Low-Power Architectures for Compressed Domain Video Coding Co-Processor

Jie Chen, *Member, IEEE*, and K. J. Ray Liu, *Senior Member, IEEE*

***Abstract*—Low power as a *de facto* is one of the most important criteria for many signal-processing system designs, particularly in multimedia cellular applications and multimedia system on chip design. There have been many approaches to achieve this design goal at many different implementation levels ranging from very-large-scale-integration fabrication technology to system design. In this paper, the multirate low-power design technique will be used along with other methods such as look-ahead, pipelining in designing cost-effective low-power architectures of compressed domain video coding co-processor. Our emphasis is on optimizing power consumption by minimizing computational units along the data path. We demonstrate both low-power and high-speed can be accomplished at algorithm/architecture level. Based on the calculation and simulation results, the design can achieve significant power savings in the range of 60%–80% or speedup factor of two at the needs of users.**

***Index Terms*—Compressed domain video coding, DCT, low-power architecture, motion estimation.**

I. INTRODUCTION

IN RECENT years, the need for personal mobile communications—"anytime, anywhere" access to multimedia and communication services—has become increasingly clear. Digital cellular telephony, such as the U.S. third-generation code-division multiple access PCS and the European GSM systems, have seen rapid acceptance and growth in the marketplace. Due to the limited power-supply capability of current battery technology, low-power design to prolong the operating time of those mobile handsets becomes vital to success. On the other hand, as the VLSI fabrication technology advances, it becomes feasible to design the entire multimedia systems on a single chip—*system on chip*. However, the high power dissipation of the chip calls for extra cooling devices and expensive packages to dissipate the generated heat. It increases both the weight and cost of those systems thus the need for low-power design becomes essential. However, the development of low-power multimedia systems is still in its infancy. The low-power video coding systems achieved at *device/process level* such as low-power video coder design has been reported in [1], which uses 0.5 μm VLSI fabrication technology. As 0.25 μm and 0.18 μm CMOS technologies become mature, people tend to consider the low-power design toward that direction. Nevertheless, the cost of device/process-level approach is the most expensive among all other low-power techniques because it requires the investment of new semiconductor equipment and technology, which is beyond the budget for most small or start-up companies. Furthermore, it takes time for the fabrication technology to be mature enough for mass production and for the computer-aided design (CAD) tools to handle those "deep submicron" effects. Other than device/process-level approach, recently wide techniques are used to achieve low-power, cost effective architectures for video coding system [2]–[4]. Those designs are achieved under the current technology without investing and waiting for the new expensive devices, advanced VLSI fabrication technology and CAD tools.

In this paper, we design the low-power video coding co-processor at the *algorithm/architecture* level, which provides the most leveraged way to achieve low-power consumption when both effectiveness and cost are taken into consideration [5]. Basically, the algorithm/architecture low-power design is achieved by reformulating the algorithms and mapping them to efficient low-power VLSI architectures to compensate for the speed loss caused by lowered supply voltage. We emphasize on optimizing the power consumption of the video coding co-processor design by minimizing computational units along the data path. Let us explain our idea in more detail. The conventional hybrid motion-compensated DCT video coding structure adopted by the standards is not optimized in terms of hardware complexity because both the motion estimation and DCT/IDCT units, which consume 80% of the design [2], [6], [7], cannot be combined together into one unit. Thus, the following question can logically be posed: "Can we estimate motions also in the compressed domain so that we can optimize the power consumption by reducing the computational units?" In the category of compressed-domain motion estimation, three-dimensional fast Fourier transform (3D-FFT) has been successfully used to estimate motion in several consecutive frames [8], [9]. But this approach is not compatible with the standards because it requires processing of several frames rather than two. Moreover, the FFT operating on complex numbers is not used in any video-coding standards and the global routing structure is undesirable in VLSI design. Fortunately, the standard complied solutions, the fully DCT-based motion compensated video coding algorithms, have been provided in [10], [11]. As we all know, the phase of Fourier transform of the shifted signal encapsulates the

Manuscript received July 20, 1999; revised March 24, 2000. This work was supported in part by the Office of Naval Research Grant N00014-93-10566 and the National Science Foundation SF NYI Award MIP9457397. The associate editor coordinating the review of this paper and approving it for publication was Prof. Mohammed Ismail.

J. Chen is with Bell Laboratories, Lucent Technologies, Murray Hill, NJ 07974 USA.

K. J. R. Liu is with the Electrical Engineering Department and Institute for Systems Research, University of Maryland, College Park, MD 20742 USA (e-mail: kjrliu@eng.umd.edu).

Publisher Item Identifier S 1520-9210(00)04965-8.

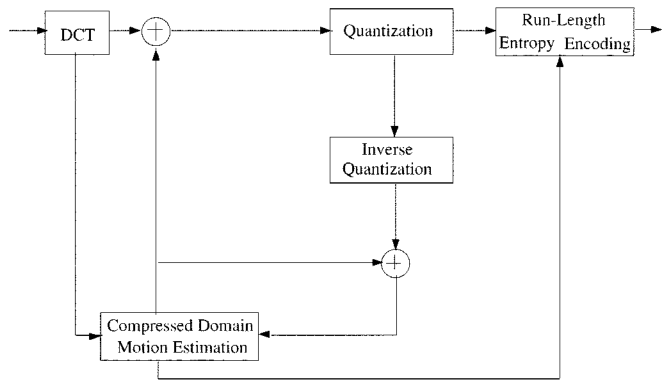

DCT
Quantization
Run-Length Entropy Encoding
Inverse Quantization
Compressed Domain Motion Estimation

Fig. 1. Fully DCT-based motion compensated video coder structure (Here motion estimation is achieved in DCT domain).

information about the shift. Based on the same argument, the authors discover that the motion information of P or B frame is actually embedded in its DCT coefficients. In other words, the motion can be extracted based on the DCT coefficients of the block in current frame (P/B) and its corresponding one in previous frame (I/P). The overall system architecture is shown in Fig. 1. The main advantages to adopt such an approach are listed as follows.

- **From the implementation viewpoint:** We can save silicon area significantly by naturally accommodating both DCT and motion estimation processors into one processing unit (based on the VLSI implementation results, the chip size of our combined design (DCT + half-pel motion estimation) under normal operating condition is smaller than or about the same as those block-matching designs alone without DCT/IDCT unit [12]). This nice property is very useful for our low-power design at algorithm/architecture level.
- **From the system delay viewpoint:** The DCT can be moved out of the feedback loop and thus the operating speed of DCT can be reduced to the data rate of the incoming video stream. Moreover, IDCT is now removed from the feedback loop thus there are only quantizers and compressed domain motion estimator in the loop. This not only reduces the complexity of the coder but also reduces the system delay without any tradeoff of performance.
- **From the algorithm viewpoint:** It reduces overall complexity significantly compared to the hybrid motion-compensated DCT video coding schemes in the standards because the overall complexity is now dominated by DCT computation instead of block matching.

Due to its DCT-based nature of the algorithm, the fully CORDIC-based (COordinate Rotation Digital Compute [13]) architectures, under normal operating condition, and its corresponding signal chip VLSI implementation have been proposed in [12], [14], [15].

In this paper, we extend the video coding architectures in [14], [15] for low-lower applications. All advantages mentioned in the CORDIC-based design, i.e., high throughput, numerical stability, multiplier-free, modular and solely local connected properties are also inherited in our low-power design. Based on the calculation and simulation results, the proposed design can be readily applied to high-speed video communication with the speedup factor of two under normal supply voltage i.e., 5 V. Or, the same design can operate at two-time slower operating frequency under lowered supply voltage (3.08 V) while retaining the original data throughput rate. It enables us to achieve significant power saving in the range of 60%–80% without sacrificing system performance (refer to the detailed discussion later). Therefore, our low-power design can smartly conquer both low-power and high-speed requirements, which are often considered to be the problems of opposite natures, at the needs of users.

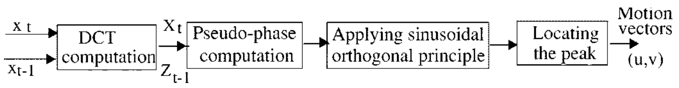


Fig. 2. Illustration of the compressed domain video co-processor design.

The multirate low-power design technique [16], [17] will be used along with other low-power design methods such as look-ahead, pipelining in our design to achieve low-power/high-speed performance. In what follows, we explain the detailed design of our compressed domain low-power video coding co-processor. Then we present the simulation results in Section III to demonstrate the performance of our design. Finally the paper is concluded in Section IV.

II. Low-Power/High-Speed Architectures

As we have pointed out earlier, the block matching approach estimates the motion by the best matching while the compressed domain approach estimates the motion by comparing the energy, in terms of the DCT coefficients, of the shifted images. Although it is not as intuitive as those block matching methods, it is helpful in understanding this scheme by considering *pseudo-phase* in compressed domain design analogous to *phase* in Fourier transform. In other words, the compressed domain approach is based on the principle that a relative shift in the spatial domain results in a linear phase shift in the Fourier domain. The proposed low-power design to realize such a compressed domain scheme has fully pipelined parallel architecture, as shown in Fig. 2. It consists of four major processing stages. Here we are only considering the combined design of DCT and motion estimation units, which serves as the computing engine or co-processor of the whole video coding system. The motion (u, v) can be estimated by taking the current and its reference blocks, x_t and x_{t-1} with the size of $N \times N$, as inputs. If these two blocks differ by a translational displacement, then the displacement can be found by locating the peak of the inverse two-dimensional DCT (2D DCT) transform of the normalized pseudo-phase function of these two blocks as follows:

$$\begin{aligned}&\text{IDCT}\{\text{normalized pseudo phases}\}\\&\qquad\text{where pseudo phases} = \mathcal{F}(X_t, Z_{t-1}) \qquad (1)\end{aligned}$$

where pseudo phases is the function of X_t and Z_{t-1}, $X_t = \text{DCT}_{\text{II}}\{x_t\}$ and $Z_{t-1} = \text{DCT}_{\text{I}}\{x_{t-1}\}$ are *type-II* and *type-I* DCT coefficients, respectively [18]. (In video coding standards, the *type-II* DCT has been used). Notice that the motion vectors

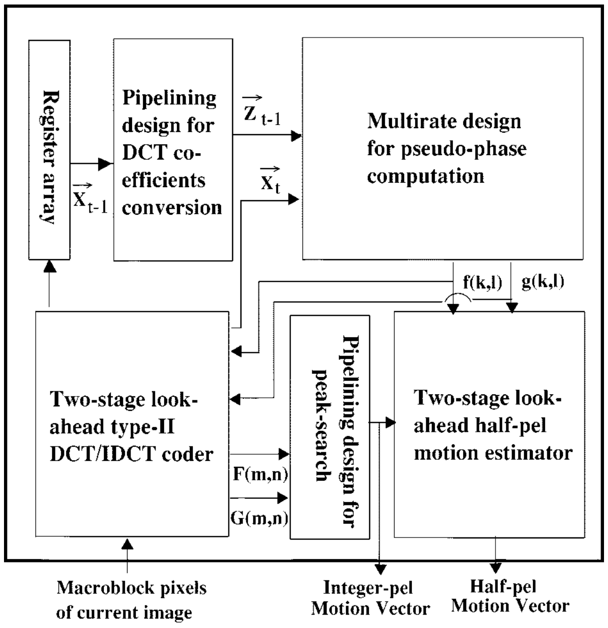

Register array
Pipelining design for DCT co-efficients conversion
$\vec{X}_{t-1}$
$\vec{Z}_{t-1}$
$\vec{X}_t$
Multirate design for pseudo-phase computation
f(k,l)
g(k,l)
Two-stage look-ahead type-II DCT/IDCT coder
Pipelining design for peak-search
Two-stage look-ahead half-pel motion estimator
F(m,n)
G(m,n)
Macroblock pixels of current image
Integer-pel Motion Vector
Half-pel Motion Vector

Fig. 3. Floor-plane of low-power compressed domain video coding co-processor.

are limited to the block size. If the motion vectors go beyond the block boundary, the motion vector of $(0,0)$ will be used, instead.

The detailed floor-plane of our low-power/high-speed video co-processor design is shown in Fig. 3. It is shown that doubling the accuracy of motion compensation from integer-pel to half-pel can reduce the bit-rate by up to 0.5 bits/sample [19], [20]. Therefore, a two-stage look-ahead half-pel motion estimator is included in our low-power design, as shown in Fig. 3. In other words, our co-processor can estimate motion at either int-pel or half-pel resolution based on the needs of users. Next, we will focus on the design of each building blocks of the compressed domain video coding co-processor.

A. Two-Stage Look-Ahead Type-II DCT/IDCT Coder

Unlike the conventional DCT coder design using matrix factorization, we adopt the time-recursive DCT [21], [22] which is able to simultaneously generate *type-II* DCT and discrete sine transform (DST) coefficients—X_t^c and X_t^s needed to compute the pseudo-phase function. Furthermore, in real-time video signal processing, the data arrive serially. The traditional transformation algorithms [23] buffer the incoming data and then perform the transformation with the complexity of $O(N\log N)$ while the time-recursive approach merges the buffering and transform operations into a single unit of total lower hardware complexity $O(N)$ (Here N is the block size). Most importantly, due to the inherent time-recursive characteristic, we can use *look-ahead* method to reduce the power consumption. In principle, the speed-up provided by look-ahead compensates the speed loss caused by reduced supply voltage at the cost of increasing hardware complexity.

1) Two-Stage Look-Ahead DCT: 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 in [21] as

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

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

where

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

Here, t is the time index.

The two-stage look-ahead time-recursive updating of DCT and DST coefficients is given by

$$\begin{aligned} X_{t+2}^c(k) &= \frac{2}{N}C(k)\sum_{n=t+2}^{t+N+1} x(n) \\ &\quad\times\cos\left[\frac{k\pi}{N}\left[(n-t-2)+\frac{1}{2}\right]\right] \\ &= \frac{2}{N}C(k)\sum_{n=t+2}^{t+N+1} x(n)\cos\left[\frac{k\pi}{N}\left[(n-t)+\frac{1}{2}\right]\right] \\ &\quad\times\cos\frac{2k\pi}{N} + \frac{2}{N}C(k)\sum_{n=t+2}^{t+N+1} x(n) \\ &\quad\times\sin\left[\frac{k\pi}{N}\left[(n-t)+\frac{1}{2}\right]\right]\sin\frac{2k\pi}{N} \end{aligned} \tag{2}$$

$$\begin{aligned} X_{t+2}^s(k) &= \frac{2}{N}C(k)\sum_{n=t+2}^{t+N+1} x(n) \\ &\quad\times\sin\left[\frac{k\pi}{N}\left[(n-t-2)+\frac{1}{2}\right]\right] \\ &= \frac{2}{N}C(k)\sum_{n=t+2}^{t+N+1} x(n)\sin\left[\frac{k\pi}{N}\left[(n-t)+\frac{1}{2}\right]\right] \\ &\quad\times\cos\frac{2k\pi}{N} - \frac{2}{N}C(k)\sum_{n=t+2}^{t+N+1} x(n) \\ &\quad\times\cos\left[\frac{k\pi}{N}\left[(n-t)+\frac{1}{2}\right]\right]\sin\frac{2k\pi}{N}. \end{aligned} \tag{3}$$

Both (2) and (3) can be combined into the following equation:

$$\begin{bmatrix} X_{t+2}^c(k) \\ X_{t+2}^s(k) \end{bmatrix} = \begin{bmatrix} \cos\frac{2k\pi}{N} & \sin\frac{2k\pi}{N} \\ -\sin\frac{2k\pi}{N} & \cos\frac{2k\pi}{N} \end{bmatrix} \begin{bmatrix} \bar{X}_{t+2}^c(k) \\ \bar{X}_{t+2}^s(k) \end{bmatrix} \tag{4}$$

where $\bar{X}_{t+2}^c(k)$ is related to $X_t^c(k)$ and $\bar{X}_{t+2}^s(k)$ is related to

$X_t^s(k)$ by

$$\begin{aligned}
\bar{X}_{t+2}^c(k) &= \frac{2}{N}C(k)\sum_{n=t+2}^{t+N+1} x(n)\cos\left[\frac{k\pi}{N}\left[(n-t)+\frac{1}{2}\right]\right]\\
&= \frac{2}{N}C(k)\sum_{n=t}^{t+N-1} x(n)\cos\left[\frac{k\pi}{N}\left[(n-t)+\frac{1}{2}\right]\right]\\
&\quad+\frac{2}{N}C(k)x(t+N+1)\cos\left[\frac{k\pi}{N}\left(N+\frac{3}{2}\right)\right]\\
&\quad+\frac{2}{N}C(k)x(t+N)\cos\left[\frac{k\pi}{N}\left(N+\frac{1}{2}\right)\right]\\
&\quad-\frac{2}{N}C(k)x(t+1)\cos\left[\frac{3k\pi}{2N}\right]\\
&\quad-\frac{2}{N}C(k)x(t)\cos\left[\frac{k\pi}{2N}\right]\\
&= X_t^c(k)+\frac{2}{N}C(k)[-x(t)+(-1)^k x(t+N)]\\
&\quad\times\cos\frac{k\pi}{2N}+\frac{2}{N}C(k)[-x(t+1)\\
&\quad+(-1)^k x(t+N+1)]\cos\frac{3k\pi}{2N},\\
\bar{X}_{t+2}^s(k) &= \frac{2}{N}C(k)\sum_{n=t+2}^{t+N+1} x(n)\sin\left[\frac{k\pi}{N}\left[(n-t)+\frac{1}{2}\right]\right]\\
&= \frac{2}{N}C(k)\sum_{n=t}^{t+N-1} x(n)\sin\left[\frac{k\pi}{N}\left[(n-t)+\frac{1}{2}\right]\right]\\
&\quad+\frac{2}{N}C(k)x(t+N+1)\sin\left[\frac{k\pi}{N}\left(N+\frac{3}{2}\right)\right]\\
&\quad+\frac{2}{N}C(k)x(t+N)\sin\left[\frac{k\pi}{N}\left(N+\frac{1}{2}\right)\right]\\
&\quad-\frac{2}{N}C(k)x(t+1)\sin\left[\frac{3k\pi}{2N}\right]\\
&\quad-\frac{2}{N}C(k)x(t)\sin\left[\frac{k\pi}{2N}\right]\\
&= X_t^s(k)+\frac{2}{N}C(k)[-x(t)+(-1)^k x(t+N)]\\
&\quad\times\sin\frac{k\pi}{2N}+\frac{2}{N}C(k)[-x(t+1)\\
&\quad+(-1)^k x(t+N+1)]\sin\frac{3k\pi}{2N}.
\end{aligned}\tag{5}$$

Based on the above derivations, we can combine those equations together and get

$$\left\{\begin{aligned}
\begin{bmatrix} X_{t+2}^c(k)\\ X_{t+2}^s(k)\end{bmatrix} &= \begin{bmatrix}\cos\frac{2k\pi}{N} & \sin\frac{2k\pi}{N}\\ -\sin\frac{2k\pi}{N} & \cos\frac{2k\pi}{N}\end{bmatrix}\\
&\quad\times\left[\begin{bmatrix} X_t^c(k)\\ X_t^s(k)\end{bmatrix}+\begin{bmatrix}\bar{X}_t^c(k)\\ \bar{X}_t^s(k)\end{bmatrix}\right]\\
\begin{bmatrix}\bar{X}_t^c(k)\\ \bar{X}_t^s(k)\end{bmatrix} &= \frac{2}{N}C(k)\begin{bmatrix}\cos\frac{k\pi}{2N} & \cos\frac{3k\pi}{2N}\\ \sin\frac{k\pi}{2N} & \sin\frac{3k\pi}{2N}\end{bmatrix}\\
&\quad\times\begin{bmatrix} -x(t)+(-1)^k x(t+N)\\ -x(t+1)+(-1)^k x(t+N+1)\end{bmatrix}.
\end{aligned}\right.\tag{6}$$

The following illustrates how this dually generated DCT and DST lattice structure works to obtain the DCT and DST with a series of input data $[x(t), x(t+1), \ldots, x(t+N), x(t+N+1), \ldots]$ for a specific k. The initial values of the transformed signals $X_t^c(k)$ and $X_t^s(k)$ are set to zero so are the initial values in the shift register in the front of the lattice module, as shown in Fig. 4. In the high-speed image system such as HDTV, digitized images are available in a sequential or stream fashion. In the conventional approaches, the serial data is buffered and then transformed. Waiting for data to become ready will cause additional delay, which is not desirable for real-time service. In our time-recursive design, those input sequence $[x(t), x(t+1), \ldots]$ shifts sequentially into the shift register. Then the output signals $X_c^{t+2}(k)$ and $X_s^{t+2}(k)$, $k = 0, 1, \ldots, N-1, N$, are updated recursively according to (6). The multiplications in the plane rotation in (6) are replaced by three CORDIC processors. After the input datum $x(t+N+1)$ shifts into the shift register, the DCT and DST coefficients are dually obtained at the output for this index k. To improve the throughput and reduce the latency, a parallel lattice array consists of N such lattice modules can be used for parallel computations.

2) Two-Stage Look-Ahead Inverse DCT: The *type-II* inverse IDCT is defined as

$$\begin{aligned}
x_t^c(n) &= \frac{2}{N}\sum_{k=t}^{t+N-1} C(k-t)X(k)\\
&\quad\times\cos\left[\left(\frac{2n+1}{2N}\right)(k-t)\pi\right];\quad n\in\{0,\ldots,N-1\}.
\end{aligned}$$

The two-stage look-ahead time-recursive updating of IDCT coefficients is given by

$$\begin{aligned}
x_{t+2}^c(n) &= \frac{2}{N}\sum_{k=t+2}^{t+N+1} C(k-t-2)X(K)\\
&\quad\times\cos\left[\left(\frac{2n+1}{2N}\right)(k-t-2)\pi\right]\\
&= \frac{2}{N}\sum_{k=t+2}^{t+N+1} C(k-t-2)X(K)\\
&\quad\times\cos\left[\left(\frac{2n+1}{2N}\right)(k-t)\pi\right]\cos\left[\frac{2n+1}{N}\pi\right]\\
&\quad+\frac{2}{N}\sum_{k=t+2}^{t+N+1} C(k-t-2)X(K)\\
&\quad\times\sin\left[\left(\frac{2n+1}{2N}\right)(k-t)\pi\right]\sin\left[\frac{2n+1}{N}\pi\right].
\end{aligned}\tag{7}$$

We can rewrite (7) as follows:

$$\begin{bmatrix} x_{t+2}^c(n)\\ x_{t+2}^{as}(n)\end{bmatrix} = \begin{bmatrix}\Gamma_c(2) & \Gamma_s(2)\\ -\Gamma_s(2) & \Gamma_c(2)\end{bmatrix}\begin{bmatrix}\bar{x}_{t+2}^c(n)\\ \bar{x}_{t+2}^{as}(n)\end{bmatrix}\tag{8}$$

where $\Gamma_c(m) = \cos(\frac{2n+1}{2N})m\pi$ and $\Gamma_s(m) = \sin(\frac{2n+1}{2N})m\pi$. The reason to introduce the auxiliary variable $x_{t+2}^{as}(n)$, which is defined as

$$\begin{aligned}
x_t^{as}(n) &= \frac{2}{N}\sum_{k=t}^{t+N-1} C(k-t)X(k)\\
&\quad\times\sin\left[\left(\frac{2n+1}{2N}\right)(k-t)\pi\right];\quad n\in\{0,\ldots,N-1\}
\end{aligned}$$

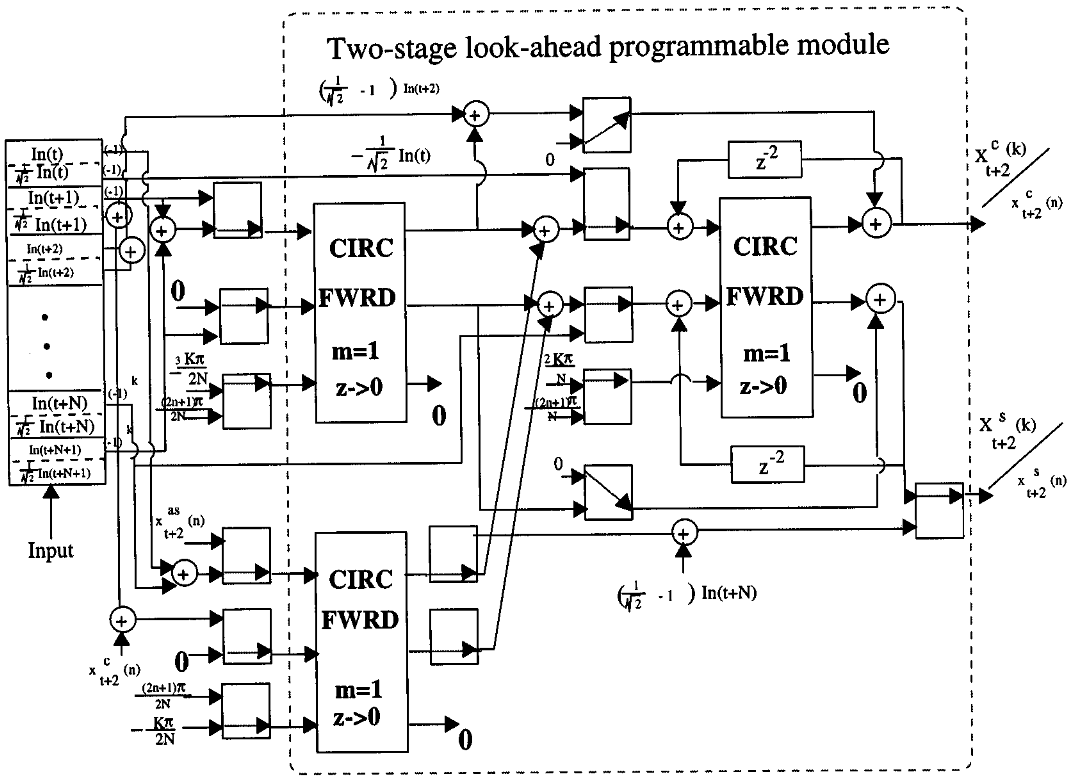


Fig. 4. Two-stage look-ahead type-II DCT/IDCT coder, the switch setting is for DCT and the complementary setting is for inverse IDCT computation. Here "CIRC FWRD" stands for circular forward rotation which is one of the CORDIC operating modes ($m = 1, z \to 0$). Z^{-2} unit stands for delaying the data by two clock cycles.

is to keep the lattice structure for numerical stability and multiplier-free architecture. And, $\bar{x}^c_{t+2}(n)$ is related to $x^c_t(n)$ and $\bar{x}^{as}_{t+2}(n)$ is related to $x^{as}_t(n)$ by

$$\begin{aligned}\bar{x}^c_{t+2}(n) &= \frac{2}{N}\sum_{k=t+2}^{t+N+1} C(k-t-2)X(k)\\ &\quad\times \cos\left[\left(\frac{2n+1}{2N}\right)(k-t)\pi\right]\\ &= x^c_t(n) - \frac{2}{N}\frac{1}{\sqrt{2}}X(t) - \frac{2}{N}X(t+1)\Gamma_c(1)\\ &\quad+\frac{2}{N}\left(\frac{1}{\sqrt{2}}-1\right)X(t+2)\Gamma_c(2)\\ &\quad-(-1)^n\frac{2}{N}X(t+N+1)\Gamma_s(1),\end{aligned}$$

$$\begin{aligned}\bar{x}^{as}_{t+2}(n) &= \frac{2}{N}\sum_{k=t+2}^{t+N+1} C(k-t-2)X(k)\\ &\quad\times \sin\left[\left(\frac{2n+1}{2N}\right)(k-t)\pi\right]\\ &= x^{as}_t(n) - \frac{2}{N}X(t+1)\Gamma_s(1) + \frac{2}{N}\left(\frac{1}{\sqrt{2}}-1\right)\\ &\quad\times X(t+2)\Gamma_s(2) + (-1)^n\frac{2}{N}X(t+N)\\ &\quad+(-1)^n\frac{2}{N}X(t+N+1)\Gamma_c(1). \qquad (9)\end{aligned}$$

Based on the above derivations, we can combine those in (9) together and get

$$\begin{aligned}\begin{bmatrix}\bar{x}^c_{t+2}(n)\\ \bar{x}^{as}_{t+2}(n)\end{bmatrix} &= \begin{bmatrix}x^c_t(n)\\ x^{as}_t(n)\end{bmatrix} + \begin{bmatrix}-\frac{2}{N}\frac{1}{\sqrt{2}}X(t)\\ 0\end{bmatrix}\\ &\quad+\begin{bmatrix}\Gamma_c(1) & -\Gamma_s(1)\\ \Gamma_s(1) & \Gamma_c(1)\end{bmatrix}\begin{bmatrix}-\frac{2}{N}X(t+1)\\ (-1)^n\frac{2}{N}X(t+N+1)\end{bmatrix}\\ &\quad+\frac{2}{N}\left(\frac{1}{\sqrt{2}}-1\right)X(t+2)\begin{bmatrix}\Gamma_c(2)\\ \Gamma_s(2)\end{bmatrix}\\ &\quad+\begin{bmatrix}0\\ (-1)^n\frac{2}{N}X(t+N)\end{bmatrix}. \qquad (10)\end{aligned}$$

As a result, we can substitute $\bar{x}^c_{t+2}(n)$ and $\bar{x}^{as}_{t+2}(n)$ in (8) and get

$$\left\{\begin{aligned}&\begin{bmatrix}x^c_{t+2}(n)\\ x^{as}_{t+2}(n)\end{bmatrix}\\ &\quad=\begin{bmatrix}\Gamma_c(2) & \Gamma_s(2)\\ -\Gamma_s(2) & \Gamma_c(2)\end{bmatrix}\\ &\quad\times\left[\begin{bmatrix}x^c_t(n)\\ x^{as}_t(n)\end{bmatrix}+\frac{2}{N}\begin{bmatrix}-\frac{1}{\sqrt{2}}X(t)\\ (-1)^nX(t+N)\end{bmatrix}\right]\\ &\quad+\frac{2}{N}\begin{bmatrix}\Gamma_c(1) & \Gamma_s(1)\\ -\Gamma_s(1) & \Gamma_c(1)\end{bmatrix}\begin{bmatrix}-X(t+1)\\ (-1)^nX(t+N+1)\end{bmatrix}\\ &\quad+\frac{2}{N}\left(\frac{1}{\sqrt{2}}-1\right)\begin{bmatrix}X(t+2)\\ 0\end{bmatrix}.\end{aligned}\right. \qquad (11)$$

Notice that $x_t^{as}(n)$ is just an auxiliary variable to keep the lattice structure. The real variable which we are interested in is $x_t^s(n)$, which is defined as

$$x_t^s(n) = \frac{2}{N} \sum_{k=t+1}^{t+N} C(k-t)X(k)\sin\left[\left(\frac{2n+1}{2N}\right)(k-t)\pi\right] \quad n \in \{0,\ldots,N-1\}. \tag{12}$$

By following the similar procedure as above, we can relate $x_{t+2}^s(n)$ to $x_{t+2}^{as}(n)$ as

$$\begin{aligned} x_{t+2}^s(n) = x_{t+2}^{as}(n)\Gamma_c(1) &+ \left[x_{t+2}^c(n)\right. \\ &+ \left.\frac{2}{N}\left(1-\frac{1}{\sqrt{2}}\right)X(t+2)\right]\Gamma_s(1) \\ &+ (-1)^n\frac{2}{N}\left(\frac{1}{\sqrt{2}}-1\right)X(t+N+1). \end{aligned}$$

Or

$$\begin{bmatrix} x_{t+2}^s(n) \\ \star \end{bmatrix} = \begin{bmatrix} \Gamma_c(1) & -\Gamma_s(1) \\ \Gamma_s(1) & \Gamma_c(1) \end{bmatrix} \begin{bmatrix} x_{t+2}^{as}(n) \\ \frac{2}{N}\left(1-\frac{1}{\sqrt{2}}\right)X(t+2) \end{bmatrix} + \begin{bmatrix} (-1)^n\frac{2}{N}\left(\frac{1}{\sqrt{2}}-1\right)X(t+N+1) \\ 0 \end{bmatrix} \tag{13}$$

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

Both two-stage look-ahead DCT computation in (6) and its inverse counterpart, IDCT computation in (11) and (13), undergo the similar computing procedure except for minor differences in the input data and rotation angles. In order to save chip area, we can interleave them into a unified structure which contains three CORDIC's, as shown in Fig. 4.

Clearly, the look-ahead system can be clocked at two-time faster rate than the original system for high-speed application. Or, by reducing the supply voltage from $V_{\rm dd}$ to $V'_{\rm dd}$, we increase the propagation delay of look-ahead system until it equals to that of the original system. The propagation delay at supply voltage of $V_{\rm dd}$ is given by

$$t_{\rm pd}(V_{\rm dd}) = \frac{C_L V_{\rm dd}}{\epsilon(V_{\rm dd}-V_t)^2} \tag{14}$$

where C_L is the capacitance along the critical path, V_t is the device threshold voltage, and ϵ is a constant which depends on the process parameters. For K-stage look-ahead system, the propagation delay is

$$\bar{t}_{\rm pd}(V'_{\rm dd}) = \frac{1}{K}\frac{C_L V'_{\rm dd}}{\epsilon(V'_{\rm dd}-V_t)^2}. \tag{15}$$

By equating $\bar{t}_{\rm pd}(V'_{\rm dd})$ in (15) to $t_{\rm pd}(V_{\rm dd})$ in (14), we get the following equation:

$$\frac{C_L}{K}\frac{V'_{\rm dd}}{\epsilon(V'_{\rm dd}-V_t)^2} = \frac{C_L V_{\rm dd}}{\epsilon(V_{\rm dd}-V_t)^2}. \tag{16}$$

TABLE I
COMPARE THE TRADE-OFF OF HARDWARE COST AND POWER SAVING FOR LOOK-AHEAD DESIGNS

Stages of look-ahead	Supply voltage	Percentage of power saving	Percentage of hardware increasing
2	$3.08V$	72%	150%
4	$2.11V$	89%	250%
8	$1.54V$	95%	450%

Substituting $V_{\rm dd} = 5$ V and all $V_t = 0.7$ V, we find that for a two-stage look-ahead system (i.e., $K = 2$) a supply voltage of $V'_{\rm dd} = 3.08$ V is necessary for the two propagation delays to equal each other. In other words, we achieve low-power design while still keep the same system throughput.

The dynamic power consumption of a CMOS circuit is given by

$$P = \alpha C_{\rm total} V_{\rm dd}^2 f \tag{17}$$

where α is the average fraction of the total node capacitance being switched (also referred to as the activity factor), $C_{\rm total}$ is the total switching capacitance, $V_{\rm dd}$ is the supply voltage and f is the clock frequency. By employing (17), we get the ratio of the power consumption of two-stage look-ahead design, $P_{2-{\rm stage}}$, to the power of original design, $P_{\rm dct}$, as

$$\frac{P_{2-{\rm stage}}}{P_{\rm dct}} = \frac{C_{2-{\rm stage}}}{C_{\rm dct}}\left(\frac{3.08\ {\rm V}}{5\ {\rm V}}\right)^2 \frac{\frac{1}{2}f}{f} = 0.28$$

where f is the original operating frequency, $C_{2-{\rm stage}}$ and $C_{\rm dct}$ represent the total switching capacitances of look-ahead and its original implementation. Provided that the capacitances due to CORDIC's are dominant in the circuit and are roughly proportional to the number of CORDIC's, we get $C_{2-{\rm stage}} \approx (3/2)C_{\rm dct}$ because the low-power design requires three CORDICs while the original design needs two CORDIC's. Overall the look-ahead design results in 72% power saving without sacrificing the system throughput at the expense of 50% hardware overhead. In essence, we trade silicon area for low-power consumption.

Based on the same approach as two-stage look-ahead design, we can extend to four-stage look-ahead design and beyond. Let us look at four-stage look-ahead design. By substituting $V_{\rm dd} = 5$ V, $V_t = 0.7$ V and $K = 4$ into (16), we get $V'_{\rm dd} = 2.11$ V. The ratio of the power consumption of four-stage look-ahead design, $P_{4-{\rm stage}}$, to $P_{\rm dct}$ is

$$\begin{aligned} \frac{P_{4-{\rm stage}}}{P_{\rm dct}} &= \frac{C_{4-{\rm stage}}}{C_{\rm dct}}\left(\frac{2.11\ {\rm V}}{5\ {\rm V}}\right)^2 \frac{\frac{1}{4}f}{f} \\ &= \frac{5}{2}\left(\frac{2.11\ {\rm V}}{5\ {\rm V}}\right)^2 \frac{1}{4} = 0.11. \end{aligned}$$

Our studies have revealed that the power saving is generally increased by employing more look-ahead stages (K). However, beyond a certain "critical point" $(K = 4)$, the percentage of further power saving is small while hardware cost is increasing drastically. The results are listed and plotted in Table I and Fig. 5, respectively. To illustrate the concept of our low-power

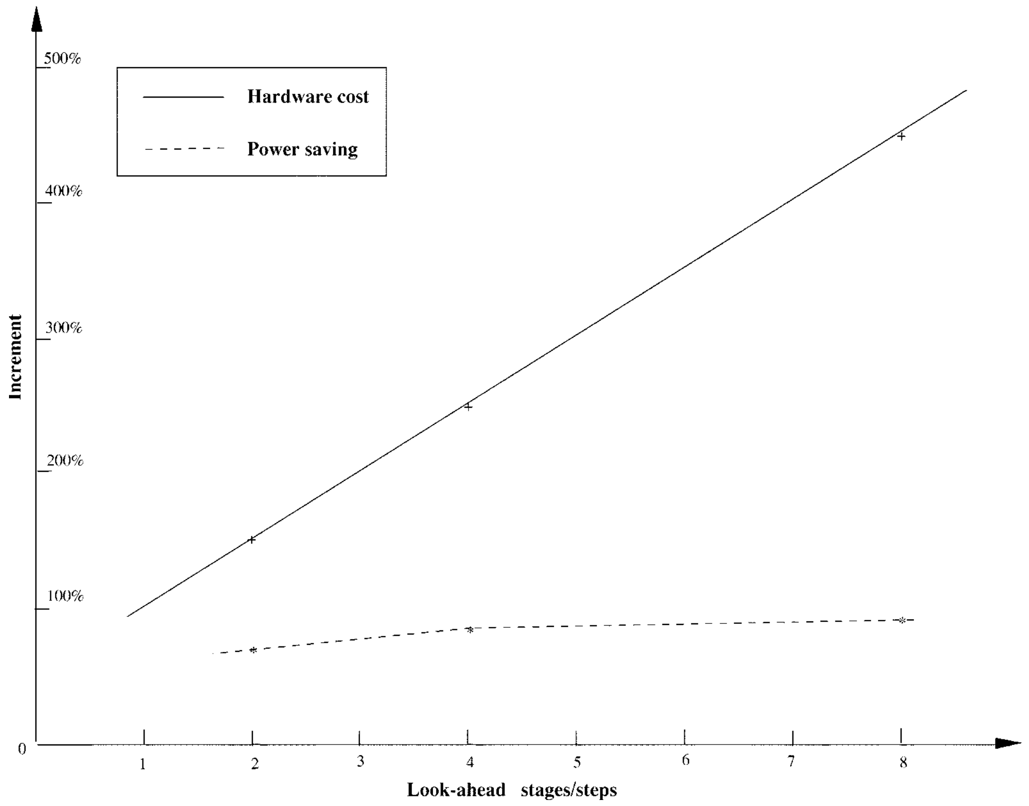


Fig. 5. Power saving and hardware cost increment for different look-ahead systems.

design, we choose two-stage look-ahead in this paper for our low-power DCT coder design.

Because 2D-DCT can be decomposed into two-stage pipelined 1-D computation, we therefore adopt the same approach as in [14] to extend our low-power DCT design to 2-D design. As a result, it is able to output four *type-II* DCT coefficients $X_t^{cc}(k,l), X_t^{cs}(k,l), X_t^{sc}(k,l)$ and $X_t^{ss}(k,l)$ such as

$$X_t^{cs}(k,l) = \frac{4}{N^2}C(k)C(l)\sum_{m,n=0}^{N-1} x_t(m,n) \times \cos\left[\frac{k\pi}{N}(m+0.5)\right]\sin\left[\frac{l\pi}{N}(n+0.5)\right] \quad \text{for } k \in \{0,\ldots,N-1\},\quad l \in \{1,\ldots,N\} \tag{18}$$

simultaneously, as shown in Fig. 6.

B. Pipelining Design for DCT Coefficients Conversion

Pipelining is the most commonly used technique to achieve high-speed. The main idea behind is to insert flip-flop between m consecutive pipeline stages so that the delay through the critical path can be shortened by a factor of m. As a result, the speed of the system is m times faster than that of the original system at the penalty of increasing system latency. On the other hand, the pipelining can be used to compensate for the delay incurred in the low-power design when supply voltage drops.

In order to calculate the pseudo phases, the *type-I* DCT coefficients of previous block, $Z_{t-1}^{cc}(k,l), Z_{t-1}^{cs}(k,l), Z_{t-1}^{sc}(k,l)$ and $Z_{t-1}^{ss}(k,l)$ such as

$$Z_{t-1}^{sc}(k,l) = \frac{4}{N^2}C(k)C(l)\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} x_{t-1}(m,n) \times \sin\left[\frac{k\pi}{N}(m)\right]\cos\left[\frac{l\pi}{N}(n)\right], \quad \text{for } k \in \{1,\ldots,N-1\},\quad l \in \{0,\ldots,N\} \tag{19}$$

are needed. However, it is undesirable to compute those *type-I* coefficients separately from *type-II* coefficients otherwise it will increases overall hardware complexity significantly. As a matter of fact, this problem can be circumvented because those *type-I* DCT coefficients can actually be obtained by the plane rotation of its counterpart *type-II* DCT coefficients, $X_{t-1}^{cc}(k,l), X_{t-1}^{cs}(k,l), X_{t-1}^{sc}(k,l)$ and $X_{t-1}^{ss}(k,l)$, which are stored in the array registers as shown in Fig. 3. Those *type-I* and *type-II* DCT coefficients are related as follows:

$$\begin{bmatrix} \cos\theta\cos\phi & \cos\theta\sin\phi & \sin\theta\cos\phi & \sin\theta\sin\phi \\ -\cos\theta\sin\phi & \cos\theta\cos\phi & -\sin\theta\sin\phi & \sin\theta\cos\phi \\ -\sin\theta\cos\phi & -\sin\theta\sin\phi & \cos\theta\cos\phi & \cos\theta\sin\phi \\ \sin\theta\sin\phi & -\sin\theta\cos\phi & -\cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix} \times \begin{bmatrix} X_{t-1}^{cc}(k,l) \\ X_{t-1}^{cs}(k,l) \\ X_{t-1}^{sc}(k,l) \\ X_{t-1}^{ss}(k,l) \end{bmatrix} = \begin{bmatrix} Z_{t-1}^{cc}(k,l) \\ Z_{t-1}^{cs}(k,l) \\ Z_{t-1}^{sc}(k,l) \\ Z_{t-1}^{ss}(k,l) \end{bmatrix} \tag{20}$$

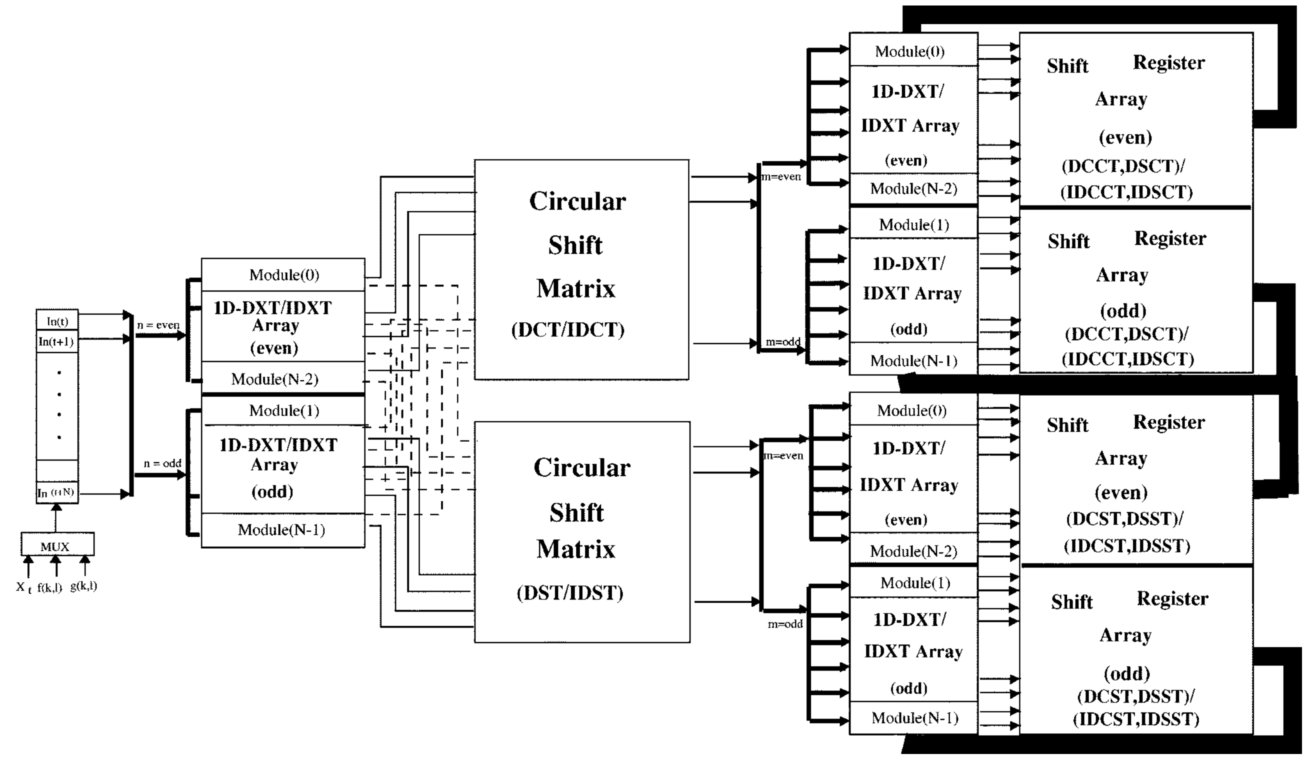

In(t)
In(t+1)
In (t+N)
MUX
X_t f(k,l) g(k,l)
n = even
n = odd
Module(0)
1D-DXT/IDXT Array (even)
Module(N-2)
Module(1)
1D-DXT/IDXT Array (odd)
Module(N-1)
Circular Shift Matrix (DCT/IDCT)
Circular Shift Matrix (DST/IDST)
m=even
m=odd
1D-DXT/ IDXT Array (even)
1D-DXT/ IDXT Array (odd)
Shift Register Array (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)

Fig. 6. The 2D-DCT computation.

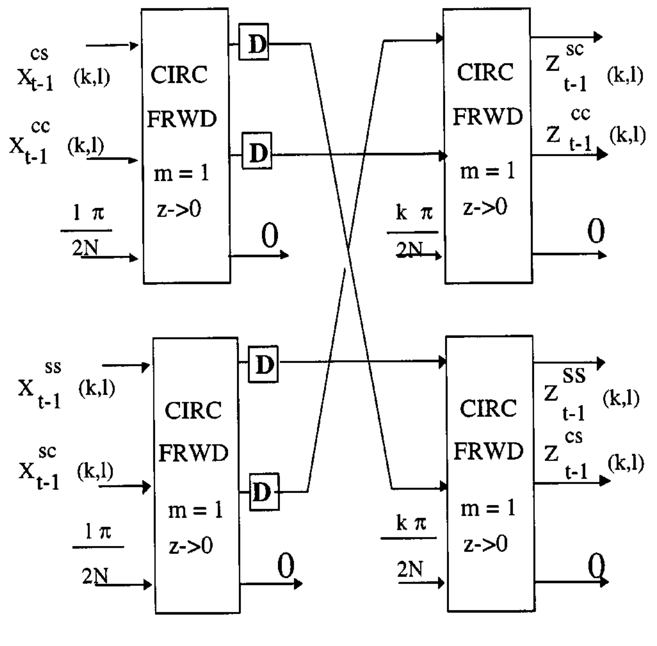


Fig. 7. Pipelining design for *type-II* to *type-I* DCT coefficients conversion.

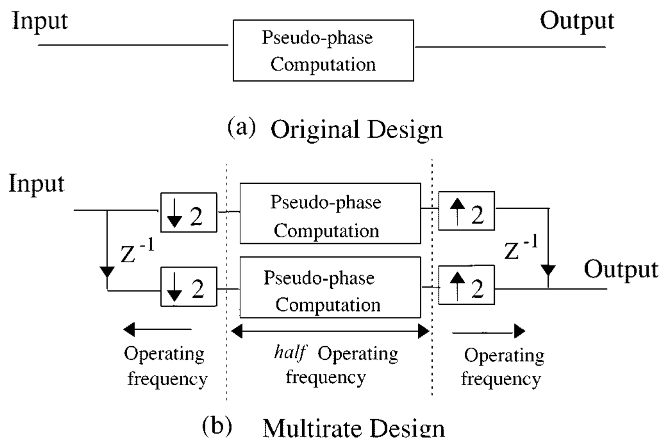


Fig. 8. Multirate design for pseudo-phase computation.

where $\theta = (k\pi/2N)$, $\phi = (l\pi/2N)$. This DCT coefficient conversion can be realized by two-stage orthogonal plan rotations as shown in Fig. 7. By inserting flip-flops $(\mathbf{D})$ across the feed-forward cut-set, we can achieve high-speed design. Now the pipelining design can run two-time faster than the original design because the critical path has been halved. Or, we can reduce the power supply voltage from 5 V to 3.08 V based on the same argument as in (16) while still maintain the original system throughput. The ratio of the power consumption of pipelining design, P_{pipe}, to the power of original design, P_{rotator}, is given by

$$\frac{P_{\text{pipe}}}{P_{\text{rotator}}} = \frac{4}{4}\left(\frac{3.08\text{ V}}{5\text{ V}}\right)^2 \frac{\frac{1}{2}f}{f} = 0.19$$

which leads to 81% power saving at the cost of increased system latency. Here the operating frequency f is the same as that in DCT coder because our low-power design is a synchronous design.

C. Multirate Design for Pseudo-Phase Computation

Other than pipelining, parallel data processing is another frequently used technique to achieve high-speed design. In principle, the desired functions are decomposed into independent and parallel small tasks. Then the small tasks are executed

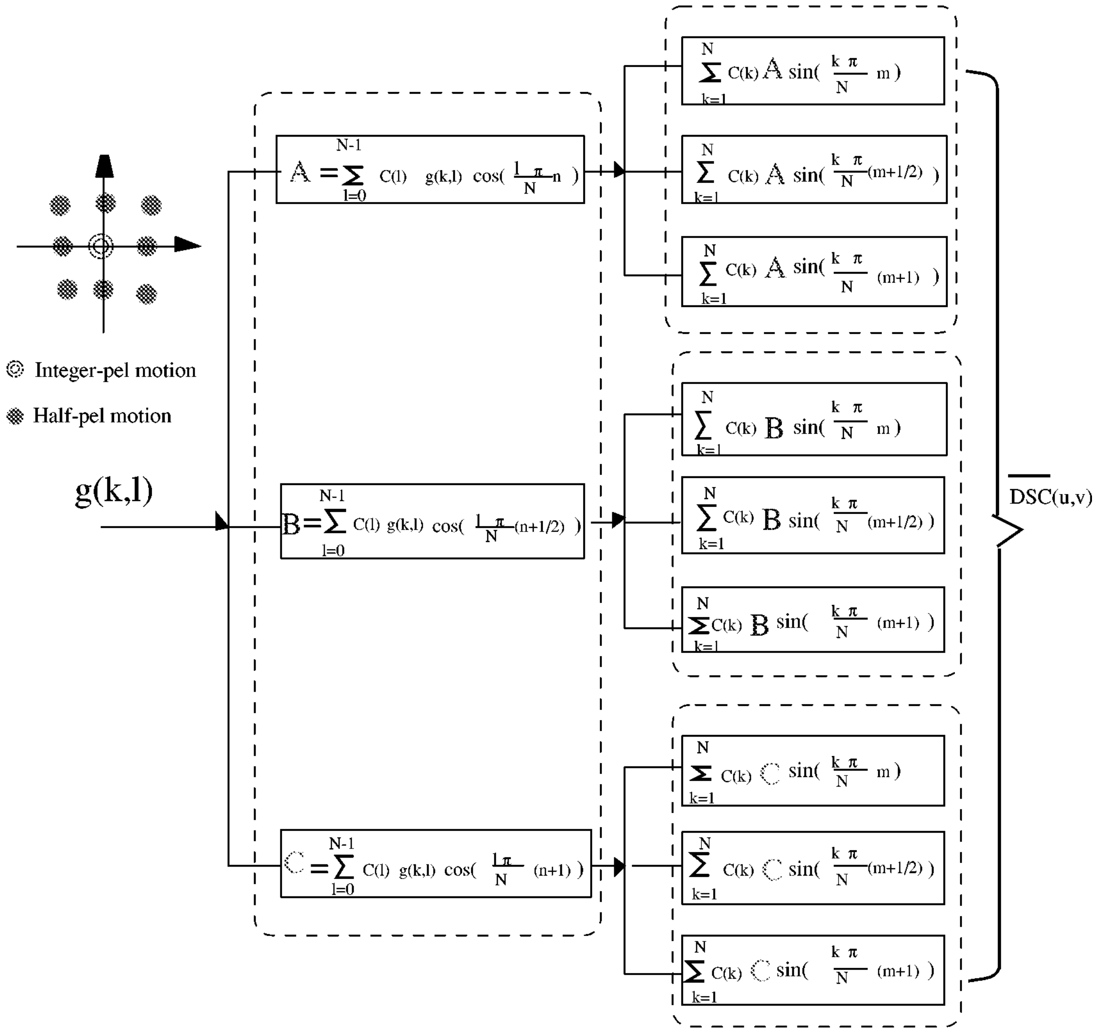

Integer-pel motion
Half-pel motion
g(k,l)
A = Σ_{l=0}^{N-1} C(l) g(k,l) cos(lπ/N n)
B = Σ_{l=0}^{N-1} C(l) g(k,l) cos(lπ/N (n+1/2))
C = Σ_{l=0}^{N-1} C(l) g(k,l) cos(lπ/N (n+1))
Σ_{k=1}^{N} C(k) A sin(kπ/N m)
Σ_{k=1}^{N} C(k) A sin(kπ/N (m+1/2))
Σ_{k=1}^{N} C(k) A sin(kπ/N (m+1))
Σ_{k=1}^{N} C(k) B sin(kπ/N m)
Σ_{k=1}^{N} C(k) B sin(kπ/N (m+1/2))
Σ_{k=1}^{N} C(k) B sin(kπ/N (m+1))
Σ_{k=1}^{N} C(k) C sin(kπ/N m)
Σ_{k=1}^{N} C(k) C sin(kπ/N (m+1/2))
Σ_{k=1}^{N} C(k) C sin(kπ/N (m+1))
DSC(u,v)

Fig. 9. Schematic diagram of decomposing $\overline{\text{DSC}}(u, v)$ computation.

concurrently and individual results are combined together. The well-known "divide-and-conquer" strategy is one of these kinds of parallel processing. The goal of parallel processing is to utilize each processing element (PE) fully to achieve maximum data throughput rate. Therefore, this feature is very suitable for high-speed data processing and its modular design is very desirable for VLSI implementation. The multirate approach used in this paper is belongs to this category.

Traditionally, multirate technique is widely used in subband coding-based compression of audio/video signals and in trans-multiplexers that convert between time and frequency division multiplexing [16]. Our interest, on the other hand, is to apply this technique to compensate the speed loss due to lowered supply voltage or to simply speed-up the design under normal condition. The pseudo-phase functions $f(k,l)$ and $g(k,l)$ of integer-pel displacements can be obtained by solving the following *system equation* [10]

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

$$\times \underbrace{\begin{bmatrix} \star \\ f(k,l) \\ g(k,l) \\ \star \end{bmatrix}}_{\vec{\theta}_{m,n}(k,l)} = \underbrace{\begin{bmatrix} X^{cc}_t(k,l) \\ X^{cs}_t(k,l) \\ X^{sc}_t(k,l) \\ X^{ss}_t(k,l) \end{bmatrix}}_{\vec{x}_t(k,l)}$$

$$\text{for } k,l \in \{1,\ldots,N-1\} \quad (21)$$

where (m,n) is the integer-pel displacement and $\vec{\theta}_{m,n}$ is the pseudo-phase vector. For illustration purpose, the pseudo-phase computation module in the original design is shown in Fig. 8(a) (please refer to [14] for the detail design). Here the processing rate of the operator has to be as fast as the input data rate.

By employing *multirate* low-power design, the pseudo-phases are computed from the reformulated circuit using the decimated sequences $(M = 2)$, as shown in Fig. 8(b). The additional concurrency is obtained by dividing the data stream into odd and even sequences. Both pseudo-phase computation modules process such sequences concurrently and individual outputs are then combined together. Now the multirate design operates at two different rates. Because the operating frequency of pseudo-phase computation is reduced to half of the input data rate while the overall throughput rate is still remained the same, the speed penalty therefore is compensated at the

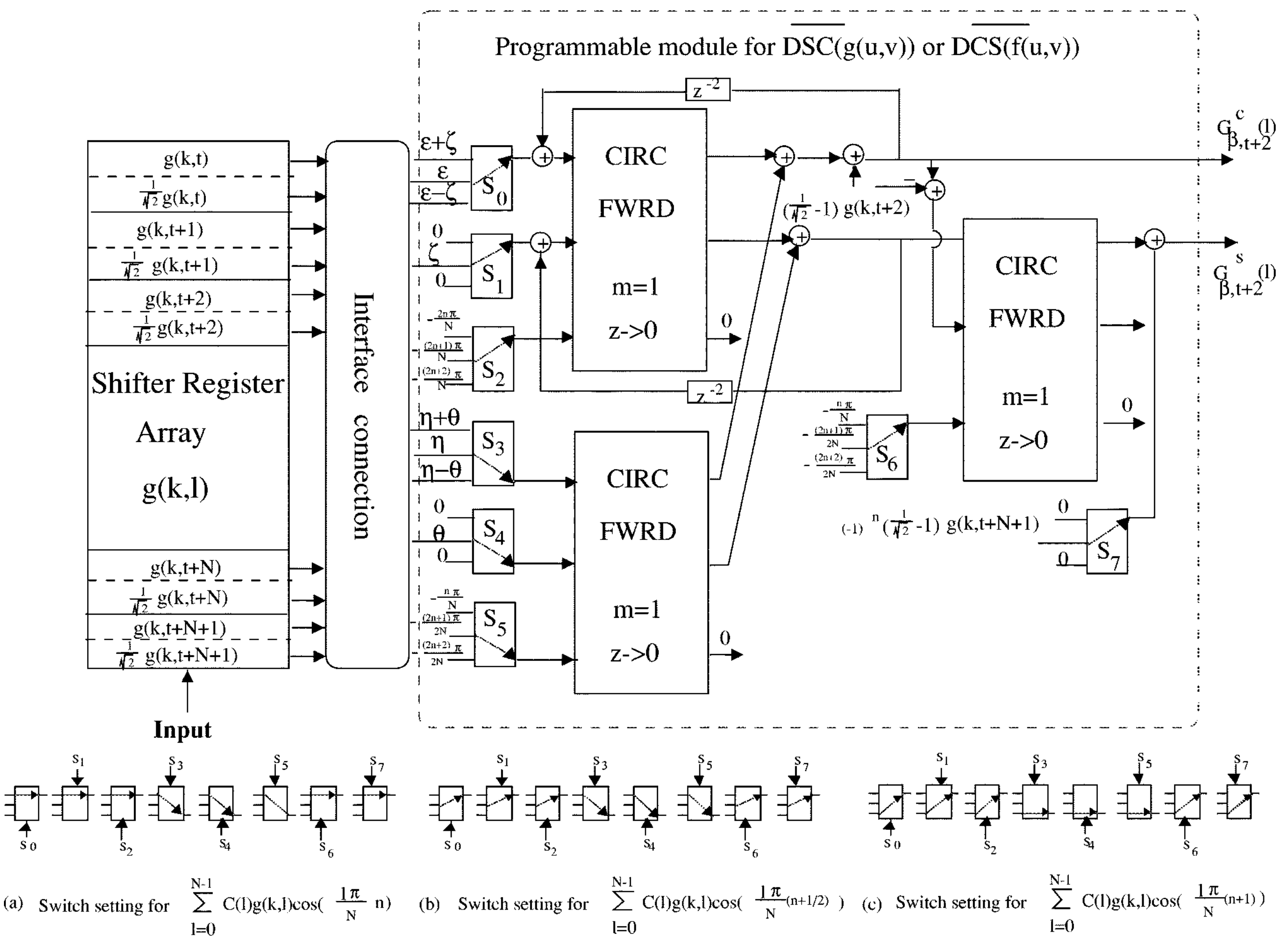

Programmable module for DSC(g(u,v)) or DCS(f(u,v))
Shifter Register Array g(k,l)
Interface connection
Input
CIRC FWRD m=1 z->0
CIRC FWRD m=1 z->0
CIRC FWRD m=1 z->0
z^-2
z^-2
S0
S1
S2
S3
S4
S5
S6
S7
g(k,t)
g(k,t+1)
g(k,t+2)
g(k,t+N)
g(k,t+N+1)
(1/√2 -1) g(k,t+2)
(-1)^n (1/√2 -1) g(k,t+N+1)
G^c_{β,t+2}(l)
G^s_{β,t+2}(l)
(a) Switch setting for Σ_{l=0}^{N-1} C(l)g(k,l)cos(lπ/N n)
(b) Switch setting for Σ_{l=0}^{N-1} C(l)g(k,l)cos(lπ/N (n+1/2))
(c) Switch setting for Σ_{l=0}^{N-1} C(l)g(k,l)cos(lπ/N (n+1))

Fig. 10. Programmable module for low-power half-pel motion estimator. Here the "Interface connection" is set based on Table II.

architectural level. With the similar argument stated previously in Section II-A2, we can keep the overall throughput rate while reduce the power supply voltage from 5 V to 3.08 V. The multirate design needs 20 CORDIC's, which is twice the number of CORDIC's in original design plus additional down/up-sampling devices. Provided that the capacitances due to CORDIC's are dominant in the circuit, the ratio of the power consumption of multirate design, P_{multi}, to the power of original design, P_{phase}, can be obtained as

$$\frac{P_{\text{multi}}}{P_{\text{phase}}} = \frac{20}{10}\left(\frac{3.08\text{ V}}{5\text{ V}}\right)^2 \frac{\frac{1}{2}f}{f} = 0.38.$$

Overall, we can achieve the power saving of 62% or the speed-up factor of two at the cost of doubled hardware complexity.

D. Pipelining Design for Peak-Search

As we have pointed out in the introduction, the translational displacement can be found by locating the peak of the inverse 2D-DCT transform of the normalized pseudo-phase function. Unlike full search block matching, this peak-search is a quite straightforward process because we only need to locate the maximum values of the 2-D matrices. The 2-D search can be simply decomposed to row-then-column 1-D search. The decomposition search looks for the peak value of each row, followed by a column search of the previous results. If we fail to locate the peak e.g. the motion vector goes beyond the block boundaries, the motion vector of $(0, 0)$ will be used, instead. Since it is fully pipelined, we can insert flip-flops after the peak search of each row. Therefore, we cut the critical path by half and get the low-power design. Based on the same argument in Section II-B, it achieves 81% power saving under 3.08 V supply-voltage.

TABLE II
PARAMETERS FOR DIFFERENT PHASES IN (40) AND (41)

β	0	$\frac{1}{2}$	1
J	$\mathbf{R}(2n)$	$\mathbf{R}(2n+1)$	$\mathbf{R}(2n+2)$
H	$\begin{bmatrix} \epsilon+\zeta \\ 0 \end{bmatrix}$	$\begin{bmatrix} \epsilon \\ \zeta \end{bmatrix}$	$\begin{bmatrix} \epsilon-\zeta \\ 0 \end{bmatrix}$
L	$\mathbf{R}(n)$	$\mathbf{R}(n+\frac{1}{2})$	$\mathbf{R}(n+1)$
M	$\begin{bmatrix} \eta+\theta \\ 0 \end{bmatrix}$	$\begin{bmatrix} \eta \\ \theta \end{bmatrix}$	$\begin{bmatrix} \eta-\theta \\ 0 \end{bmatrix}$
N	$\mathbf{0}$	$\begin{bmatrix} \kappa \\ 0 \end{bmatrix}$	$\mathbf{0}$

$$\mathbf{R}(m) = \begin{bmatrix} cos\frac{m\pi}{N} & sin\frac{m\pi}{N} \\ -sin\frac{m\pi}{N} & cos\frac{m\pi}{N} \end{bmatrix},$$
$$\epsilon = -\frac{1}{\sqrt{2}}\frac{2}{N}g(k,t); \zeta = (-1)^n\frac{2}{N}g(k,t+N); \eta = -\frac{2}{N}g(k,t+1);$$
$$\theta = (-1)^n\frac{2}{N}g(k,t+N+1); \kappa = (-1)^n\frac{2}{N}(\frac{1}{\sqrt{2}}-1)g(k,t+N+1);$$

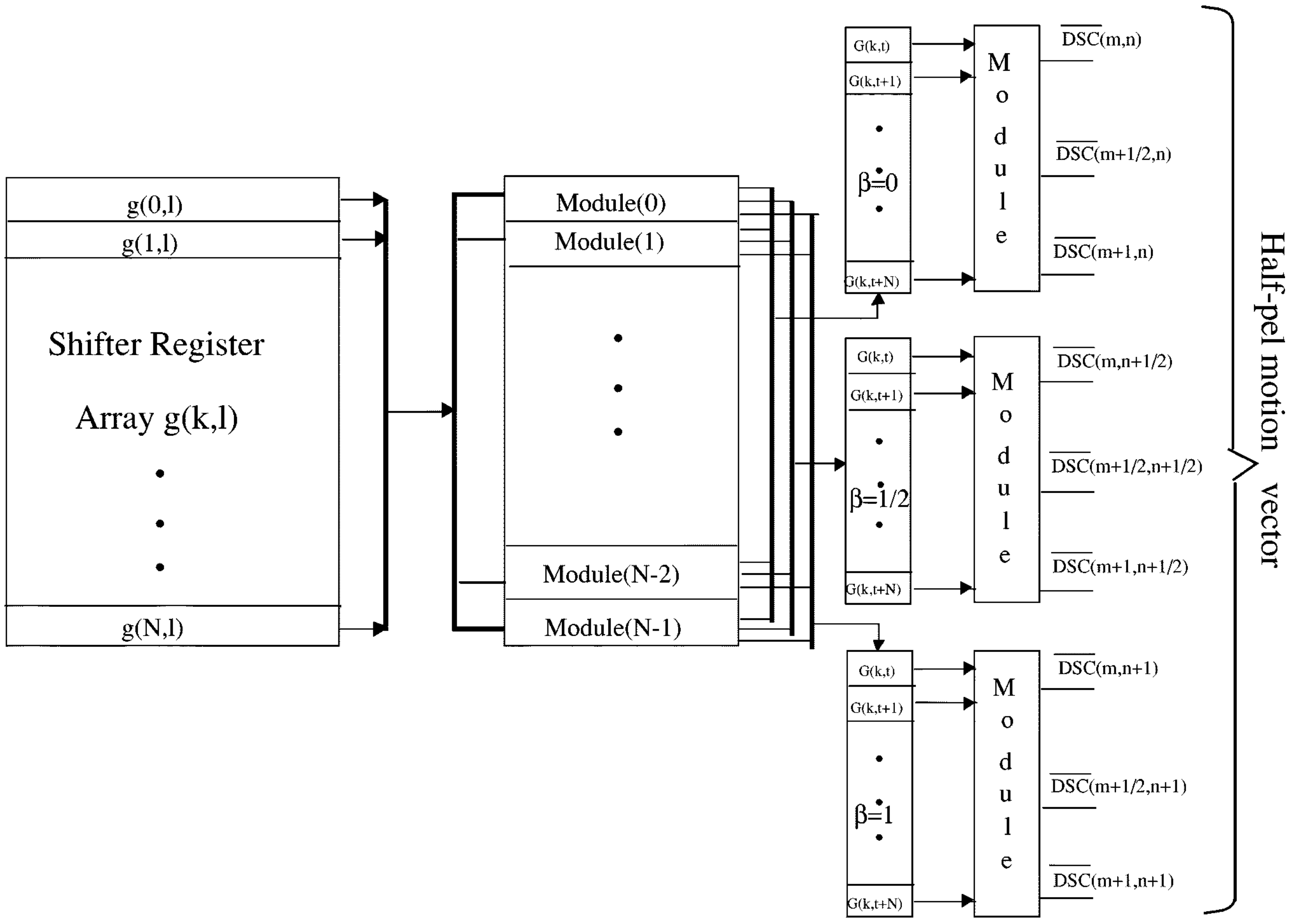

g(0,l)
g(1,l)
Shifter Register
Array g(k,l)
g(N,l)
Module(0)
Module(1)
Module(N-2)
Module(N-1)
G(k,t)
G(k,t+1)
β=0
G(k,t+N)
Module
$\overline{\mathrm{DSC}}$(m,n)
$\overline{\mathrm{DSC}}$(m+1/2,n)
$\overline{\mathrm{DSC}}$(m+1,n)
β=1/2
$\overline{\mathrm{DSC}}$(m,n+1/2)
$\overline{\mathrm{DSC}}$(m+1/2,n+1/2)
$\overline{\mathrm{DSC}}$(m+1,n+1/2)
β=1
$\overline{\mathrm{DSC}}$(m,n+1)
$\overline{\mathrm{DSC}}$(m+1/2,n+1)
$\overline{\mathrm{DSC}}$(m+1,n+1)
Half-pel motion vector

Fig. 11. Block diagram for computing $\overline{\mathrm{DSC}}(u, v)$. Here the "module" refers to the one in Fig. 10.

E. Two-Stage Look-Ahead Half-Pel Motion Estimator

To obtain motion at half-pel accuracy, we first compute the integer-pel motion vectors (m, n) then use "two-stage look-ahead half-pel motion estimator" in Fig. 3 to compute the half-pel motion vectors. With such an approach, we can avoid conventional interpolation procedure to determine the half-pel motion vectors by only considering the nine positions $u \in \{m - 0.5, m, m + 0.5\}$ and $v \in \{n - 0.5, n, n + 0.5\}$ surrounding integer-pel motion vectors (m, n) [11]. As a result, it decreases the overall complexity and avoids undesirable data flow. In other words, the peak positions among $\overline{\mathrm{DCS}}(u, v)$ and $\overline{\mathrm{DSC}}(u, v)$ indicates the half-pel motion as illustrated at the upper right corner of Fig. 9. Next we will explain how to adopt look-ahead approach mentioned previously to achieve the low-power half-pel motion estimator architecture.

$$\overline{\mathrm{DCS}}(u,v) = \frac{4}{N^2}\sum_{k=0}^{N-1}\sum_{l=1}^{N} C(k)C(l)f(k,l) \times \cos\frac{k\pi}{N}\left(u+\frac{1}{2}\right)\sin\frac{l\pi}{N}\left(v+\frac{1}{2}\right) \quad (22)$$

$$\overline{\mathrm{DSC}}(u,v) = \frac{4}{N^2}\sum_{k=1}^{N}\sum_{l=0}^{N-1} C(k)C(l)g(k,l) \times \sin\frac{k\pi}{N}\left(u+\frac{1}{2}\right)\cos\frac{l\pi}{N}\left(v+\frac{1}{2}\right) \quad (23)$$

By taking a close look at (22) and (23), we observe that both $\overline{\mathrm{DSC}}(u, v)$ and $\overline{\mathrm{DCS}}(u, v)$ computations are similar. Here we use $\overline{\mathrm{DSC}}(u, v)$ computation as an example, the same approach can be applied to $\overline{\mathrm{DCS}}(u, v)$. In order to figure out $\overline{\mathrm{DSC}}(u, v)$, we can decompose its 2-D computation into two-stage hierarchic 1-D calculations as illustrated in Fig. 9. As a matter of fact, those computations encircled by dot-boxes, such as

$$\begin{aligned} A &= \sum_{l=0}^{N-1} C(l)g(k,l)\cos\left(\frac{l\pi}{N}n\right) \\ B &= \sum_{l=0}^{N-1} C(l)g(k,l)\cos\left(\frac{l\pi}{N}\left(n+\frac{1}{2}\right)\right) \\ C &= \sum_{l=0}^{N-1} C(l)g(k,l)\cos\left(\frac{l\pi}{N}(n+1)\right) \end{aligned} \quad (24)$$

in the middle level of Fig. 9, are similar except the phase differences such as $(l\pi/N)n$, $(l\pi/N)(n+(1/2))$ and $(l\pi/N)(n+1)$. Therefore, those computations can actually be integrated and realized by a programmable structure, as shown in Fig. 10.

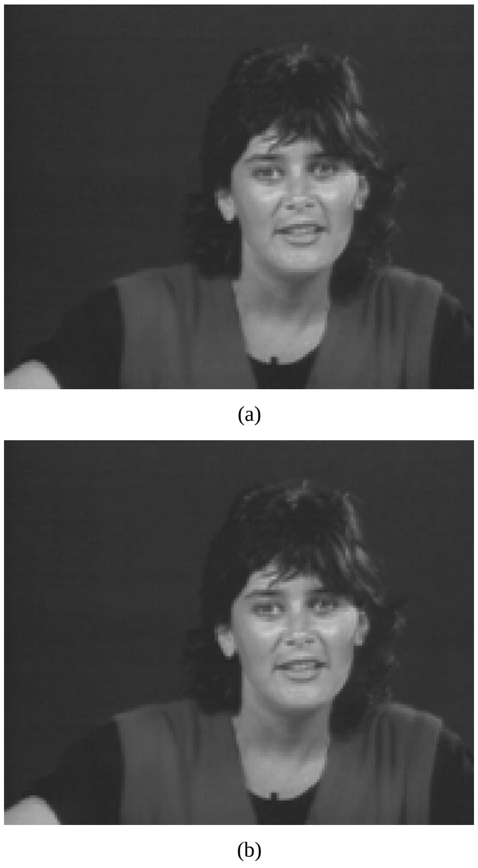


Fig. 12. "Miss America" frame 91: (a) original frame and (b) reconstructed frame.

Before we proceed our discussion, let us define

$$G^c_{\beta,t}(l) = \frac{2}{N}\sum_{l=t}^{t+N-1} C(l-t)g(k,l)\cos\left[\frac{(l-t)\pi}{N}(n+\beta)\right]$$
$$G^s_{\beta,t}(l) = \frac{2}{N}\sum_{l=t+1}^{t+N} C(l-t)g(k,l)\sin\left[\frac{(l-t)\pi}{N}(n+\beta)\right]$$

where

$$C(l) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{for } l = 0 \text{ or } N \\ 1 & \text{otherwise.}\end{cases} \tag{25}$$

Here, n is one of the integer-pel motion vector (m, n) and $\beta \in \{0, (1/2), 1\}$ indicates different phases. The time index t in $G^c_{\beta,t}(l)$ and $G^s_{\beta,t}(l)$ denotes the transform starting from $g(k,t)$ which are pseudo phase functions derived in Section II-C. In addition, an auxiliary variable

$$G^{as}_{\beta,t}(l) = \frac{2}{N}\sum_{l=t}^{t+N-1} C(l-t)g(k,l) \times \sin\left[\frac{(l-t)\pi}{N}(n+\beta)\right] \tag{26}$$

is introduced to maintain the lattice structure similar to that of DCT computation in Section II-A2. To achieve low-power design, we need to find out two-stage look-ahead coefficients $G^c_{\beta,t+2}(l)$ and $G^{as}_{\beta,t+2}(l)$ in terms of $G^c_{\beta,t}(l)$ and $G^{as}_{\beta,t}(l)$. Furthermore, $G^s_{\beta,t+2}(l)$ can be obtained indirectly through $G^{as}_{\beta,t+2}(l)$. Next, we will derive those relationships.

$\boxed{\text{For phase } (l\pi/N)(n+1) \text{ or } \beta = 1}$

$$\begin{aligned} G^c_{1,t+2}(l) &= \frac{2}{N}\sum_{l=t+2}^{t+N+1} C(l-t-2)g(k,l) \times\cos\left[\frac{(l-t-2)\pi}{N}(n+1)\right] \\ &= \frac{2}{N}\sum_{l=t+2}^{t+N+1} C(l-t-2)g(k,l) \times\cos\left[\frac{(l-t)\pi}{N}(n+1)\right]\cos\left[\frac{2\pi}{N}(n+1)\right] \\ &\quad + \frac{2}{N}\sum_{l=t+2}^{t+N+1} C(l-t-2)g(k,l) \times\sin\left[\frac{(l-t)\pi}{N}(n+1)\right]\sin\left[\frac{2\pi}{N}(n+1)\right] \end{aligned} \tag{27}$$

$$\begin{aligned} G^{as}_{1,t+2}(l) &= \frac{2}{N}\sum_{l=t+2}^{t+N+1} C(l-t-2)g(k,l) \times\sin\left[\frac{(l-t-2)\pi}{N}(n+1)\right] \\ &= \frac{2}{N}\sum_{l=t+2}^{t+N+1} C(l-t-2)g(k,l) \times\sin\left[\frac{(l-t)\pi}{N}(n+1)\right]\cos\left[\frac{2\pi}{N}(n+1)\right] \\ &\quad - \frac{2}{N}\sum_{l=t+2}^{t+N+1} C(l-t-2)g(k,l) \times\cos\left[\frac{(l-t)\pi}{N}(n+1)\right]\sin\left[\frac{2\pi}{N}(n+1)\right] \end{aligned} \tag{28}$$

Both (27) and (28) can be combined into the following equation:

$$\begin{bmatrix} G^c_{1,t+2}(l) \\ G^{as}_{1,t+2}(l)\end{bmatrix} = \begin{bmatrix} \cos\frac{2(n+1)\pi}{N} & \sin\frac{2(n+1)\pi}{N} \\ -\sin\frac{2(n+1)\pi}{N} & \cos\frac{2(n+1)\pi}{N}\end{bmatrix} \times \begin{bmatrix} \bar{G}^c_{1,t+2}(l) \\ \bar{G}^{as}_{1,t+2}(l)\end{bmatrix}. \tag{29}$$

$\bar{G}^c_{1,t+2}(l)$ can be rewritten as

$$\begin{aligned} \bar{G}^c_{1,t+2}(l) &= \frac{2}{N}\sum_{l=t+2}^{t+N+1} C(l-t-2)g(k,l) \times\cos\left[\frac{(l-t)\pi}{N}(n+1)\right] \\ &= \frac{2}{N}\frac{1}{\sqrt{2}}g(k,t+2)\cos\left[\frac{2(n+1)\pi}{N}\right] \\ &\quad + \frac{2}{N}\sum_{l=t+3}^{t+N-1} g(k,l)\cos\left[\frac{(l-t)\pi}{N}(n+1)\right] \\ &\quad + (-1)^{n+1}\frac{2}{N}g(k,t+N) + (-1)^{n+1} \times\frac{2}{N}g(k,t+N+1)\cos\left[\frac{\pi}{N}(n+1)\right]. \end{aligned} \tag{30}$$

$G^c_{1,t}(l)$ can be rewritten as

$$\begin{aligned}G^c_{1,t}(l) &= \frac{2}{N}\sum_{l=t}^{t+N-1} C(l-t)g(k,l)\cos\left[\frac{(l-t)\pi}{N}(n+1)\right]\\ &= \frac{2}{N}\frac{1}{\sqrt{2}}g(k,t) + \frac{2}{N}\sum_{l=t+3}^{t+N-1} g(k,l)\\ &\quad\times\cos\left[\frac{(l-t)\pi}{N}(n+1)\right]\\ &\quad+\frac{2}{N}g(k,t+1)\cos\left[\frac{\pi}{N}(n+1)\right]\\ &\quad+\frac{2}{N}g(k,t+2)\cos\left[\frac{2\pi}{N}(n+1)\right].\end{aligned}\tag{31}$$

By combining (30) and (31), we can express $\bar{G}^c_{1,t+2}(l)$ in terms of $G^c_{1,t}(l)$ as

$$\begin{aligned}\bar{G}^c_{1,t+2}(l) &= G^c_{1,t}(l) + \frac{2}{N}\left(\frac{1}{\sqrt{2}}-1\right)\\ &\quad\times g(k,t+2)\cos\left[\frac{2(n+1)\pi}{N}\right]\\ &\quad-\frac{2}{N}g(k,t+1)\cos\frac{\pi}{N}(n+1)\\ &\quad-\frac{2}{N}\frac{1}{\sqrt{2}}g(k,t)+(-1)^{n+1}\frac{2}{N}g(k,t+N)\\ &\quad+(-1)^{n+1}\frac{2}{N}g(k,t+N+1)\\ &\quad\times\cos\left[\frac{\pi}{N}(n+1)\right].\end{aligned}\tag{32}$$

We can also relate $\bar{G}^{as}_{1,t+2}(l)$ to $G^{as}_{1,t}(l)$ as

$$\begin{aligned}\bar{G}^{as}_{1,t+2}(l) &= \frac{2}{N}\sum_{l=t+2}^{t+N+1} C(l-t-2)\\ &\quad\times g(k,l)\sin\left[\frac{(l-t)\pi}{N}(n+1)\right]\\ &= G^{as}_{1,t}(l)+\frac{2}{N}\left(\frac{1}{\sqrt{2}}-1\right)\\ &\quad\times g(k,t+2)\sin\left[\frac{2(n+1)\pi}{N}\right]\\ &\quad-\frac{2}{N}g(k,t+1)\sin\frac{\pi}{N}(n+1)+(-1)^{n+1}\\ &\quad\times\frac{2}{N}g(k,t+N+1)\sin\left[\frac{\pi}{N}(n+1)\right].\end{aligned}\tag{33}$$

Both (32) and (33) can be merged as

$$\begin{aligned}&\begin{bmatrix}\bar{G}^c_{1,t+2}(l)\\ \bar{G}^{as}_{1,t+2}(l)\end{bmatrix}\\ &= \begin{bmatrix}G^c_{1,t}(l)\\ G^{as}_{1,t}(l)\end{bmatrix} + \frac{2}{N}[(-1)^{n+1}g(k,t+N+1)\\ &\quad-g(k,t+1)]\begin{bmatrix}\cos\frac{\pi}{N}(n+1)\\ \sin\frac{\pi}{N}(n+1)\end{bmatrix}\\ &\quad+\frac{2}{N}\left(\frac{1}{\sqrt{2}}-1\right)g(k,t+2)\begin{bmatrix}\cos\frac{2(n+1)\pi}{N}\\ \sin\frac{2(n+1)\pi}{N}\end{bmatrix}\\ &\quad+\frac{2}{N}\begin{bmatrix}-\frac{1}{\sqrt{2}}g(k,t)+(-1)^{n+1}g(k,t+N)\\ 0\end{bmatrix}.\end{aligned}\tag{34}$$

Let us define

$$\mathbf{R}(2n+1) = \begin{bmatrix}\cos\frac{(2n+1)\pi}{N} & \sin\frac{(2n+1)\pi}{N}\\ -\sin\frac{(2n+1)\pi}{N} & \cos\frac{(2n+1)\pi}{N}\end{bmatrix}.\tag{35}$$

Then (29) can be expressed as

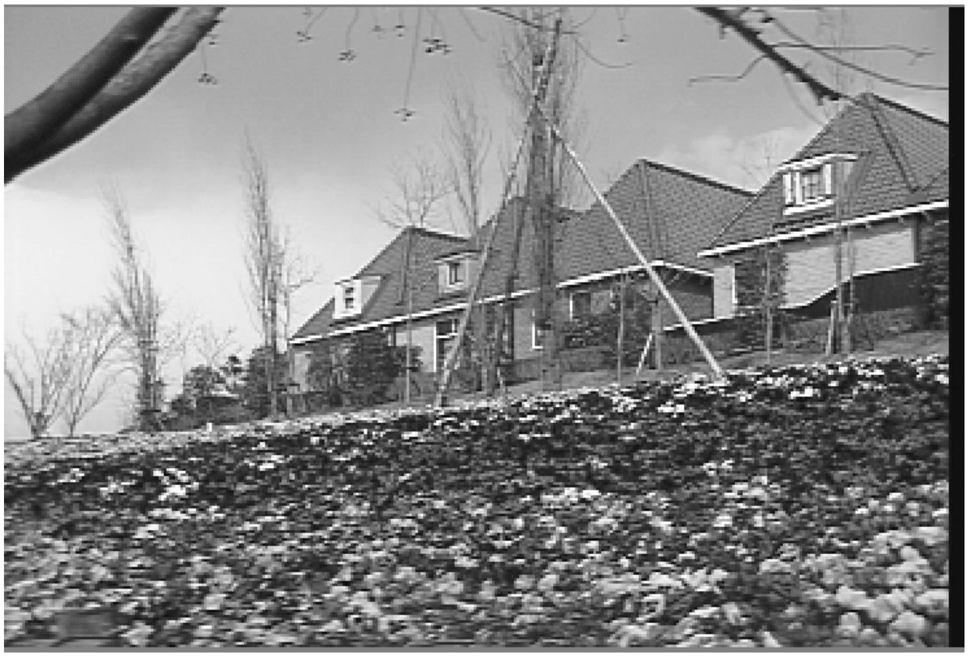
$$\begin{aligned}&\begin{bmatrix}G^c_{1,t+2}(l)\\ G^{as}_{1,t+2}(l)\end{bmatrix}\\ &= \mathbf{R}(2n+2)\begin{bmatrix}\bar{G}^c_{1,t+2}(l)\\ \bar{G}^{as}_{1,t+2}(l)\end{bmatrix}\\ &= \mathbf{R}(2n+2)\\ &\quad\times\begin{bmatrix}G^c_{1,t}(l)-\frac{2}{N}\frac{1}{\sqrt{2}}g(k,t)+(-1)^{n+1}\frac{2}{N}g(k,t+N)\\ G^{as}_{1,t}(l)\end{bmatrix}\\ &\quad+\frac{2}{N}\begin{bmatrix}\left(\frac{1}{\sqrt{2}}-1\right)g(k,t+2)\\ 0\end{bmatrix}+\frac{2}{N}\mathbf{R}(n+1)\\ &\quad\times\begin{bmatrix}(-1)^{n+1}g(k,t+N+1)-g(k,t+1)\\ 0\end{bmatrix}.\end{aligned}\tag{36}$$

By following the similar procedure, we can express $G^s_{1,t+2}(l)$ in terms of $G^{as}_{1,t+2}(l)$

$$\begin{aligned}G^s_{1,t+2}(l) &= G^{as}_{1,t+2}(l)\cos\left[\frac{\pi}{N}(n+1)\right]\\ &\quad+\left[G^c_{1,t+2}(l)+\frac{2}{N}\left(1-\frac{1}{\sqrt{2}}\right)g(k,t+2)\right]\\ &\quad\times\sin\left[\frac{\pi}{N}(n+1)\right].\end{aligned}$$

In summary,

$$\begin{cases}\begin{bmatrix}G^c_{1,t+2}(l)\\ G^{as}_{1,t+2}(l)\end{bmatrix}\\ \quad= \mathbf{R}(2n+2)\\ \qquad\times\begin{bmatrix}G^c_{1,t}(l)-\frac{2}{N}\frac{1}{\sqrt{2}}g(k,t)+(-1)^{n+1}\frac{2}{N}g(k,t+N)\\ G^{as}_{1,t}(l)\end{bmatrix}\\ \qquad+\frac{2}{N}\begin{bmatrix}\left(\frac{1}{\sqrt{2}}-1\right)g(k,t+2)\\ 0\end{bmatrix}\\ \qquad+\frac{2}{N}\mathbf{R}(n+1)\\ \qquad\times\begin{bmatrix}(-1)^{n+1}g(k,t+N+1)-g(k,t+1)\\ 0\end{bmatrix}\\ G^s_{1,t+2}(l)\\ \quad= G^{as}_{1,t+2}(l)\cos\left[\frac{\pi}{N}(n+1)\right]\\ \qquad+\left[G^c_{1,t+2}(l)+\frac{2}{N}\left(1-\frac{1}{\sqrt{2}}\right)g(k,t+2)\right]\\ \qquad\times\sin\left[\frac{\pi}{N}(n+1)\right].\end{cases}\tag{37}$$


(a)

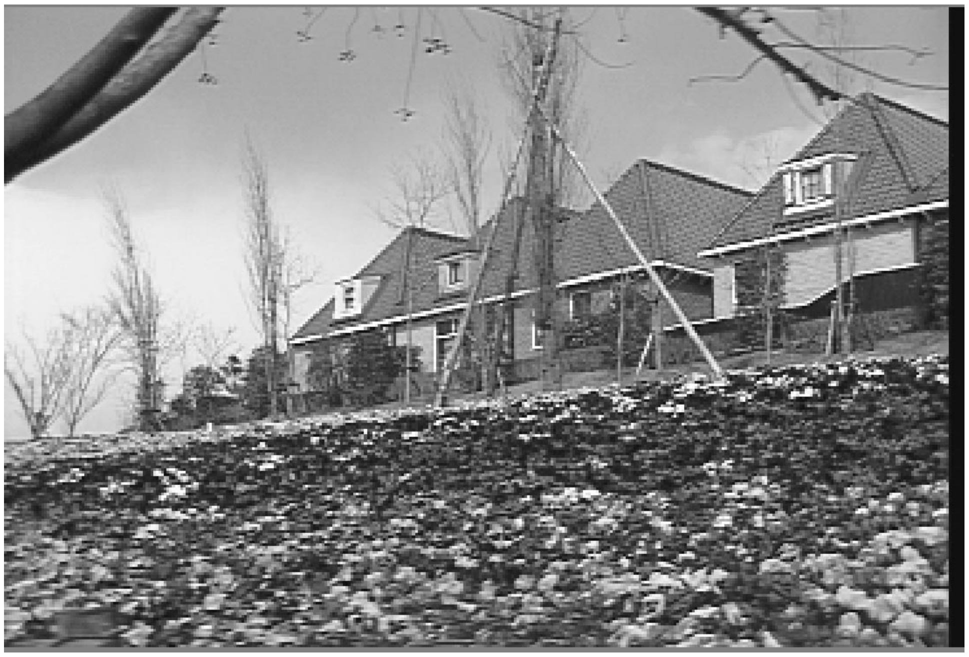

(b)

Fig. 13. "Flower Garden" frame 57: (a) original frame and (b) reconstructed frame.

TABLE III
SPEED-UP FACTORS OF DIFFERENT MODULES IN OUR DESIGN

Unit	Look-ahead type-II DCT/ IDCT coder	Pipelining DCT coefficients conversion	Multirate pseudo-phase computation	Pipelining peak search	Look-ahead half-pel estimator
Speed-up factor	1.87	1.95	1.81	1.96	1.85

With the same approach, we can get two-stage look-ahead updated relations for other phases as follows:

$\boxed{\text{For phase } (l\pi/N)(n+(1/2)) \text{ or } \beta=(1/2)}$

$$\left\{\begin{array}{l}
\begin{bmatrix} G^c_{\frac{1}{2},t+2}(l) \\ G^{as}_{\frac{1}{2},t+2}(l) \end{bmatrix} \\
\quad = \mathbf{R}(2n+1)\begin{bmatrix} G^c_{\frac{1}{2},t}(l) - \frac{2}{N}\frac{1}{\sqrt{2}}g(k,t) \\ G^{as}_{\frac{1}{2},t}(l) + (-1)^n\frac{2}{N}g(k,t+N) \end{bmatrix} \\
\quad +\frac{2}{N}\begin{bmatrix} \left(\frac{1}{\sqrt{2}}-1\right)g(k,t+2) \\ 0 \end{bmatrix} \\
\quad +\frac{2}{N}\mathbf{R}\left(n+\frac{1}{2}\right)\begin{bmatrix} -g(k,t+1) \\ (-1)^n g(k,t+N+1) \end{bmatrix} \\
G^s_{\frac{1}{2},t+2}(l) \\
\quad = G^{as}_{\frac{1}{2},t+2}(l)\cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)\right] \\
\quad + \left[G^c_{\frac{1}{2},t+2}(l) + \frac{2}{N}\left(1-\frac{1}{\sqrt{2}}\right)g(k,t+2)\right] \\
\quad \times \sin\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)\right] \\
\quad +(-1)^n\frac{2}{N}\left(\frac{1}{\sqrt{2}}-1\right)g(k,t+N+1).
\end{array}\right. \tag{38}$$

$\boxed{\text{For phase } (l\pi/N)n \text{ or } \beta=0}$

$$\left\{\begin{array}{l}
\begin{bmatrix} G^c_{0,t+2}(l) \\ G^{as}_{0,t+2}(l) \end{bmatrix} \\
\quad = \mathbf{R}(2n) \\
\quad \times \begin{bmatrix} G^c_{0,t}(l) + (-1)^n\frac{2}{N}g(k,t+N) - \frac{2}{N}\frac{1}{\sqrt{2}}g(k,t) \\ G^{as}_{0,t}(l) \end{bmatrix} \\
\quad +\frac{2}{N}\begin{bmatrix} \left(\frac{1}{\sqrt{2}}-1\right)g(k,t+2) \\ 0 \end{bmatrix} \\
\quad +\frac{2}{N}\mathbf{R}(n)\begin{bmatrix} (-1)^n g(k,t+N+1) - g(k,t+1) \\ 0 \end{bmatrix} \\
G^s_{0,t+2}(l) \\
\quad = G^{as}_{0,t+2}(l)\cos\frac{\pi n}{N} + \left[G^c_{0,t+2}(l)\right. \\
\quad \left. + \frac{2}{N}\left(1-\frac{1}{\sqrt{2}}\right)g(k,t+2)\right]\sin\frac{\pi n}{N}.
\end{array}\right. \tag{39}$$

Those look-ahead updating equations (37), (38) and (39) are alike except for some minor differences in the data paths and the rotation angles. Therefore, we can combine them to obtain the following unified equation to simultaneously generate $G^c_{\beta,t+2}(l)$ and $G^s_{\beta,t+2}(l)$:

$$\begin{bmatrix} G^c_{\beta,t+2}(l) \\ G^{as}_{\beta,t+2}(l) \end{bmatrix} = \mathbf{J}\left[\begin{bmatrix} G^c_{\beta,t}(l) \\ G^{as}_{\beta,t}(l) \end{bmatrix} + \mathbf{H}\right] + \mathbf{L}\cdot\mathbf{M} + \frac{2}{N}\begin{bmatrix} \left(\frac{1}{\sqrt{2}}-1\right)g(k,t+2) \\ 0 \end{bmatrix}. \tag{40}$$

The auxiliary variable $G^{as}_{\beta,t+2}(l)$ is related to $G^s_{\beta,t+2}(l)$ by

$$\begin{bmatrix} G^s_{\beta,t+2}(l) \\ \star \end{bmatrix} = \mathbf{L}\begin{bmatrix} G^{as}_{\beta,t+2}(l) \\ G^c_{\beta,t+2}(l) + \frac{2}{N}\left(1-\frac{1}{\sqrt{2}}\right)g(k,t+2) \end{bmatrix} + \mathbf{N}. \tag{41}$$

The corresponding parameters $\mathbf{H}, \mathbf{J}, \mathbf{L}, \mathbf{M}$ and $\mathbf{N}$ in (40) and (41) depending on the different phases are listed in Table II. The unified programmable module requires three CORDIC's, as shown in Fig. 10.

Based on the previous assumption, we get the ratio of power consumption of look-ahead design, $P_{\text{look-ahead}}$, to the power of original design, $P_{\text{half-pel}}$, as follows:

$$\frac{P_{\text{look-ahead}}}{P_{\text{half-pel}}} = \frac{C_{\text{look-ahead}}}{C_{\text{half-pel}}}\left(\frac{3.08\text{ V}}{5\text{ V}}\right)^2\frac{\frac{1}{2}f}{f} = 0.28$$

where $C_{\text{look-ahead}} \approx (3/2)C_{\text{half-pel}}$. Therefore, we can achieve 72% power saving at the expense of 50% hardware overhead.

$N+3$ such programmable modules can be used for parallel computing of $\overline{\text{DSC}}(u,v)$ for different channels (k,l), as shown in Fig. 11. The peak position among those values indicates half-pel motion vector. Overall the two-stage look-ahead half-pel motion estimator needs a total of $3N+9$ CORDIC's and $7N+33$ adders, as listed in Table IV.

TABLE IV
HARDWARE COST AND THROUGHPUT OF LOW-POWER/HIGH-SPEED DESIGN

Component	CORDICs	Adders	Registers	Throughput
Type-II DCT/IDCT	9N	27N+12	$N+6N^2$	$O(N)$
Type Conversion	4N	0	0	$O(N)$
Pseudo Phase	20N	2N	0	$O(N)$
Peak Searching	0	0	$2N^2$	$O(N)$
Half-pel Motion Estimator	3N+9	7N+33	$3N+N^2$	$O(N)$
Total	36N+9	36N+45	$4N+9N^2$	$O(N)$

III. SIMULATION RESULTS AND HARDWARE COST

We implement the low-power architectures for video coding co-processor using both C and Verilog. Simulations are made to verify the behavior of our design by taking "Miss America" and "Flower Garden" etc. as the test sequences. The original frames and reconstructed frames using our proposed low-power design are shown in Fig. 12(a) and (b) and Fig. 13(a) and (b), respectively. The simulation results demonstrate that our low-

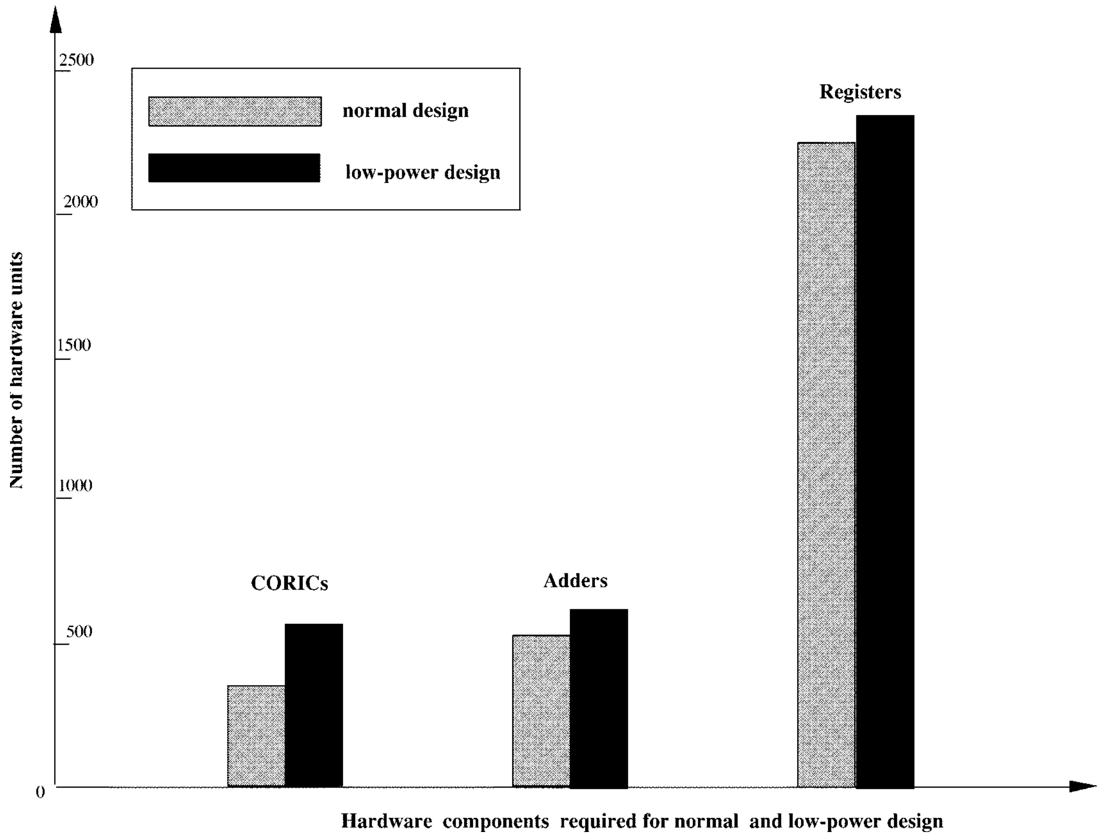


Fig. 14. Hardware cost for both low-power and normal design. Here we use block size $N = 16$.

TABLE V
POWER CONSUMPTION OF DIFFERENT MODULES IN OUR NORMAL AND LOW-POWER DESIGNS

Unit	Look-ahead type-II DCT/ IDCT coder	Pipelining DCT coefficients conversion	Multirate pseudo-phase computation	Pipelining peak search	Look-ahead half-pel estimator
Normal design	$63.67mw$	$17.31mw$	$46.17mw$	$1.65mw$	$18.98mw$
Low-power design	$22.06mw$	$4.27mw$	$19.08mw$	$0.36mw$	$6.67mw$
Power-saving factor	65.36%	75.33%	58.67%	78.18%	64.86%

power design can achieve comparable video quality as the original ones. We also compare the speed of normal and our low-power/high-speed design of individual module in Fig. 3. Here the simulation is performed at the gate level with 0.8 μm CMOS technology. The speed-up factors are listed in Table III. Based on the simulation results, we observe that our design can operate at about two-time faster clock rate than the original design, which is corresponding to our discussion/derivations in Section II.

To process the video sequence, each frame is divided into nonoverlapped block which contains $N \times N$ pixels as input to our low-power/high-speed design. The hardware cost and throughput of each building block in Fig. 3 to process those blocks are summarized in Table IV. Our design is flexible and scalable because it requires $33N$ CORDIC processors, $29N + 12$ adders to get motion vectors at integer-pel accuracy and additional $3N + 9$ CORDIC's, $7N + 33$ adders for those at half-pel accuracy. The number of hardware components needed in low-power design compared to that in original design [14], [15] is also plotted in Fig. 14.

A common question is: "How good is our design compared to those traditional block matching motion estimation methods?" Without low-power design, we have implemented our combined motion estimation and DCT units into a single chip [12]. The video coding system works at 20 MHz clock rate under 5 V supply voltage with 0.8 μm CMOS technology. By comparing to the traditional (full search or hierarchical search) motion estimation approaches [7], [24]–[30], our design are smaller than or about the same as those block-matching designs with respect to both power and area. It is important to note that our chip naturally accommodates both DCT and motion estimation units while the others may require multiple chips. By applying the *Powermill* developed by Synopsys, we compare the power consumption of different modules in our designs under both normal and low-power operating conditions as listed in Table V. Our low-power design operates at 10 MHz at 3.3 V supply voltage. It achieves the same throughput rate 157.82 Mbps as the normal design with the same 0.8 μm CMOS technology. From the simulation results, we observe the following: Compared to the normal design, our low-power design achieves the power saving of 64.51% with the power consumption of 52.44 mw. It corresponds to our previous calculations that our low-power design saves the power at the range of 60%–80% at the cost of much less than doubled hardware complexity, as shown in Fig. 14. As a result, low-power design at the *algorithm/architecture* level is the most leveraged way to achieve low-power consumption when both effectiveness and cost are taken into consideration.

IV. Conclusion

Anticipating the future trend of running multimedia applications on the portable personal devices, we propose the cost-effective low-power/high-speed co-processor architectures for video coding systems. Unlike the low-power video codec design using the costly advanced deep submicron fabrication technology, our low-power design is achieved at the algorithmic/architectural levels. Our emphasis is on optimizing power consumption by minimizing computational units along the data path. Compared with other approaches, our algorithmic/architectural low-power approach is one of the most economical ways to save power. Techniques such as look-ahead, multirate, pipelining have been used in our design. Based on the calculation and simulation results, our power saving is in the range of 60%–80% or speed-up factor of two at the needs of users.

References

[1] K. Hasegawa and K. Ohara *et al.*, "Low-power video encoder/decoder chip set for digital VCR's," *IEEE J. Solid-State Circuits*, vol. 31, pp. 1780–1788, Nov. 1996.

[2] T. Xanthopoulos and A. P. Chandrakasan, "A low-power IDCT macrocell for MPEG-2 MP@ML exploiting data distribution properties for minimal activity," *IEEE J. Solid-State Circuits*, vol. 34, pp. 693–703, May 1999.

[3] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. New York: Wiley, 1999.

[4] T. Xanthopoulos, Y. Yaoi, and A. Chandrakasan, "Architectural exploration using Verilog-based power estimation: A case study of the IDCT," in *Proc. 1997. Design Automation Conf. 34th DAC*, vol. 2, 1997, pp. 415–420.

[5] A. P. Chandrakasan and R. W. Brodersen, *Low Power Digital CMOS Design*. Norwell, MA: Kluwer, 1995.

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

[7] K. K. Chan and C.-Y. Tsui, "Exploring the power consumption of different motion estimation architectures for video compression," in *Proc. 1997 IEEE Int. Symp. Circuits and Systems*, July 1997, pp. 1217–1220.

[8] A. Kojima, N. Sakurai, and J. Kishigami, "Motion detection using 3D-FFT spectrum," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, Apr. 1993, pp. V213–V216.

[9] B. Porat and B. Friedlander, "A frequency domain algorithm for multiframe detection and estimation of dim targets," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, pp. 398–401, Apr. 1990.

[10] U. V. Koc and K. J. R. Liu, "Dct-based motion estimation," *IEEE Trans. Image Processing*, vol. 7, pp. 948–965, July 1998.

[11] ——, "Interpolation-free subpixel motion estimation techniques in dct domain," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, pp. 460–487, Aug. 1998.

[12] J. Chen and K. J. R. Liu, "Efficient architecture and design for an embedded video coding engine," , submitted for publication.

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

[14] J. Chen and K. J. R. Liu, "A complete pipelined parallel CORDIC architecture for motion estimation," *IEEE Trans. Circuits Syst. II*, vol. 45, pp. 653–660, May 1998.

[15] ——, "A fully pipelined parallel CORDIC architecture for half-pel motion estimation," in *Proc. IEEE Int. Conf. Image Processing*, vol. 2, Santa Barbara, CA, Oct. 1997, pp. 574–577.

[16] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*. Englewood Cliffs, NJ: Prentice-Hall, 1993.

[17] A.-Y. Wu, K. J. R. Liu, Z. Zhang, K. Nakajima, and A. Raghupathy, "Low-power design methodology for DSP systems using multirate approach," in *Proc. IEEE Int. Symp. Circuits and Systems*, Atlanta, GA, May 1996, pp. IV.292–295.

[18] Z. Wang, "Fast algorithms for the discrete W transform and for the discrete Fourier transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 803–816, Aug. 1984.

[19] B. Girod, "Why B-pictures work: A theory of multi-hypothesis motion-compensated prediction," in *Proc. 1998 IEEE Int. Conf. Image Processing*, Oct. 1998, pp. 213–217.

[20] S.-L. Iu, "Comparison of motion compensation using different degrees of sub-pixel accuracy for interfield/interframe hybrid coding of HDTV image sequences," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, vol. 3, 1992, pp. 465–468.

[21] 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.

[22] 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.

[23] A. V. Oppenheim and R. W. Schafer, *Discret-Time Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[24] R. Dianysian and R. L. Baker, "Bit-serial architecture for real-time motion compensation," *Proc. SPIE Visual Commun. and Image Proc.*, vol. 1001, no. 2, pp. 900–907, Oct. 1988.

[25] K.-M. Yang, M.-T. Sun, and L. Wu, "A family of VLSI designs for the motion compensation block-matching algorithms," *IEEE Trans. Circuits. Syst. II*, vol. 36, pp. 1317–1325, Oct. 1989.

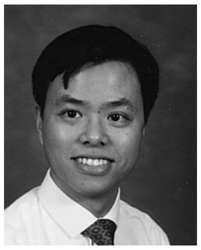
[26] O. Colavin, A. Artieri, J. F. Naviner, and R. Pacalet, "A dedicated circuit for real-time motion estimation," presented at the EuroASIC, Feb. 1991.

[27] P. Ruetz, P. Tong, D. Bailey, D. Luthi, and P. Ang, "A high-performance full-motion video compression chip set," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, June 1992.

[28] S. Uramoto *et al.*, "A half-pel precision motion estimation processor for ntsc-resolution video," *IEICE Trans. Electron.*, vol. E-77, no. 12, pp. 1930–1936, Dec. 1994.

[29] Y. Tokuno *et al.*, "A motion video compression LSI with distributed arithmetic architecture," presented at the IEEE Custom Integ. Circ. Conf., June 1993.

[30] R. Hervigo, J. Kowalczuk, and D. Mlynek, "A multiprocessor architecture for HDTV motion estimation system," *IEEE Trans. Consumer Electron.*, vol. 38, pp. 690–697, Aug. 1992.



Jie Chen (S'95–M'97) received the B.S. degree in electrical engineering and the M.S. degree in physics from Fudan University, Shanghai, China, in 1987 and 1990, respectively, and the M.S. and Ph.D. degrees in electrical engineering from the University of Maryland, College Park, in 1992 and 1998, respectively.

From 1992 to 1995, he was with Hughes Network System research group, Germantown, MD, and participated in TDMA and wireless local-loop system design and implementation. Later, he joined the working group of ITU-T third-generation CDMA standard. From 1998 to 2000, he was with Bell Laboratories, Lucent Technologies, Murray Hill, NJ, in research and development of optical network integrated circuits. He currently is Chief Hardware Architect and System Engineer of a start-up company, working on digital audio broadcasting. His research interests include multimedia signal processing, broadband networking, and wireless communications. He has published over 15 papers and holds two U.S. patents. He is the co-author of the book *Compressed Domain Video Coding System: Algorithm, Architecture and Co-design* (Marcel Dekker Inc.). His web page is http://dspserv.eng.umd.edu/alumni/chenjie.html.

Dr. Chen is a guest editor of a Special Issue on Multimedia over IP of the IEEE Transactions on Multimedia. He is the special chair of Multimedia over Networks in the International Symposium of Circuits and System'2000 (ISCAS'2000). He also served as technical member of multimedia and communications committee for ISCAS'2000.

K. J. Ray Liu (S'86–M'90–SM'93) received the B.S. degree from the National Taiwan University, Taipei Taiwan, R.O.C., in 1983, and the Ph.D. degree from the University of California, Los Angeles, in 1990, both in electrical engineering.

Since 1990, he has been with Electrical Engineering Department and Institute for Systems Research, University of Maryland, College Park, where he is a Professor. During his sabbatical leave in 1996–1997, he was Visiting Associate Professor at Stanford University, Stanford, CA. His research interests span various aspects of signal/image processing and communications. He has published over 180 papers, of which over 65 are in archival journals and book chapters. He is the co-editor of the books, *High Performance VLSI Signal Processing: Volume I: System Design and Methodology; and Vol II: Algorithms, Architectures, and Applications* (New York: IEEE Press, 1998).

Dr. Liu has received numerous awards, including the 1994 National Science Foundation Young Investigator Award, the IEEE Signal Processing Society's 1993 Senior Award (Best Paper Award), the IEEE Benelux Joint Chapter on Vehicular Technology and Communications 1999 Award, the George Corcoran Award in 1994 for outstanding contributions to electrical engineering education and the 1995–96 Outstanding Systems Engineering Faculty Award in recognition of outstanding contributions in interdisciplinary research, both from the University of Maryland, and many others. He was an Associate Editor of IEEE TRANSACTIONS ON SIGNAL PROCESSING, a Guest Editor of special issues on Multimedia Signal Processing of the PROCEEDINGS OF THE IEEE, and a Guest Editor of special issue on Signal Processing for Wireless Communications of IEEE JOURNAL OF SELECTED AREAS IN COMMUNICATIONS. He currently serves as the Chair of Multimedia Signal Processing Technical Committee of IEEE Signal Processing Society, a Guest Editor of a Special Issue on Multimedia over IP of the IEEE TRANSACTIONS ON MULTIMEDIA, a Guest Editor of Special Issue on Robust Multimedia Transmission of the IEEE SIGNAL PROCESSING MAGAZINE, and editor of the JOURNAL OF VLSI SIGNAL PROCESSING SYSTEMS, and the series editor of Marcel Dekker series on signal processing.