig.4b. Each PE of linear array-V is functionally similar to that of linear array-I except that each PE of linear array –V has two input and two output terminals and two accumulators t_1 and t_2 corresponding to the two output terminals of the last PE of each linear array-II and linear array-IV. Each output sequence of terminals 1 of last PE of linear array-II and linear array –IV is delayed by (*N*+1) time-steps. Alternate pairs of input data from terminals 1 and 2 of each PE of linear array –V are processed and routed to accumulators t_1 and t_2, respectively. The 3-D DCT components are obtained from the PEs of linear array-V.

IV. HARDWARE AND THROUGHPUT CONSIDERATIONS

Each of the linear arrays-I, - II and –V for computing *N*-point DCT require *N* number of identical PEs. In every computational cycle each PE of linear arrays-I and –II performs a complex addition followed by a complex multiplication and each PE of linear array-V performs two complex additions and two complex multiplications. The duration of each time-step is, therefore, the time required for computing a complex addition and a complex multiplication by the PEs. The actual duration of the time-step will, however, depend on the hardware used in the PEs for implementing the complex additions and multiplications. In the linear array-I, the first output is obtained after (*N*+1) time-steps and the rest (*N*-1) output are obtained in subsequent (*N*-1) time-steps. In linear array-II, the first output is obtained after *N* time-steps and the rest (*N*-1) output are obtained in subsequent (*N*-1) time-steps. However, successive sets of *N*-point DCTs are obtained from both the linear arrays-I and-II in every *N* time-steps. However, 2*N* outputs are obtained from linear array-V in every *N* time-steps. The first output of the lower layer is obtained after (*N*+1) time-steps. The first output of the upper layer is obtained after (2*N*+1) time-steps. The first set of 2-D DCT component is obtained in (4*N*-1) time-steps. However, successive sets of 2D-DCTs may be obtained in every *N* time-steps. The throughput rate of the proposed 2-D DCT structure would, therefore, be $R = (N / T)$ where *T* is the duration of a time-step, given by $T = T_m + T_a$. T_m and T_a are, respectively, the time required for performing a complex multiplication and a complex addition in the PEs. The first two 3-D DCT components are obtained from the first PE of linear array-V after (4*N* +3) time-steps. The complete set of 3-D DCT components is obtained in 7*N* time-steps. However, successive sets of 3-D DCT may be obtained in every *N* time-steps. The throughput rate of the proposed 3-D DCT structure is $R = (N / T)$. Numbers of multipliers and adders required by the proposed architecture are equal to 4(4*N* +1) and 16(*N* +1), respectively. The area complexity (A), computation time (τ) and VLSI performance measure ($A\tau^2$) are $2N^2$, 3*N* and $18N^4$, respectively.

The proposed systolic architecture for implementation of 2-D DCT (Fig.3f) consists of two blocks of identical linear arrays for two distinct stages of computation. The processing in the arrays of two blocks takes place in mutually orthogonal directions. Both the blocks consist of *N* number of linear arrays. Each linear array of both blocks consists of *N* number of locally connected PEs. All the PEs of each linear array of both blocks are identical except the last PE. Each PE and each terminal cell of the linear arrays require one accumulator storage. Each of the identical PEs consists of one multiplier and one adder for performing one multiplication and two additions in every computational cycle. The last PE and the terminal cells of the arrays require one multiplier and one adder to perform one multiplication and one addition in every computational cycle. Apart from this, each PE of the linear arrays requires two latches for data transfer. The first output of the first layer of the first block is obtained after (*N* +1) time-steps. The first output of the first layer of the second block is obtained after 2(*N* +1) time-steps. All the *N* sets of *N*–point DCTs are obtained in 4*N* time-steps. However, successive sets of DCTs may be obtained in every *N* time-steps. The throughput rate of the proposed 2-D architecture would, therefore, be $R = (N / T)$ where *T* is the duration of a time-step, given by $T = T_m + T_a$.

Table I comparison of hardware requirements of the proposed architectures with the architecture of [5].

Features	ARCHITECTURE OF [5]	Proposed architecture using complex arithmetic	Proposed architecture using real arithmetic
Number of Multipliers	$\frac{7}{8}\log_2 N$	$4N^2(3N+1)$	$3N(N+1)$
Number of Adders	$\frac{9}{2}N^3\log_2 N - 3N^3 + 3N^2$	$16(N+1)$	$2N(5N+4)$

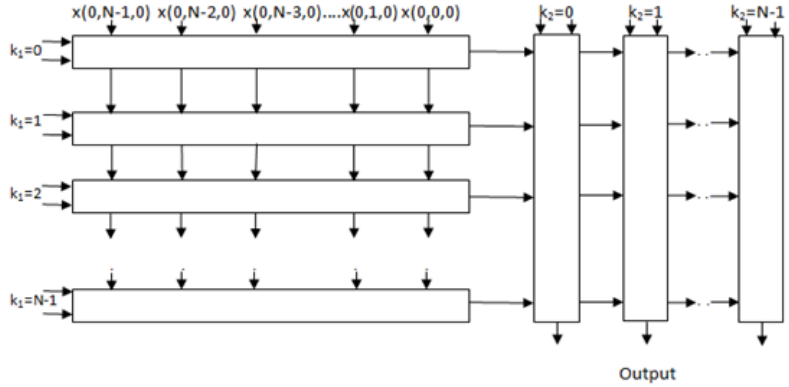


Fig.3f: Systolic architecture for implementation of 2-D DCT.

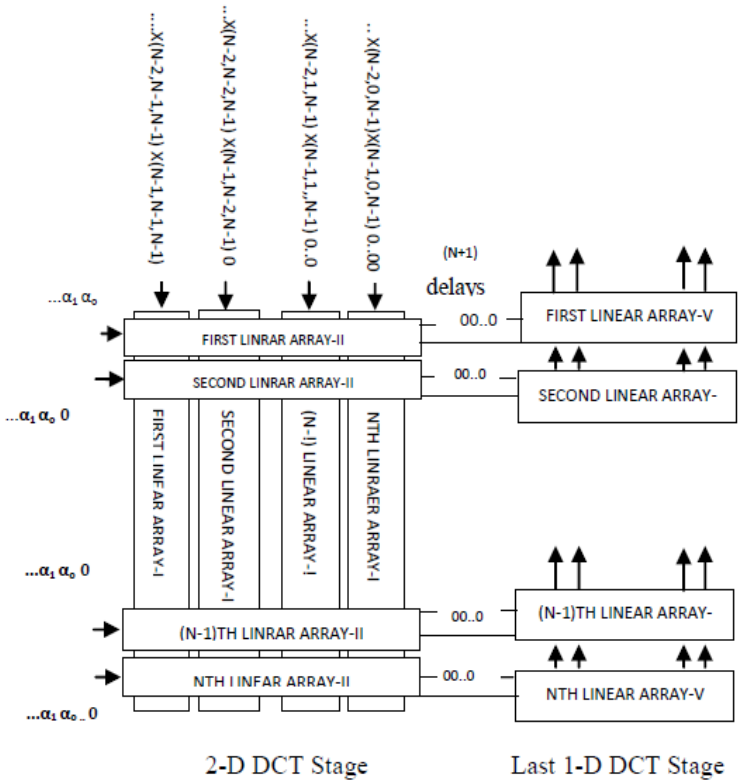


Fig 4a: Systolic architecture for implementation of 3-D DCT

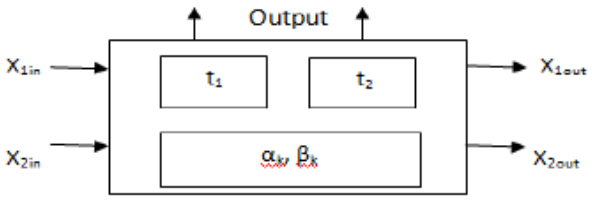


Fig.4b: Function of each PE of linear array-V.

V. CONCLUSION

I In this paper, a systolic architecture for computation of 3-D DCT and two systolic architectures for computation of 2-D DCT are proposed. These architectures do not require any extra hardware / time for the transposition of the intermediate output. The proposed architecture provides high throughput of computation due to fully pipelined processing and massive parallelism employed. The architectures are relatively easy to implement and highly suitable for VLSI implementation. The proposed architectures have been implemented using C programming using P-4 processor with speed of 1000 M Hz and 256 MB RAM.

REFERENCE

[1] N.Ahmed, T.Natarajan, and K.R.Rao," Discrete cosine transform, "*IEEE Trans. on Computers*, vol.C-23, pp.90-93, Jan.1974.

[2] R.J.Clark, "Relation between the Kahunen-Loeve and cosine transform,"*Proc.IEE*, vol.128, pp.359-360, Nov. 1981.

[3] S.C.Chan and K.L.Ho,"Direct methods for computing discrete sinusoidal transforms," *in Proc. Inst. Elect. Eng. Radar Signal Process.*, vol.137, pp.433-442, Dec.1990.

[4] H.S.Hou,"A first recursive algorithm for computing the discrete cosine transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol.ASSP-35, pp.1455-1461, Oct.1987.

[5] S. Boussakta and H. O. Alshibami, "Fast algorithm for the 3-D DCT –II,"*IEEE Trans. on Signal Processing*, vol.52, No.4, pp.992-1001 , April 2004

[6] A. Aggoun,"Three dimensional DCT / IDCT architecture," *International Journal of Advances in Engineering and Technology*, vol.6, No.2, pp.648-658, May 2013.

[7] I. Jollah, A. Aggoun, and M. Mc Cormick, "A 3-D DCT architecture for compression of integral 3-D images," *Proceedings of IEEE Workshop on Signal Processing Systems*, pp.238-244, 2000.

[8] I. Jollah, and A. Aggoun," A parallel 3-D DCT architecture for the compression of integral 3-D images," *Proceedings of the 8th IEEE International Conference on Electronics, Circuits and Systems*, pp.229-232, 2001.

[9] S.Saponra, L. Fanucci, and P. Terreni, "Low-power VLSI architectures for 3-D discrete cosine transform (DCT)," *Proceedings of the 46th IEEE International Midwest Symposium on Circuits and Systems*, 2003, vol.3, pp.1567-1570, Dec.2003.

