A Complete Pipelined Parallel CORDIC Architecture for Motion Estimation

Jie Chen and K. J. Ray Liu

Department of Electrical Engineering and Institute for Systems Research
University of Maryland, College Park, MD 20742, USA

ABSTRACT

We propose a novel fully-pipelined parallel CORDIC architecture (CORDIC-DXT-ME) employing the DCT Pseudo-Phase Techniques for Motion Estimation. Its low computational complexity, $O(N^2)$ as compared with $O(N^4)$ of BKM-ME, makes it fascinating in real time applications. In addition, the DCT-based nature enables us to replace all multipliers by CORDICs with simple shift-and-add operations and to incorporate its implementation with the DCT codec design to gain further savings in hardware complexity. The architecture is regular, modular, and has solely local connection and is suitable for MPEG2 compatible video codec design on a dedicated single chip.

I. Introduction

Extensive research has been done in the past in designing cost-effective MPEG compatible video codec architectures [1]. The most commonly used Block Matching Algorithm (BKM-ME) motion estimation architectures are based on the matching of blocks between the current and a reference frame in terms of the mean absolute difference (MAD). Exhaustive search block matching architectures implemented on VLSI chips [2] decompose four dimensional search onto lower dimension systolic arrays. In spite of the simplicity, the computational complexity is very high, i.e. $O(N^4)$ for a $N \times N$ block. To reduce the computational complexity, some hierarchical search structures which require two or more sequential steps to find suboptimal estimates have been proposed. Besides the block-based approaches, the pel recursive motion estimation architecture proposed in [3] is very vulnerable to noise and may suffer from the instability problem.

In this paper, we propose a novel fully-pipelined parallel CORDIC (COoridinate Rotation Digital Computer) [5] architecture (CORDIC-DXT-ME) employing the DCT Pseudo-Phase Techniques for Motion Estimation. Unlike fast block search motion estimation methods which simply pick several displacement candidates out of all possible displacement values in terms of minimum MAD values of a reduced number of pixels, the DCT Pseudo-Phase Techniques employ the sinusoidal orthogonal principles to directly extract displacement information from the discrete sinusoidal DXT(DCT/DST) transform coefficients of images. The hybrid DCT motion-compensated video coder structure used in MPEG2, H.261 and H.263 is an undesirable coder architecture because the throughput of the coder is limited by the processing speed of the feedback loop which is the major bottleneck of the entire digital video system for high-end real-time applications. On the other hand, the DCT-Based Motion Estimation (DXT-ME) algorithm [4] works in the DCT transform domain so that we can moved DCT/IDCT out of the loop. This not only reduces the complexity of the coder but also achieves higher system throughput by resolving the bottleneck problem without any tradeoff of the performance. From the implement point of view, we can get rid of the IDCT used for DXT-ME by interleaving it with the DCT. Furthermore, now the DCT and motion estimation are combined into a single component of relatively low complexity. This major breakthrough makes the MPEG2 compatible video codec design on a single dedicated chip feasible in no sacrifice of the performance.

The matrix rotational property of the DXT-ME algorithm motivates us to choose CORDIC as the basic computation units instead of conventional multiplier-and-adder units (MAC). Because CORDIC with simple shift-and-add is more efficient in evaluating square roots, divisions, trigonometric functions and their inverse, and, somewhat less often, hyperbolic transformations. In addition, CORDIC processing elements are extremely simple and quite compact to realize, while being no slower than the bit serial multipliers widely proposed for VLSI array structures.

In the next section, we introduce the time-recursive CORDIC 2D-DCT/2D-IDCT and the type conversion structures. In Section III. we present the pipelined CORDIC Pseudo-Phase computation structure and the peak searching architecture along with the simulation results. Finally, the paper is concluded in Section IV.

II. 2D-DCT/2D-IDCT and Type Conversion

A. Time-Recursive 2D-DXT and its inverse Structure

The one-dimensional DCT/DST (1D-DXT-II) of a sequential input data starting from $x(t)$ and end ending with $x(t+N)$ is defined as

$$X_t^c(k) = \frac{2}{N}C(k)\sum_{n=t}^{t+N-1} x(n)\cos[\frac{k\pi}{N}[(n-t)+\frac{1}{2}]];$$

$$X_t^s(k) = \frac{2}{N}C(k)\sum_{n=t}^{t+N-1} x(n)\sin[\frac{k\pi}{N}[(n-t)+\frac{1}{2}]];$$

$$\text{where } C(k) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{for } k=0 \text{ or } N, \\ 1, & \text{otherwise,} \end{cases}$$

Here the time index t in $X_t^c(k)$ and $X_t^s(k)$ denotes that the transform starts from $x(t)$. The time-recursive updating is derived in [7] as:

For $k = 1 \ldots N-1$

$$\begin{cases} \begin{bmatrix} X_{t+1}^c(k) \\ X_{t+1}^s(k) \end{bmatrix} = \mathbf{R}(2)\left[\begin{bmatrix} X_t^c(k) \\ X_t^s(k) \end{bmatrix} + \begin{bmatrix} \overline{X}_t^c(k) \\ \overline{X}_t^s(k) \end{bmatrix} \right] \\ \begin{bmatrix} \overline{X}_t^c(k) \\ \overline{X}_t^s(k) \end{bmatrix} = \mathbf{R}(1)\begin{bmatrix} \frac{2}{N}(-x(t)+(-1)^k x(t+N)) \\ 0 \end{bmatrix} \end{cases} \quad (1)$$

$$\text{where } \mathbf{R}(m) = \begin{bmatrix} \cos\frac{mk\pi}{2N} & \sin\frac{mk\pi}{2N} \\ -\sin\frac{mk\pi}{2N} & \cos\frac{mk\pi}{2N} \end{bmatrix}$$

*This work is supported in part by the NSF NYI award MIP9457397 and the ONR grant N00014-93-10566.

The commonly used row-column 2D-DCT structures are not able to simultaneously generate dual DCT and DST outputs which will be used for computing the Pseudo Phases. The multiplications in the plane rotation matrix $\mathbf{R}(m)$ can be implemented by CORDICs shown in Fig. 1. For $k = 0$

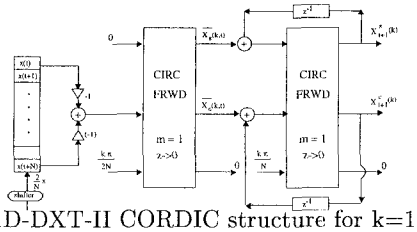


Fig. 1. 1D-DXT-II CORDIC structure for k=1,...,N-1

in the DCT and $k = N$ in the DST, $\mathbf{R}(m)$ can be reduced to addition and subtraction. If N is even, we can save one CORDIC in the $\frac{N}{2}$th channel, because $\mathbf{R}(2)$ can be simplified to addition and subtraction.

In the same fashion, the one-dimensional time-recursive inverse IDCT/IDST (1D-IDXT-II) updating are derived in [7]. The purpose to introduce the auxiliary variable $x_t^{as}(n)$ is to maintain the lattice structure same as in 1D-DXT-II case. By spreading out $x_t^s(n), x_t^{as}(n)$ and canceling the common terms, we get

$$x_t^s(n) = x_t^{as}(n) + (-1)^n \frac{1}{\sqrt{2}} X(t+N);$$

Therefore, similar to the DCT/DST case, the inverse transform IDCT/IDST actually can be combined and dully generated shown in Fig. 2. From the above discussion, we find

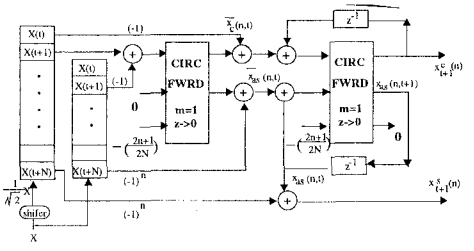


Fig. 2. The CORDIC structure for 1D-IDXT-II

both 1D-DXT-II and inverse 1D-IDXT-II modules share the similar CORDIC structure. N such modules can be used for parallel computing 1D-DXT-II/1D-IDXT-II coefficients for different channels. It takes N clock cycles to dually generate $X_t^c(k), X_t^s(k)$ and their inverse $x_t^c(n), x_t^s(n)$. So the computational complexity is $O(N^2)$ and it needs $2(N-1)$ CORDICs for 1D-DXT-II and $2N$ for its inverse 1D-IDXT-II .

We extend and modify the two dimensional DCT structure in [6] to simultaneously generate DCCT, DCST, DSCT and DSST coefficients by eliminating the *circular shift array*, because it is superfluous and only causes extra $O(N)$ delay for the *Shift Register Array* to rewind the final results. Using the similar approach, we can get its inverse counterpart structure shown in Fig. 3. Both 2D-DXT-II and inverse 2D-IDXT-II share the same structure shown in Fig. 3 except the *transform module* used in front of *1D-DXT/1D-IDXT array*. 2D-DXT-II/2D-IDXT-II is composed of three 1D-DXT-II/1D-IDXT-II thus the coder needs total of $6(N-1)$ CORDICs for 2D-DXT-II and $6N$ for 2D-IDXT-II. (If N is even, we can save three CORDICs for the $\frac{N}{2}$th channel in 2D-DXT-II). Since it is fully pipelined, the computational complexity is still $O(N^2)$ and the latency is $2N$.

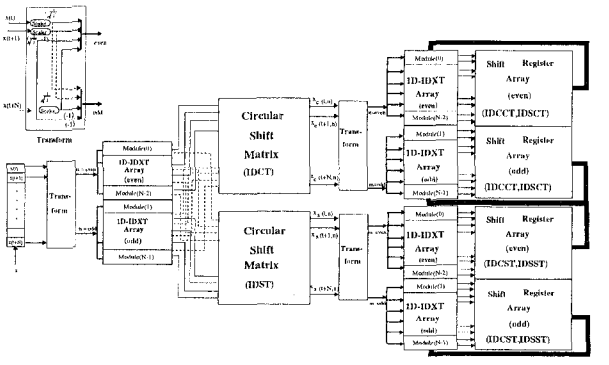
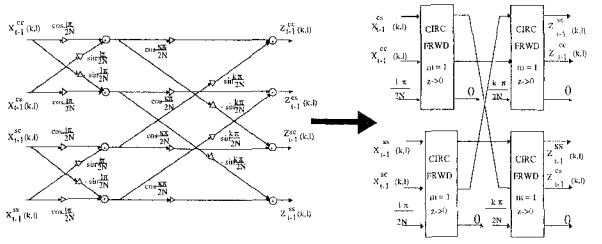


Fig. 3. inverse 2D-IDXT-II corder structure

B. 2D-DXT-II to 2D-DXT-I Type Conversion

The DXT-ME algorithm requires type I 2D-DXT-I coefficients $Z_{t-1}^{cc}(k,l)$, $Z_{t-1}^{cs}(k,l)$, $Z_{t-1}^{sc}(k,l)$ and $Z_{t-1}^{ss}(k,l)$ to generate the Pseudo Phases. Actually the 2D-DXT-I functions can be obtained by the plane rotation of the 2D-DXT-II kernels.

The lattice and its corresponding CORDIC structure for the conversion kernel is depicted in Fig. 4. The conversion



(a) conversion kernel (b) its CORDIC structure

Fig. 4. The structure of the conversion kernel

at the block boundary can be simplified to one dimensional relationship. Thus the structure for the conversion at the block boundary is similar to the one in Fig. 4 but only has one stage. A parallel lattice array consists of $N-1$ conversion kernel modules in Fig. 4 along with one conversion boundary module can be used for parallel transformation and it improves the computational speed drastically. The whole structure requires $4N$ CORDICs and takes $O(N)$ time to finish 2D-DXT-II to 2D-DXT-I transformation. The computational complexity is $O(N^2)$.

III. Pseudo Phase Computation & Peak search

A. Pipelined Pseudo-Phase Computation Structure

In a two dimensional space, the two Pseudo Phase functions $f(k,l)$ and $g(k,l)$ are defined in [4] by taking the block boundary into consideration. The $\mathbf{Z}_{t-1}(k,l) \in R^{4\times 4}$ is the *system matrix* of the DXT-ME algorithm at (k,l). At the boundaries of each block in the transform domain, the DCT coefficients of $x_{t-1}(m,n)$ and $x_t(m,n)$ have only one dimensional relationship.

For $k = 0, l = 1 \ldots N-1$

To solve

$$f(0,l) = \frac{1}{\sqrt{2}} \frac{Z_{t-1}^{cc}(0,l)X_t^{cs}(0,l) - Z_{t-1}^{cs}(0,l)X_t^{cc}(0,l)}{(Z_{t-1}^{cc}(0,l))^2 + (Z_{t-1}^{cs}(0,l))^2}$$

is equivalent to solve the following linear equation

$$\begin{bmatrix} Z_{t-1}^{cc}(0,l) & Z_{t-1}^{cs}(0,l) \\ -Z_{t-1}^{cs}(0,l) & Z_{t-1}^{cc}(0,l) \end{bmatrix} \begin{bmatrix} f(0,l) \\ \star \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} X_t^{cs}(0,l) \\ X_t^{cc}(0,l) \end{bmatrix} \quad (2)$$

Here $\star$ stands for *don't care*.

Let $K = \sqrt{(Z^{cc}_{t-1}(0,l))^2 + (Z^{cs}_{t-1}(0,l))^2}$ and $\alpha = tan^{-1}(\frac{Z^{cs}_{t-1}(0,l)}{Z^{cc}_{t-1}(0,l)})$, then (2) can be converted to

$$\begin{bmatrix} f(0,l) \\ \star \end{bmatrix} = \frac{1}{\sqrt{2}}\frac{1}{K}\begin{bmatrix} cos\alpha & -sin\alpha \\ sin\alpha & cos\alpha \end{bmatrix}\begin{bmatrix} X^{cs}_t(0,l) \\ X^{cc}_t(0,l) \end{bmatrix} \quad (3)$$

We can cascade circular backward rotation CORDIC to compute K and α and circular forward CORDIC to calculate $f(0,l)$ followed by a linear backward rotation to scale $f(0,l)$ by $\frac{1}{\sqrt{2}K}$ as shown in Fig. 5. The similar approach can be applied to compute

$f(k,N) = \frac{1}{\sqrt{2}}\frac{Z^{cs}_{t-1}(k,N)X^{cs}_t(k,N)+Z^{ss}_{t-1}(k,N)X^{ss}_t(k,N)}{(Z^{cs}_{t-1}(k,N))^2+(Z^{ss}_{t-1}(k,N))^2}$

For $k = 0, l = N$

$$f(0,N) = \frac{1}{2}\frac{X^{cs}_t(0,N)}{Z^{cc}_{t-1}(0,N)} = \frac{1}{2}\frac{X^{cs}_t(0,N)}{X^{cs}_{t-1}(0,N)} \quad (4)$$

This is much easier to compute by using only one linear backward rotation CORDIC followed by a right shifter to half the previous result. We can get Pseudo Phase $g(k,l)$ at the block boundary with the same method.

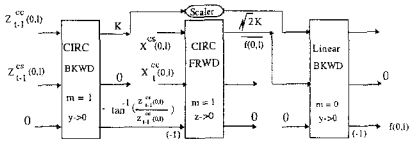


Fig. 5. The Pseudo Phase f(k,l) at block boundary

To find the Pseudo Phase $f(k,l)$ and $g(k,l)$ for $k,l \in \{1,\ldots,N-1\}$ [4], it requires to solve the following linear equation with $\mathbf{Z}_{t-1}(k,l) \in R^{4\times 4}$.

$$\mathbf{Z}_{t-1}(k,l)\begin{bmatrix} \star \\ f(k,l) \\ g(k,l) \\ \star \end{bmatrix} = \begin{bmatrix} X^{cc}_t(k,l) \\ X^{cs}_t(k,l) \\ X^{sc}_t(k,l) \\ X^{ss}_t(k,l) \end{bmatrix}, \quad (5)$$

If we adopt Gauss elimination or Givens rotation method, it incurs a heavy computational burden. However, by exploring the structure embedded in the $\mathbf{Z}_{t-1}(k,l)$, we can reduce the two dimensional problem to one dimension. Let us first introduce the following Lemma which inspires us to find the way of solving (5).

Lemma 1 *When* $\frac{Z^{sc}}{Z^{cc}} = \frac{Z^{ss}}{Z^{cs}}$, $\mathbf{Z}_{t-1}(k,l)$ *in (5) is a orthogonal matrix*

In this situation, we can adopt the similar CORDIC structure in Fig. 4 to find the Pseudo Phases $f(k,l)$ and $g(k,l)$

When $\frac{Z^{sc}}{Z^{cc}} \neq \frac{Z^{ss}}{Z^{cs}}$

By multiplying both sides of (5) by $\mathbf{Z}^T_{t-1}(k,l)$, we get

$$\begin{bmatrix} K & 0 & 0 & G \\ 0 & K & G & 0 \\ 0 & G & K & 0 \\ G & 0 & 0 & K \end{bmatrix}\begin{bmatrix} \star \\ f(k,l) \\ g(k,l) \\ \star \end{bmatrix} = \mathbf{Z}^T_{t-1}(k,l)\begin{bmatrix} X^{cc}_t(k,l) \\ X^{cs}_t(k,l) \\ X^{sc}_t(k,l) \\ X^{ss}_t(k,l) \end{bmatrix} \quad (6)$$

where

$$K = (Z^{cc}_{t-1}(k,l))^2 + (Z^{cs}_{t-1}(k,l))^2 + (Z^{sc}_{t-1}(k,l))^2 + (Z^{ss}_{t-1}(k,l))^2,$$

$$G = 2 * [Z^{cs}_{t-1}(k,l)Z^{sc}_{t-1}(k,l) - Z^{cc}_{t-1}(k,l)Z^{ss}_{t-1}(k,l)]$$

Here we are only interested in solving Pseudo Phase $f(k,l)$ and $g(k,l)$, so (6) can be reduced to 2×2 matrix equation.

$$\begin{bmatrix} K & G \\ G & K \end{bmatrix}\begin{bmatrix} f(k,l) \\ g(k,l) \end{bmatrix} = \underbrace{\begin{bmatrix} Z^{cs}_{t-1}(k,l) & Z^{sc}_{t-1}(k,l) \\ Z^{sc}_{t-1}(k,l) & Z^{cs}_{t-1}(k,l) \end{bmatrix}\begin{bmatrix} -X^{cc}_t(k,l) \\ X^{ss}_t(k,l) \end{bmatrix}}_{\Gamma_1}$$

$$+ \underbrace{\begin{bmatrix} Z^{cc}_{t-1}(k,l) & -Z^{ss}_{t-1}(k,l) \\ -Z^{ss}_{t-1}(k,l) & Z^{cc}_{t-1}(k,l) \end{bmatrix}\begin{bmatrix} X^{cs}_t(k,l) \\ X^{sc}_t(k,l) \end{bmatrix}}_{\Gamma_2} = \begin{bmatrix} P \\ Q \end{bmatrix} \quad (7)$$

Then $f(k,l)$ and $g(k,l)$ can be found as:

$$\begin{bmatrix} f(k,l) \\ g(k,l) \end{bmatrix} = \frac{1}{\Delta}\begin{bmatrix} K & -G \\ -G & K \end{bmatrix}\begin{bmatrix} P \\ Q \end{bmatrix} \quad (8)$$

where $\Delta = K^2 - G^2$.

Based on the previous derivation, a 2-stage pipelined structure can be designed. In the first stage, we can compute K and G, Γ_1 and Γ_2 in parallel. Then the results are fed coincidentally into the second stage to compute the Pseudo Phases. Here we need to employ the somewhat less often, hyperbolic transformations to evaluate Γ_1, Γ_2 and the Pseudo Phases $f(k,l)$ and $g(k,l)$. In summary, we can parallelize $N-1$ kernel modules with a boundary module to speed up the Pseudo Phases computation. It needs $(14N-6)$ CORDICs and $O(N)$ time to accomplish the work. And the computational complexity is $O(N^2)$

B. Peak Searching & Simulation Results

The two-dimensional search for the peak value among $F(m,n)$ and $G(m,n)$ in [4] where $F(m,n) = IDCSTII(f(k,l))$ and $G(m,n) = IDSCII(g(k,l))$ can be reduced to the one-dimensional search by the row-column decomposition search which looks for the peak value of each row, followed by a vertical search of the previous results shown in Fig. 6 (here we use $G(m,n)$ as an example, it is also applicable to $F(m,n)$). The $\hat{d}$ is the peak among $G(m,n)$. The flow diagram for

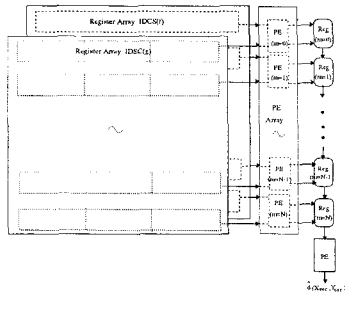
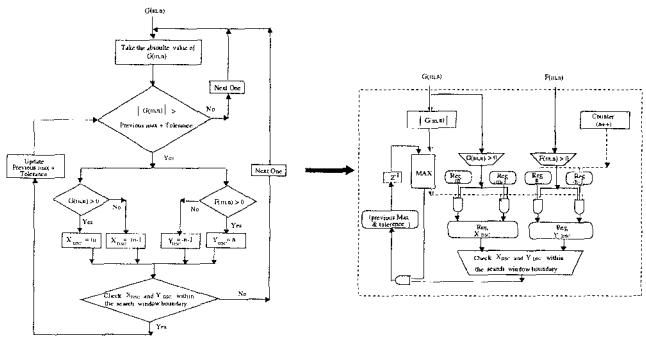


Fig. 6. The two dimensional peak search structure

designing the process element (PE) and the corresponding hardware structure are shown in Fig. 7. The peak search needs $2(N+1)$ PEs and takes $2N$ time to finish. The computational complexity is $O(N^2)$

The assemble CORDIC-DXT-ME architecture block diagram depicted in Fig. 8. The current frame x_t are fed into 2D-DXT-II which computes four coefficients $X^{cc}_t(k,l)$, $X^{cs}_t(k,l)$, $X^{sc}_t(k,l)$ and $X^{ss}_t(k,l)$ meanwhile the coefficients of previous frame x_{t-1} are transformed to 2D-DXT-I coefficients $Z^{cc}_{t-1}(k,l)$, $Z^{cs}_{t-1}(k,l)$, $Z^{sc}_{t-1}(k,l)$ and $Z^{ss}_{t-1}(k,l)$. The *Pseudo Phases Computation* module utilizes all the previous parameters and takes N times to produce two Pseudo-Phase functions $f(k,l)$ and $g(k,l)$. Then $f(k,l)$ and $g(k,l)$ undergo in-



(a) flowchart (b) corresponding structure

Fig. 7. The process element for peak search

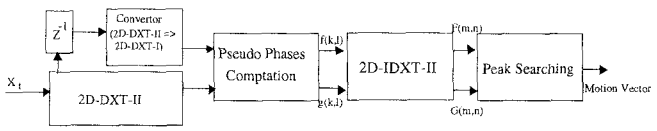


Fig. 8. The assemble CORDIC-DXT-ME block diagram

verse IDCST-II and IDSCT-II transform to generate $F(m,n)$ and $G(m,n)$. Finally we search the peak values among $F(m,n)$ and $G(m,n)$ to determine the motion vector. So the CORDIC-DXT-ME are fully-pipelined while it has massively parallel local operations. As we mentioned in the introduction, we can eliminate inverse 2D-IDXT-II by interleaving it with 2D-DXT-II to trade the speed for the silicon area. The computational complexity and hardware cost of each stage are summarized in Table 1. In Fig. 9, we move the image

Table 1: Computational complexity and number of CORDICs

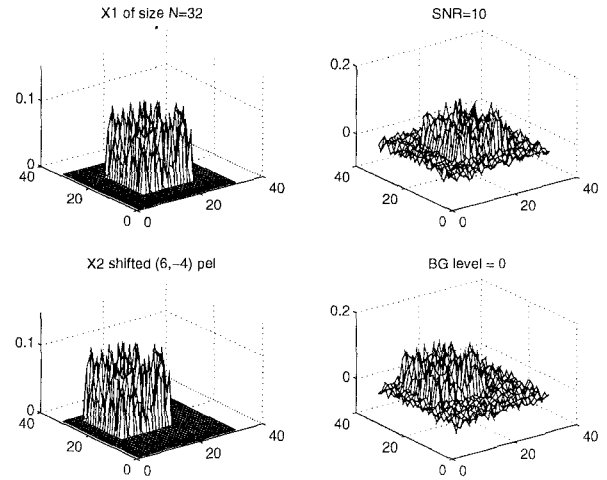
Stage	Component	CORDICs	Computational Complexity
1	2D-DXT-II	6N-6	$O(N^2)$
	Conversion	4N	$O(N^2)$
2	Pseudo Phase	14N-6	$O(N^2)$
3	2D-IDXT-II	6N	$O(N^2)$
4	Peak Searching	0	$O(N^2)$

X1 in the direction (6,-4) corresponding to image X2 with additive Gaussian noise at SNR=10dB. Our simulation of the designed CORDIC-DXT-ME shows that it estimates the correct motion shown in Fig. 10

IV. Conclusion

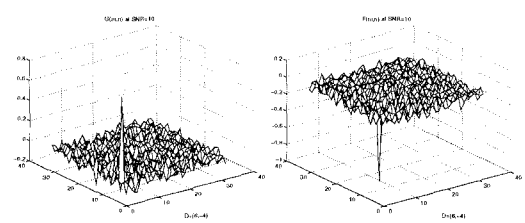
In this paper, we propose a novel fully-pipelined parallel CORDIC architecture (CORDIC-DXT-ME) employing DCT Pseudo-Phase Techniques for Motion Estimation. The CORDIC-DXT-ME architecture requires only $30N - 12$ CORDIC processors. In addition to its low computational complexity, the DCT-based nature enables us to replace all multipliers by CORDIC with simple shift-and-add operations and to incorporate its implementation with the DCT codecs design to gain further savings in hardware complexity.

The CORDIC-DXT-ME architecture efficiently combines both the DCT and motion estimation into the Fully DCT-Based Video Coder. Thus the performance-critical feedback loop contains only transform-domain motion estimation unit instead of three major components as in the conventional hybrid DCT motion-compensated video coder design thus it achieves higher throughput and lower system complex-



(a) inputs x1 and x2 (b) inputs with added noise

Figure 9: A movement (6,-4) with additive with Gaussian noise at SNR=10 dB



(f) G(m,n) and F(m,n)

Figure 10: CORDIC-DXT-ME architecture estimates the correct motion (6,-4)

ity. The CORDIC-DXT-ME architecture has inherently massively parallel local operations and fully-pipelined global manipulations. Its regular and modular structure along with only local interconnection makes it feasible for MPEG2 compatible video codec design on a dedicated single chip.

References

[1] P. Pirsch, N. Demassieux and W. Gehrke, "VLSI Architectures for Video Compression – A Survey", *Proceedings of the IEEE*, vol. 83, no. 2, pp. 220–245, Feburay 1995.

[2] T. Komarek and P. Pirsch, "Array architectures for block-matching algorithms", *IEEE Trans. Circuits and Systems*, vol. 36, no. 10, pp. 1301–1308, October 1989.

[3] R. C. Kim and S. U. Lee, "A VLSI architecture for a pel recursive motion estimation algorithm", *IEEE Trans. Circuits and Systems*, vol. 36, no. 10, pp. 1291–1300, October 1989.

[4] U. -V. Koc and K. J. R. Liu, "Discrete-Cosine/Sine-Transform based motion estimation", *Proceedings of IEEE International Conference on Image Processing (ICIP)*, Austin, Texas, November 1994, vol. 3, pp. 771-775

[5] Y. H. Hu "CORDIC-Based VLSI Architectures for Digital Signal Processing", *IEEE Signal Processing Magine*, pp. 16–35, July 1992.

[6] C. T. Chiu and K. J. R. Liu, "Real-time parallel and fully pipelined two-dimensional DCT lattice structures with applications to HDTV systems", *IEEE Trans. Circuits and Systems for Video Technology*, vol. 2, no. 1, pp. 25–37, March 1992.

[7] K. J. R. Liu and C. T. Chiu, "Unified parallel lattice structures for time-recursive Discrete Cosine/Sine/Hartley transforms", *IEEE Trans. Signal Processing*, vol. 41, no. 3, pp. 1357–1377, March 1993.