# A FULLY PIPELINED PARALLEL CORDIC ARCHITECTURE FOR HALF-PEL MOTION ESTIMATION*

*Jie Chen* and *K. J. Ray Liu*

Electrical Engineering Department and Institute for Systems Research
University of Maryland at College Park

## ABSTRACT

*Based on the concept of pseudo-phase and the sinusoidal orthogonal principles, we design a novel low-complexity and high throughput CORDIC (COordinate Rotation Digital Computer) architecture for half-pel motion estimation. The proposed multiplier-free and fully-pipelined parallel architecture works solely at DCT domain without interpolation of input images to meet the needs of high-quality high-bitrate video communications. In addition, the DCT-based nature enables us to efficiently combine the DCT and motion estimator into one single component of relatively low complexity. Its regular and modular structure along with only local interconnection provides a low-complexity solution for MPEG-2 and H.263 compatible video codec design on a dedicated single chip.*

## 1. INTRODUCTION

To further improve the compression rate and resolution, motion estimation with sub-pixel accuracy is essential because movements in a video sequence are not necessarily multiples of the sampling grid distance. As a result, motion compensation with half-pel accuracy is recommended in MPEG-2 and H.263 standards. In [1], the exhaustive block matching is implemented with a 1-D systolic array. Then a half-pel precision processing unit is introduced to estimate half-pel motion vectors based on integer-pel accuracy using bilinear interpolation method. However interpolation not only increases the complexity and data flow of a coder but also may adversely affect the accuracy of motion estimated from the interpolated images. Therefore, the drawbacks of the implementation in [1] are the loss of accuracy and increased computational complexity.

Based on the concept of pseudo-phase and the sinusoidal orthogonal principles [2],[3], we propose a novel low-complexity, and high throughput fully pipelined parallel CORDIC [4] architecture (CORDIC-HDXT-ME) for half-pel motion estimation. This multiplier-free architecture works solely at DCT domain without interpolation of input images to meet the needs of high-quality high-bitrate video communications. To obtain an estimation of half-pel accuracy, we can have better flexibility and scalability by first computing the integer-pel motion vectors and then considering only those around them to determine the half-pel motion vectors. Its low complexity, $O(N^2)$ as compared with $O(N^4)$ for commonly used half-pel Block Matching architecture (HBKM-ME) [1], makes it attractive in real-time multimedia applications. In addition, its ability to estimate motion completely in DCT domain enables us to replace all multiply-and-add (MAC) operations in plane rotation with CORDIC processors, and to efficiently combine the two major processing components, the DCT and motion estimator, into one single component of relatively low complexity. Furthermore, from the implementation point of view, we can get rid of the IDCT module used for half-pel motion estimation by interleaving it with the DCT module. Those major features along with the regular, modular and only local connected properties of the proposed CORDIC-HDXT-ME architecture make the MPEG-2, H.263 compatible real-time video codec design on a single dedicated chip feasible without sacrifice of the performance.

In this paper we first discuss the CORDIC architecture for half-pel motion estimation based on pseudo-phase techniques. We then present the simulation results of our design. The paper is concluded in Section 3.

## 2. HIGH-PERFORMANCE HALF-PEL MOTION ESTIMATOR DESIGN

According to the SDXT-ME algorithm in [3], we derive the assembled block diagram of CORDIC-HDXT-ME architecture outlined in Fig. 1. It has four major processing stages/phases:

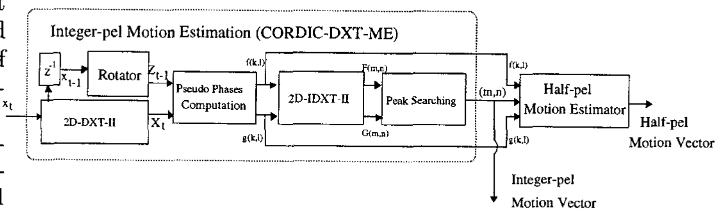


Figure 1: The assembled block diagram of CORDIC-HDXT-ME

1. The $N \times N$ block of pixels at the current frame $x_t$ are fed into *type II DCT/DST* (2D-DXT-II) coder which computes four coefficients $X_t^{cc}(k,l)$, $X_t^{cs}(k,l)$, $X_t^{sc}(k,l)$ and $X_t^{ss}(k,l)$ such as:

$$X_t^{cc}(k,l) = \frac{4}{N^2}C(k)C(l)\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} x_t(m,n)\cos[\frac{k\pi}{N}(m+\frac{1}{2})]\cos[\frac{l\pi}{N}(n+\frac{1}{2})].$$

Meanwhile the $N \times N$ *type II DCT/DST* coefficients of the previous frame $x_{t-1}$ are transformed to *type I DCT/DST* (2D-DXT-I) coefficients $Z_{t-1}^{cc}(k,l)$, $Z_{t-1}^{cs}(k,l)$, $Z_{t-1}^{sc}(k,l)$ and $Z_{t-1}^{ss}(k,l)$ such as:

*This work is supported in part by the ONR grant N00014-93-1-0566, NSF grant MIP9309506, and MIPS/MicroStar.

0-8186-8183-7/97 $10.00 © 1997 IEEE

$$Z^{cc}_{t-1}(k,l) = \frac{4}{N^2}C(k)C(l)\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} x_{t-1}(m,n)\cos[\frac{k\pi}{N}(m)]\cos[\frac{l\pi}{N}(n)].$$

2. The pseudo phases computation module utilizes all the previous parameters and takes $O(N)$ times to produce two pseudo-phase functions $f(k,l)$ and $g(k,l)$.

3. $f(k,l)$ and $g(k,l)$ undergo *type II inverse* IDCT/IDST (2D-IDXT-II) transform to generate $F(m,n)$ and $G(m,n)$.

4. Finally we search the peak values based on sinusoidal orthogonal principles among $F(m,n)$ and $G(m,n)$ to determine the integer-pel motion vector $(m,n)$.

To obtain an estimation at half-pel accuracy, we can have better flexibility and scalability by first computing the integer-pel motion vectors $(m,n)$ and then utilizing the half-pel motion estimator block to obtain the estimation at half-pel accuracy. The half-pel motion vector is determined by only considering the nine possible positions around the integer-pel displacement $(m,n)$.

In what follows, we describe the detailed design of each block in Fig. 1.

## *A. Time-Recursive Programmable Module for 2D-DXT-II and 2D-IDXT-II*

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

$$X^c_t(k) = \frac{2}{N}C(k)\sum_{n=t}^{t+N-1} x(n)\cos[\frac{k\pi}{N}[(n-t)+\frac{1}{2}]];$$
$$X^s_t(k) = \frac{2}{N}C(k)\sum_{n=t}^{t+N-1} x(n)\sin[\frac{k\pi}{N}[(n-t)+\frac{1}{2}]];$$
$$\text{where } C(k) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{for } k=0 \text{ or } N, \\ 1, & \text{otherwise,} \end{cases}$$

Here the time index $t$ in $X^c_t(k)$ and $X^s_t(k)$ denotes that the transform starts from $x(t)$. Unlike the commonly used 2D-DCT structures (such as matrix factorization, systolic structure implementation, *etc*), the time-recursive DCT architecture derived in [5] is able to simultaneously generate DCT and DST outputs, which can be used in computing pseudo phases later. The time-recursive updating of 1D-DXT-II is given by:

$$\begin{cases} \begin{bmatrix} X^c_{t+1}(k) \\ X^s_{t+1}(k) \end{bmatrix} = \mathrm{R}(2)\left[\begin{bmatrix} X^c_t(k) \\ X^s_t(k) \end{bmatrix} + \begin{bmatrix} \overline{X}^c_t(k) \\ \overline{X}^s_t(k) \end{bmatrix}\right] \\ \begin{bmatrix} \overline{X}^c_t(k) \\ \overline{X}^s_t(k) \end{bmatrix} = \mathrm{R}(1)\begin{bmatrix} \frac{2}{N}(-x(t)+(-1)^k x(t+N)) \\ 0 \end{bmatrix} \end{cases} \tag{1}$$

where $\mathrm{R}(m) = \begin{bmatrix} \cos\frac{mk\pi}{2N} & \sin\frac{mk\pi}{2N} \\ -\sin\frac{mk\pi}{2N} & \cos\frac{mk\pi}{2N} \end{bmatrix}$. Note that the orthogonal rotation of the lattice structure in (1) is numerical stable. This nice numerical property is very useful in finite-precision implementation.

Similarly, the updating of *inverse 1D-DXT-II* (1D-IDXT-II) is given by:

$$\begin{bmatrix} x^c_{t+1}(n) \\ x^{as}_{t+1}(n) \end{bmatrix} = \begin{bmatrix} cos(\frac{2n+1}{2N})\pi & sin(\frac{2n+1}{2N})\pi \\ -sin(\frac{2n+1}{2N})\pi & cos(\frac{2n+1}{2N})\pi \end{bmatrix} \left[\begin{bmatrix} X^c_t(n) \\ X^{as}_t(n) \end{bmatrix} + \begin{bmatrix} -\frac{1}{\sqrt{2}}X(t) \\ (-1)^n X(t+N) \end{bmatrix}\right] + \begin{bmatrix} (\frac{1}{\sqrt{2}}-1)X(t+1) \\ 0 \end{bmatrix}. \tag{2}$$

The auxiliary variable $x^{as}_t(n)$ is related to $x^s_t(n)$ by: $x^s_t(n) = x^{as}_t(n) + (-1)^n\frac{1}{\sqrt{2}}X(t+N)$. From the above discussion, we observe that both 1D-DXT-II and its inverse 1D-IDXT-II share a common computational module with only some minor differences in the data paths and the rotation angles. We thus can integrate both modules into one programmable module as shown in Fig. 2. The setting of switches

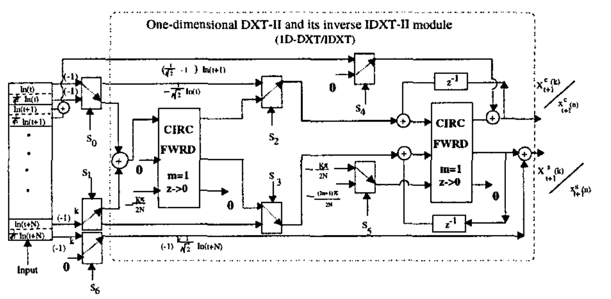


Figure 2: 1D-DXT/IDXT-II programmable module

$\mathrm{S} \doteq [s_0 s_1 s_2 s_3 s_4 s_5 s_6]$ in Fig. 2 is for the 1D-DXT-II and the complementary setting is for its inverse 1D-IDXT-II.

In the third stage of Fig. 1, two pseudo-phase functions, $f(k,l)$ and $g(k,l)$, pass through 2D-IDXT-II module (*type II inverse IDCST* and *IDSCT*) to generate $F(m,n)$ and $G(m,n)$ in view of the orthogonal property:

$$F(m,n) = \frac{4}{N^2}\sum_{k=0}^{N-1}\sum_{l=1}^{N} C(k)C(l)f(k,l)\cos\frac{k\pi}{N}(m+\frac{1}{2})\sin\frac{l\pi}{N}(n+\frac{1}{2})$$
$$G(m,n) = \frac{4}{N^2}\sum_{k=1}^{N}\sum_{l=0}^{N-1} C(k)C(l)g(k,l)\sin\frac{k\pi}{N}(m+\frac{1}{2})\cos\frac{l\pi}{N}(n+\frac{1}{2})$$

In order to interleave the DCT module used in the first stage with the IDCT module in the third stage of Fig. 1, we extend and modify two dimensional DCT structure in [6] to simultaneously generate *type II* DCCT, DCST, DSCT and DSST coefficients when the pixels of the current frame $x_t$ are selected, or $F(m,n)$ and $G(m,n)$ when $f(k,l)$ and $g(k,l)$ are in turn selected shown in Fig. 3. To dually generate $F(m,n)$

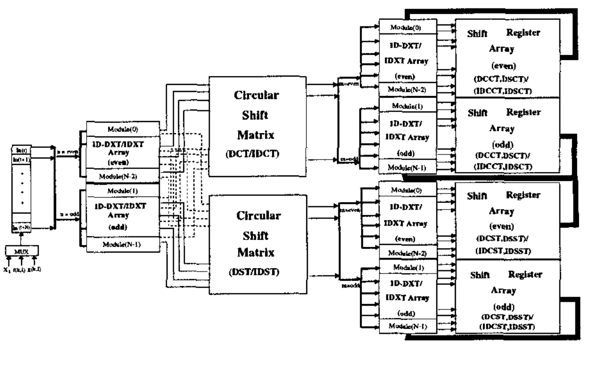


Figure 3: Time-recursive structure for 2D-DXT/IDXT-II

and $G(m,n)$, we use a multiplexer to control the input data flow by choosing $N$ of $f(k,l)$ $l \in \{1,\ldots,N\}$ followed by $g(k,l)$ $l \in \{0,\ldots,N-1\}$ alternatively for different $k$. As a result, the module generates one-dimensional $IDST(f(k,l))$ which enters the lower Circular Shift Matrix in Fig. 3 and $IDCT(g(k,l))$ which enters the upper Matrix in turn. The two-dimensional $IDCST(f(k,l))$ and $IDSCT(g(k,l))$ can be obtained independently afterwards.

Note that the *type I* 2D-DXT-I coefficients $Z^{cc}_{t-1}(k,l)$, $Z^{cs}_{t-1}(k,l)$, $Z^{sc}_{t-1}(k,l)$ and $Z^{ss}_{t-1}(k,l)$ required by the pseudo-phase computation in the second stage of Fig. 1 can actually be obtained by the plane rotation of the *type II* 2D-DXT-II kernels as shown in Fig. 4.

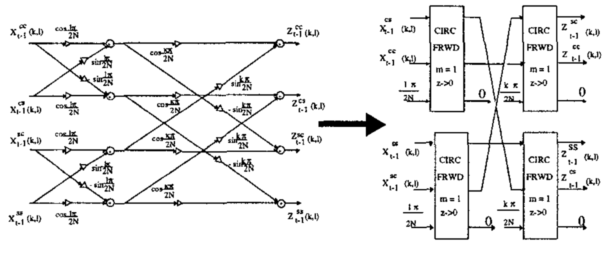


Figure 4: 2D-DXT-II to 2D-DXT-I type conversion

### B. Pseudo-Phase Computation and Peak Searching

As mentioned in [2], the pseudo phases of integer-pel displacements between two frames $x_{t-1}$ and $x_t$ are computed by solving the *system equation*: $\mathbf{Z}_{t-1}(k,l)\cdot\vec{\theta}_{m,n}(k,l) = \vec{\mathbf{x}}_t(k,l)$, for $k,l \in \{1,\ldots,N-1\}$ where $(m,n)$ is the displacement, $\vec{\theta}_{m,n}$ is the pseudo-phase vector $[\star, f(k,l), g(k,l), \star]^{\mathrm{T}}$. Here $\star$ stands for *don't care*. If we adopt Gauss elimination or Givens rotation method to solve for pseudo-phase $f(k,l)$ and $g(k,l)$ in the *system equation*, 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. Because we are only interested in solving pseudo-phase $f(k,l)$ and $g(k,l)$, so by multiplying both sides of *system equation* by $\mathbf{Z}^T_{t-1}(k,l)$, we can reduce the *system equation* to $2 \times 2$ linear equation

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

where $K = (Z^{cc}_{t-1}(k,l))^2 + (Z^{cs}_{t-1}(k,l))^2 + (Z^{sc}_{t-1}(k,l))^2 + (Z^{ss}_{t-1}(k,l))^2$, $J = 2*[Z^{cs}_{t-1}(k,l)Z^{sc}_{t-1}(k,l) - Z^{cc}_{t-1}(k,l)Z^{ss}_{t-1}(k,l)]$. Then $f(k,l)$ and $g(k,l)$ can be found as:

$$\begin{bmatrix} f(k,l) \\ g(k,l) \end{bmatrix} = \frac{1}{H}\begin{bmatrix} K & -J \\ -J & K \end{bmatrix}\begin{bmatrix} A \\ B \end{bmatrix}, \quad \text{where } H = K^2 - J^2.$$

Based on the previous derivation, a pipelined structure can be designed shown in Fig. 5. Here some CORDIC processors are operated in the somewhat less often, *hyperbolic rotation modes* to evaluate $A, B$ and the pseudo phases $f(k,l)$ and $g(k,l)$.

The two-dimensional search for the peak value among $F(m,n)$ and $G(m,n)$ can be reduced to the one-dimensional search by the row-column decomposition search which first looks for the peak value of each row, followed by a vertical search of the previous results.

### C. Time-Recursive Programmable Module for half-pel motion estimator

To obtain an estimation at half-pel accuracy, we can first computing the integer-pel motion estimate $(m,n)$ and then utilize the half-pel motion estimator block produce $\overline{DCS}(u,v)$ and $\overline{DSC}(u,v)$ for $u \in \{m-0.5, m, m+0.5\}$ and $v \in \{n-0.5, n, n+0.5\}$ as shown in Fig. 1. The $\overline{DCS}(u,v)$ and $\overline{DSC}(u,v)$ are defined as follows [3]:

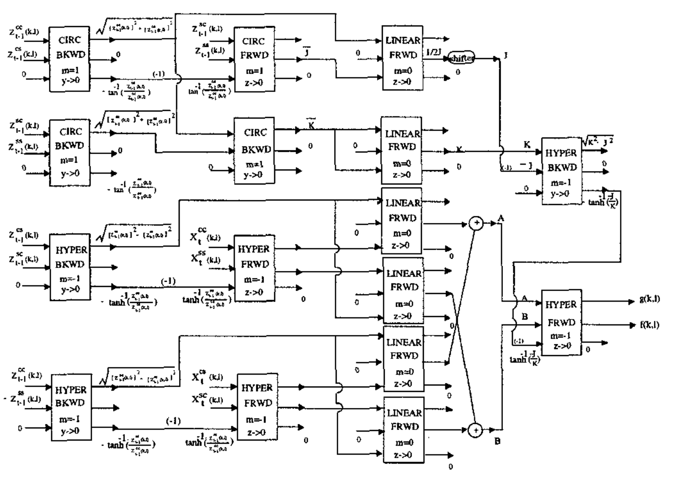


Figure 5: Pipelined structure for pseudo-phase computation

$$\overline{DCS}(u,v) = \sum_{k=0}^{N-1}\sum_{l=1}^{N} C(k)C(l)f(k,l)\cos\frac{k\pi}{N}(m+\mu_u+\frac{1}{2})\sin\frac{l\pi}{N}(n+\mu_v+\frac{1}{2})$$

$$\overline{DSC}(u,v) = \sum_{k=1}^{N}\sum_{l=0}^{N-1} C(k)C(l)g(k,l)\sin\frac{k\pi}{N}(m+\mu_u+\frac{1}{2})\cos\frac{l\pi}{N}(n+\mu_v+\frac{1}{2})$$

where $\mu_u, \mu_v \in \{-0.5, 0, 0.5\}$. The half-pel motion vector is determined by only considering the nine possible positions around the integer-pel displacement $(m,n)$. The peak position of either $\overline{DCS}(u,v)$ or $\overline{DSC}(u,v)$ determines the half-pel motion estimation.

For the computation of two-dimensional $\overline{DSC}(u,v)$, we can decompose its computation into tree-structured hierarchical *one-dimensional type II inverse IDCT/IDST* as shown in Fig. 6 (here we use $\overline{DSC}(u,v)$ as an example, the same approach can be applied to $\overline{DCS}(u,v)$). Because the

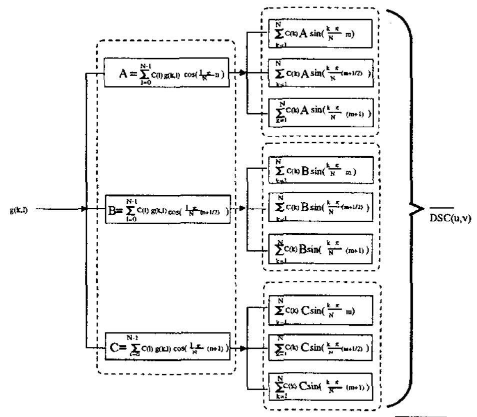


Figure 6: Tree-structured computation for $\overline{DSC}(u,v)$

computations encircled by dot-boxes in Fig. 6 are the same except for the phase differences. Let us define

$$G^c_t(l) = \frac{2}{N}\sum_{l=t}^{t+N-1} C(l-t)g(k,l)\cos\phi; \; G^s_t(l) = \frac{2}{N}\sum_{l=t+1}^{t+N} C(l-t)g(k,l)\sin\phi;$$

where $\phi$ stands for different phases such as $\frac{l\pi}{N}n$, $\frac{l\pi}{N}(n+\frac{1}{2})$ and $\frac{l\pi}{N}(n+1)$ in the middle level of Fig. 6. Therefore, by using the same time-recursive approach for inverse 1D-IDXT-II described previously, we can get the time-recursive updating similar to (2) for different $\phi$. Those updating equations are similar but with only some minor differences in the data paths and the rotation angles. Therefore, we can combine them to the following unified equation

$$\begin{bmatrix} G^{c}_{t+1}(l) \\ G^{as}_{t+1}(l) \end{bmatrix} = \Delta \left[ \begin{bmatrix} G^{c}_{t}(l) \\ G^{as}_{t}(l) \end{bmatrix} + \Gamma \right] + \begin{bmatrix} (\frac{1}{\sqrt{2}} - 1)g(k, t+1) \\ 0 \end{bmatrix} \quad (3)$$

The auxiliary variable $G^{s}_{t}(l)$ is related to $G^{as}_{t}(l)$ by

$$G^{s}_{t}(l) = G^{as}_{t}(l) + \alpha. \quad (4)$$

The corresponding parameters, $\mathbf{\Delta}, \mathbf{\Gamma}$ and $\alpha$ which depend on the different phases are listed in Table 1. Furthermore

| | $n+\frac{1}{2}$ | $n$ | $n+1$ |
|---|---|---|---|
| $\Delta$ | $\Lambda$ | $\Lambda \cdot \Upsilon^{-1}$ | $\Lambda \cdot \Upsilon$ |
| $\Gamma$ | $\begin{bmatrix} A \\ B \end{bmatrix}$ | $\begin{bmatrix} A+B \\ 0 \end{bmatrix}$ | $\begin{bmatrix} A-B \\ 0 \end{bmatrix}$ |
| $\alpha$ | $(-1)^n \frac{1}{\sqrt{2}} g(k, t+N)$ | 0 | 0 |

$$\Lambda = \begin{bmatrix} \cos\frac{(n+\frac{1}{2})\pi}{N} & \sin\frac{(n+\frac{1}{2})\pi}{N} \\ -\sin\frac{(n+\frac{1}{2})\pi}{N} & \cos\frac{(n+\frac{1}{2})\pi}{N} \end{bmatrix} \quad \Upsilon = \begin{bmatrix} \cos\frac{\pi}{N} & \sin\frac{\pi}{N} \\ -\sin\frac{\pi}{N} & \cos\frac{\pi}{N} \end{bmatrix}$$

$$A = -\frac{1}{\sqrt{2}} g(k, t); \qquad B = (-1)^n g(k, t+N)$$

Table 1: Components of different phases in (3) and (4)

the unified equation can be realized by one programmable module as shown in Fig. 7 and the switches are set depending on different phases.

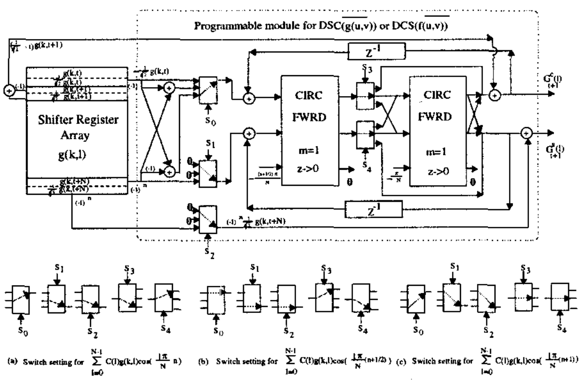

Programmable module for DSC(g(u,v)) or DCS(f(u,v))
Shifter Register Array g(k,l)
CIRC FWRD m=1 z->0
CIRC FWRD m=1 z->0
$z^{-1}$
$z^{-1}$
$S_0$
$S_1$
$S_2$
$S_3$
$S_4$
(a) Switch setting for $\sum_{l=0}^{N-1} C(l)g(k,l)\cos(\frac{l\pi}{N}n)$
(b) Switch setting for $\sum_{l=0}^{N-1} C(l)g(k,l)\cos(\frac{l\pi}{N}(n+1/2))$
(c) Switch setting for $\sum_{l=0}^{N-1} C(l)g(k,l)\cos(\frac{l\pi}{N}(n+1))$

Figure 7: Programmable module

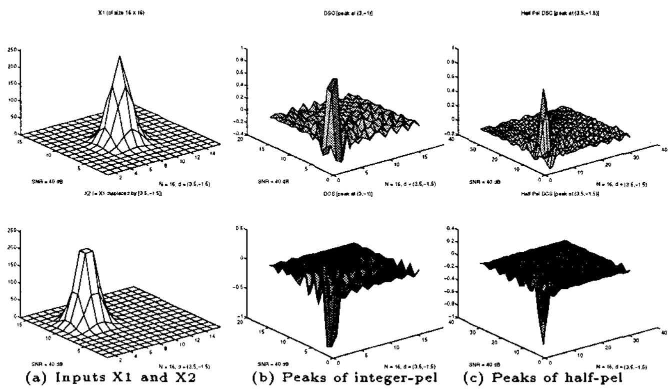
$N+3$ such programmable modules can be used for parallel computing of $\overline{DSC}(u, v)$ for different channels.

### D. Hardware cost and Simulation Results

Based on the above discussion, the hardware cost and throughput of each stage in Fig. 1 are summarized in Table 2. The structure is a scalable design that uses $26N-2$ CORDIC processors, $14N$ adders to perform an integer-pel motion estimation, and it requires additional $2N+6$ CORDICs, $6N+18$ adders to perform half-pel motion estimation. In Fig. 8 (a), we move the image $X1$ in the direction (3.5,-1.5) corresponding to image $X2$ with additive Gaussian noise at SNR=40dB. Our simulation of the designed CORDIC-DXT-ME shows that it estimates the correct integer-pel motion (3,-1) shown in Fig. 8 (b) and half-pel motion (3.5,-1.5)

| Component | CORDICs | Adders | Complexity | Throughput |
|---|---|---|---|---|
| 2D-DXT/IDXT | 6N | 12N+2 | $O(N^2)$ | $O(N)$ |
| Conversion | 4N | 0 | $O(N^2)$ | $O(N)$ |
| Pseudo Phase | 16N-2 | 2N-2 | $O(N^2)$ | $O(N)$ |
| Peak Searching | 0 | 0 | $O(N^2)$ | $O(N)$ |
| Half-pel Estimator | 2N+6 | 6N+18 | $O(N^2)$ | $O(N)$ |
| Total | 28N+4 | 20N+18 | $O(N^2)$ | $O(N)$ |

Table 2: Hardware cost and throughput



(a) Inputs X1 and X2 (b) Peaks of integer-pel (c) Peaks of half-pel

Figure 8: CORDIC-HDXT-ME estimates a movement (3.5, -1.5) with additive white Gaussian noise at SNR = 40 dB

in Fig. 8 (c).

## 3. CONCLUSION

The CORDIC-HDXT-ME architecture can estimate motion with half-pel accuracy without interpolation of input images to avoid the deterioration of the estimation accuracy. Meanwhile the structure has flexibility and scalability by using $26N-2$ CORDIC processors, $14N$ adders to get an integer-pel motion estimation and additional $2N+6$ CORDICs, $6N+18$ adders to obtain half-pel motion estimation. Finally the CORDIC-HDXT-ME architecture efficiently combines both the DCT and motion estimation into the Fully DCT-Based Video Coder. As a result, it achieves higher throughput $O(N)$ and lower system complexity $O(N^2)$.

## REFERENCES

[1] K. Ishihara and *et al.*, "A half-pel precision MPEG2 motion-estimation processor with concurrent three-vector search," *IEEE J. Solid-State Circuits*, vol. 30, pp. 1502–1509, Dec. 1995.

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

[3] U. V. Koc and K. J. R. Liu, "DCT-based subpixel motion estimation," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, (Altanta, GA), pp. 1930–1933, 1996.

[4] Y. H. Hu, "CORDIC-Based VLSI architectures for digital signal processing," *IEEE Signal Processing Magazine*, pp. 16–35, July 1992.

[5] 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. 30, pp. 1357–1377, Mar. 1993.

[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, pp. 25–37, Mar. 1992.