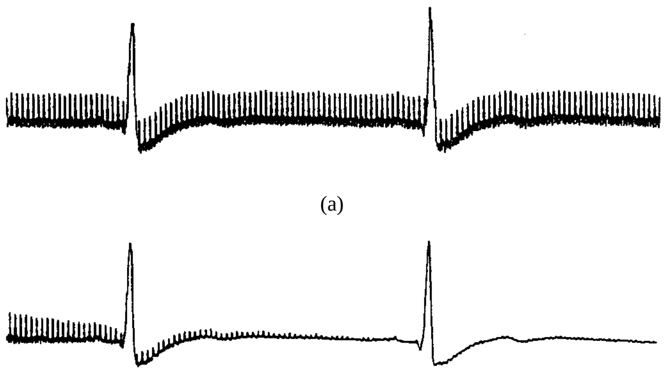


Fig. 8. Power line interference removal in ECG signal. (a) Input waveform of the comb filter. (b) Output waveform of the comb filter.

IV. Application Example

A major problem in the recording of ECG is that the measurement signals degraded by the power line interference. One source of interference is electrical field characterized by noise concentrated at the fundamental frequency 60 Hz. The other source is magnetic field which is characterized by high harmonic content. The harmonics are due to the nonlinear characteristics of transformer cores in the power supply [8]. Thus, to use comb filter to reduce interference becomes an important subject in ECG measurement.

In this example, we utilize the comb filter designed by the method in Example 4 to remove power line interference. The samples used here have 8 bits and the sampling rate is 600 Hz. Fig. 8(a) shows the input waveform that is ECG signal corrupted by harmonic interference with fundamental frequency 60 Hz. The specification of comb filter is chosen as

$$H_d(\omega) = \begin{cases} 0, & \omega = 0.2k\pi \quad k = 0, 1, \cdots, 5 \\ 1, & \text{otherwise.} \end{cases} \tag{29}$$

Fig. 8(b) shows the waveform of comb filter output with zero initial. From this result, it is obvious the interference has been removed by our comb filter except some transient states appear at the beginning.

V. Conclusion

In this paper, a new comb filter design method using fractional sample delay has been presented. First, the specification of the comb filter design is transformed into that of fractional delay filter design. Then, the FIR and allpass filter design techniques are directly used to design fractional delay filter with transformed specification. Next, we develop a constrained fractional delay filter design approach to improve the performance of the direct design method. Finally, several design examples and an experiment of the power line interference removal in ECG signal are demonstrated to illustrate the effectiveness of this new design approach.

References

[1] J. A. Van Alste and T. S. Schilder, "Removal of based-line wander and power-line interference from the ECG by an efficient FIR filter with reduced number of taps," *IEEE Trans. Biomed. Eng.*, vol. BME-32, pp. 1052–1060, Dec. 1985.

[2] J. D. Wang and H. J. Trussell, "Adaptive harmonic noise cancellation with an application to distribution power line communication," *IEEE Trans. Commun.*, vol. 36, pp. 875–884, July 1988.

[3] A. Nehorai and B. Porat, "Adaptive comb filtering for harmonic signal enhancement," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-24, pp. 1124–1138, Nov. 1986.

[4] Y. K. Jang and J. F. Chicharo, "Adaptive IIR comb filter for harmonic signal cancellation," *Int. J. Electron.*, vol. 75, pp. 241–250, 1993.

[5] S. C. Pei and C. C. Tseng, "Elimination of AC interference in electrocardiogram using IIR notch filter with transdient suppression," *IEEE Trans. Biomed. Eng.*, vol. 42, pp. 1128–1132, Nov. 1995.

[6] T. I. Laakso, V. Valimaki, M. Karjalainen, and U. K. Laine, "Splitting the unit delay: Tools for fractional delay filter design," *IEEE Signal Processing Mag.*, pp. 30–60, Jan. 1996.

[7] M. Lang and T. I. Laakso, "Simple and robust method for the design of allpass filters using least squares phase error criterion," *IEEE Trans. Circuits Syst. II*, vol. 41, pp. 40–48, Jan. 1994.

[8] C. D. McManus, D. Neubert, and E. Crama, "Characterization and elimination of AC noise in electrocardiograms: A comparison of digital filtering methods," *Comput. Biomed. Res.*, vol. 26, pp. 48–67, 1993.

A Complete Pipelined Parallel CORDIC Architecture for Motion Estimation

Jie Chen and K. J. Ray Liu

***Abstract*—In this paper, a novel fully pipelined parallel CORDIC architecture is proposed for motion estimation. Unlike other block matching structures, it estimates motion in the discrete cosine transform (DCT) transform domain instead of the spatial domain. As a result, it achieves high system throughput and low hardware complexity as compared to the conventional motion estimation design in MPEG standards. That makes the proposed architecture very attractive in real-time high-speed video communication. Importantly, the DCT-based nature enables us not only to efficiently combine DCT and motion estimation units into a single component but also to replace all multiply-and-add operations in plane rotation by CORDICs to gain further savings in hardware complexity. Furthermore this multiplier-free architecture is regular, modular, and has solely local connection suitable for VLSI implementation. The goal of the paper is to provide a solution for MPEG compatible video codec design on a dedicated single chip.**

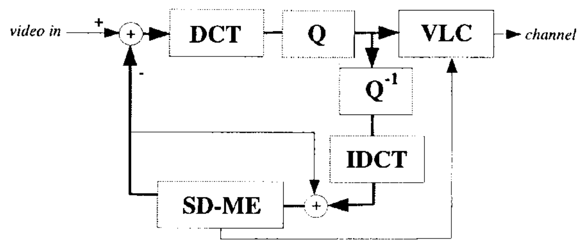
I. Introduction

Because of the simplicity of the block matching motion estimation (BKM-ME), it has been adopted in MPEG and H.263 standards. However, the computational complexity of BKM-ME is very high, i.e. $O(N^4)$ for a $N \times N$ block, hence high hardware complexity. To reduce the computational complexity, some simplified block search methods (such as logarithmic search, three-step search, etc.) and the corresponding structures have been proposed. Those methods pick several displacement candidates out of all possible displacement values in terms of minimum mean absolute difference values of the reduced number of pixels and still require two or more sequential steps to find suboptimal estimates. A good review paper about VLSI architectures for video compression can be found in [1]. Besides

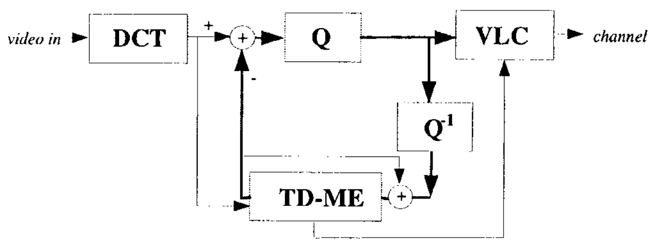
Manuscript received October 17, 1997; revised February 27, 1998. This work was supported in part by the Office of Naval Research under Grant N00014-93-10566 and by the National Science Foundation under NYI Award MIP9457397. This paper was recommended by Guest Editors F. Maloberti and W. C. Siu.

The authors are with the Electrical Engineering Department and Institute for Systems Research, University of Maryland, College Park, MD 20742 USA.

Publisher Item Identifier S 1057-7130(98)03964-0.



(a)



(b)

Fig. 1. Coder structures: SD-ME—spatial-domain motion estimation. TD-ME—Transform-domain motion estimation. (a) Conventional hybrid coder structure. (b) Fully DCT-based coder structure.

the block-based approaches, the manipulation of video data in the DCT domain has been recognized important in many advanced video applications [2]. Many manipulation functions can be performed in the DCT domain more efficiently than in the spatial domain due to a much lower data rate. Therefore, how to design and implement such a system in a cost-effective way have become a big challenge to us.

In this paper, a novel fully pipelined parallel CORDIC [3] architecture (CORDIC-DXT-ME) is proposed. Unlike any block search motion estimation structures, the CORDIC-DXT-ME estimates motions in the DCT transform domain instead of the spatial domain. As a result, it achieves high system throughput and low hardware complexity as compared to the conventional motion estimation design shown in Fig. 1(a) in MPEG, H.263 standards. The throughput of the conventional coder is limited by the processing speed of the feedback loop which is the major bottleneck of the entire digital video system for high-speed real-time applications. On the other hand, the CORDIC-DXT-ME works solely in the DCT transform domain so that we can move DCT/IDCT out of the loop as shown in Fig. 1(b). Now the performance-critical feedback loop of DCT-based coder contains only transform-domain motion estimation unit (TD-ME) instead of three major components (DCT, IDCT, SD-ME). This not only reduces the complexity of the coder but also achieves higher system throughput. In addition, the DCT-based nature of CORDIC-DXT-ME enables us not only to efficiently combine the two major processing components, the DCT and motion estimation, into one single component of relatively low complexity but also to replace all multiply-and-add operations in plane rotation with CORDIC processors. CORDIC with simple shift-and-add is more efficient in evaluating trigonometric functions and hyperbolic transformations, it is extremely simple and quite compact to realize while being no slower than the bit serial multipliers widely proposed for VLSI array structures. Those major advantages along with the regular, modular and only local connected properties of the proposed architecture make the MPEG, H.263 compatible real-time video codec design on a single dedicated chip feasible.

In the next section, we will describe the CORDIC-DXT-ME design in detail. We then present simulation results in Section III to show the performance of the proposed architecture. Finally, this paper is concluded in Section IV.

II. CORDIC-DXT-ME Design

As well known, Fourier transform (FT) of a signal $x(t)$ is related to FT of its shifted (or delayed if t represents time) version, $x(t-\tau)$, by this equation:

$$\mathcal{F}\{x(t-\tau)\} = e^{-j\omega\tau}\mathcal{F}\{x(t)\}, \tag{1}$$

where $\mathcal{F}\{\cdot\}$ denotes Fourier transform. The phase of the FT of the shifted signal contains the information about the amount of shift τ, which can easily be extracted. However, the discrete cosine transform (DCT) or its counterpart, the discrete sine transform (DST), does not have any phase components as usually found in discrete Fourier transform (DFT), but this shift information is actually preserved in the *pseudo phases* of DCT (or DST) coefficients of moving images [4]. Therefore, DCT pseudo-phase technique [4] can exact motion displacement directly from the consecutive images and results in low computational complexity $O(N^2)$ as opposed to $O(N^4)$ for BKM-ME. Based on this technique, we design a fully pipelined parallel CORDIC-DXT-ME architecture. It employs a highly hierarchical and modular design. The actual design proceeds in a bottom–up manner, but for clarity of presentation, we have used the top–down method of description.

A two-dimensional image, depending upon the application, will have a certain number of pixels in every row and column. For the sake of simplicity, take an image whose pixels are encoded in gray-scales as an example. Each pixel typically would have 256 gray levels represented by an 8-b word. For complicated video sequences in which objects may move across the border of blocks in nonuniform background, preprocessing can be employed to enhance the features of motion objects and keeps the computational complexity of the overall motion estimation low [4]. Under that circumstance, the CORDIC-DXT-ME takes either edge extraction or frame differentiation of the consecutive frames as input frame. The input frame is further divided into blocks of $N \times N$ pixels, and the motion estimation is computed for each of the blocks. The input data is fed in serially to CORDIC-DXT-ME staring from the top left pixel to the top right pixel, and then on to the next row, and so on for every block, until the entire image is estimated.

The assembled block diagram of CORDIC-DXT-ME is outlined in Fig. 2. It has four major processing stages:

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 $X_t^{ss}(k,l)$:

$$X_t^{cc}(k,l) = \frac{4}{N^2}C(k)C(l)\sum_{m,n=0}^{N-1} x_t(m,n) \cdot \cos\left[\frac{k\pi}{N}(m+0.5)\right]\cos\left[\frac{l\pi}{N}(n+0.5)\right], \quad \text{for } k,l \in \{0,\cdots,N-1\} \tag{2}$$

$$X_t^{cs}(k,l) = \frac{4}{N^2}C(k)C(l)\sum_{m,n=0}^{N-1} x_t(m,n)\cos \cdot \left[\frac{k\pi}{N}(m+0.5)\right]\sin\left[\frac{l\pi}{N}(n+0.5)\right], \quad \text{for } k \in \{0,\cdots,N-1\}, l \in \{1,\cdots,N\} \tag{3}$$

$$X_t^{sc}(k,l) = \frac{4}{N^2}C(k)C(l)\sum_{m,n=0}^{N-1} x_t(m,n) \cdot \sin\left[\frac{k\pi}{N}(m+0.5)\right]\cos\left[\frac{l\pi}{N}(n+0.5)\right], \quad \text{for } k \in \{1,\cdots,N\}, l \in \{0,\cdots,N-1\} \tag{4}$$

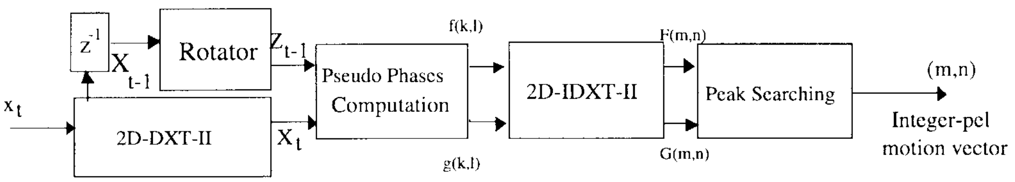


Fig. 2. The assembled block diagram of CORDIC-DXT-ME.

$$X_t^{ss}(k,l) = \frac{4}{N^2}C(k)C(l)\sum_{m,n=0}^{N-1} x_t(m,n) \cdot \sin\left[\frac{k\pi}{N}(m+0.5)\right]\sin\left[\frac{l\pi}{N}(n+0.5)\right], \quad \text{for } k,l \in \{1,\cdots,N\} \tag{5}$$

where

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

Meanwhile, the $N \times N$ *type II DCT/DST* coefficients of the previous frame 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 $Z_{t-1}^{ss}(k,l)$ defined similarly to *type II* kernels in (2)–(5) such as

$$Z_{t-1}^{cs}(k,l) = \frac{4}{N^2}C(k)C(l)\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} x_{t-1}(m,n) \cdot \cos\left[\frac{k\pi}{N}(m)\right]\sin\left[\frac{l\pi}{N}(n)\right], \quad k \in \{0,\cdots,N\}, \quad l \in \{1,\cdots,N-1\}. \tag{6}$$

2) The pseudophases computation module utilizes all the previous computed parameters and takes $O(N)$ times to produce two pseudo-phase functions $f(k,l)$ and $g(k,l)$, which will be discussed in details later.
3) Then $f(k,l)$ and $g(k,l)$ undergo *type II inverse IDCT/IDST* (2D-IDXT-II) transform to generate $F(m,n)$ and $G(m,n)$ in view of the orthogonal property

$$\begin{aligned} F(m,n) &= IDCSTII(f(k,l)) \\ &= \frac{4}{N^2}\sum_{k=0}^{N-1}\sum_{l=1}^{N} C(k)C(l)f(k,l) \cdot \cos\frac{k\pi}{N}\left(m+\frac{1}{2}\right)\sin\frac{l\pi}{N}\left(n+\frac{1}{2}\right) \\ &= [\delta(m-m_u)+\delta(m+m_u+1)] \cdot [\delta(n-m_v)-\delta(n+m_v+1)] \end{aligned} \tag{7}$$

$$\begin{aligned} G(m,n) &= IDSCTII(g(k,l)) \\ &= \frac{4}{N^2}\sum_{k=1}^{N}\sum_{l=0}^{N-1} C(k)C(l)g(k,l) \cdot \sin\frac{k\pi}{N}\left(m+\frac{1}{2}\right)\cos\frac{l\pi}{N}\left(n+\frac{1}{2}\right) \\ &= [\delta(m-m_u)-\delta(m+m_u+1)] \cdot [\delta(n-m_v)+\delta(n+m_v+1)] \end{aligned} \tag{8}$$

where m_v, m_u are the peak positions used in determining the integer-pel motion vectors.
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).

In what follows, we describe the detailed design of each block given in Fig. 2.

A. Time-Recursive 2D-DXT/IDXT-II Programmable Module and Type Transformation Module

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 generateDCT and DST outputs, which can be used in computing pseudo phases later. The time-recursive updating of 1D-DXT-II is given by (9) at the bottom of the page. Here the time index t in $X_t^c(k)$ and $X_t^s(k)$ denotes that the transform starts from $x(t)$ and ends with $x(t+N)$. Both conventional MAC-based (multiplication and accumulation) lattice structure and its corresponding CORDIC-based architecture are shown in Fig. 3. It is well known that the orthogonal rotation of the lattice structure in (9) is numerically stable so that the roundoff errors will not be accumulated. This nice numerical property is very useful in finite-precision implementation. The multiplications in the plane rotation encircled by dot-boxes in Fig. 3(a) are replaced by CORDIC processors shown in Fig. 3(b).

The type II one-dimensional time-recursive inverse IDCT/IDST (1D-IDXT-II) is defined as

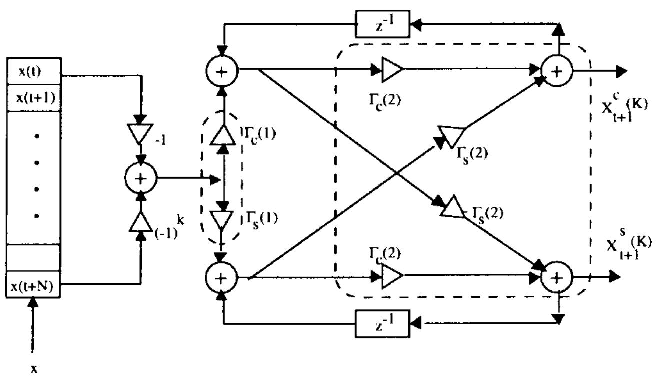
$$x_t^c(n) = \sum_{n=t}^{t+N-1} C(k-t)X(k)\cos\left[\frac{(k-t)(2n+1)\pi}{2N}\right], \quad n \in \{0,\cdots,N-1\}$$

$$x_t^s(n) = \sum_{n=t+1}^{t+N} C(k-t)X(k)\sin\left[\frac{(k-t)(2n+1)\pi}{2N}\right], \quad n \in \{0,\cdots,N-1\}$$

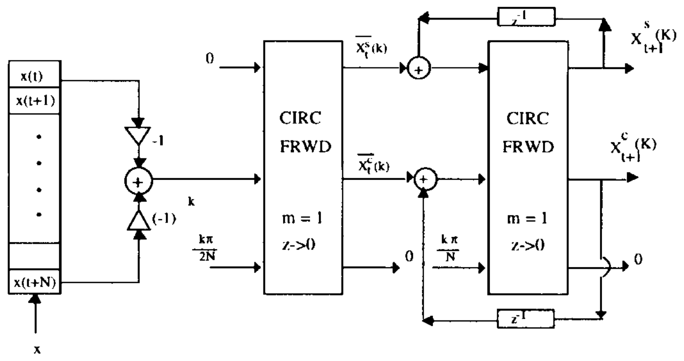
and the time-recursive updating is given by

$$\begin{bmatrix} x_{t+1}^c(n) \\ x_{t+1}^{as}(n) \end{bmatrix} = \begin{bmatrix} \cos\left(\frac{2n+1}{2N}\right)\pi & \sin\left(\frac{2n+1}{2N}\right)\pi \\ -\sin\left(\frac{2n+1}{2N}\right)\pi & \cos\left(\frac{2n+1}{2N}\right)\pi \end{bmatrix} \cdot \left[\begin{bmatrix} x_t^c(n) \\ x_t^{as}(n) \end{bmatrix} + \begin{bmatrix} -\frac{1}{\sqrt{2}}X(t) \\ (-1)^n X(t+N) \end{bmatrix}\right] + \begin{bmatrix} \left(\frac{1}{\sqrt{2}}-1\right)X(t+1) \\ 0 \end{bmatrix}. \tag{10}$$

$$\begin{bmatrix} X_{t+1}^c(k) \\ X_{t+1}^s(k) \end{bmatrix} = \begin{bmatrix} \cos\frac{k\pi}{N} & \sin\frac{k\pi}{N} \\ -\sin\frac{k\pi}{N} & \cos\frac{k\pi}{N} \end{bmatrix} \cdot \left[\begin{bmatrix} X_t^c(k) \\ X_t^s(k) \end{bmatrix} + \begin{bmatrix} \cos\frac{k\pi}{2N} & \sin\frac{k\pi}{2N} \\ -\sin\frac{k\pi}{2N} & \cos\frac{k\pi}{2N} \end{bmatrix} \cdot \begin{bmatrix} \frac{2}{N}(-x(t)+(-1)^k x(t+N)) \\ 0 \end{bmatrix}\right]. \tag{9}$$



(a)



(b)

Fig. 3. 1D-DXT-II structure with $\Gamma_c(m) = \cos(mk\pi/2N)$ and $\Gamma_s(m) = \sin(mk\pi/2N)$. (a) lattice structure. (b) corresponding CORDIC structure.

The purpose to introduce the auxiliary variable $x_t^{as}(n)$ in (10) is to maintain the same lattice structure as in 1D-DXT-II case and $x_t^{as}(n)$ is defined as

$$x_t^{as}(n) = \sum_{k=t}^{t+N-1} C(k-t)X(k)\sin\left[\frac{(k-t)(2n+1)\pi}{2N}\right], \quad n \in \{0,\cdots,N-1\}. \tag{11}$$

The auxiliary variable $x_t^{as}(n)$ is related to $x_t^s(n)$ by

$$x_t^s(n) = x_t^{as}(n) + (-1)^n \frac{1}{\sqrt{2}} X(t+N), \qquad n \in \{0,\cdots,N-1\}.$$

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 inputs and the rotation angles. Therefore, we can integrate both modules into one programmable module as shown in Fig. 4. The setting of switches $\boldsymbol{S} \doteq [s_0 s_1 s_2 s_3 s_4 s_5 s_6]$ in Fig. 4 is for the 1D-DXT-II and the complementary setting is for its inverse 1D-IDXT-II. N such module can be used for parallel computing 1D-DXT/IDXT-II coefficients for different channels.

Based on the same approach as in one-dimensional case, we extend and modify the two-dimensional design in [6] to interleave 2D-DXT-II with its inverse 2D-IDXT-II as shown in Fig. 5. The following is an example to calculate DCCT, DSCT, DCST and DSST coefficients. When row data vectors of $N \times N$ block of pixels arrive, the left 1D-DXT/IDXT Array in Fig. 5 performs DCT and DST transform on them. After N clock cycles, the 1D-DCT-II coefficients are available and stored in the upper Circular Shift Matrix meanwhile its 1D-DST-II counterpart are stored in the lower Matrix. The operations of the right 1D-DXT/IDXT Array and Shift Register Array in Fig. 5 are the same as those in the left. Therefore, after $2N$ clock cycles, the 2D-DXT-II coefficients can be simultaneously generated. By resetting the switches of the programmable module in Fig. 4, which corresponds to those modules for different channels in 1D-DXT/IDXT Array in Fig. 5, we can also simultaneously generate the inverse 2D-IDXT-II coefficients. The multiplexer (MUX) in Fig. 5 is used to control the input data flow.

A closer look at the right-hand side of (7) and (8) tells us that both $F(m,n)$ and $G(m,n)$ can be dually generated by using the time-recursive structure in Fig. 5. We can use MUX to select N of $f(k,l), l \in \{1,\cdots,N\}$, followed by $g(k,l), l \in \{0,\cdots,N-1\}$, alternatively for different k. As a result, the left 1D-DXT/IDXT Array in Fig. 5 generates one-dimensional inverse $IDST(f(k,l))$ which enter the lower Circular Shift Matrix and then $IDCT(g(k,l))$ which enter the upper Matrix in turn. The two-dimensional $IDCST(f(k,l))$ and $IDSCT(g(k,l))$ can be obtained independently afterwards. Therefore, from implementation point of view, we can interleave the 2D-DXT-II module used in the first stage with the 2D-IDCT-II module in the third stage of Fig. 2. Overall $X_t^{cc}(k,l), X_t^{cs}(k,l), X_t^{sc}(k,l)$ and $X_t^{ss}(k,l)$ can be simultaneously generated when the pixels of the current frame x_t are selected as input, or $F(m,n)$ and $G(m,n)$ are generated when $f(k,l)$ and $g(k,l)$ are in turn selected by MUX in Fig. 5.

Note that the *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 $Z_{t-1}^{ss}(k,l)$ required by the pseudo-phase computation in the second stage of Fig. 2 can actually be obtained by the plane rotation of the *type II* 2D-DXT-II kernels as follows:

$$\begin{bmatrix} \cos\theta\cos\phi & \cos\theta\sin\phi & \sin\theta\cos\phi & \sin\theta\sin\phi \\ -\cos\theta\sin\phi & \cos\theta\cos\phi & -\sin\theta\sin\phi & \sin\theta\cos\phi \\ -\sin\theta\cos\phi & -\sin\theta\sin\phi & \cos\theta\cos\phi & \cos\theta\sin\phi \\ \sin\theta\sin\phi & -\sin\theta\cos\phi & -\cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix} \cdot \begin{bmatrix} X_{t-1}^{cc}(k,l) \\ X_{t-1}^{cs}(k,l) \\ X_{t-1}^{sc}(k,l) \\ X_{t-1}^{ss}(k,l) \end{bmatrix} = \begin{bmatrix} Z_{t-1}^{cc}(k,l) \\ Z_{t-1}^{cs}(k,l) \\ Z_{t-1}^{sc}(k,l) \\ Z_{t-1}^{ss}(k,l) \end{bmatrix} \tag{12}$$

where $\theta = (k\pi/2N), \phi = (l\pi/2N)$. The lattice and its corresponding CORDIC structures are depicted in Fig. 6(a) and (b). N such modules can be used for parallel transformation for different channels. It requires a total of $4N$ CORDICs and takes $O(N)$ times to perform rotation from *type II* 2D-DXT-II to *type I* 2D-DXT-I.

B. Pseudo-Phase Computation and Peak Searching

In a two-dimensional space, the two pseudo-phase functions $f(k,l)$ and $g(k,l)$ are defined by taking the block boundary into consideration [4], as shown in (13) and (14) at the bottom of page 658. The pseudo-phase computation kernel, $g^{CS}(k,l)$ and $g^{SC}(k,l)$ for $k,l \in \{1,\cdots,N-1\}$ which will be discussed in details later, have two-dimensional relationship and are more complicated to solve than those at block boundary.

1) Pseudo-Phase Functions at Block Boundary: In what follows, we use $f(k,l)$ computation at the block boundary as an example. It is also applicable to $g(k,l)$ at the block boundary.

To solve $f(0,l)$ in (13) is equivalent to solve the following linear equation

$$\begin{bmatrix} Z_{t-1}^{cc}(0,l) & Z_{t-1}^{cs}(0,l) \\ -Z_{t-1}^{cs}(0,l) & Z_{t-1}^{cc}(0,l) \end{bmatrix} \begin{bmatrix} f(0,l) \\ * \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} X_t^{cs}(0,l) \\ X_t^{cc}(0,l) \end{bmatrix} \tag{15}$$

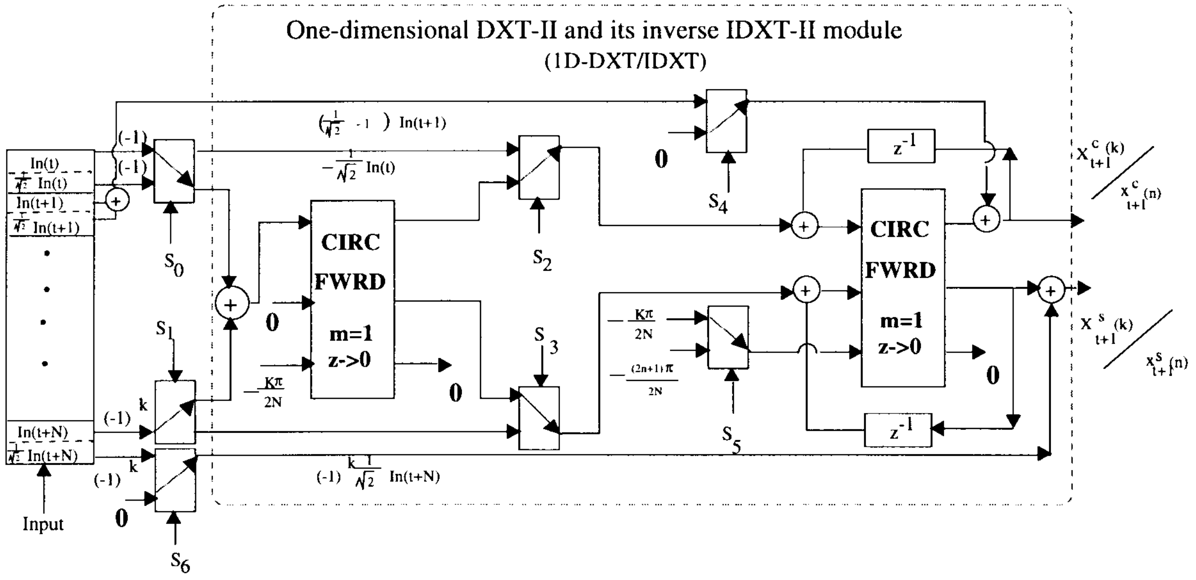


Fig. 4. 1D-DXT/IDXT-II programmable module.

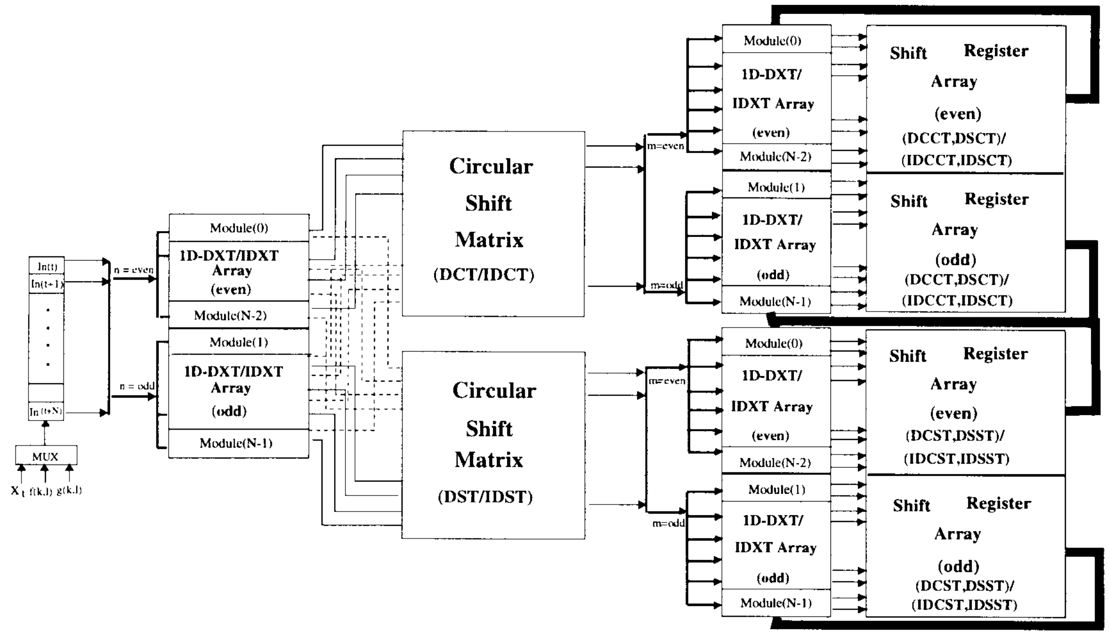

Circular Shift Matrix (DCT/IDCT)
Circular Shift Matrix (DST/IDST)
1D-DXT/IDXT Array (even)
1D-DXT/IDXT Array (odd)
Module(0)
Module(N-2)
Module(1)
Module(N-1)
Shift Register Array (even) (DCCT,DSCT)/ (IDCCT,IDSCT)
Shift Register Array (odd) (DCCT,DSCT)/ (IDCCT,IDSCT)
Shift Register Array (even) (DCST,DSST)/ (IDCST,IDSST)
Shift Register Array (odd) (DCST,DSST)/ (IDCST,IDSST)
n = even
n = odd
m=even
m=odd
MUX

Fig. 5. Time-recursive structure for 2D-DXT-II and its inverse.

where $*$ stands for *don't care*. (15) can be converted to

$$\begin{bmatrix} f(0,l) \\ * \end{bmatrix} = \frac{1}{\sqrt{2}}\frac{1}{H}\begin{bmatrix} \cos\gamma & -\sin\gamma \\ \sin\gamma & \cos\gamma \end{bmatrix}\begin{bmatrix} X_t^{cs}(0,l) \\ X_t^{cc}(0,l) \end{bmatrix} \tag{16}$$

where $H = \sqrt{(Z_{t-1}^{cc}(0,l))^2 + (Z_{t-1}^{cs}(0,l))^2}$ and $\gamma = \tan^{-1}(Z_{t-1}^{cs}(0,l)/Z_{t-1}^{cc}(0,l))$. The corresponding CORDIC structure is shown in Fig. 7. The similar approach can be applied to compute $f(k,N)$.

It is much easier to compute $f(0,N)$ by using only one CORDIC operating in *linear backward rotation mode* followed by a right shifter to halve the result.

2) Pseudo-Phase Kernel Functions: As mentioned in [4], the pseudo-phase kernel functions $f(k,l)$ and $g(k,l)$ of integer-pel displacements between previous frames x_{t-1} and current frame x_t are computed by solving the *system equation*:

$$\underbrace{\begin{bmatrix} Z_{t-1}^{cc}(k,l) & -Z_{t-1}^{cs}(k,l) & -Z_{t-1}^{sc}(k,l) & Z_{t-1}^{ss}(k,l) \\ Z_{t-1}^{cs}(k,l) & Z_{t-1}^{cc}(k,l) & -Z_{t-1}^{ss}(k,l) & -Z_{t-1}^{sc}(k,l) \\ Z_{t-1}^{sc}(k,l) & -Z_{t-1}^{ss}(k,l) & Z_{t-1}^{cc}(k,l) & -Z_{t-1}^{cs}(k,l) \\ Z_{t-1}^{ss}(k,l) & Z_{t-1}^{sc}(k,l) & Z_{t-1}^{cs}(k,l) & Z_{t-1}^{cc}(k,l) \end{bmatrix}}_{\boldsymbol{Z}_{t-1}(k,l)}$$

$$\cdot \underbrace{\begin{bmatrix} g^{CC}(k,l) \\ g^{CS}(k,l) \\ g^{SC}(k,l) \\ g^{SS}(k,l) \end{bmatrix}}_{\vec{\theta}_{m,n}(k,l)} = \underbrace{\begin{bmatrix} X_t^{cc}(k,l) \\ X_t^{cs}(k,l) \\ X_t^{sc}(k,l) \\ X_t^{ss}(k,l) \end{bmatrix}}_{\vec{x}_t(k,l)},$$

$$\text{for } k,l \in \{1,\cdots,N-1\} \tag{17}$$

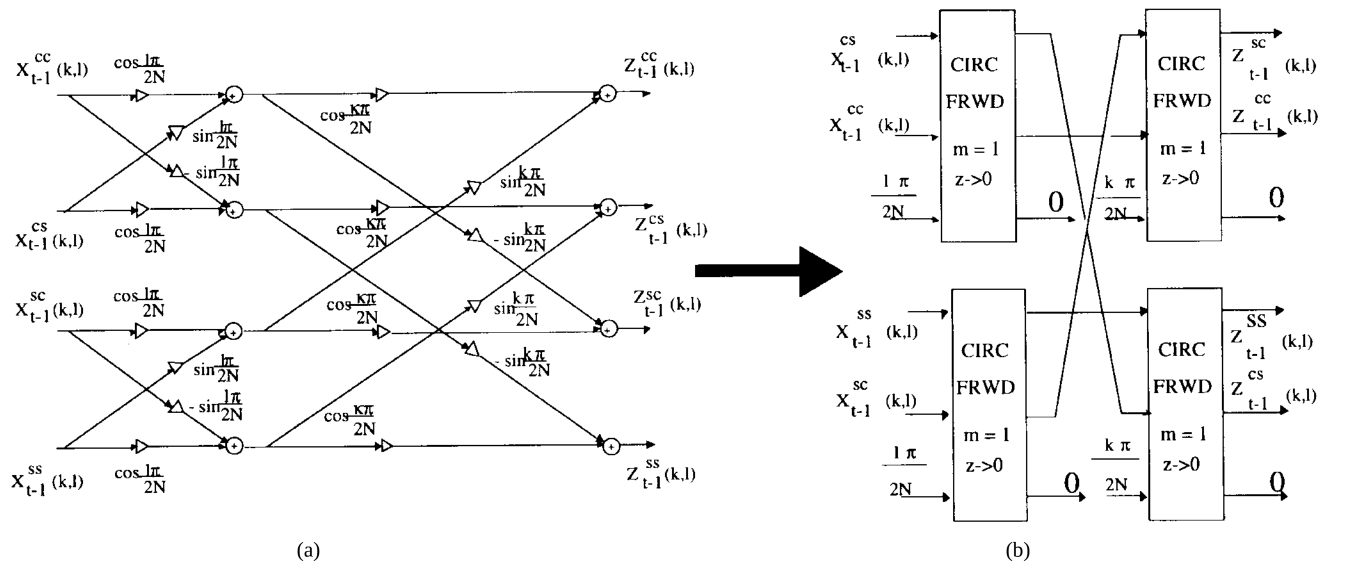


Fig. 6. The structure of *type II* to *type I* coefficients conversion. (a) Lattice structure. (b) Its CORDIC structure.

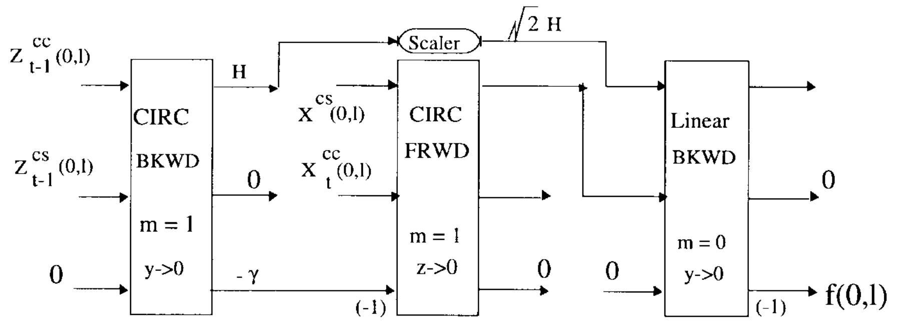


Fig. 7. The pseudo-phase $f(k,l)$ computation at block boundary.

where (m,n) is the displacement and $\vec{\theta}_{m,n}$ is the pseudo-phase vector which also equals to $[*, f(k,l), g(k,l), *]^T$. Note that $\boldsymbol{Z}_{t-1}(k,l) \in \boldsymbol{R}^{4\times 4}$ in (17). If we adopt Gauss elimination or Givens rotation method to solve for pseudo-phase $f(k,l)$ and $g(k,l)$, it incurs a heavy computational burden. However, by exploring the structure embedded in the $\boldsymbol{Z}_{t-1}(k,l)$, we can reduce the 4×4 problem in (17) to 2×2 case. Let us first introduce the following lemma which leads to an efficient way of solving (17).

Lemma 1: If $(Z^{sc}_{t-1}(k,l)/Z^{cc}_{t-1}(k,l)) = (Z^{ss}_{t-1}(k,l)/Z^{cs}_{t-1}(k,l))$, then $\boldsymbol{Z}_{t-1}(k,l)$ is an orthogonal matrix.

Proof: Let

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

$$\alpha = \tan^{-1}\left(\frac{Z^{sc}_{t-1}(k,l)}{Z^{cc}_{t-1}(k,l)}\right), \quad \beta = \tan^{-1}\left(\frac{Z^{ss}_{t-1}(k,l)}{Z^{sc}_{t-1}(k,l)}\right). \tag{18}$$

$$f(k,l) = \begin{cases} g^{CS}(k,l), & \text{for } k,l \in \{1,\cdots,N-1\} \\ \dfrac{1}{\sqrt{2}}\,\dfrac{Z^{cc}_{t-1}(k,l)X^{cs}_t(k,l) - Z^{cs}_{t-1}(k,l)X^{cc}_t(k,l)}{(Z^{cc}_{t-1}(k,l))^2 + (Z^{cs}_{t-1}(k,l))^2}, & \text{for } k=0, l \in \{1,\cdots,N-1\} \\ \dfrac{1}{\sqrt{2}}\,\dfrac{Z^{cs}_{t-1}(k,l)X^{cs}_t(k,l) + Z^{ss}_{t-1}(k,l)X^{ss}_t(k,l)}{(Z^{cs}_{t-1}(k,l))^2 + (Z^{ss}_{t-1}(k,l))^2}, & \text{for } l=N, k \in \{1,\cdots,N-1\} \\ \dfrac{1}{2}\,\dfrac{X^{cs}_t(k,l)}{Z^{cc}_{t-1}(k,l)}, & \text{for } k=0, l=N. \end{cases} \tag{13}$$

$$g(k,l) = \begin{cases} g^{SC}(k,l), & \text{for } k,l \in \{1,\cdots,N-1\} \\ \dfrac{1}{\sqrt{2}}\,\dfrac{Z^{cc}_{t-1}(k,l)X^{sc}_t(k,l) - Z^{sc}_{t-1}(k,l)X^{cc}_t(k,l)}{(Z^{cc}_{t-1}(k,l))^2 + (Z^{sc}_{t-1}(k,l))^2}, & \text{for } l=0, k \in \{1,\cdots,N-1\} \\ \dfrac{1}{\sqrt{2}}\,\dfrac{Z^{sc}_{t-1}(k,l)X^{ss}_t(k,l) + Z^{ss}_{t-1}(k,l)X^{sc}_t(k,l)}{(Z^{sc}_{t-1}(k,l))^2 + (Z^{ss}_{t-1}(k,l))^2}, & \text{for } k=N, l \in \{1,\cdots,N-1\} \\ \dfrac{1}{2}\,\dfrac{X^{sc}_t(k,l)}{Z^{cc}_{t-1}(k,l)}, & \text{for } k=N, l=0. \end{cases} \tag{14}$$

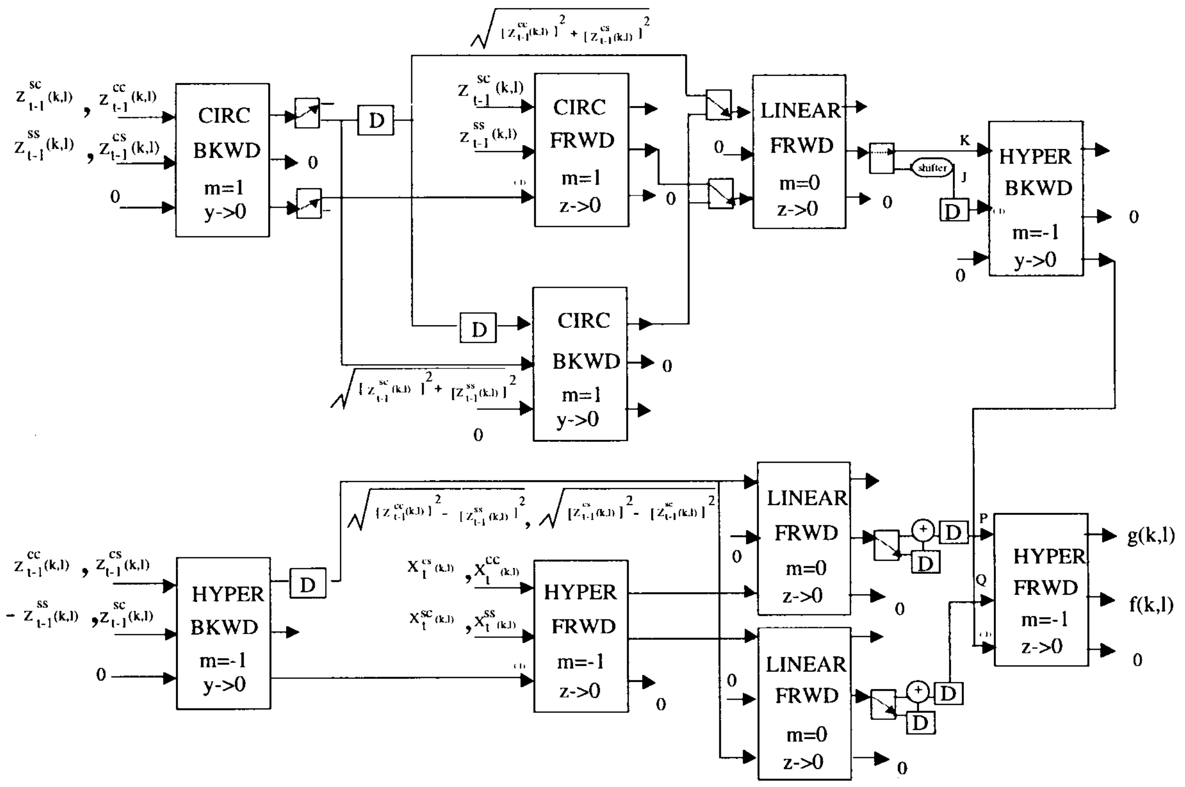


Fig. 8. Folded pipeline structure for pseudo-phase computation.

Then,

$$\frac{Z^{cc}_{t-1}(k,l)}{\sqrt{K}} = \cos\alpha\cos\beta$$
$$\frac{Z^{cs}_{t-1}(k,l)}{\sqrt{K}} = \cos\alpha\sin\beta$$
$$\frac{Z^{sc}_{t-1}(k,l)}{\sqrt{K}} = \sin\alpha\cos\beta$$
$$\frac{Z^{ss}_{t-1}(k,l)}{\sqrt{K}} = \sin\alpha\sin\beta.$$

Therefore $\boldsymbol{Z}_{t-1}(k,l)$ can be rewritten as the matrix shown at the bottom of the page. The matrix $\boldsymbol{A}$ is an orthonormal matrix because $\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{I}$. Hence,

$$\boldsymbol{Z}_{t-1}(k,l)\boldsymbol{Z}^T_{t-1}(k,l) = K\boldsymbol{I}\square$$

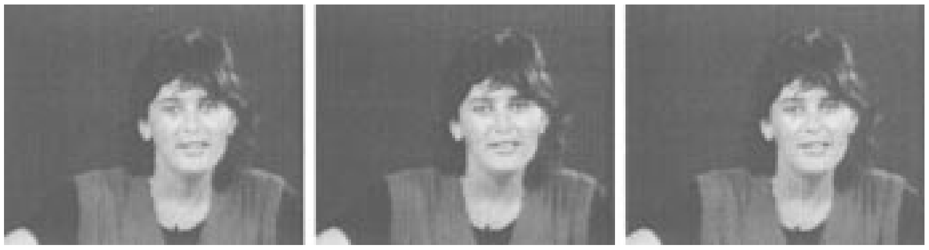
By apply the lemma to the *system equation*, we can solve for the pseudo-phase information as (19) at the bottom of the page. Both lattice and its corresponding CORDIC structures to solve for $f(k,l)$ and $g(k,l)$ in (19) are similar to those in Fig. 6.

If $(Z^{sc}_{t-1}(k,l)/Z^{cc}_{t-1}(k,l)) \neq (Z^{ss}_{t-1}(k,l)/Z^{cs}_{t-1}(k,l))$, it is more difficult to find the pseudo-phase functions. Fortunately, we are only interested in solving pseudo-phase $f(k,l)$ and $g(k,l)$, so we can multiply both sides of the *system equation* by $\boldsymbol{Z}^T_{t-1}(k,l)$ and obtain the desired reduced 2×2 equation:

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

$$\boldsymbol{Z}_{t-1}(k,l) = \sqrt{K}\underbrace{\begin{bmatrix} \cos\alpha\cos\beta & -\cos\alpha\sin\beta & -\sin\alpha\cos\beta & \sin\alpha\sin\beta \\ \cos\alpha\sin\beta & \cos\alpha\cos\beta & -\sin\alpha\sin\beta & -\sin\alpha\cos\beta \\ \sin\alpha\cos\beta & -\sin\alpha\sin\beta & \cos\alpha\cos\beta & -\cos\alpha\sin\beta \\ \sin\alpha\sin\beta & \sin\alpha\cos\beta & \cos\alpha\sin\beta & \cos\alpha\cos\beta \end{bmatrix}}_{\boldsymbol{A}}.$$

$$\begin{bmatrix} * \\ f(k,l) \\ g(k,l) \\ * \end{bmatrix} = \frac{1}{K}\underbrace{\begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta & \sin\alpha\cos\beta & \sin\alpha\sin\beta \\ -\cos\alpha\sin\beta & \cos\alpha\cos\beta & -\sin\alpha\sin\beta & \sin\alpha\cos\beta \\ -\sin\alpha\cos\beta & -\sin\alpha\sin\beta & \cos\alpha\cos\beta & \cos\alpha\sin\beta \\ \sin\alpha\sin\beta & -\sin\alpha\cos\beta & -\cos\alpha\sin\beta & \cos\alpha\cos\beta \end{bmatrix}}_{\boldsymbol{A^T}}\begin{bmatrix} X^{cc}_t(k,l) \\ X^{cs}_t(k,l) \\ X^{sc}_t(k,l) \\ X^{ss}_t(k,l) \end{bmatrix} \qquad (19)$$



(a) (b) (c)

Fig. 9. Simulation on the "Miss America" sequence. (a) Frame 90. (b) Frame 91. (c) Reconstructed frame.

$$+ \underbrace{\begin{bmatrix} Z_{t-1}^{cc}(k,l) & -Z_{t-1}^{ss}(k,l) \\ -Z_{t-1}^{ss}(k,l) & Z_{t-1}^{cc}(k,l) \end{bmatrix} \begin{bmatrix} X_t^{cs}(k,l) \\ X_t^{sc}(k,l) \end{bmatrix}}_{\Gamma_2}$$

$$\triangleq \begin{bmatrix} P \\ Q \end{bmatrix} \tag{20}$$

where K is defined in (18) and $J = 2 \times [Z_{t-1}^{cs}(k,l) Z_{t-1}^{sc}(k,l) - Z_{t-1}^{cc}(k,l) Z_{t-1}^{ss}(k,l)]$.

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

$$\begin{bmatrix} f(k,l) \\ g(k,l) \end{bmatrix} = \frac{1}{\Delta} \begin{bmatrix} K & -J \\ -J & K \end{bmatrix} \begin{bmatrix} P \\ Q \end{bmatrix}, \quad \text{where} \quad \Delta = K^2 - J^2. \tag{21}$$

From the above discussion, we observe that the computations of Γ_1 and Γ_2 defined in (20) are similar. We thus employ the folding technique [7] to reduce the number of the CORDICs in the design. The folded architecture given in Fig. 8 requires 10 CORDICs which is almost half of the number of CORDICs used in the unfolded pipeline architecture. This reduction in hardware is achieved at the expense of increasing the latency of that module. The switch settings in Fig. 8 are for computing K, Γ_1 and the complementary settings are for computing J, Γ_2. The latches, Ds in Fig. 8, driven by two nonoverlap clocks are introduced across the feed-forward cut-set not only to reduce the critical path but also to synchronize the pseudo-phase computation. Here some CORDIC processors are operated in the somewhat less often, *hyperbolic rotation modes* to evaluate P, Q and the pseudo phases $f(k,l)$ and $g(k,l)$. Overall the structure is fully pipelined and it takes $O(N)$ time to compute two pseudo-phase functions $f(k,l)$ and $g(k,l)$.

The two-dimensional search for the peak value among $F(m,n)$ and $G(m,n)$ in [4] 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. In summary, to process $N \times N$ block of pixel, the hardware cost of each component and throughput of the system in Fig. 2 are summarized in Table I.

TABLE I
HARDWARE COST AND THROUGHPUT

Stage	Component	CORDICs	Adders	Throughput
1/3	2D-DXT/IDXT-II	$6N$	$25N$	$O(N)$
1	Conversion	$4N$	0	$O(N)$
2	Pseudo Phase	$10N$	$2N$	$O(N)$
4	Peak Searching	0	0	$O(N)$
	Total	$20N$	$27N$	$O(N)$

III. SIMULATION RESULTS

Simulations have been performed on the "Miss America" sequence in QCIF format with frame size 176×144 using the proposed architecture. The original frame 90 and 91 are shown in Fig. 9(a) and (b) and the reconstructed frame 91 based on the moving vectors obtained from CORDIC-DXT-ME is shown in Fig. 9(c).

IV. CONCLUSION

In this paper, we propose a novel low-complexity high throughput CORDIC architecture (CORDIC-DXT-ME) for motion estimation. This proposed multiplier-free and fully pipelined parallel structure requires $20N$ CORDICs and $27N$ adders to process $N \times N$ block of pixels. In addition, this fully DCT-based coder exhibits accurate motion estimate. Its regular and modular structure along with only local connection makes MPEG, H.263 compatible video codec design on a dedicated single chip feasible.

REFERENCES

[1] P. Pirsch, N. Demassieux, and W. Gehrke, "VLSI architectures for video compression—A survey," *Proc. IEEE*, vol. 83, pp. 220–245, Feb. 1995.

[2] S.-F. Chang and D. G. Messerschmidt, "Manipulation and compositing of MC–DCT compressed video," *IEEE J. Select. Areas Commun.*, vol. 13, pp. 1–11, Jan. 1995.

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

[4] U.-V. Koc and K. J. R. Liu, "Discrete cosine/sine-transform based motion estimation," in *Proc. IEEE Int. Conf. on Image Processing*, Austin, TX, vol. 3, Nov. 1994, pp. 771–775; also, *IEEE Trans. Image Processing*, to be published.

[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 Syst. Video Technol.*, vol. 2, pp. 25–37, Mar. 1992.

[7] K. K. Parhi, C.-Y. Wang, and A. P. Brown, "Synthesis of control circuits in folded pipelined DSP architectures," *IEEE J. Solid-State Circuits*, vol. 27, pp. 29–43, Jan. 1992.