A CORDIC Based Programmable DXT Processor Array

V. K. Anuradha

Electrical Communication Engg. Dept.
Indian Institute of Science
Bangalore 560012, India

V. Visvanathan

Computer-Aided Design Laboratory
Supercomputer Education and Research Centre
Indian Institute of Science
Bangalore 560012, India

Abstract

A CORDIC based processor array which can be programmed by switch settings to compute the Discrete Hartley, Cosine or Sine Transforms or their inverses is described. Through a novel formulation of the transform computations in the CORDIC framework, N-point transforms are mapped on to a linear array of $\lfloor \frac{N}{2} \rfloor + 1$ CORDIC processors with minimal control overhead to incorporate the programmability.

1 Introduction

Discrete orthogonal transforms are computation intensive. For real time applications it is necessary to develop special purpose architectures consisting of local interconnections incorporating both parallel processing and pipelining. Discrete transforms like DCT, DHT and DST are widely used in many applications. Though separate architectures exist for each of these transforms, a programmable architecture across these transforms is yet to be proposed. CORDIC based arrays have been reported for DFT and DHT [1]. But CORDIC based DCT/IDCT and DST/IDST discussed here is a novel idea. CORDIC based arrays are attractive for orthogonal transforms since they have less roundoff error when compared to multiplication and accumulation based arrays [2].

We propose a CORDIC based architecture that is programmable across the orthogonal transforms DCT/IDCT, DHT/IDHT and DST/IDST. We refer to these transforms as DXT. It is a linear processor array with $\lfloor N/2 \rfloor + 1$ CORDIC processors (where N is the size of the transform) with local interconnections. The CORDIC based Programmable DXT (PDXT) has minimal control overhead circuitry to bring about programmability.

The rest of the paper is organized as follows. In Section 2, we begin with a discussion on CORDIC arithmetic. In Section 3, the computations of DXT are put in the CORDIC framework and the concept of CORDIC based DXT is introduced. Programmable DXT is presented in Section 4. In Section 5, we summarize our work and discuss future scope of work.

This work was supported in part by the Dept. of Electronics, Govt. of India under the "LSI/VLSI Design Centre" project.
V. K. Anuradha is presently at Motorola India Electronics Pvt. Ltd., Bangalore.

2 CORDIC Arithmetic

2.1 Principle of CORDIC

In the CORDIC computation [1], a vector $[x, y]^t$ is rotated over an angle $(+\theta)$ to a new vector $[x', y']^t$, with simple shift and add operations. The basic concept of the CORDIC computation is to decompose the desired rotation angle θ into the sum/difference of a set of predefined elementary rotation angles θ_i's such that the rotation through each of them can be accomplished with shift and add. In order to achieve n bits of accuracy, the coordinate rotation angle θ is split into a sequence of n sections [1]:

$$\theta = \pm\theta_0 \pm\theta_1 \cdots \pm\theta_{n-1} = \sum_{i=0}^{n-1} \xi_i \theta_i \qquad (1)$$

Where $\theta_i > 0$, $cos\theta_i \neq 0$ and $\xi_i = \pm 1$ depending on the direction of rotation. Though CORDIC arithmetic can be operated in different coordinate systems [1], we shall confine ourselves to the circular coordinate system in the context of PDXT. The CORDIC computation is given below.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \prod_{i=0}^{n-1} \begin{bmatrix} cos\xi_i\theta_i & -sin\xi_i\theta_i \\ sin\xi_i\theta_i & cos\xi_i\theta_i \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$= K \prod_{i=0}^{n-1} \begin{bmatrix} 1 & -\xi_i 2^{-S(i)} \\ \xi_i 2^{-S(i)} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \qquad (2)$$

where $K = \prod_{i=0}^{n-1} cos\theta_i$. The θ_i's are chosen such that $tan(\theta_i) = 2^{-S(i)}$ where $S(i)$ is the shift factor. The shift factors are given in [3]. Equation(2) can be summarized as n iterative equations:

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} 1 & -\xi_i 2^{-S(i)} \\ \xi_i 2^{-S(i)} & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \qquad (3)$$

The output vector in CORDIC rotation is to be multiplied by a constant K as this is not being taken into

0-8186-4990-9/94 $3.00 © 1994 IEEE

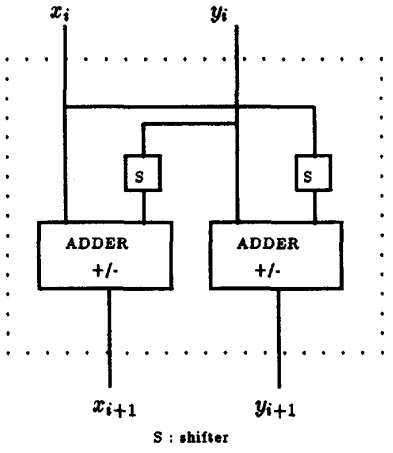

Figure 1: CORDIC Micro-rotation Module (CMM)

account in eq(3). Various techniques have been proposed to reduce this computation overhead [1].

The symmetry properties of transform kernels of the DXT are exploited to implement DXT efficiently in the CORDIC framework. Given the transform size N of DXT, corresponding rotation angles θ's are known. We can therefore evaluate the corresponding unique ξ_i's that satisfy eq(1). Only ξ_i's are going to vary across the DXT. They can be stored in the CORDIC module to realize the transform of interest, which is computed by the iterative equations (3). This forms the basic principle of Programmable DXT.

2.2 CORDIC Architectures

We can describe the CORDIC iterations as a sequence of micro-rotation operations [4] where the computation described by eq(3) is called a micro-rotation. CORDIC Micro-rotation Module (CMM) is shown in Figure 1. One macro-rotation requires n micro-rotations where a macro-rotation is the computation of eq(2) excluding the multiplicative constant K.

The ξ_i's corresponding to the rotation angle θ and the shift factors are stored in the CMM. If the clock period (T_{clock}) is matched to the time taken for each micro-rotation then the total time taken for a macro-rotation (T_{macro}) = $n\,T_{clock}$.

3 CORDIC Based DXT

In this section we rewrite the computations of each of the DXTs such that they can be mapped efficiently onto a CORDIC processor array. Later a unified CORDIC processor array is derived which can be programmed to obtain any one of the DXTs.

3.1 CORDIC Based DHT/IDHT

CORDIC based solution for DHT has been proposed in [5]. In this section we improve upon this and increase the CORDIC processor utilization. This results in the reduction of the number of CORDIC processors required by 50%.

Discrete Hartley transform: The Discrete Hartley N-point transform given by $H(r)$ is defined [5] as follows:

$$\begin{aligned} X(r) &= \sum_{k=0}^{N-1} x(k)[C(r,k)+S(r,k)] \qquad (4)\\ & \quad r = 0,1,\ldots,N-1.\\ H(r) &= (1/N)\,X(r) \end{aligned}$$

Here, $C(r,k) = cos(2\pi rk/N)$ and $S(r,k) = sin(2\pi rk/N)$. From now onward, we work with $X(r)$ only. The multiplicative constant $(1/N)$ can be subsumed with the CORDIC post multiplicative constant K. Equation(4) can be written as follows:

$$\begin{bmatrix} X(r) \\ - \end{bmatrix} = \sum_{k=0}^{N-1} \begin{bmatrix} C(r,k) & S(r,k) \\ -S(r,k) & C(r,k) \end{bmatrix} \begin{bmatrix} x(k) \\ x(k) \end{bmatrix} \qquad (5)$$

$r = 0,1,\ldots,N-1$. Chang and Lee [5] have mapped this computation on to a linear processor array using N CORDIC processing elements. We can see from the above that the CORDIC computation is not being used efficiently. We propose the following rearrangement to improve the efficiency. Consider the following:

$$\begin{aligned} X(N-r) &= \sum_{k=0}^{N-1} x(k)[cos(2\pi(N-r)k/N)\\ &\quad +sin(2\pi(N-r)k/N)] \qquad (6)\\ &= \sum_{k=0}^{N-1} x(k)\,[C(r,k)-S(r,k)] \qquad (7)\\ &\quad r = 0,1,\ldots,N-1 \end{aligned}$$

Then, we can write the expression for DHT as follows:

$$\begin{bmatrix} X(r) \\ X(N-r) \end{bmatrix} = \sum_{k=0}^{N-1} \begin{bmatrix} C(r,k) & S(r,k) \\ -S(r,k) & C(r,k) \end{bmatrix} \begin{bmatrix} x(k) \\ x(k) \end{bmatrix} \qquad (8)$$

where $r = 0,1,\ldots,N/2$. Comparing eq(5) with eq(8) we can observe that two transform points are computed by each macro-rotation. This results in a 100% gain in the utilization of the CORDIC computation when compared to the implementation in [5].

Inverse Discrete Hartley Transform: IDHT is defined as

$$x(k) = \sum_{r=0}^{N-1} H(r)\,[C(r,k)+S(r,k)] \qquad (9)$$

Here, $k = 0,1,\ldots,N-1$ Except for the multiplicative constant $(1/N)$, the expression for IDHT is same as

the DHT, hence the same algebra holds. Therefore, eq(9) can be written as follows:

$$\begin{bmatrix} x(k) \\ x(N-k) \end{bmatrix} =$$

$$\sum_{r=0}^{N-1} \begin{bmatrix} C(r,k) & S(r,k) \\ -S(r,k) & C(r,k) \end{bmatrix} \begin{bmatrix} H(r) \\ H(r) \end{bmatrix} \quad (10)$$

where $k = 0, 1, \ldots, N/2$.

3.2 CORDIC Based DCT/IDCT

So far many VLSI implementations of DCT have been reported [6][7] using techniques other than CORDIC arithmetic. In this section we rearrange the DCT/IDCT computation to be able to realize it efficiently using CORDIC macro-rotation operation.

Discrete Cosine Transform: The N-point DCT given by $G(r)$ is defined [6] as follows:

$$X(r) = \sum_{k=0}^{N-1} x(k) cos(\pi(2k+1)r/2N) \quad (11)$$
$$r = 0, 1, \ldots, N-1$$
$$G(r) = c(r)\,X(r) \quad (12)$$

c(r)=$1/\sqrt{2}$ for r=0 and c(r)=1 otherwise. We exploit the symmetry property of $X(r)$ and $X(N-r)$ to be able to use the CORDIC unit effectively for DCT computation.

$$X(N-r) = \sum_{k=0}^{N-1} x(k)\, cos(\pi(2k+1)(N-r)/2N)$$
$$= \sum_{k=0}^{N-1} x(k)(-1)^k\, sin(\pi(2k+1)r/2N) \quad (13)$$

From eq(11) and eq(13) we can write the following:

$$\begin{bmatrix} X(r) \\ X(N-r) \end{bmatrix} =$$

$$\sum_{k=0}^{N-1} \begin{bmatrix} C'(r,k) & (-1)^{k+1}S'(r,k) \\ (-1)^k S'(r,k) & C'(r,k) \end{bmatrix} \begin{bmatrix} x(k) \\ 0 \end{bmatrix} \quad (14)$$

Here, $r = 0, 1, \ldots, N/2$. $C'(r,k) = cos(\pi(2k+1)r/2N)$ and $S'(r,k) = sin(\pi(2k+1)r/2N)$. Note that for alternate k, the data vector $[x(k), 0]^t$ is rotated either by $(2k+1)\pi r/2N$ or $-(2k+1)\pi r/2N$.

Inverse Discrete Cosine Transform: The IDCT is defined [6] as follows:

$$x(k) = \sum_{r=0}^{N-1} c(r)\, G(r) cos(\pi(2k+1)r/2N) \quad (15)$$

$k = 0, 1, \ldots, N-1$. $c(r) = 1/\sqrt{2}$ if $r = 0$ and $c(r) = 1$ otherwise. Unlike the DCT, in IDCT, $c(r)$ manifests as a pre-multiplication, hence it has to be dealt with separately. This is taken care of in the CORDIC based IDCT by absorbing it as a CORDIC macro-rotation. The symmetry between $x(k)$ and $x(N-k-1)$ is used to express the computation of IDCT in the CORDIC form effectively.

$$x(N-k-1) =$$

$$\sum_{r=0}^{N-1} c(r)(-1)^r G(r) cos(\pi(2k+1)r/2N) \quad (16)$$

Since rotation by 45° corresponds to multiplication by $\frac{1}{\sqrt{2}}$, we can write eq(15) and eq(16) as

$$\begin{bmatrix} x(k) \\ - \end{bmatrix} = \begin{bmatrix} cos(45°) & sin(45°) \\ -sin(45°) & cos(45°) \end{bmatrix} \begin{bmatrix} G(0) \\ 0 \end{bmatrix} +$$

$$\sum_{r=1}^{N-1} \begin{bmatrix} C'(r,k) & S'(r,k) \\ -S'(r,k) & C'(r,k) \end{bmatrix} \begin{bmatrix} G(r) \\ 0 \end{bmatrix} \quad (17)$$

$$\begin{bmatrix} x(N-k-1) \\ - \end{bmatrix} =$$

$$\begin{bmatrix} cos(45°) & sin(45°) \\ -sin(45°) & cos(45°) \end{bmatrix} \begin{bmatrix} G(0) \\ 0 \end{bmatrix} +$$

$$\sum_{r=1}^{N-1} (-1)^r \begin{bmatrix} C'(r,k) & S'(r,k) \\ -S'(r,k) & C'(r,k) \end{bmatrix} \begin{bmatrix} G(r) \\ 0 \end{bmatrix} \quad (18)$$

As can be observed from the above two equations, it is enough to compute $x(k)$ for $k = 0, 1, \ldots, (N/2-1)$. $x(N-k-1)$ can be computed from that.

After rotating the vector $[G(r), 0]^t$ by an angle $(2k+1)\pi r/2N$ we accumulate the result over $r = 0, 1, \ldots, N-1$ to obtain the corresponding $x(k)$. After each rotation the result is either added or subtracted depending on whether r is even or odd to obtain $x(N-k-1)$.

3.3 CORDIC Based DST/IDST

The technique of rearranging the computation to obtain CORDIC based IDCT can be easily extended to DST/IDST.

Discrete Sine Transform: The N-point discrete sine transform given by $G(r)$ is defined [6] as follows:

$$X(r) = \sum_{k=1}^{N} x(k) sin(\pi kr/N + 1) \quad (19)$$
$$r = 1, \ldots, N$$
$$G(r) = c(r)\,X(r) \quad (20)$$

c(r) = $\sqrt{2/(N+1)}$. Note that for DST indexes are defined from 1 to N instead of 0 to $N-1$. In DST, the relationship between $X(r)$ and $X(N+1-r)$ is used

to rewrite the computation to put it in the CORDIC framework as follows.

$$\begin{bmatrix} X(r) \\ - \end{bmatrix} =$$

$$\sum_{k=1}^{N} \begin{bmatrix} C''(r,k) & S''(r,k) \\ -S''(r,k) & C''(r,k) \end{bmatrix} \begin{bmatrix} 0 \\ x(k) \end{bmatrix} \quad (21)$$

$$\begin{bmatrix} X(N+1-r) \\ - \end{bmatrix} =$$

$$\sum_{k=1}^{N} (-1)^{k+1} \begin{bmatrix} C''(r,k) & S''(r,k) \\ -S''(r,k) & C''(r,k) \end{bmatrix} \begin{bmatrix} 0 \\ x(k) \end{bmatrix} \quad (22)$$

for $r = 1, \ldots, N/2$.
Here, $C''(r,k) = cos(\pi kr/N + 1)$ and $S''(r,k) = sin(\pi kr/N + 1)$. From eq(21) and eq(22) we can conclude that, it is sufficient to evaluate $X(r)$ for $r = 1, 2, \ldots, N/2$. $X(N+1-r)$ can be computed from this by appropriately adding/subtracting the resulting macro-rotation depending on whether k is odd or even.

Inverse Discrete Sine Transform: IDST given by $x'(k)$ is defined [6] as

$$x(k) = \sum_{r=1}^{N} G(r) sin(\pi kr/N + 1) \quad (23)$$
$$x'(k) = (\sqrt{2/(N+1)})x(k)$$

It has the same form as DST. Hence the algebra developed for DST is applicable for IDST also. The expression for CORDIC based IDST is given below.

$$\begin{bmatrix} x(k) \\ - \end{bmatrix} =$$

$$\sum_{r=1}^{N} \begin{bmatrix} C''(r,k) & S''(r,k) \\ -S''(r,k) & C''(r,k) \end{bmatrix} \begin{bmatrix} 0 \\ G(r) \end{bmatrix} \quad (24)$$

$$\begin{bmatrix} x(N+1-k) \\ - \end{bmatrix} =$$

$$\sum_{r=1}^{N} (-1)^{r+1} \begin{bmatrix} C''(r,k) & S''(r,k) \\ -S''(r,k) & C''(r,k) \end{bmatrix} \begin{bmatrix} 0 \\ G(r) \end{bmatrix} \quad (25)$$

for $k = 1, \ldots, N/2$.

In this section we have rearranged the computation of each of the DXTs to fit it in the CORDIC framework efficiently. From eq(8), eq(10), eq(14), eq(17), eq(18), eq(21), eq(22), eq(24), and eq(25) we can see that the CORDIC macro-rotation is the basic computation in all of them. This paves way for a unified CORDIC based DXT.

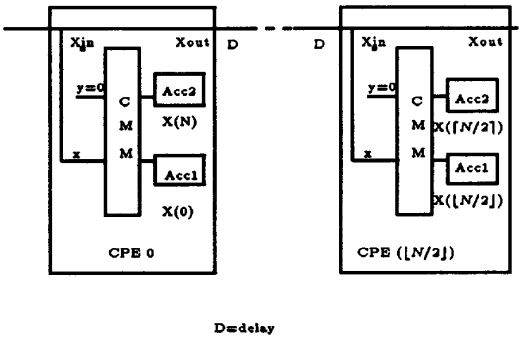

Figure 2: CORDIC Based Systolic Array for DCT

4 Programmable DXT

In this section we derive the systolic processor array for DCT based on eq(14), and for IDCT based on eq(17) and eq(18). Note that an efficient CORDIC based architecture for DCT or IDCT has not been reported so far. CORDIC based systolic arrays for the other DXTs are also derived similarly. The concept of Programmable DXT (PDXT) is introduced based on this.

4.1 DCT Array

Following a straight forward application of the canonical mapping methodology discussed in [8], the computation of eq(14) is mapped on to the systolic processor array shown in Figure 2. We call the basic computational module, CORDIC Processing Element (CPE). Each CPE is responsible for computing $X(r)$ and $X(N-r)$. The CPE consists of a CMM and two accumulators Acc1 & Acc2. After each macro-rotation, the result is accumulated in Acc1 and Acc2. After the N^{th} macro-rotation Acc1 and Acc2 contain $X(r)$ and $X(N-r)$ respectively. For an N-point transform we need to compute $X(0), X(1), \ldots, X(N-1)$. Depending on whether N is even or odd we have the following two cases.

For N even: With $(N/2+1)$ CPEs we obtain all the transform points. $X(0)$ and $X(N)$ are available from the 0^{th} CPE. But $X(N)$ is not required. We get $X(N/2)$ at both the outputs of the $(N/2)^{th}$CPE. Any one of them can be used.

For N odd: Here there will be $(\lfloor N/2 \rfloor + 1)$ CPEs. $X(0)$ and $X(N)$ are available from the 0^{th} CPE. $X(N)$ is not required. But both the outputs of the $(\lfloor N/2 \rfloor)^{th}$ CPE are used.

Combining both odd and even cases, we can say that, there will in general be $(\lfloor N/2 \rfloor + 1)$ CPEs in the CORDIC based systolic array for DCT.

Each processor computes $X(r)$ and $X(N-r)$. The output won't be in the sequential order if we take them out using a single link. So two serial links are used for this purpose. The interconnection between the neigh-

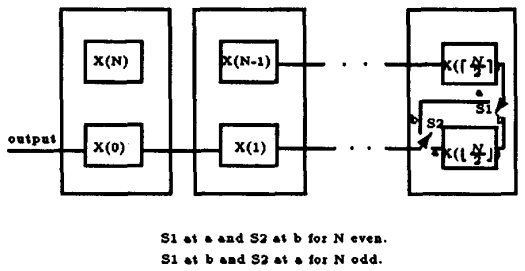

Figure 3: Output phase for DCT

boring processors in the output phase is depicted in Figure 3 for N even and odd cases. Note that the output phase takes N clock cycles.

Let the basic clock cycle (T_{clock}) be matched to the time taken for one micro-rotation in a CPE. $T_{clock} \approx T_{add} + T_{shift}$ where T_{add} = time taken for addition; T_{shift} = time taken for shifting. Having two additional registers per CPE, the output phase can be overlapped with the computation phase. With this Latency is $(Nn + \lfloor N/2 \rfloor + N)T_{clock}$ and throughput is $1/(nN)T_{clock}$.

4.2 IDCT Array

From eq(17) and eq(18), using the same mapping methodology, we can obtain the systolic processor array for IDCT (Figure 4). Here the vector $[G(r), 0]^t$ is rotated by an angle $(2k+1)\pi r/2N$ and the result is accumulated over $r = 0, 1, \ldots, N-1$ in Acc1 to obtain x(k). Acc2, on the other hand, takes the same input as Acc1 and alternately adds it to or subtracts it from the accumulated value (depending on whether r is even or odd) in order to obtain $x(N-k-1)$. Depending on N being even or odd we have the following two cases.

For N even: We need $(N/2)$ processors to compute all the transform points. All the outputs from all the CPEs are used (Figure 4.a).

For N odd: Here $\lfloor N/2 \rfloor + 1$ CPEs are used (Figure 4.b). We obtain x($\lfloor N/2 \rfloor$) at both the outputs of the $(\lfloor N/2 \rfloor + 1)^{th}$ CPE. Any one of them can be used.

4.3 DHT, IDHT, DST and IDST Arrays

Comparing eq(14) and eq(8) we can observe that, the DHT array will be similar to the DCT array (Figure 2). Here, ξ_i's corresponding to $2\pi kr/N$ have to be programmed in the CPE. At the input we have $[x(i), x(i)]^t$ instead of $[x(i), 0]^t$ as is the case in DCT. CORDIC based IDHT is similar to DHT/DCT array.

Equation(21) and Equation(22) are used to derive the systolic array for DST. By noting the similarity of these equations with eq(17) and eq(18) we can conclude that, the CORDIC based systolic array for DST will resemble the IDCT array. The ξ_i's correspond to $\pi kr/N + 1$. Input vector for the CPE is $[0, x(k)]^t$. IDST will be similar to the DST/IDCT array.

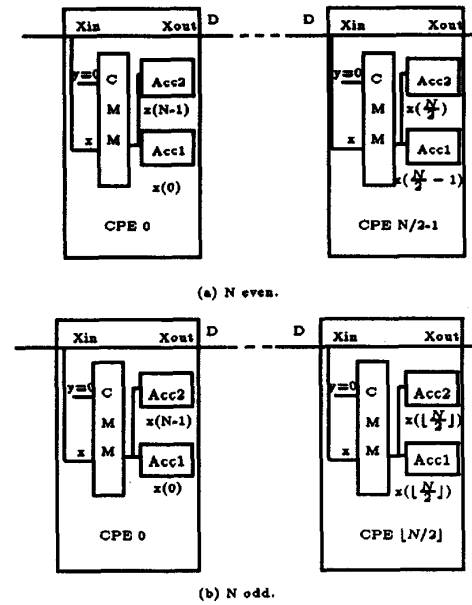

Figure 4: CORDIC Based Systolic Array for IDCT for (a)N even (b)N odd

4.4 The PDXT

We observed that the DXTs can be mapped on to a CORDIC linear systolic array with minimal variation across the transforms. Those variations that remain, and the appropriate control circuitry, which is to be programmed to realize any one of the DXTs is given below.

- The θ's, and hence the ξ_i's are going to vary across the DXTs. Since the θ's are known, the corresponding ξ_i's are evaluated off line and stored in the CMM.
- The interconnection of the registers containing the transform values in the output phase is going to be different for the DXTs. This is taken care of, by providing switches which can be programmed.
- The accumulator Acc1 will add and accumulate the result of the macro-rotations. Acc2 differs from this slightly as it has to either be a straight forward adder for some DXTs (DHT, IDHT and DCT), or alternately add or subtract for other DXTs (IDCT, DST and IDST). This can be brought forth by having a flip-flop to control this functioning.

With this, the concept of PDXT is substantiated and the final PDXT along with the switches required for the programmability is shown in Figure 5. Depending on the position of these switches, the DXT of interest

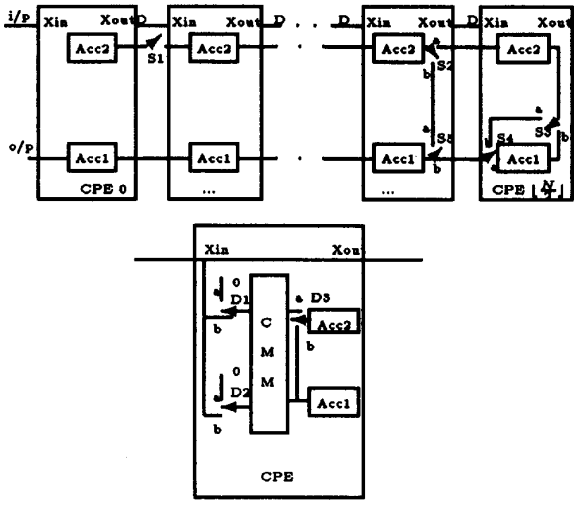

Figure 5: PDXT and the details of the CPE

DXT	N	S1	S2	S3	S4	S5	D1	D2	D3
DCT	even	off	a	a	b	b	a	b	a
	odd	off	a	b	a	b	a	b	a
IDCT	even	on	b	-	-	a	a	b	b
	odd	on	a	a	b	b	a	b	b
DHT	even	off	a	a	b	b	b	b	a
	odd	off	a	b	a	b	b	b	a
IDHT	even	off	a	a	b	b	b	b	a
	odd	off	a	b	a	b	b	b	a
DST	even	on	b	-	-	a	b	a	b
	odd	on	a	a	b	b	b	a	b
IDST	even	on	b	-	-	a	b	a	b
	odd	on	a	a	b	b	b	a	b

Table 1: Various switch positions for PDXT

can be obtained. The switch positions for the various DXTs are listed in Table 1. The throughput and latency of the PDXT is the same as that derived for the DCT in Section 4.1 Note that the PDXT array has to have a post multiplicative unit to take care of the CORDIC constant K and the transform constant $c(r)$.

The basic θ_i's and the shift factors are available in [3]. They are used to determine the ξ_i's (which take on values +1 or -1). The number n of θ_i's chosen depends on the accuracy requirements. The operation of the PDXT has been verified through a software simulation of finite precision CORDIC arithmetic for all the DXTs with various values of N and n. The results compare favorably with the input-output behavior of the "exact" (double precision) DXTs.

5 Summary & Future work

In sum, we have rearranged the computations of the Discrete Hartley, Cosine and Sine Transforms and their inverses to be able to realize each of them using CORDIC arithmetic efficiently. We refer to these transforms as DXT. We derived the CORDIC based systolic processor arrays for the DXTs. For an N point DXT, we need a maximum of $(\lfloor \frac{N}{2} \rfloor + 1)$ CORDIC processors to implement any of the DXTs. Based on this we introduced the concept of programmable DXT. With minimal control circuitry, a $(\lfloor \frac{N}{2} \rfloor + 1)$ CORDIC processor array can be used to implement any of the N point DXTs. We have discussed the programmability issues and the required control circuitry in detail.

Each macro-rotation operation in the CORDIC computation is achieved by n micro-rotation operations mapped onto the same CORDIC Micro-rotation Module (CMM). We can obtain a highly pipelined implementation of the CORDIC module by having n different stages to do the n micro-rotations [4]. We can trade-off area for throughout in this implementation using the pipelined clustering technique developed in [9]. We are presently working out the details of this. The possibility of an efficient implementation of the post multiplicative module required for the PDXT is under investigation.

References

[1] Y.H. Hu, "CORDIC algorithm and CORDIC based processor array," *IEEE Signal Processing Magazine*, pp. 17-35, July 1992.

[2] L.W. Chang, "Roundoff Error Problem of the Systolic Array for DFT," *IEEE Trans. Signal Processing*, vol. 41, no. 1, pp. 395-398, Jan 1993.

[3] X.H. Hu, "Expanding the range of convergence of the CORDIC algorithm," *IEEE Trans. Computers*, vol. 40, pp. 13-21, Jan. 1991.

[4] A.A.J.De lange, "Design and implementation of highly parallel and pipelined VLSI systems," *M. S. Thesis*, Delft University of Technology, 1991.

[5] L.W. Chang and S.W. Lee, "Systolic Arrays for Discrete Hartley Transform," *IEEE Trans. Signal Processing*, vol. 39, no. 11, Nov. 1991.

[6] L.W. Chang et al, "A Unified systolic Array for Discrete Cosine and Sine Transforms," *IEEE Trans. Signal Processing*, vol. 39, pp. 192-194, Jan. 1991.

[7] Chaitali Chakrabarti, "VLSI Architectures for Real time Signal Processing," Ph. D. Dissertation, University of Maryland, 1990.

[8] S.Y.Kung, "VLSI Array Processors," Prentice Hall, Englewood Cliffs, New Jersey.

[9] V. Visvanathan, N. Mohanty and S. Ramanathan, "An Area Efficient Systolic Architecture for Real Time VLSI Finite Impulse Response Filters," *Proc. VLSI Design '93* pp. 166-171.