Algorithm-Based Low-Power and High-Performance Multimedia Signal Processing

K. J. RAY LIU, SENIOR MEMBER, IEEE, AN-YEU WU, MEMBER, IEEE,
ARUN RAGHUPATHY, STUDENT MEMBER, IEEE, AND JIE CHEN, STUDENT MEMBER, IEEE

Invited Paper

Low power and high performance are the two most important criteria for many signal-processing system designs, particularly in real-time multimedia applications. There have been many approaches to achieve these two design goals at many different implementation levels ranging from very-large-scale-integration fabrication technology to system design. However, the major drawback is that present approaches are either too costly or not efficient enough. In this paper, we review the works that have been done at various levels and focus on the algorithm-based approaches for low-power and high-performance design of signal-processing systems. We present the concept of multirate computing that originates from filterbank design, then show how to employ it along with the other algorithmic methods to develop low-power and high-performance signal-processing systems. The proposed multirate design methodology is systematic and applicable to many problems. We demonstrate in this paper that multirate computing is a powerful tool at the algorithmic level that enables designers to achieve either significant power reduction or high throughput depending on their choice. Design examples on basic multimedia processing blocks such as filtering, source coding, and channel coding are given. A digital-signal-processing engine that is an adaptive reconfigurable architecture is also derived from the common features of our approach. Such an architecture forms a new generation of high-performance embedded signal processor based on the adaptive computing model. The goal of this paper is to demonstrate the flexibility and effectiveness of algorithm-based approaches and to show that the multirate approach is an effective and systematic design methodology to achieve low-power and high-throughput signal processing at the algorithmic and architectural level.

***Keywords*—** *Adaptive filters, discrete cosine transforms, FIR digital filters, IIR digital filters, low-power VLSI design, motion estimation, multirate processing, parallel architectures, Reed–Solomon codes, video coding, Viterbi decoding.*

Manuscript received July 14, 1997; revised December 17, 1997. The Guest Editor coordinating the review of this paper and approving it for publication was T. Chen. This work was supported in part by the Office of Naval Research under Grant N00014-93-10566 and the National Science Foundation under NYI Award MIP9457397.

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

A.-Y. Wu is with the Electrical Engineering Department, National Central University, Chung-li, Taiwan R.O.C.

Publisher Item Identifier S 0018-9219(98)03522-1.

I. INTRODUCTION

Multimedia—a versatile presentation of speech, audio, video, graphics, and text—has become the major theme in today's information technology that merges traditional giants of communication, computing, and information processing into an interdisciplinary field. Nowadays, people enjoy multimedia in almost every aspect of their daily life such as communication, data search, presentation, advertisement, games, etc. As a consequence, it has been emerging as a new technology that has changed and is still changing our lifestyle. One major feature of multimedia data processing is that it calls for computationally intensive data processing at millions to billions of operations per second. Hence, how to handle the demanding computational tasks in real time and how to implement them in a cost-effective way have become a big challenge.

On the other hand, with the advent of personal communications services (PCS's) and personal data assistant (PDA's), the future trend is to run multimedia applications on those portable devices. The need for high-speed data/signal processing will lead to much higher power consumption than traditional portable applications. Due to the limited power-supply capability of current battery technology, we are facing a dilemma that has two design problems to conquer, and they are considered to be problems of opposite natures. One is to explore high-performance design and implementation that can meet the stringent speed constraint for real-time multimedia applications. The other is to consider low-power design so as to prolong the operating time of the PCS/PDA devices. The following question can be posed. Can we design the system smartly so that we can develop a solution that can conquer both problems depending on the needs of the users? We will answer this question by working at the algorithmic level.

In this paper, we will review the design of low-power and high-performance multimedia signal-processing systems and focus on the treatment of algorithmic and ar-

chitectural level approaches. We will introduce a new algorithm-based low-power and high-performance design technique—*the multirate approach*—and combine it along with other digital signal processing (DSP) techniques such as look-ahead and folding to design several major multimedia DSP processing blocks, including digital finite/infinite impulse response (FIR/IIR) filtering, source/channel coding processing kernels for low-power/high-speed multimedia systems. The resulting very-large-scale-integration (VLSI) architectures can perform the DSP functions at M times slower operating frequency while retaining the same data throughput rate. This feature can help achieve significant power saving under low supply voltage without loss of speed performance. Also, the proposed VLSI architectures can be readily applied to high-speed signal processing with a speedup factor of M.

In what follows, we first briefly review existing design approaches as well as DSP techniques that achieve low-power/high-performance data processing. Then we will introduce the concept of multirate computing and other low-power design methodologies.

A. Low-Power VLSI Design Approaches

The power dissipation in a well-designed digital complementary metal–oxide–semiconductor (CMOS) circuit can be modeled as [1]

$$P \approx \alpha \cdot C_{\text{eff}} \cdot V_{\text{dd}}^2 \cdot f_{\text{clk}} \tag{1}$$

where α is the average fraction of the total node capacitance being switched (also referred to as the activity factor), C_{eff} is the effective loading capacity, V_{dd} is the supply voltage, and f_{clk} is the operating frequency. On the other hand, the delay of the CMOS device can be approximated as

$$T_D \approx \frac{C_L \cdot V_{\text{dd}}}{I} = \frac{C_L \cdot V_{\text{dd}}}{\epsilon(V_{\text{dd}} - V_t)^2} \tag{2}$$

where C_L is the capacitance along the critical path, ϵ is the device parameter, and V_t is the threshold voltage of the devices. Equations (1) and (2) play the essential roles in low-power VLSI designs. Namely, in order to lower the total power consumption of the CMOS circuits, we want to reduce the values of α, V_{dd}, C_{eff}, and f_{clk} by applying all possible techniques at all levels of the VLSI system without sacrificing T_D when those parameters change [2]–[6]. The existing low-power design approaches are summarized below.

1) Device/VLSI Technology Level: Over the last decade, the CMOS feature size has been reduced from more than 3 μm to about 0.25 μm now. The advance in integrated circuit (IC) fabrication technology also leads to low-power design. Smaller transistor size not only improves the device/circuit speed performance but also reduces the total silicon area and capacitances and hence the total power consumption. Besides, the increased level of integration allows designers to use the vacated area for extra circuits to compensate for the degraded device speed due to lowered supply voltage (as we will discuss later).

In addition to the reduction of the feature size, lowering the threshold voltage V_t is another commonly used approach to achieve low-power consumption at the technology level [3], [7]. From (2), we can see that the delay of the circuit is inversely proportional to $(V_{\text{dd}} - V_t)^2$. Thus, it is desirable to reduce the magnitude of V_t either to minimize the degradation of speed caused by lowered V_{dd} or to allow further reduction in V_{dd}.

On the other hand, it is predicted that 1.5 V (or lower) operation will be needed by the year 2001 for portable products [5]. Since the supply voltage of the conventionally scaled CMOS technology will reach its limit at a supply voltage of 1.5 V, an alternate process technology such as silicon-on-insulator (SOI) is suggested to replace CMOS technology [5], [8]. The SOI technology allows power supply reduction to 1 V or less and also greatly simplifies the fabrication process. Those merits have made SOI the best candidate for future low-power fabrication technology. Nevertheless, the cost of the technology/device approach is the most expensive among all low-power techniques since it requires the investment of new semiconductor equipment and technology. Furthermore, it takes time for a fabrication technology to be mature enough for mass industrial applications.

2) Circuit Approach: High-speed/low-power VLSI operators (adders and multiplies) are usually developed at the circuit/gate levels (e.g., look-ahead adders, Wallace-tree multiplier, etc. [9]). By employing some special DSP treatments of the basic VLSI operators, the hardware cost and the speed performance can be further improved. One example is *distributed arithmetic* [10], which is widely used in discrete cosine transform (DCT) chip designs [11]–[13]. Other approaches include a *residue number system* that performs multiplication in the modulo domain [14]–[16], *power-of-two coefficients* that employ only adders to perform multiplication [17]–[19], and a *delta-sigma modulator* that makes one bit quantization feasible in the analog–digital, digital–analog circuit designs [20]. Each approach provides different tradeoffs in terms of the power/speed/area of the design.

At the CMOS circuit level, various circuit design techniques are also available; e.g., dynamic versus static CMOS logic, conventional static versus pass-transistor logic, and synchronous versus asynchronous design [2]. As far as power consumption is concerned, the static CMOS logic and asynchronous design are preferable because their switching activities are less. The pass-transistor logic family is also a promising candidate since it uses lesser number of transistors when compared with conventional static CMOS circuits for implementing the same logic function [1, ch. 5]. Recently, the low-power digital circuits based on adiabatic-switching technique were introduced [21]. By employing the adiabatic-switching circuits, the signal energies stored on circuit capacitances can be recycled instead of dissipated as heat, which provides a promising power-saving technology at the circuit level.

3) Logic-Level Approach: In CMOS circuits with negligible leakage current, power is dissipated only when there

is a transition at the output of the gate (zero to one or one to zero in logic value). In the logic-level, low-power design, the major focus is to reduce the frequency of energy consumed in transitions for given logic functions, i.e., the activity factor α in (1). A variety of existing approaches can be found in [22]–[25]. They basically examine the given logic functions and perform logic optimization/synthesis in such a way that the total number of logic transitions can be minimized for most input signals. By doing so, α, and hence the total power consumption of the circuits, can be reduced. In general, the power saving of the logic approach is in the range of 20–75%.

4) Architectural/Algorithmic Approach: From (1), we can see that the reduction of $V_{\rm dd}$ is the leveraged way to reduce the total power consumption due to its quadratic dependence. However, the delay will drastically increase as $V_{\rm dd}$ approaches V_t [see (2)]. That is, we suffer from a *speed penalty* as $V_{\rm dd}$ is decreased. To meet the low-power/high-throughput constraint in most DSP applications, the key issue in algorithmic/architectural-level low-power design is to "compensate" for the increased delay caused by the lowered supply voltage. Current approaches to compensate for the increased delay include the techniques of "parallel processing" and "pipelining" [2], [6]. In this paper, we will present a new compensation technique based on the multirate approach, which will be discussed in detail later.

5) System-Level Approach: The system-level, low-power VLSI design evolves from the power-saving techniques that are frequently used in laptop/notebook computers as well as the energy-saving "green products." When one of the subsystems is idle for a period of time, it may switch to one of the modes—*doze*, *nap*, *sleep*—to save system power. Recent state-of-the-art central processing unit (CPU) designs have employed the same design concept to manage both dynamic and static power of the CPU [26]. The embedded activity-management circuit of the CPU provides the capability to shut down portions of subsystems that are not required in current or impending operations. Therefore, significant power saving can be achieved. There are two design issues involved in the system-level low-power designs. One is the partitioning of the system into submodules that have high interconnection density within themselves. By doing so, the influence of shutting down one submodule on other submodules can be minimized. For application-specific (AS)IC designs that use multichip modules (MCM's) implementation, the partitioning can be done by employing the systematic approach discussed in [27]. The other is the design of additional hardware/software for monitoring the working status of each submodule of the chip, which will introduce extra cost and weight to the system.

Among these low-power techniques, the algorithmic/architectural approach is the most promising one [6]. First, the algorithmic/architectural low-power design is achieved by reformulating the algorithms and mapping them to efficient low-power VLSI architectures to compensate for the speed penalty caused by low supply voltage. Basically, we only trade more silicon area for low-power consumption under current technology, without invoking dedicated circuit design, new expensive devices, and advanced VLSI fabrication technology. Compared with other approaches, the algorithmic/architectural low-power design is one of the most economical ways to save power. Second, the power saving of the algorithmic/architectural approach is in the range of 70–90%[1] (as we will show it in this paper). Therefore, the algorithmic/architectural-level approach provides the most leveraged way to achieve low-power consumption when both effectiveness and cost are taken into consideration.

B. Algorithm/Architecture-Based Low-Power/High-Performance Approaches

In the literature, there have been many existing research efforts to perform low-power/high-performance data processing at the architectural/algorithmic level for DSP applications. Basically, they can be categorized as follows.

1) Approximation Approach: Sometimes, the direct implementation of a given DSP algorithm is very costly and inefficient. To reduce the implementation cost, we may relax some constraint (or make some approximation) in the DSP algorithms. As long as the relaxed/approximation approaches yield acceptable performance in the statistical sense, they can generally lead to much simplified architectures, which then add to the saving of total implementation cost. For example, in motion estimation (ME) schemes, the three-step ME [28] is usually employed to reduce the computational complexity while maintaining a performance comparable with the full-search ME schemes [29]. The *split recursive least squares (RLS)* [30] algorithm is an approximate RLS that can perform RLS for nonstructured data and has only $O(N)$ complexity. Examples of approximate signal processing based on incremental refinement can be found in [31].

2) Model-Based Approaches: Model-based signal processing explores the inherent properties of signals, then uses only some parameters to represent the desired signal. Examples are linear prediction coding (LPC) [32] in digital speech processing and autoregressive (AR), moving average (MA), and ARMA models in stochastic data processing. Also, in image/video processing, the model-based approach has been used in very-low-bit-rate coding [33]. The advantage of such an approach is that it helps to capture the signal characteristics through those parameters (e.g., LPC coefficients are related to the vocal track model). On the other hand, it can be applied to efficient data transmission since only the parameters need to be transmitted. A good review of model-based signal processing can be found in [34].

3) Pipelining: Pipelining is the most frequently used technique to achieve high-speed data processing in VLSI signal processing. The main idea behind pipelining a design is to insert latches between m consecutive pipeline stages so that the delay through the critical path can be shortened

[1]The current goal is to reduce the total power dissipation of the electronics systems to two orders of magnitude less than what would have been with the conventional technology [5].

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 the latency. On the other hand, the pipelining can be used to compensate for the delay incurred in the low-power design when the supply voltage drops significantly. Some derivations of the pipelining techniques are asynchronous circuit pipelining [35], two-level (word/bit-level) pipelining [36], and bit-level pipelining approaches [37].

4) Parallel Processing: Parallel data processing is the technique that decomposes the desired DSP function into independent and parallel small tasks. Then the small tasks are executed concurrently, and individual results are combined together to obtain the desired computation result. It is a form of "divide-and-conquer" strategy in dealing with DSP problems. One famous example is the fast Fourier transform (FFT) [38]. It first decomposes an N-point FFT into two $N/2$-point FFT's, and the whole computation is expanded recursively to form a parallel computing network. A systolic array, on the other hand, is a parallel structure by nature. The goal of parallel processing is to utilize each processing element (PE) fully to achieve maximum data throughput rate. This feature is very suitable for high-speed data processing and VLSI implementations. The multirate approach proposed in this paper belongs to this category. We will show a systematic way to convert a given DSP algorithm into its equivalent multirate realization. The advantage of the multirate approach will be illustrated in the next section.

In summary, these design approaches are frequently used for building low-complexity, low-cost, high-throughput-rate DSP systems or for low-power implementation. In what follows, we will introduce the concept of look-ahead and multirate computing. These approaches provide a systematic and efficient way to achieve low-power/high-performance DSP computations at the algorithmic level.

C. Look-Ahead and Multirate Computing Concepts

As we discussed in the preceding subsection, "pipelining" and "parallel processing" are frequently used techniques to achieve high-speed data processing in VLSI design. These two techniques are also applied to architectural-level low-power design [2]. In [2], these two techniques were suggested to compensate for the speed penalty, and a simple comparator circuit was used to demonstrate how parallel independent processing of the data can achieve good compensation at the architectural level. In most DSP applications, however, the problems encountered are much more complex. It is almost impossible to directly decompose the problems into independent and parallel tasks. Therefore, the properties of the DSP algorithms should be fully exploited in order to develop efficient techniques to compensate for the loss of speed performance caused by low-voltage operations. The main issue here is to reformulate the algorithms so that the desired outputs can be obtained without hindering the system performance such as data throughput rate.

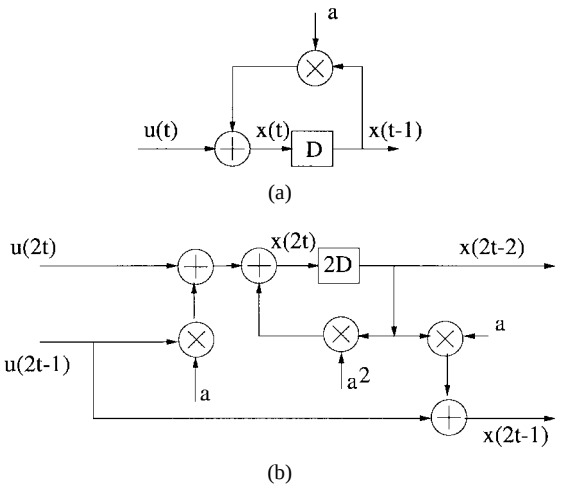


Fig. 1. (a) First-order recursive digital filter. (b) Two-stage look-ahead equivalent.

1) Look-Ahead: VLSI implementations of recursive algorithms are constrained by the speed of the feedback loop. The look-ahead transformation [39]–[41] modifies the algorithm so that the recursive bottleneck is eased. The speed-up provided by the application of look-ahead comes at the cost of increased hardware complexity. We will illustrate the idea of look-ahead using a simple first-order recursive filter [41]. The input-output equation can be written as

$$x(t) = ax(t-1) + u(t) \tag{3}$$

where $x(t)$ is the desired recursive output, $u(t)$ represents the current input, and a is a constant. The computations involved in (3) are shown in Fig. 1(a).

Applying a two-step look-ahead converts (3) into

$$x(t) = a^2x(t-2) + au(t-1) + u(t). \tag{4}$$

The look-ahead transformation in (4) can be exploited in many ways to obtain high-speed designs [39], [40]. Fig. 1(b) shows how the look-ahead transformation enables a block filter implementation. Note that there are two delay elements in the feedback loop in Fig. 1(b), whereas the feedback loop in Fig. 1(a) has only one delay element. Let the critical path propagation delay of the original circuit be T. The feed-forward computation in Fig. 1 (i.e., the computation of $u(2t) + au(2t-1)$ and the computation of $x(2t-1)$ from $x(2t-2)$) can be easily pipelined [41] so that the critical path of the modified design is determined by the feedback loop. Since the feedback loops in the original and look-ahead designs each consist of one multiplier and one adder, the critical path delay of the look-ahead circuit T' also equals T. In the look-ahead design, *two* outputs are computed in parallel. This means that the throughput of the look-ahead design is twice that of the original design. Alternately, low-power operation is possible with the look-ahead implementation. This low-power operation can be obtained while maintaining the same throughput as the original implementation. The supply voltage of the look-ahead implementation is scaled from $V_{\rm dd}$ down to $V'_{\rm dd}$.

The voltage $V'_{\rm dd}$ can be chosen so that the critical path delay, which is $T' = T$ at $V_{\rm dd}$, increases to $T'' = 2T$ at $V'_{\rm dd}$. This is because the look-ahead design can provide the same throughput as the original design while working at half the clock speed. The supply voltage $V'_{\rm dd}$ can be computed using the delay model (2) as

$$\frac{C_L V'_{\rm dd}}{\epsilon(V'_{\rm dd} - V_t)^2} = 2\frac{C_L V_{\rm dd}}{\epsilon(V_{\rm dd} - V_t)^2}. \tag{5}$$

Substituting $V_{\rm dd} = 5$ V and $V_t = 0.7$ V, we get $V'_{\rm dd} = 3.08$ V. Here, the capacitance along the critical path C_L remains the same after the look-ahead transformation. The ratio of the power consumption of the look-ahead implementation $P_{\rm la}$ to the power of the original implementation P_o can be written as

$$\frac{P_{\rm la}}{P_o} = \frac{C_{\rm la}}{C_o}\left(\frac{V'_{\rm dd}}{V_{\rm dd}}\right)^2 \frac{\frac{1}{2}f_o}{f_o} \tag{6}$$

where C_o and $C_{\rm la}$ represent the effective capacitances of the original and look-ahead designs, respectively, and f_o is the original operating frequency. If we neglect the complexity of latches, we can approximate the complexity of the look-ahead system to be thrice that of the original system (i.e., $C_{\rm la} \approx 3C_o$). We then obtain $P_{\rm la} = 0.57P_o$ or a power saving of 43%.

We have discussed the specific case of two-step look-ahead for a simple first-order recursive filter. In general, look-ahead by a factor of M may be applied to a recursive system to obtain an M-fold speed-up or a low-power design. Consider the following system:

$$\mathbf{x}(t+1) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t) \tag{7}$$

where $\mathbf{x}(t+1)$ represents the present state vector, $\mathbf{u}(t)$ is the input, and $\mathbf{A}(t)$ and $\mathbf{B}(t)$ are inputs that may or may not depend on external inputs. To achieve low-power design, we can apply an M-step look-ahead [42]. This is equivalent to expressing $\mathbf{x}(t+M)$ in terms of $\mathbf{x}(t)$, which leads to

$$\begin{aligned}\mathbf{x}(t+M) = &\left[\prod_{i=0}^{M-1}\mathbf{A}(t+i)\right]\mathbf{x}(t) \\ &+ \sum_{i=0}^{M-1}\left[\prod_{j=1}^{i}\mathbf{A}(t+M-j)\right] \\ &\cdot \mathbf{B}(t+M-i-1)\cdot\mathbf{u}(t+M-i-1).\end{aligned} \tag{8}$$

The corresponding power reduction can be found using arguments similar to those used above.

2) Multirate: In this paper, we propose a new technique—*the multirate approach*—to compensate the aforementioned speed penalty for low-power design. At the same time, it can be used for high-speed design as well. Traditionally, multirate signal processing has focused on perfect reconstruction systems that are widely used in subband coding-based compression of audio/video signals and in transmultiplexers that convert between time and frequency division multiplexing [43]. The goal in perfect reconstruction systems is to design the multirate filterbank so that the equivalent transfer function, when the analysis and synthesis are combined together, is a simple delay. Such a perfect reconstruction property can make a distortion-free data transmission in communication systems.

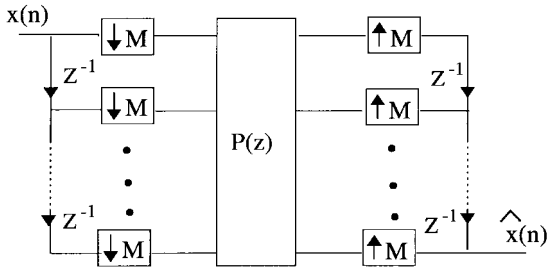


Fig. 2. An M-channel alias-free filterbank.

Our interest, on the other hand, is to apply the multirate network to realize a specific system-transfer function. That is, if the system distortion is equivalent to the desired system-transfer function, it becomes a computing system that produces the desired transfer function under the multirate environment. Hence, we can consider an alias-free M channel maximally decimated filterbank as a linear time-invariant (LTI) system [43, ch. 5]. If the filterbank needs to operate at a throughput of R, then the operators (that perform the computations indicated by $P(z)$ in Fig. 2) need to operate only at the rate of R/M. This means that by using slower operators that work at the rate of R/M, we can achieve low-power design while still maintaining the overall throughput rate R.

In this paper, we will show how this multirate filterbank idea can be generalized to a wide class of algorithms. Specifically, we will discuss how the algorithms can be modified/reformulated so that the actual operators can perform computation at a slower rate. Fig. 3 shows the generalized multirate approach for $M = 2$. The additional concurrency that is required in the reformulated algorithm can be obtained by applying appropriate transformations that take advantage of the properties of the particular algorithm. For example, let us consider the discrete cosine transform (DCT) architecture in Fig. 4. For most of the existing serial-input parallel-output (SIPO) DCT architectures [44], [45], the processing rate of the operators must be as fast as the input data rate [see Fig. 4(a)]. In employing multirate low-power design, the DCT is computed from the reformulated circuit using the decimated sequences [Fig. 4(b)]. It is now a multirate system that operates at two different sample rates. Since the operating speed of the processing elements is reduced to half of the original data rate while the data throughput rate is still maintained, the speed penalty is compensated at the architectural level. Suppose that the $C_{\rm eff}$ is approximately doubled due to the hardware overhead in the reformulated circuit. Since all the operations are at half of the original speed, the lowest possible voltage can be reduced from $V_{\rm dd} = 5$ V to $V'_{\rm dd} = 3.08$ V [from (5)]. Using the CMOS power dissipation model of (1), the overall power consumption of

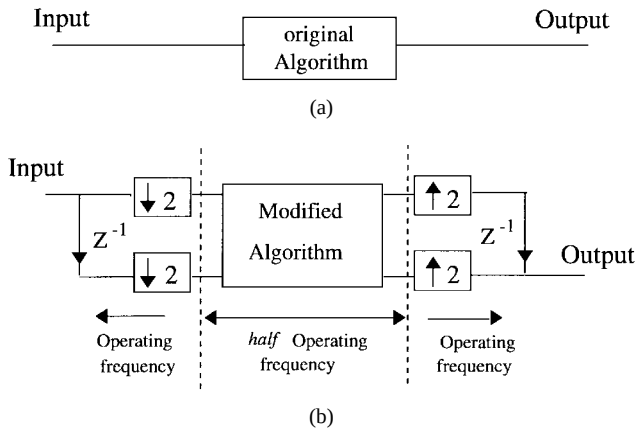


Fig. 3. The idea of multirate algorithm modification. (a) Original design. (b) Multirate design.

the multirate design can be estimated as

$$(2C_{\text{eff}})(V'_{\text{dd}})^2\left(\frac{1}{2}f\right) \approx 0.38P_0 \tag{9}$$

where P_0 denotes the power consumption of the original system.

We also show that the look-ahead idea can be used in conjunction with the multirate concept. Look-ahead can be applied to reformulate certain recursive algorithms within the multirate system (Fig. 3).

3) Dual Low-Power/High-Speed Features: In addition to low-power implementation, the other attractive application of the look-ahead and the proposed multirate architectures is in very-high-speed signal processing. In most VLSI designs, the input data rate is limited by the speed of the adders and multipliers in the circuit. In the video-rate multimedia applications, the speed constraint results in the use of expensive high-speed operators or full-custom designs. Thus, the manufacturing cost as well as the design cycle will increase drastically. Since the look-ahead and multirate architectures are running at an M times slower operating frequency than the input data rate, they can process data at a rate that is M times faster than the maximum speed of the processing elements. For example, if we want to perform DCT for serial data at 100 MHz, we may use the parallel architecture in Fig. 4, in which only 50-MHz adders and multipliers are required. Therefore, we can perform very-high-speed DCT by using only low-cost and low-speed processing elements. Hence, by employing the look-ahead and multirate parallel architectures discussed in this paper, the above-mentioned speed constraint can be resolved at the architectural level with the same design environment and fabrication technology.

D. Paper Outline

In this paper, we will demonstrate the effectiveness of the algorithm-based approach based on the look-ahead, pipelining, parallel processing, and multirate approaches. We will use several major multimedia DSP processing blocks as examples, including digital DSP and FIR/IIR filtering, source/channel coding kernels for low-power/high-speed

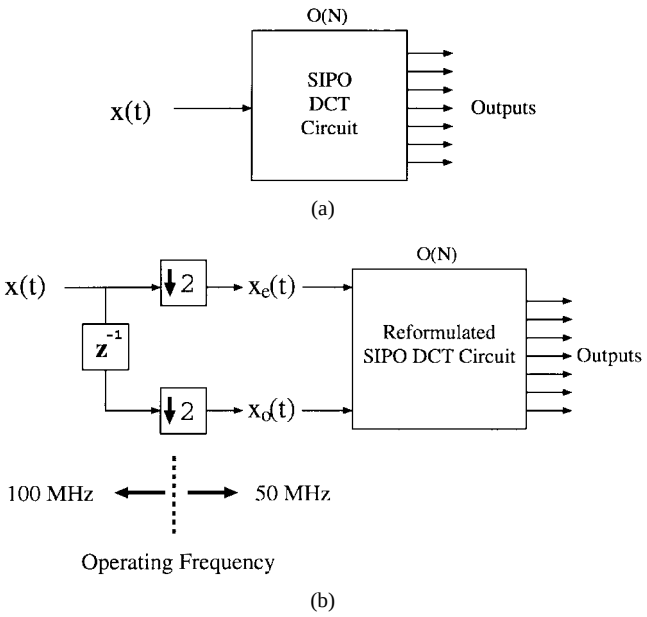


Fig. 4. (a) Original SIPO DCT circuit. (b) Low-power DCT circuit using the multirate approach.

multimedia systems, and adaptive filtering. We will also perform a finite-precision analysis and develop a DSP computing engine. The topics that we will discuss are summarized below:

1) DSP and Filtering: We present a systematic design methodology for the multirate realization of a given arbitrary LTI DSP system. The users can simply follow the design steps to convert a speed-demanding system function into its equivalent multirate transfer function. Since the data rate in the resulting multirate filtering architecture is M times slower than the original data rate while maintaining the same throughput rate, we can apply this feature either to low-power implementation or to the speed-up of the DSP systems. The proposed design methodology provides VLSI designers with a systematic tool to design low-power DSP systems at the algorithmic/architectural level. Furthermore, it can be incorporated into the design of high-level synthesis computer-aided-design (CAD) tools for power minimization.

2) Video and Data Compression: The DCT and ME are two major building blocks in most current video standards such as Motion Pictures Experts Group (MPEG)-1, MPEG-2, H.261, and H.263. We will therefore use DCT and ME as examples to illustrate the concept of algorithm-based low-power and high-speed design for source coding.

a) Transform-coding kernel design: We demonstrate how the multirate approach can be applied to low-power but high-speed transform-coding architectures. We start with the derivations of the multirate DCT architectures. The resulting multirate low-power architectures are regular, modular, and free of global communications. Also, the compensation capability is achieved at the expense of locally increased hardware and data paths. As a consequence, they are very suitable for VLSI implementation. The multirate DCT design is later extended

to a unified low-power transform coding architecture that can perform most of the existing discrete sinusoidal transforms based on the same processing elements.

b) Motion estimation: Based on the concept of pseudophase and the sinusoidal orthogonal principles [46], [47], we design a novel low-power, low-complexity, and high-throughput coordinate rotation digital computer (CORDIC) [48] architecture for half-pel motion estimation. Techniques such as look-ahead, multirate, pipelining, and folding have been combined and used in the design. The proposed multiplier-free and fully pipelined parallel architecture works solely in the DCT domain without interpolation of input images to meet the needs of portable, high-quality, high-bit-rate picture transmission. Its low computational complexity—$O(N^2)$ as compared with $O(N^4)$ for commonly used half-pel block matching architecture (HBKM-ME)—makes it attractive in real-time multimedia applications.

3) Channel Coding:

a) Reed–Solomon (RS) decoder: With the widespread use of RS codes in portable applications, low-power RS decoder design has become an important issue. We discuss how the Berlekamp algorithm can be modified in order to obtain a low-power architecture. In addition, modifications are also proposed for the syndrome and error computations. The power reduction that can be achieved by the modified design is estimated based on the VLSI synthesis results.

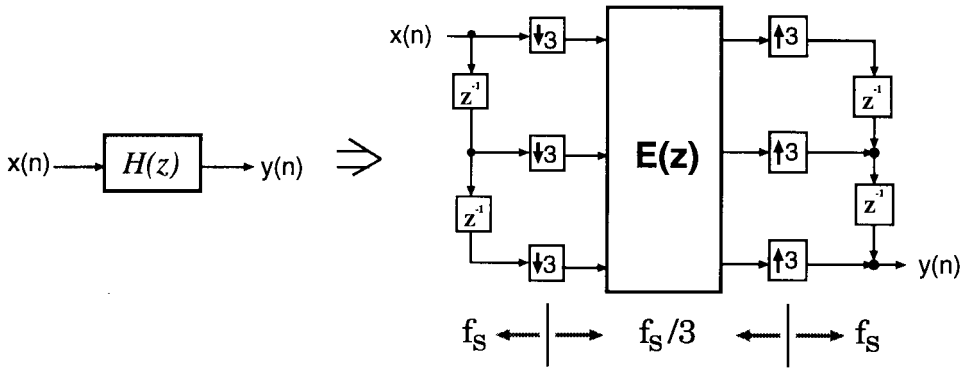
b) Viterbi decoder: The speed of a high-speed Viterbi decoder is constrained by the recursive add-compare-select (ACS) computation loop. In the literature [49]–[51], the look-ahead transformation has been applied to break the ACS loop and obtain high-speed operation. We illustrate how this transformation can be applied to the ACS unit to achieve low-power operation. We also review various techniques that have been used in the design of high-speed Viterbi decoders in the literature.

4) Adaptive Filtering: The throughput of an adaptive filter is restricted by feedback loops. For example, linear transversal least-mean-square (LMS) filters have the weight update loop and the error feedback loop. Various transformations such as look-ahead [42], [52], [53] have been applied to these feedback loops to obtain high-speed lattice and transversal adaptive filters. We review these techniques within the context of our high-speed/low-power approaches.

5) Finite-Precision Analysis of Multirate Implementation: When the VLSI implementations of the multirate architectures are taken into consideration, the choice of word length becomes an important design issue. In general, shorter word lengths will result in fewer switching events, lower capacitance, and shorter average routing length in the system. Hence, it helps to improve the power saving and speed performance of VLSI designs. On the other hand, if the word lengths are too short, the rounding error caused by fixed-point operations can be severe enough to hazard the signal-to-noise ratio (SNR). Thus, choosing minimum word lengths without degrading the SNR requirement is an important issue in the low-power/high-speed VLSI design. Motivated by this, we investigate the finite-word-length effect of the multirate DCT architectures. Our study can precisely predict the finite-precision behavior under different block sizes and decimation factors. By using the analytical results, we can assign the optimal word length for each DCT channel given the SNR constraint. Moreover, our analyses show that the average SNR's of the low-power DCT architectures are better than that of the normal design given the same word-length assignment. This indicates that multirate designs have better numerical properties than the normal design under fixed-point arithmetic.

6) A Reconfigurable Multirate DSP Computing Engine: We map an important set of DSP algorithms onto a computing engine that is of CORDIC nature. The proposed system is a massively parallel architecture that is capable of performing most low-level computationally intensive tasks including FIR/IIR filtering, subband filtering, discrete orthogonal transforms (DT's), and adaptive filtering in multimedia applications. We also show that the system can be easily reconfigured to perform multirate FIR/IIR/DT operations with negligible hardware overhead. Hence, we can double the processing speed on the fly based on the same processing elements. Since the properties of each programmed DSP function such as parallelism and pipelinability have been fully exploited in this design, the computational speed of this computing engine can be as fast as that of application-specific integrated circuit (ASIC) designs that are optimized for individual specific applications. The programmable/high-speed properties of this unified VLSI architecture make it very suitable for cost-effective video-rate multimedia applications.

The rest of this paper is organized as follows. In Section II, the design methodology for deriving multirate DSP and FIR/IIR filtering architectures is presented. The design and implementation issues of the low-power quadrature mirror filter (QMF) chips as well as the simulation/testing results are also discussed. In Section III, source-coding algorithms/architectures are discussed. In Section III-A, we present multirate transform-coding architectures starting with the derivation of multirate DCT VLSI architectures. In Section III-B, we extend the multirate DCT design to a unified transform-coding architecture. It can perform most of the existing orthogonal transforms based on the same processing elements. In Section III-C, a fully pipelined parallel CORDIC architecture for half-pel motion estimation is presented. It provides a low-power, low-complexity solution for MPEG-2, H.263 compatible video codec design on a dedicated single chip. In Section IV, channel-coding algorithms/architectures are discussed. In Section IV-A, a low-power/high-performance Reed–Solomon decoder based on the modified Berlekamp's algorithm is presented. In Section IV-B, we review high-speed Viterbi decoder design techniques and show how the look-ahead technique can be used in the low-power design of the ACS unit. In Section V, we review the current research results in low-power/high-speed adaptive filter design. In Section VI, we perform the finite-word-length analysis for the multirate DCT architectures. The rounding errors and dynamic range of the VLSI architectures under fixed-



(a) (b)

Fig. 5. (a) An LTI FIR/IIR system. (b) Its equivalent multirate implementation, where f_s is the data sampling rate.

point arithmetic are analyzed. In Section VII, we present the system architecture of a reconfigurable adaptive DSP computing engine that is capable of performing various tasks in multimedia applications. The speed-up of the system based on the multirate approach is also considered. We then conclude our work in Section VIII.

II. DSP and Digital Filtering

In this section, we present a systematic approach for the low-power design of general LTI DSP systems based on the multirate approach. In general, the direct implementation of the system transfer function $H(z)$ [see Fig. 5(a)] has the constraint that the speed of the processing elements must be as fast as the input data rate. As a result, it cannot compensate for the speed penalty under low supply voltage [2]. On the other hand, the multirate system in Fig. 5(b) requires only low-speed processing elements at one-third of the original clock rate to maintain the same throughput. Hence, we can apply this feature to either low-power implementation of the DSP system or speed-up of the system.

Motivated by this, we derive a design methodology for the design of low-power digital filtering systems. The users can simply follow the design steps to convert a speed-demanding DSP transfer function into its equivalent multirate realization. We also verify the effectiveness of the proposed multirate low-power design by the implementation of two FIR VLSI chips with different architectures. One is the normal pipelined design and the other is the multirate design with downsampling rate equal to two. We implemented both chips using the same CAD synthesis tool and the same VLSI technology. The only difference lies in the architectural design. Therefore, the effectiveness of the algorithm-based low-power design can be observed. The simulation results show that by trading 50% more silicon area, we can save up to 71% of the total power consumption without sacrificing the data-throughput rate. This observation is later verified by the testing of these two FIR chips.

In what follows, we first review some basic multirate operations that are useful for our derivations. We then present the design methodology as well as the verification of the multirate chip design.

A. Basic Multirate Operations

Given an FIR transfer function

$$H(z) = \sum_{n=0}^{N-1} h(n) z^{-n} \tag{10}$$

and an arbitrary integer M, we can rewrite $H(z)$ as

$$H(z) = \sum_{i=0}^{M-1} z^{-l} E_i(z^M) \tag{11}$$

where

$$E_i(z) = \sum_{n=0}^{\lfloor N/M \rfloor} h(Mn+i) z^{-n} \tag{12}$$

for $i = 0, 1, \cdots, M-1$. Equation (11) is referred to as the *Type I polyphase representation* with respect to M. $E_i(z)$, $i = 0, 1, \cdots, M-1$, are the *polyphase components* of $H(z)$. Fig. 6(a) shows the implementation of $H(z)$ based on the polyphase representation.

Two multirate operations will be used in our derivation. One is *noble identities* [see Fig. 6(b)]. It describes that M unit delays at the original clock rate is equivalent to one delay at an M times slower clock rate, and vice versa. The other multirate operation is the equivalent implementation of an $(M-1)$-delay element, as shown in Fig. 6(c). The multirate structure at the right is also known as the *delay chain perfect reconstruction system* [43, ch. 5]. These two basic multirate operations provide very efficient tools in the theory and implementation of multirate systems [43]. Take the *decimation circuit* depicted in Fig. 7(a), for example. The decimation filter $H(z)$ is in general a low-pass filter preventing the aliasing effect after the decimation operation. The direct implementation of Fig. 7(a) requires N multipliers and N adders, and the operating frequency of the processing operators needs to be as fast as the input data rate. Instead, we can have a more efficient implementation of Fig. 7(a) by applying the above-mentioned multirate operations: First, we replace the transfer function $H(z)$ in

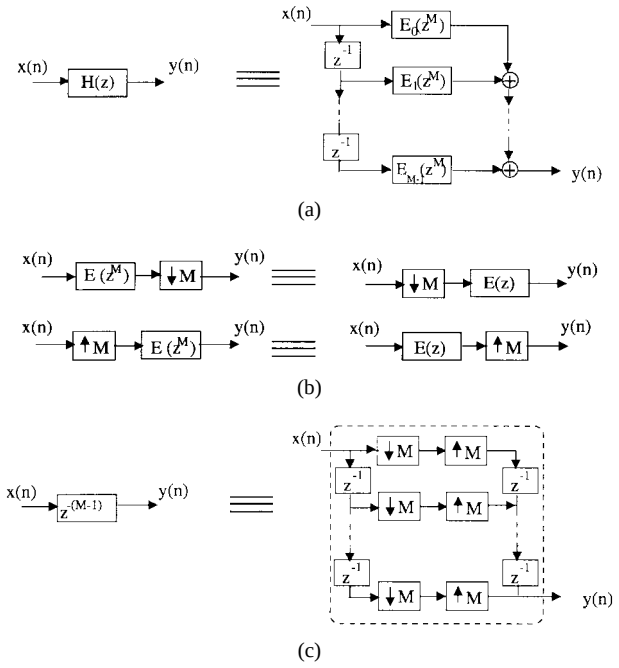


Fig. 6. Basic multirate operations. (a) Polyphase decomposition. (b) Noble identities. (c) Equivalent multirate implementation of an $(M-1)$ delay element.

Fig. 7(a) with that in Fig. 6(a), which results in Fig. 7(b). Then, we move the decimation operation toward the middle of the parallel paths [see Fig. 7(c)]. After applying the noble identity, we have the *polyphase implementation* of the decimation circuit as shown in Fig. 7(d). The total hardware cost is still N multipliers and N adders, whereas the operating frequency has been reduced to only one-third of the original clock rate.

Similarly, we can find the polyphase implementations of the decimation circuits with transfer function $z^{-1}H(z)$ and $z^{-2}H(z)$. Fig. 8(a) and (b) shows the polyphase decimation circuits with transfer function $z^{-1}H(z)$ and $z^{-2}H(z)$, respectively, by following the steps in Fig. 7. The resulting architecture is similar to that of Fig. 7(d) except that the polyphase components $E_i(z)$ have rotated, and some extra delay elements are added to $E_i(z)$. In the following derivation, the polyphase implementations in Figs. 7 and 8 will be used.

B. Design Methodology for Multirate FIR/IIR Systems

In this section, we present the design methodology to derive the multirate LTI system of Fig. 5. Without loss of generality, we assume that $M = 3$ in our derivation. The design procedure can be easily extended for any arbitrary M.

1) The Design Procedure: Given an LTI system with transfer function $H(z)$ with order N and decimation factor M, the design procedure is as follows (see Fig. 9).

Step a) Insert $M-1$ unit delays after the transfer function $H(z)$.

Step b) Replace the delay element with its equivalent "delay chain perfect reconstruction system."

Step c) Move $H(z)$ to the right until reaching the decimation operators.

Step d) Merge the delay elements with the transfer functions. Group the resulting new transfer functions ($H(z)$, $z^{-1}H(z)$, and $z^{-2}H(z)$) with their associated decimation operators.

Step e) Replace each decimation circuit in the circle shown in Fig. 9(d) with its *polyphase implementation* in Figs. 7 and 8. Then we have Fig. 9(e).

Step f) Note that the data inputs at points designated by $\mathbf{a}$ are the same, and so are those at points $\mathbf{b}$ and $\mathbf{c}$. After merging the common data paths in Fig. 9(e), we obtain Fig. 9(f), in which

$$\mathbf{E}(z) \triangleq \begin{bmatrix} E_0(z) & E_1(z) & E_2(z) \\ z^{-1}E_2(z) & E_0(z) & E_1(z) \\ z^{-1}E_1(z) & z^{-1}E_2(z) & E_0(z) \end{bmatrix}. \tag{13}$$

The general form of $\mathbf{E}(z)$ with an arbitrary decimation factor M can be shown to be

$$\mathbf{E}(z) = \begin{bmatrix} E_0(z) & E_1(z) & \cdots & E_{M-1}(z) \\ z^{-1}E_{M-1}(z) & E_0(z) & \cdots & E_{M-2}(z) \\ \vdots & \vdots & \ddots & \vdots \\ z^{-1}E_1(z) & z^{-1}E_2(z) & \cdots & E_0(z) \end{bmatrix} \tag{14}$$

which is also known as the *pseudocirculant matrix* in the context of alias-free QMF filter banks [43].

The design procedure described in Fig. 9 provides a systematic way to design a low-power DSP system. In the implementation of the FIR system, each $E_i(z)$ in Fig. 9(f) represents a subfilter of order N/M. It can be shown that the total hardware complexity to realize the multirate FIR system is MN multipliers and $(MN+M^2)$ adders. Basically, we pay a linear increase of hardware overhead in exchange for the advantage of an M-times slower processing speed.

In the multirate IIR system design, we first find out the polyphase components $E_i'(z)$, $i = 0, 1, \cdots, M-1$, of the given IIR function $H'(z)$. After replacing each $E_i(z)$ in Fig. 9 with its corresponding $E_i'(z)$, for $i = 0, 1, \cdots, M-1$, we can apply the aforementioned design methodology to convert $H'(z)$ into its equivalent multirate transfer function. Note that the complexity of each $E_i'(z)$ can be as high as that of the original transfer function $H'(z)$. Hence, we may pay up to $O(M^2)$ hardware overhead for the implementation of the multirate IIR filter. For detailed derivation of the multirate IIR system design, the readers may refer to [54].

2) Diagonalization of the Pseudocirculant Matrix: Although the multirate implementation of Fig. 9(f) can be readily applied to low-power design, the global communication of this structure is not desirable in the VLSI implementation. Therefore, we want to diagonalize the pseudocirculant matrix of (14) so as to eliminate global communication in the multirate implementation.

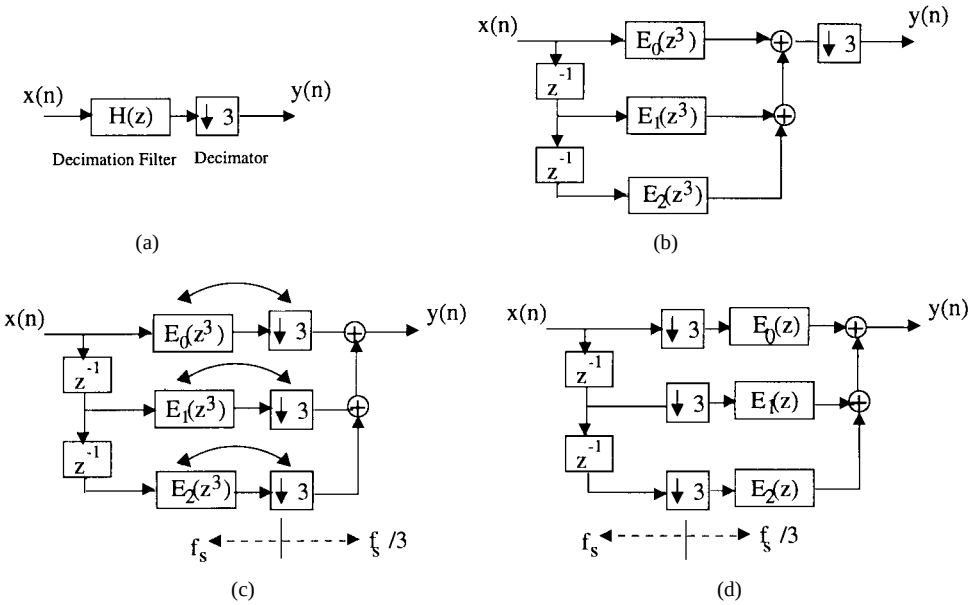


Fig. 7. Derivation of the polyphase decimation circuit, where f_s denotes the data sample rate. (a) The original decimation circuit. (b) Representing the decimation filter using Fig. 6(a). (c) Applying the noble identity. (d) The resulting polyphase implementation.

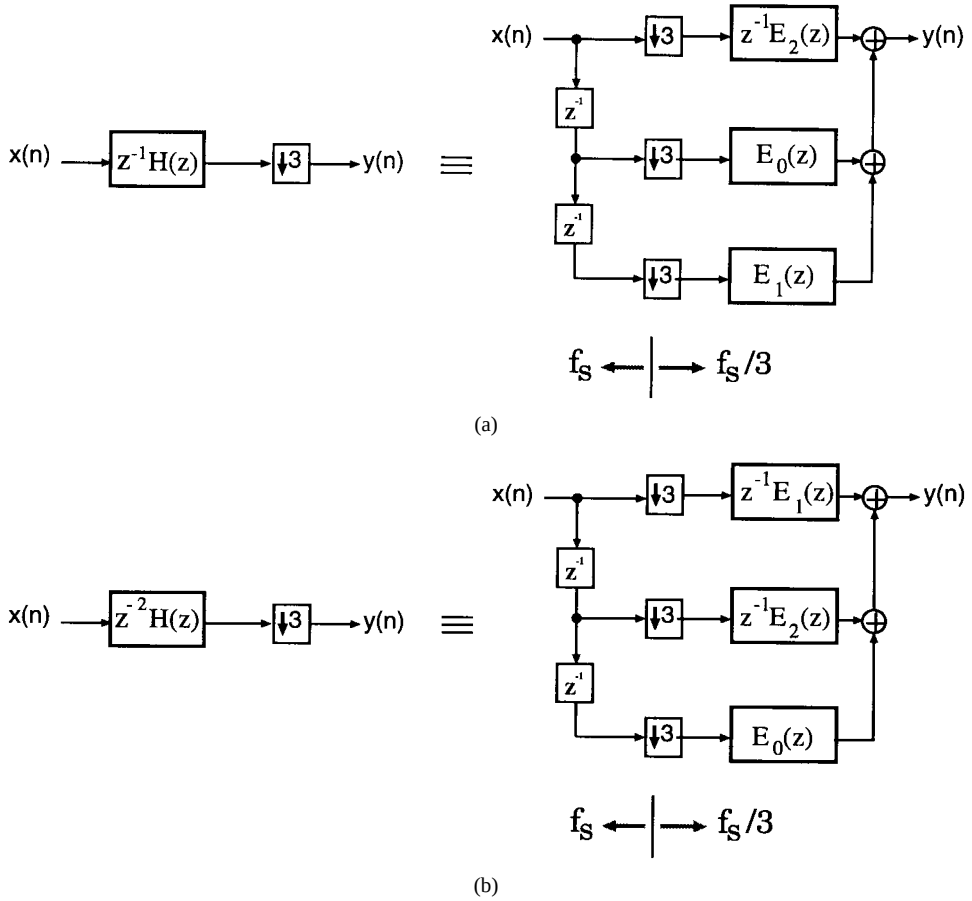


Fig. 8. Polyphase decimation circuit with transfer function (a) $z^{-1}H(z)$ and (b) $z^{-2}H(z)$, where f_s is the data sample rate.

There are two ways to diagonalize $\mathbf{E}(z)$ of (14). One is to use the DFT approach [55]. The DFT approach requires complex-number operations for the filtering operations. In addition, it still has global communications in the DFT/inverse (I)DFT networks. The second diagonalization approach is based on polynomial convolution techniques

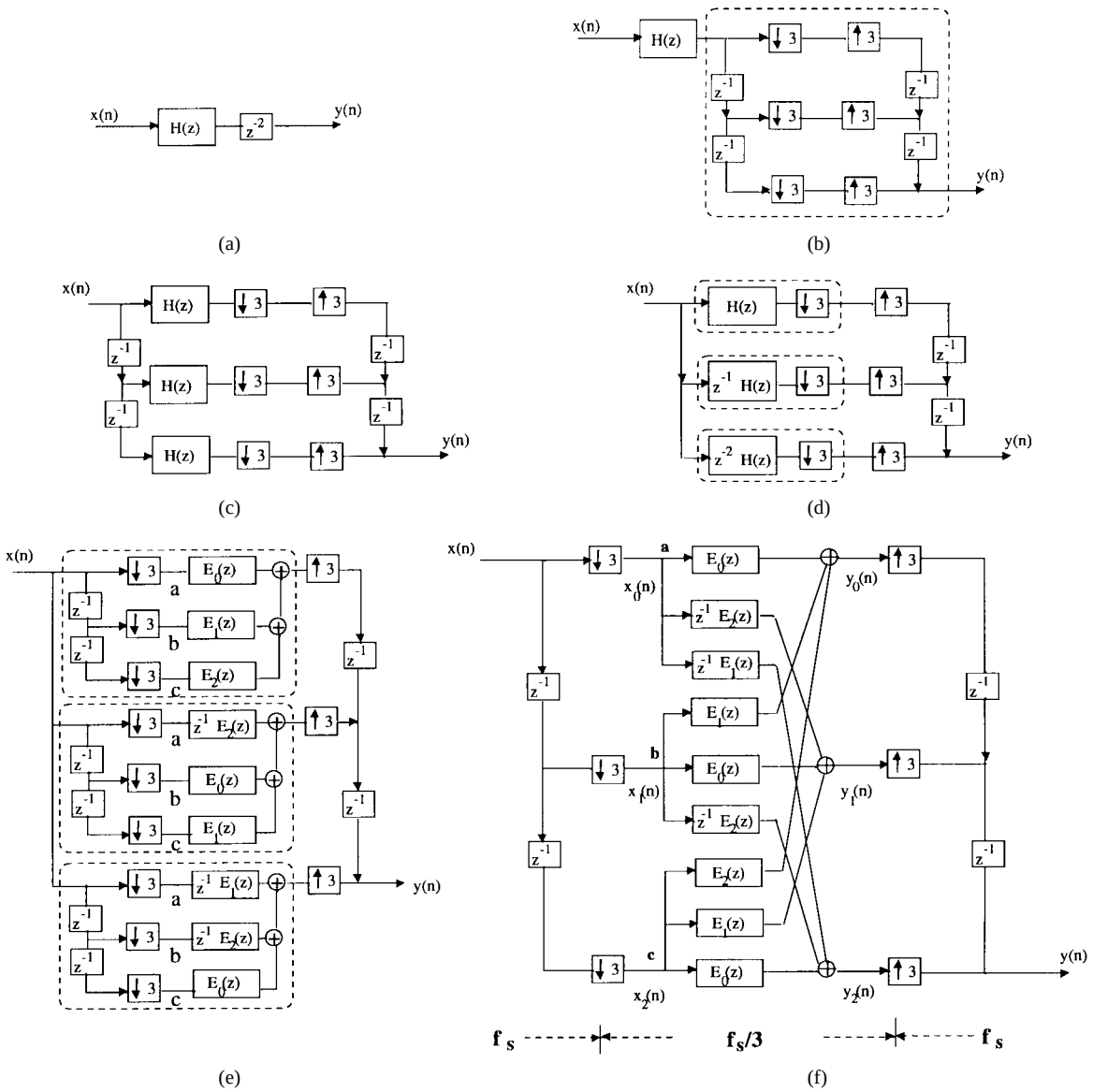


Fig. 9. Design procedure for the multirate LTI system.

[56]. As an example, for the case of $M = 2$, (14) can be rewritten as

$$\begin{bmatrix} E_0(z) & E_1(z) \\ z^{-1}E_1(z) & E_0(z) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}}_{\mathbf{B}} \cdot \begin{bmatrix} E_0(z) & 0 & 0 \\ 0 & E_0(z)+E_1(z) & 0 \\ 0 & 0 & E_1(z) \end{bmatrix} \underbrace{\begin{bmatrix} 1 & -1 \\ 0 & 1 \\ z^{-1} & -1 \end{bmatrix}}_{\mathbf{A}}. \tag{15}$$

The corresponding structure is depicted in Fig. 10. This diagonalization approach involves only real-number operations to process the decimated sequences. However, as M increases, the derivation becomes complicated and the resulting architecture is highly irregular (as opposed to the DFT approach) [56]. In the verification part, we will use the multirate FIR structure in Fig. 10 for our low-power FIR chip design.

3) Power Estimation for the Multirate FIR Architecture: Before we proceed to the chip design, we consider the power dissipation of the multirate design. From (2), it can be shown that the lowest possible supply voltage $V'_{\rm dd}$ for a device running at an M-times slower clock rate can be approximated by

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

where V_t is the threshold voltage of the device.

Assume that $V_{\rm dd} = 5$ V and $V_t = 0.7$ V in the original system. From (2), it can be shown that $V'_{\rm dd}$ can be as low as 3.08 V for the case of $M = 2$. For the direct-form FIR realization, it requires N multipliers and N adders. For the low-power multirate FIR architecture depicted in Fig. 10 (where $M = 2$), $3N/2$ multipliers and $3N/2$ adders are required. Provided that the capacitance

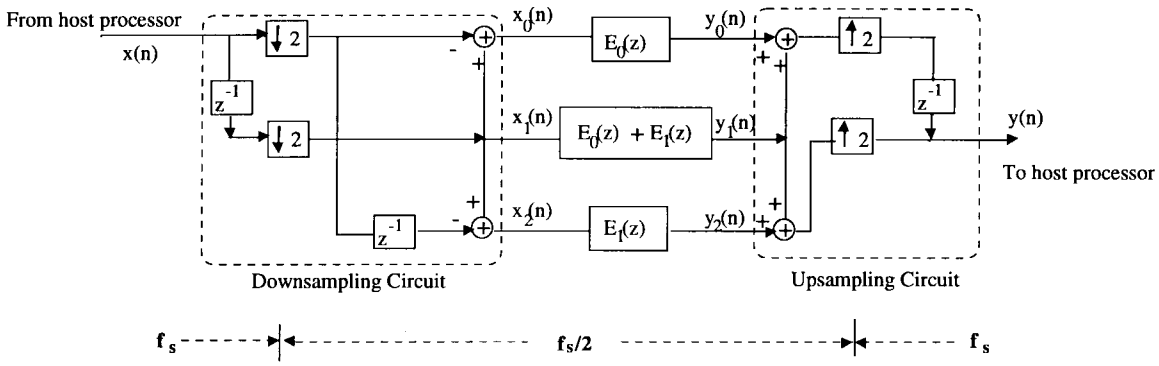


Fig. 10. Multirate FIR/IIR architecture with $M = 2$.

due to the multipliers is dominant in the circuit and is roughly proportional to the number of multipliers, the power consumption of the multirate FIR design can be estimated as

$$\left(\frac{\frac{3N}{2}}{N}C_{\text{eff}}\right)(V'_{\text{dd}})^2\left(\frac{1}{2}f\right) \approx 0.29P_0 \tag{17}$$

where P_0 denotes the power consumption of the normal FIR design. Although the multirate architecture requires about 50% hardware overhead, it consumes only 29% power of the original pipelined design. In essence, we trade hardware complexity for low-power consumption.

Another attractive application of the multirate design is in the very-high-speed filtering. If we do not lower down the supply voltage to save chip power consumption, the multirate FIR structure of Fig. 10 can process data at a rate that is twice as fast as the maximum speed of the processing elements. We will also verify this issue later.

C. Verification of the Multirate Low-Power Design

We now verify the power saving of the multirate low-power FIR structure. The selected FIR design is a QMF, which is widely used in the image compression and subband coding [57].

1) Selection of System Parameters: First, we consider the design and implementation issues of the QMF. The implementation of the QMF filter requires a large number of multipliers. To save the chip area, we choose the QMF design with power-of-two (POT) coefficients [19] for our chip implementation. For the purpose of further lowering the hardware complexity, we modify the QMF design of [19] by truncating some boundary tap coefficients and by dropping some relatively small components in each coefficient. Shown below are the QMF coefficients used in our FIR chip design:

$$\begin{aligned}
h(10) = h(11) &= 2^{-2} + 2^{-4} + 2^{-6} \\
h(9) = h(12) &= 2^{-4} + 2^{-5} \\
h(8) = h(13) &= -2^{-4} - 2^{-7} \\
h(7) = h(14) &= -2^{-5} \\
h(6) = h(15) &= 2^{-5} + 2^{-7} \\
h(5) = h(16) &= 2^{-7} + 2^{-8}
\end{aligned}$$

Table 1 PSNR Results for Different QMF's

Filter type	Filter length	PSNR (dB)	Coeffi -cient type	Fixed-point adders
Filter 32D in [57]	32	44.8	Float	N/A
Filter in [19]	32	38.8	POT	84
Filter of Eq. (18)	22	37.0	POT	36

$$\begin{aligned}
h(4) = h(17) &= -2^{-6} - 2^{-7} \\
h(3) = h(18) &= -2^{-8} \\
h(2) = h(19) &= 2^{-6} \\
h(1) = h(20) &= 0, \\
h(0) = h(21) &= -2^{-7}.
\end{aligned} \tag{18}$$

To verify the performance of this modified QMF, we carried out simulations by passing the LENA image through the subband coding structure (see [19, Fig. 5]). Table 1 lists the peak SNR (PSNR) results. Compared with the original design of [19], our modified design suffers from little degradation in PSNR but with much less hardware complexity.

Next, we want to determine the system word length to be used in our chip design. Since the word length would directly affect the resulting chip area as well as the total number of switching events in the logic circuits, it is important to determine the minimum word length without degrading the PSNR performance. We conducted computer simulations by feeding the LENA image into the subband coding structure under fixed-point arithmetic. The results are shown in Fig. 11. Since the PSNR curve saturates around $B = 12$, we use the word length of 12 bits in our design.

2) Chip Design: Having decided the system specifications of our design, we use PARTHENON [58] to design/synthesize both the normal and multirate FIR filters. The resulting chip layouts are shown in Fig. 12 [54], [59]. In the multirate design, the upper right module realizes the upsampling and downsampling circuits of Fig. 10. The other three modules realize the three $N/2$-tap FIR filters—$E_0(z)$, $E_1(z)$, and $E_0(z) + E_1(z)$ of Fig. 10—at the operating frequency of $f_s/2$. Their output signals are sent back to the up/downsampling module to reconstruct the filtering output $y(n)$ running at f_s. The chip area of the

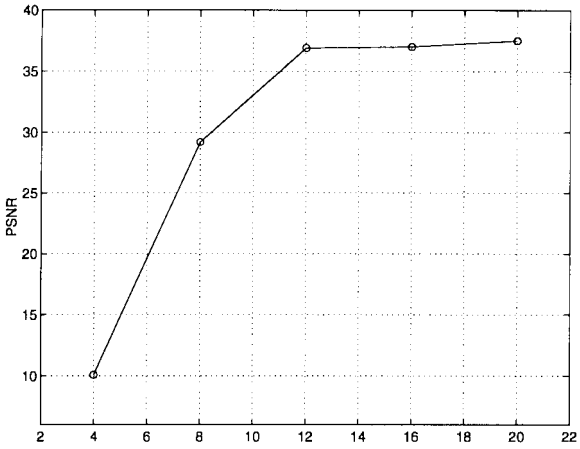


Fig. 11. PSNR results for the modified QMF as a function of system word length B.

multirate design is about 50% more than that of the normal design, as we expected. Therefore, our estimation of the effective capacitance in (17) is very accurate.

To see the effect of supply voltage on the speed of the FIR design, we conducted SPICE simulation for the critical path of the FIR structure. The simulation results depicted in Fig. 13 show that the propagation delay is approximately doubled as the supply voltage reduces from 5 to 3.08 V. This is consistent with the results presented in [2]. Since the delay in the critical path generally determines the maximum clock rate of the chip, we can predict that the performance of the filtering operations will be degraded by 50% under the 3.08-V supply voltage. Nevertheless, the data-throughput rate of the multirate FIR will not be affected by such a speed penalty since the slowed-down processing elements are in the $f_s/2$ region (see Fig. 10). The I/O data rate will remain at f_s, which is the same as the normal FIR design operated at 5 V.

3) Testing Results: These two chip designs have been fabricated by MOSIS (a VLSI fabrication service) using 2μ double metal CMOS technology without the use of any low-power cells. The chips were tested under the same test environments: the same input data sequence and the same test equipment (HP 82 000 IC evaluation system) [59].

The measurements of the power dissipation and maximum speed of both chips are listed in Table 2. As we can see, at the same 20-MHz data rate, the multirate FIR chip can operate at a lower supply voltage of 3.2 V and consumes only 21% of the power of the normal FIR chip. These results agree with our arguments for the supply voltage and power consumption of the low-power design. Under the normal 5-V supply voltage, the multirate FIR chip can operate at 31.3 MHz, which is 56% faster than the normal FIR chip.

III. Video Coding and Compression

A. Multirate Low-Power DCT Architectures

The one-dimensional (1-D) time-recursive DCT of a series of input data starting from $x(t-N+1)$ and ending at $x(t)$ is defined as

$$X_{\mathrm{DCT},k}(t) = C(k)\sum_{n=0}^{N-1}\cos\left[(2n+1)\frac{k\pi}{2N}\right]x(t+n-N+1) \quad (19)$$

for $k = 0,1,2,\cdots,N-1$, where $C(0) = \sqrt{(1/N)}$ and $C(k) = \sqrt{(2/N)}$, $k = 1,2,\cdots,N-1$, are the scaling factors. In [44]–[46], the DCT transform operator is considered as a linear shift invariant (LSI) system with a second-order IIR transfer function that maps the serial input data into their transform coefficients. The transfer function can be obtained from (19) as

$$H_{\mathrm{DCT},k}(z) = \frac{X_{\mathrm{DCT},k}(z)}{X(z)} = \left((-1)^k - z^{-N}\right)\frac{C(k)\cos\omega_k(1-z^{-1})}{1-2\cos 2\omega_k z^{-1}+z^{-2}} \quad (20)$$

where $\omega_k \triangleq \frac{k\pi}{2N}$ and $X_{\mathrm{DCT},k}(z)$ and $X(z)$ denote the z-transforms of $X_{\mathrm{DCT},k}(t)$ and $x(t)$, respectively. The corresponding IIR structure to compute the kth frequency component of the DCT is shown in Fig. 14, where $\Gamma_c(m) \triangleq (-1)^k C(k)\cos m\omega_k$. It is regular, modular, and fully pipelined. The IIR DCT works in a SIPO way. It also avoids the input buffers as well as the index mapping operation that are required in most parallel-input parallel-output (PIPO) DCT architectures [61], [62]. One disadvantage of the IIR structure is that the operation speed is constrained by the recursive loops. In what follows, we will reformulate the transfer function using the multirate approach so that speed constraint can be alleviated.

Splitting the input data sequence into the *even* sequence

$$x_e(t,n) = x(t+2n-N+1), \qquad n = 0,1,\cdots,N/2-1 \quad (21)$$

and the *odd* sequence

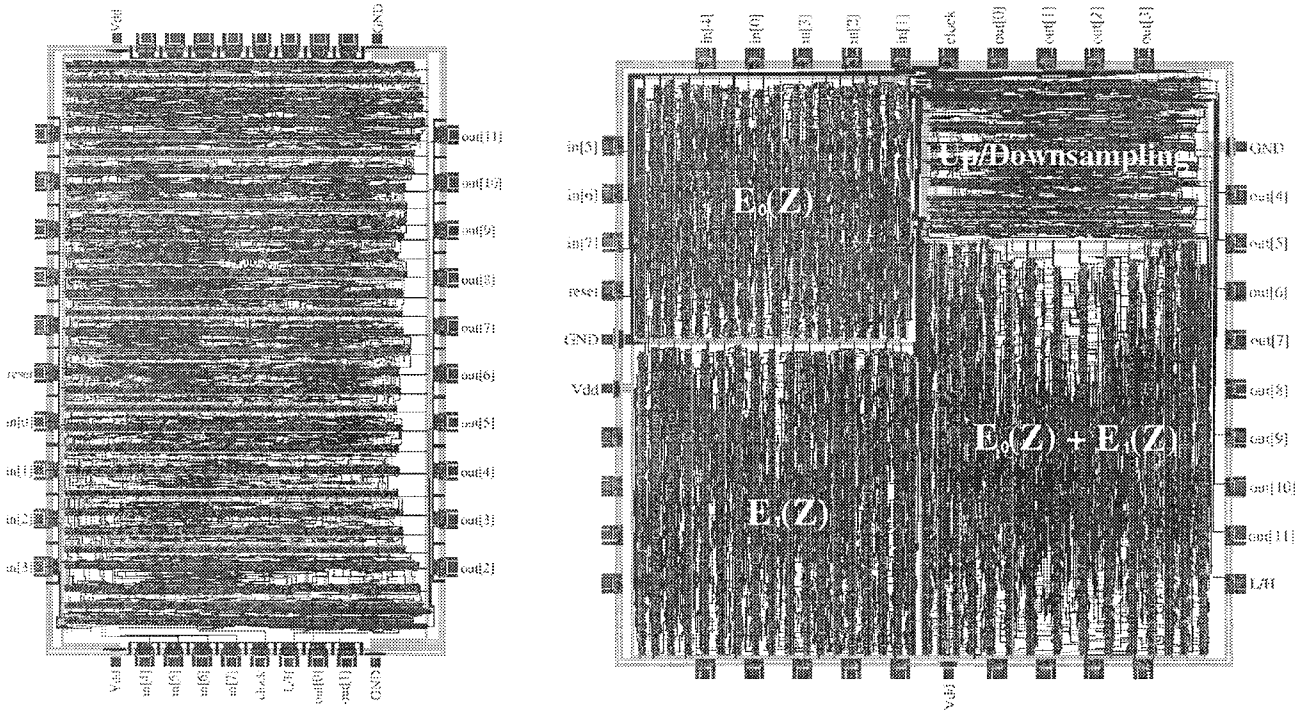
$$x_o(t,n) = x(t+2n-N+2), \qquad n = 0,1,\cdots,N/2-1 \quad (22)$$

(19) becomes

$$X_{\mathrm{DCT},k}(t) = C(k)\sum_{n=0}^{N/2-1}\cos\left[(4n+1)\frac{k\pi}{2N}\right]x_e(t,n) + C(k)\sum_{n=0}^{N/2-1}\cos\left[(4n+3)\frac{k\pi}{2N}\right]x_o(t,n). \quad (23)$$

Taking the z-transform on both sides of (23) and rearranging, we have

$$X_{\mathrm{DCT},k}(z) = \frac{C(k)\left((-1)^k - z^{-N/2}\right)}{1-2\cos 4\omega_k z^{-1}+z^{-2}} \cdot \left(\left[X_e(z)-X_o(z)z^{-1}\right]\cos 3\omega_k + \left[X_o(z)-X_e(z)z^{-1}\right]\cos\omega_k\right) \quad (24)$$



(a) (b)

Fig. 12. Final layout of (a) normal FIR filter (dimension $= 4400 \times 6600\ \lambda^2$) and (b) multirate FIR filter (dimension $= 6500 \times 6600\lambda^2$) [54].

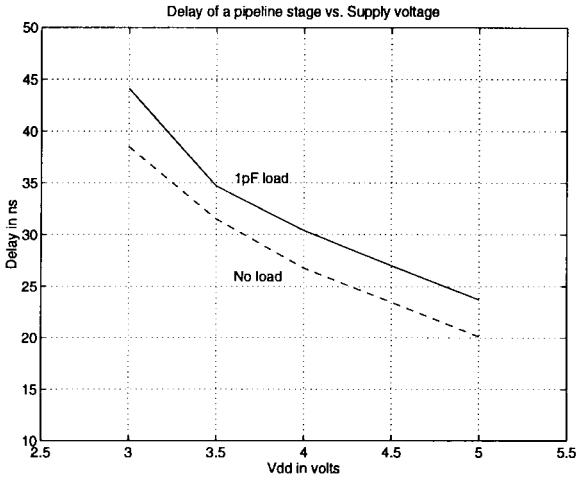


Fig. 13. Timing analysis for one pipelined stage in the FIR design.

Table 2 Testing Results of the Normal and Multirate FIR Chips

	Voltage (V)	Current (mA)	Max speed (MHz)	Power (mW)
Normal FIR	5.0	98.5	20.0	492.5
Multirate FIR	5.0	141.8	31.3	709.0
	4.5	106.4	26.0	478.8
	4.0	76.9	24.4	307.6
	3.5	42.4	21.7	148.4
	3.2	32.6	20.0	104.3

where $X_e(z)$ and $X_o(z)$ are the z-transforms of $x_e(t,n)$ and $x_o(t,n)$, respectively. The parallel architecture to realize (24) is depicted in Fig. 15, where M denotes the decimation factor.

To achieve downsampling by the factor of four $(M = 4)$, we can split the input data sequence into four decimated sequences $g_i(t,n) \triangleq x(t+(4n+i)-N+1)$, $i = 0,1,2,3$. Then $X_{\mathrm{DCT},k}(z)$ can be written as

$$X_{\mathrm{DCT},k}(z) = \frac{C(k)\left((-1)^k - z^{-N/4}\right)}{1 - 2\cos 8\omega_k z^{-1} + z^{-2}} \cdot \left(\left[G_0(z) - G_3(z)z^{-1}\right]\cos 7\omega_k + \left[G_1(z) - G_2(z)z^{-1}\right]\cos 5\omega_k + \left[G_2(z) - G_1(z)z^{-1}\right]\cos 3\omega_k + \left[G_3(z) - G_0(z)z^{-1}\right]\cos \omega_k\right) \quad (25)$$

where $G_i(z)$ is the z-transform of $g_i(t,n)$. The corresponding multirate architecture is shown in Fig. 16. Similarly, the multirate IIR IDCT architectures can be derived (for details, see [63]).

From Figs. 15 and 16, we can see that basically the multirate DCT architectures retain all advantages of the original IIR structure in [45] such as modularity, regularity, and local interconnections. This makes the proposed architectures very suitable for VLSI implementations. It is also interesting to see that the speed-compensation capability of our architectures is achieved at the expense of "locally" increased hardware complexity and routing paths. This feature of local interconnection and local hardware overhead is especially important when the transformation size is large (e.g., the MPEG audio codec in which a 32-point DCT/IDCT is used [64]).

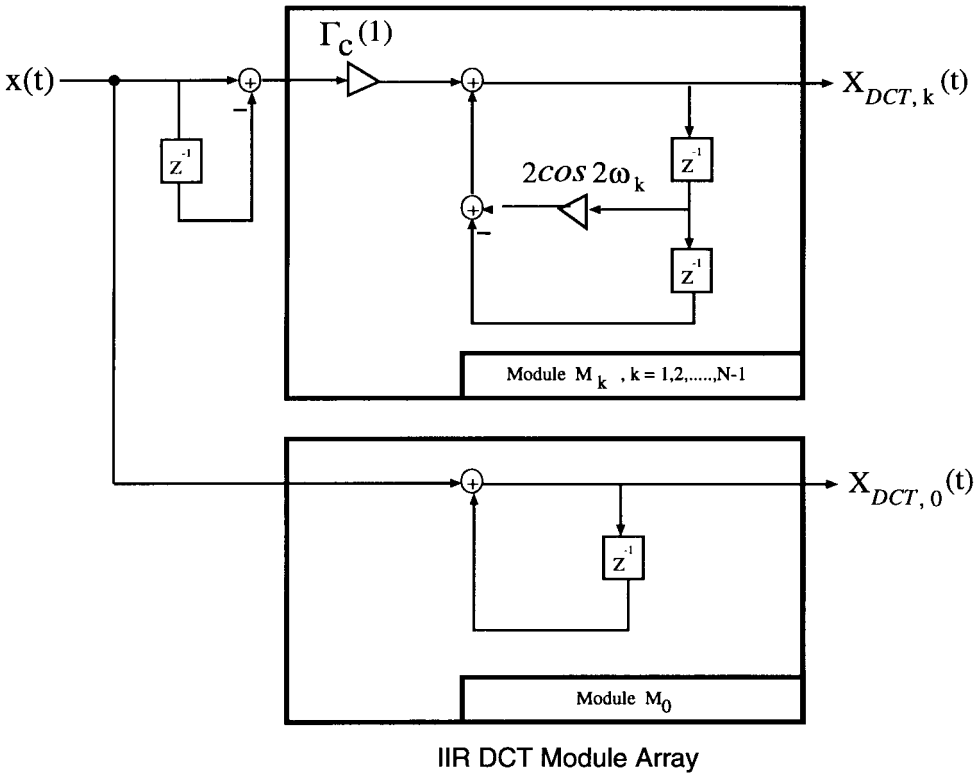


Fig. 14. IIR DCT architecture.

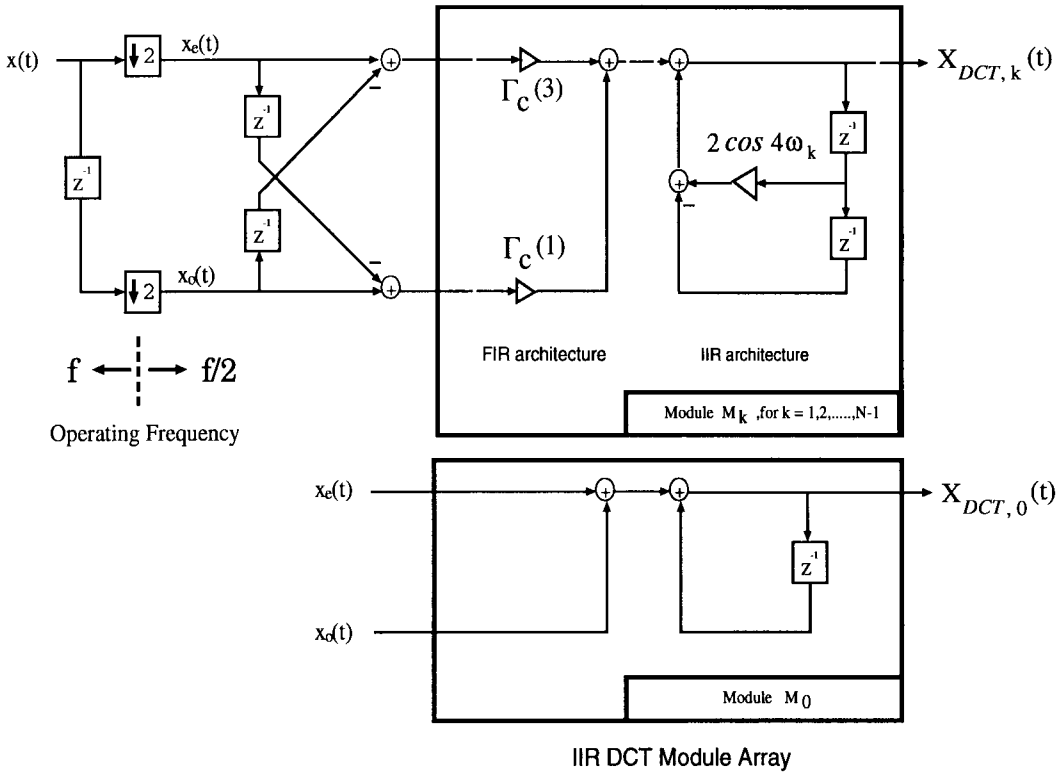


Fig. 15. Low-power DCT architecture with $M = 2$.

1) Power Estimation for the Multirate DCT Design: Next, let us consider the power dissipation of the low-power DCT architectures. For the 16-point DCT under normal operation, it requires 30 multipliers and 32 adders. For the multirate 16-point DCT with $M = 2$, 45 multipliers and 49 adders are required. Following the arguments in Section

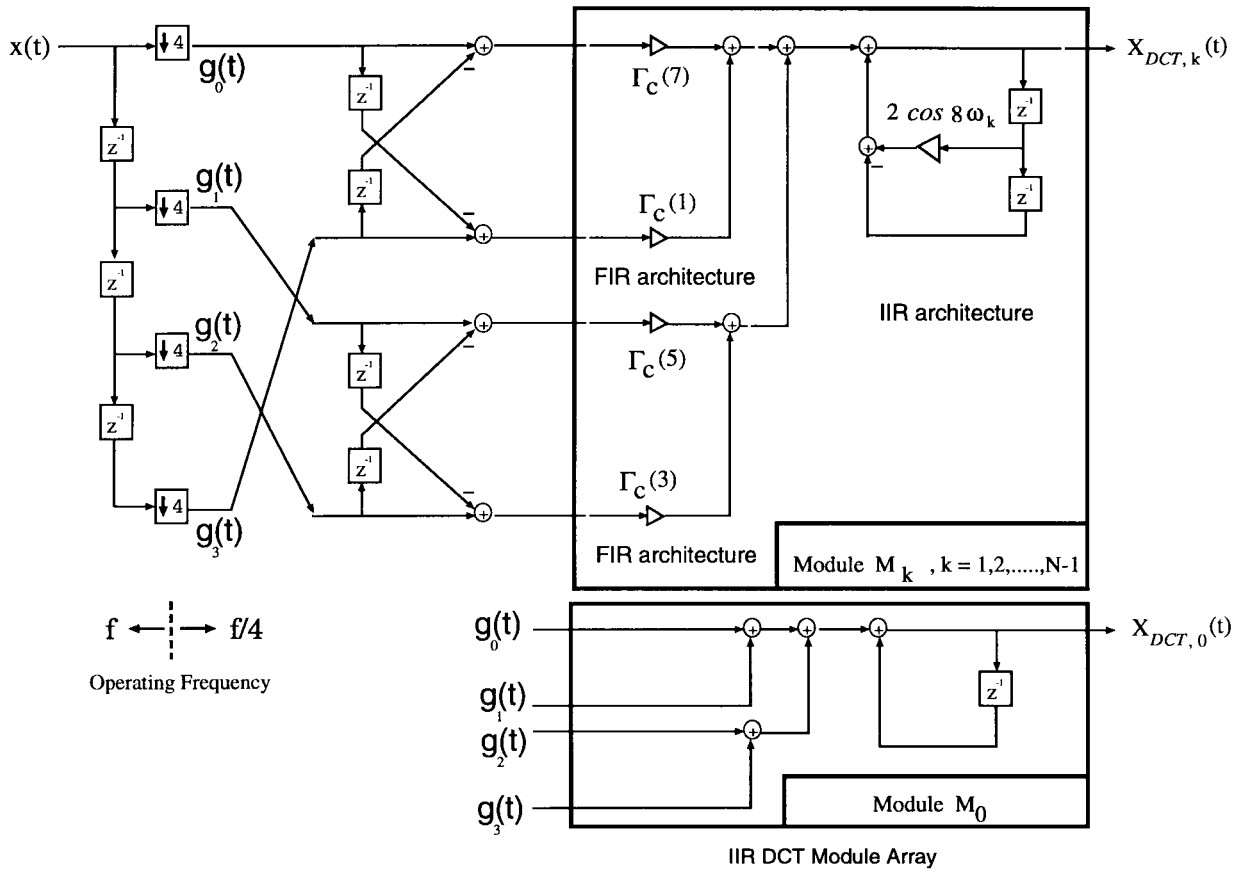


Fig. 16. Low-power DCT architecture with $M = 4$.

Table 3 Comparison of Hardware Cost for the DCT, IDCT, MLT, and ELT with Their Multirate Designs in Terms of Two-Input Adder/Multipliers

	Normal Operation		Downsampling by 2		Downsampling by 4	
	Multiplier	Adder	Multiplier	Adder	Multiplier	Adder
IIR DCT	$2N-2$	$2N$	$3N-3$	$3N+1$	$5N-5$	$5N+3$
IIR IDCT	$2N+1$	$3N$	$3N+1$	$4N+1$	$5N+1$	$6N+2$
IIR MLT	$5N$	$5N$	$10N$	$11N$	$20N$	$23N$
IIR ELT	$6N$	$6N$	$11N$	$12N$	$21N$	$24N$

II-B3, it can be shown that the power consumption for the multirate design can be roughly estimated as

$$\left(\frac{45}{30}C_{\text{eff}}\right)(3.08V)^2\left(\frac{1}{2}f\right) \approx 0.29P_0 \tag{26}$$

where P_0 denotes the power consumption of the original DCT design. Similarly, for the case $M = 4$, the 16-point DCT needs a total of 75 multipliers and 83 adders, and the lowest possible voltage supply can be 2.08 V [from (16)]. The total power can be reduced to about

$$\left(\frac{75}{30}C_{\text{eff}}\right)(2.08V)^2\left(\frac{1}{4}f\right) \approx 0.11P_0. \tag{27}$$

Therefore, we can achieve low-power consumption at the expense of reasonable complexity overhead.

2) Comparison of Architectures: Table 3 summarizes the hardware cost for the proposed architectures under normal operation and under multirate operation (for $M = 2, 4$). As we can see, the hardware overhead for the low-power design is linear complexity increase for the speed compensation. Next, we compare our low-power DCT architecture with those proposed in [45] (SIPO approach) and [62] (PIPO approach). From Table 4, we can see that our multirate SIPO approach is a good compromise between the other two approaches. Basically, the multirate approach inherits all the advantages of the existing SIPO approach; Meanwhile, it can compensate the speed penalty at the expense of "locally" increased hardware and routing, which is not the case in the PIPO approach. Although some restriction is imposed on the data size N due to the downsampling operation, the choice of N is much more flexible compared with the PIPO algorithms.

In addition to the IIR-based approach, we can also apply *backward Chebyshev recursion* to derive the multirate DCT/IDCT architectures [65]. In computing the IDCT, the Chebyshev approach requires less hardware complexity

Table 4 Comparisons of Different DCT Architectures, Where f_s Denotes the Data Sample Rate, M Denotes the Programmable Downsampling Factor, and N Is the Block Size

	Liu *et. al.* [45]	Proposed multirate IIR DCT with $M = 4$	Lee [62]
Data processing rate	f_s	f_s/M	f_s/N
No. of Multipliers	$2N-2$	$(M+1)N$ (in order)	$\left(\frac{3N}{2}\right)\log_2 N$ (in order)
No. of Adders	$2N$	$(M+1)N$ (in order)	$\left(\frac{N}{2}\right)\log_2 N$
Latency	N	N	$[\log_2 N(\log_2 N-1)]/2$
Restriction on transform size N	No	$Mk, k \in Z^+$	$2^k, k \in Z^+$
Requirement for input buffer	No	No	Yes
Index mapping	No	No	Yes
Communication	Local	Local	Global
I/O operation	SIPO	SIPO	PIPO
Speed compensation capability	N/A	Good (at the expense of locally increased hardware overhead and local routing)	Good (at the expense of globally increased hardware overhead and global routing)
Power consumption in routing	Negligible	Negligible	Noticeable as N increases
Application to pruning DCT [66]	Direct	Direct	Needs many modifications and global interconnections

than the IIR-based approach presented here. Moreover, to further reduce the hardware overhead, the multirate low-power DCT architecture with logarithmic complexity can be derived. The readers may refer to [63] for details.

B. Other Transform Coding Module Designs

In this section, we extend the multirate DCT design to a larger class of orthogonal transforms. We start with the multirate design of the modulated lapped transform (MLT) and extended lapped transform (ELT) [67]–[69]. Later, based on the derivations of MLT and ELT, we derive a unified transform coding architecture that is capable of performing most of the discrete orthogonal transforms.

1) The IIR MLT Structure: The time-recursive MLT operates on segments of data of length $2N$, $x(t+n-2N+1)$, $n = 0, 1, \cdots, 2N-1$, and produces N output coefficients, $X_{\mathrm{MLT},k}(t)$, $k = 0, 1, \cdots, N-1$, as follows [67]:

$$X_{\mathrm{MLT},k}(t) = S(k)\sqrt{\frac{2}{N}}\sum_{n=0}^{2N-1}\sin\frac{\pi}{2N}\left(n+\frac{1}{2}\right) \cdot\cos\left[\frac{\pi}{N}\left(k+\frac{1}{2}\right)\left(n+\frac{1}{2}+\frac{N}{2}\right)\right] \cdot x(t+n-2N+1) \tag{28}$$

where $S(k) = (-1)^{(k+2)/2}$ if k is even, and $S(k) = (-1)^{(k-1)/2}$ if k is odd. After some algebraic manipulations, the MLT can be decomposed into [70]

$$X_{\mathrm{MLT},k}(t) = -S(k)[X_{C,k+1}(t) + X_{S,k}(t)] \tag{29}$$

where

$$X_{C,k}(t) \triangleq \beta_1 \sum_{n=0}^{L-1}\cos[(2n+1)\omega_k + \theta_k]x(t+n-2N+1) \tag{30}$$

$$X_{S,k}(t) \triangleq \beta_1 \sum_{n=0}^{L-1}\sin[(2n+1)\omega_k + \theta_k]x(t+n-2N+1) \tag{31}$$

with block size $L = 2N$ and

$$\beta_1 \triangleq \frac{1}{\sqrt{2N}}, \quad \omega_k \triangleq \frac{\pi k}{2N}, \quad \text{and} \quad \theta_k \triangleq \frac{\pi}{2}\left(k+\frac{1}{2}\right). \tag{32}$$

The IIR transfer functions for (30) and (31) can be computed as

$$H_{C,k}(z) = \beta_1(1-z^{-L}) \cdot\frac{\cos((2L-1)\omega_k+\theta_k) - \cos((2L+1)\omega_k+\theta_k)z^{-1}}{1-2\cos 2\omega_k z^{-1}+z^{-2}} \tag{33}$$

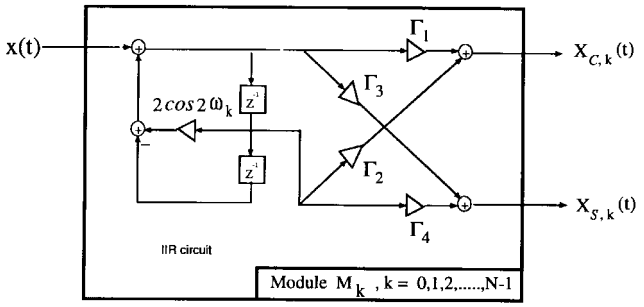
$$H_{S,k}(z) = \beta_1(1-z^{-L}) \cdot\frac{\sin((2L-1)\omega_k+\theta_k) - \sin((2L+1)\omega_k+\theta_k)z^{-1}}{1-2\cos 2\omega_k z^{-1}+z^{-2}}. \tag{34}$$

The corresponding IIR module for the dual generation of $X_{C,k}(t)$ and $X_{S,k}(t)$ is depicted in Fig. 17(a), where

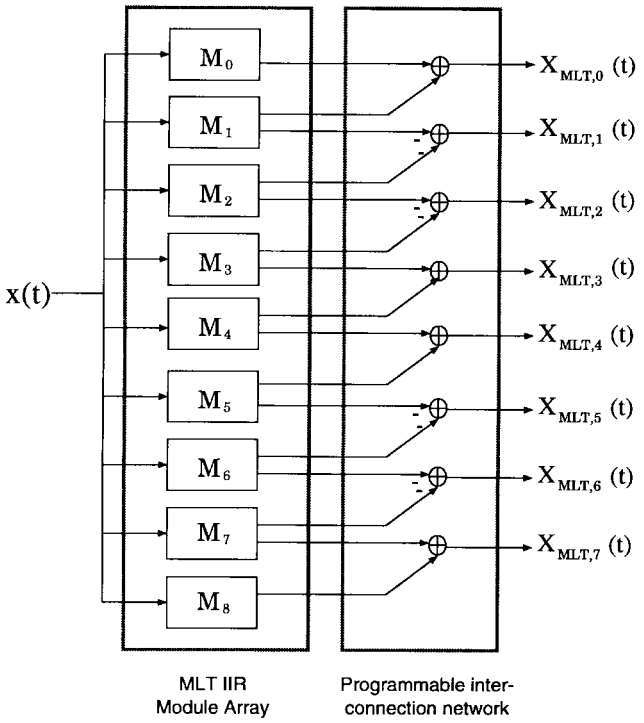
$$\begin{aligned}\Gamma_1 &\triangleq \beta_1\cos((2L-1)\omega_k+\theta_k)\\ \Gamma_2 &\triangleq -\beta_1\cos((2L+1)\omega_k+\theta_k)\\ \Gamma_3 &\triangleq \beta_1\sin((2L-1)\omega_k+\theta_k)\\ \Gamma_4 &\triangleq -\beta_1\sin((2L+1)\omega_k+\theta_k).\end{aligned} \tag{35}$$

This IIR module can be used as a basic building block to implement MLT according to (29). Fig. 17(b) illustrates the overall time-recursive MLT architecture for the case $N = 8$. It consists of two parts: one is the *IIR module array*, which computes $X_{C,k}(t)$ and $X_{S,k}(t)$ with different index k in parallel. The other is the *programmable interconnection network*, which selects and combines the outputs of the IIR array to generate the MLT coefficients.

2) Low-Power Design of the MLT: As with the low-power DCT, we can have a low-power MLT architecture if each MLT module can compute $X_{C,k}(t)$ and $X_{S,k}(t)$ from the decimated input sequences. After performing the polyphase decomposition on (33) and (34), we can



(a)



(b)

Fig. 17. (a) IIR MLT module. (b) The time-recursive MLT architecture.

compute the multirate IIR transfer functions for $H_{C,k}(z)$ and $H_{S,k}(z)$ as

$$\begin{aligned} H_{C,k}(z) = & \frac{\beta_1(1-z^{-L/2})}{1-2\cos(4\omega_k)z^{-1}+z^{-2}} \\ & \cdot([\cos((2L-3)\omega_k+\theta_k) \\ & \quad -\cos((2L+1)\omega_k+\theta_k)z^{-1}]X_e(z) \\ & +[\cos((2L-1)\omega_k+\theta_k) \\ & \quad -\cos((2L+3)\omega_k+\theta_k)z^{-1}]X_o(z)) \end{aligned} \tag{36}$$

and

$$\begin{aligned} H_{S,k}(z) = & \frac{\beta_1(1-z^{-L/2})}{1-2\cos(4\omega_k)z^{-1}+z^{-2}} \\ & \cdot([\sin((2L-3)\omega_k+\theta_k) \\ & \quad -\sin((2L+1)\omega_k+\theta_k)z^{-1}]X_e(z) \\ & +[\sin((2L-1)\omega_k+\theta_k) \\ & \quad -\sin((2L+3)\omega_k+\theta_k)z^{-1}]X_o(z)). \end{aligned} \tag{37}$$

The parallel architecture to realize (36) and (37) is shown in Fig. 18, where

$$\begin{aligned} \Gamma_{1,e} &= \beta_1\cos((2L-3)\omega_k+\theta_k) \\ \Gamma_{2,e} &= -\beta_1\cos((2L+1)\omega_k+\theta_k) \\ \Gamma_{3,e} &= \beta_1\sin((2L-3)\omega_k+\theta_k) \\ \Gamma_{4,e} &= -\beta_1\sin((2L+1)\omega_k+\theta_k) \\ \Gamma_{1,o} &= \beta_1\cos((2L-1)\omega_k+\theta_k) \\ \Gamma_{2,o} &= -\beta_1\cos((2L+3)\omega_k+\theta_k) \\ \Gamma_{3,o} &= \beta_1\sin((2L-1)\omega_k+\theta_k) \\ \Gamma_{4,o} &= -\beta_1\sin((2L+3)\omega_k+\theta_k). \end{aligned} \tag{38}$$

It consists of two MLT modules in Fig. 17(a). The upper module computes part of the $X_{C,k}(t)$ and $X_{S,k}(t)$ from the even sequence, while the lower one computes the remaining part from the odd sequence. The two adders at the right end are used to combine the even and odd outputs. Through such manipulation, only decimated sequences are processed inside the module. Hence, the MLT module can operate at half of the original clock rate by doubling the hardware complexity. The comparison of hardware cost is shown in Table 3. Suppose that P_0 denotes the power consumption of the MLT module in Fig. 17(a). From the CMOS power model, it can be shown that the power consumption for the low-power MLT modules is reduced to $0.38P_0$ and $0.17P_0$ for the case of $M=2$ and $M=4$, respectively.

3) Low-Power Design of the ELT: The ELT with basis length equal to $4N$ operates on a data segment of length $4N$, $x(t+n-4N+1), n=0,1,\cdots,4N-1$, and produces N output coefficients $X_{\mathrm{ELT},k}(t)$, $k=0,1,\cdots,N-1$. One good choice of the ELT is as follows [71]:

$$\begin{aligned} X_{\mathrm{ELT},k}(t) = & \sqrt{\frac{2}{N}}\sum_{n=0}^{4N-1}\left[\frac{1}{2\sqrt{2}}-\frac{1}{2}\cos\frac{\pi}{N}\left(n+\frac{1}{2}\right)\right] \\ & \cdot\cos\left[\frac{\pi}{N}\left(k+\frac{1}{2}\right)\left(n+\frac{1}{2}+\frac{N}{2}\right)\right] \\ & \cdot x(t+n-4N+1). \end{aligned} \tag{39}$$

By the use of some trigonometric identities, we can rewrite (39) as

$$X_{\mathrm{ELT},k}(t) = -\tilde{X}_{S,k+1}(t)+\sqrt{2}\tilde{X}_{C,k}(t)+\tilde{X}_{S,k-1}(t) \tag{40}$$

where

$$\tilde{X}_{C,k}(t) \triangleq \beta_2\sum_{n=0}^{L-1}\cos[(2n+1)\omega_k'+\theta_k']x(t+n-4N+1) \tag{41}$$

$$\tilde{X}_{S,k}(t) \triangleq \beta_2\sum_{n=0}^{L-1}\sin[(2n+1)\omega_k'+\theta_k']x(t+n-4N+1) \tag{42}$$

with

$$L=4N, \quad \beta_2 \triangleq \frac{1}{2\sqrt{2N}}, \quad \omega_k' \triangleq \frac{\pi}{2N}\left(k+\frac{1}{2}\right)$$

and

$$\theta_k' \triangleq \frac{\pi}{2}\left(k+\frac{1}{2}\right). \tag{43}$$

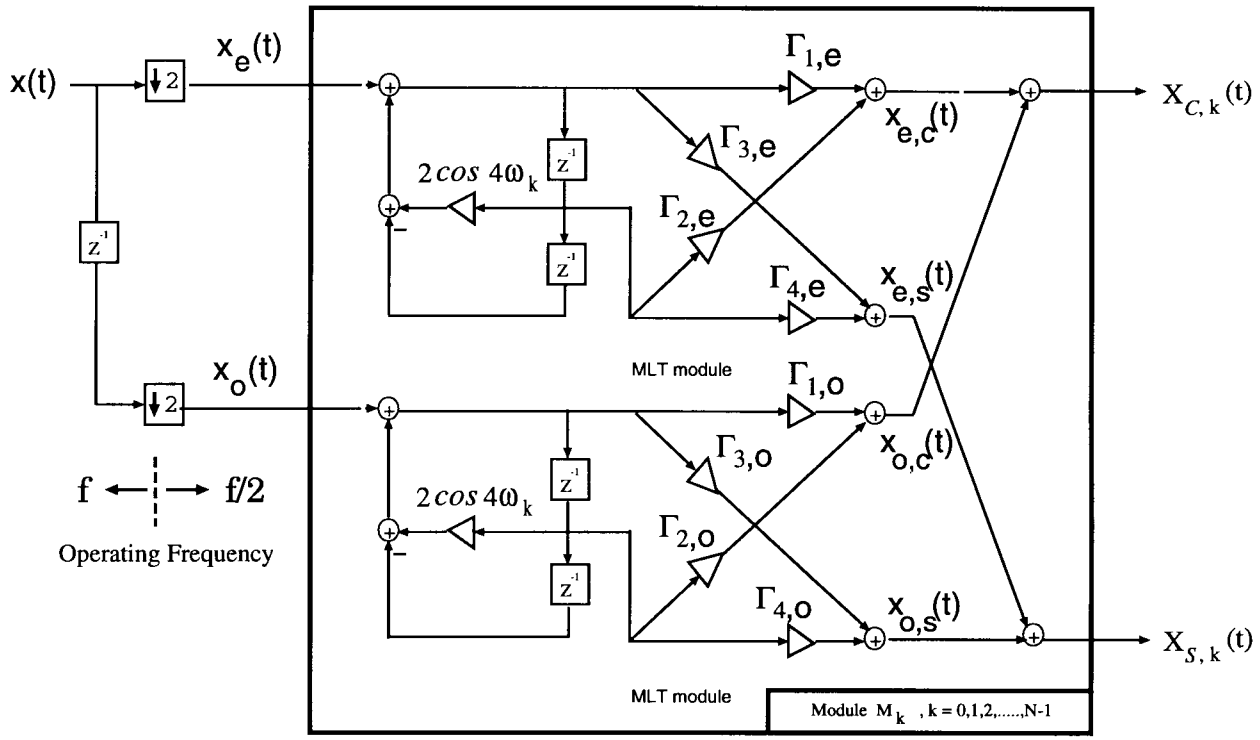


Fig. 18. Low-power IIR MLT module design.

Define the relationship in (29) and (40) as the *combination functions*. After comparing (29)–(32) with (40)–(43), we see that the MLT and ELT have identical mathematical structures except for the definitions of parameters and the combination functions. Therefore, the IIR MLT module in Fig. 17(a), as well as the low-power MLT module in Fig. 18, can be readily applied to ELT by simply modifying those multiplier coefficients. Also, the overall ELT architecture is similar to the MLT architecture in Fig. 17(b) except that the programmable interconnection network performs according to the combination function defined in (40). The hardware cost for the ELT can be found in Table 3. Since the number of multipliers of the ELT is about the same as that of the MLT, the power savings for both transforms are similar. Although the proposed architecture requires more hardware complexity compared with the single-output regressive DCT architecture in [72], the decomposition of the MLT/ELT into $\tilde{X}_{C,k}(t)$ and $\tilde{X}_{S,k}(t)$ in (29) and (40) is more systematic and fits into our unified programmable design very well (as we will discuss in the next section and Section VII). We therefore use the two-output MLT/ELT architectures for integrity of the presentation.

4) Unified Low-Power IIR Transform Module Design: From the transform functions described in (29)–(32) and (40)–(43), we observe that the low-power MLT architecture in Figs. 17 and 18 can be used to realize most existing discrete sinusoidal transforms by suitably setting the parameters and defining the combination functions. For example, $X_{C,k}(t)$ in (30) is equivalent to the DCT by setting

$$L = N, \quad \beta_1 = C(k), \quad \omega_k = \frac{k\pi}{2N}, \quad \text{and} \quad \theta_k = 0. \tag{44}$$

As a result, the multirate MLT module in Fig. 18 can compute the DCT with different index k in parallel.

The other example is the DFT with real-valued inputs. With the following parameter setting:

$$L = N, \quad \beta_1 = \frac{1}{\sqrt{N}}, \quad \omega_k = \frac{-k\pi}{N}, \quad \text{and} \quad \theta_k = -\omega_k \tag{45}$$

(30) and (31) become

$$X_{C,k}(t) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} \cos\left(\frac{-2\pi}{N} kn\right) x(t+n-N+1) \tag{46}$$

$$X_{S,k}(t) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} \sin\left(\frac{-2\pi}{N} kn\right) x(t+n-N+1) \tag{47}$$

which are the real part and the imaginary part of the DFT, respectively. The discrete Hartley transform (DHT) can be computed using the same parameter setting as the DFT except that the programmable interconnection network in Fig. 17(b) performs as

$$X_{\mathrm{DHT},k}(t) = X_{C,k}(t) + X_{S,k}(t). \tag{48}$$

The parameter settings as well as the corresponding combination functions for other orthogonal transforms are summarized in Table 5.

The programmable feature of the unified low-power module design makes it very attractive in transform-coding applications. First, the unified structure can be implemented as a high-performance programmable coprocessor, which performs various discrete transforms for the host processor

Table 5 Parameter Setting for the Unified Low-Power IIR Transform Coding Architecture

	Data Length	β_1	ω_k	θ_k	Combination Function
DCT	N	$C(k)$	$\frac{k\pi}{2N}$	0	$X_{DCT,k}(t) = X_{C,k}(t)$
IDCT	N	$C(1)$	$\frac{\pi}{2N}(n+\frac{1}{2})$	$-\omega_k$	$X_{IDCT,k}(t) = X_{C,k}(t) + (C(0) - C(1))x(n-N+1)$
DST-IV in [73]	N	$C(1)$	$\frac{\pi}{2N}(k+\frac{1}{2})$	0	$X_{DST,k}(t) = X_{S,k}(t)$
IDST-IV in [73]	N	$C(1)$	$\frac{\pi}{2N}(k+\frac{1}{2})$	0	$X_{IDST,k}(t) = X_{S,k}(t)$
MLT	$2N$	$\frac{1}{\sqrt{2N}}$	$\frac{k\pi}{2N}$	$\frac{\pi}{2}(k+\frac{1}{2})$	$X_{MLT,k}(t) = -S(k)[\, X_{C,k+1}(t) + X_{S,k}(t)\,]$
ELT	$4N$	$\frac{1}{2\sqrt{2N}}$	$\frac{\pi}{2N}(k+\frac{1}{2})$	$\frac{\pi}{2}(k+\frac{1}{2})$	$X_{ELT,k}(t) = -X_{S,k+1}(t) + \sqrt{2}X_{C,k}(t) + X_{S,k-1}(t)$
DFT	N	$\frac{1}{\sqrt{N}}$	$\frac{-k\pi}{N}$	$-\omega_k$	$Re\{X_{DFT,k}(t)\} = X_{C,k}(t),\ Im\{X_{DFT,k}(t)\} = X_{S,k}(t).$
DHT	N	$\frac{1}{\sqrt{N}}$	$\frac{-k\pi}{N}$	$-\omega_k$	$X_{DHT,k}(t) = X_{C,k}(t) + X_{S,k}(t).$

by loading the suitable parameters. Second, by hard wiring the multiplier coefficients of the modules to preset values according to the transformation type, we can perform any one of the discrete sinusoidal transforms based on the same architecture. This can significantly reduce the design cycle as well as the manufacturing cost.

C. Motion Estimation

In video coding, motion estimation has been shown to be very useful in reducing temporal redundancy but requires very high computational complexity. Therefore, numerous VLSI dedicated—or function-specific—architectures have been designed solely for motion estimation [74] with applications to high-definition television (HDTV) [75], video telephony, multimedia systems [76], asynchronous transfer mode of video [77], etc. The most commonly used motion-estimation architectures are based on the block matching motion estimation (BKM-ME) algorithm, which performs the matching of blocks between the current frame and the reference frame in terms of either mean square error (MSE) [29] or mean absolute difference (MAD) [28]. Extensive investigation has shown that using the MAD leads to a simpler implementation, while the performance is as good as when the MSE is used in motion-compensated prediction [78]. Hence, the MAD is usually employed as the cost function in practical designs.

Some exhaustive-search block-matching architectures have been proposed [79]–[81]. Due to the complexity of block matching, the hardware complexity is also high. Architectures for block matching based on hierarchical search strategies to reduce the computational complexity were presented in [82]–[85]. These structures require two or more sequential steps/stages to find suboptimal estimates. To avoid the blocking effect in BKM-ME, motion estimation based on lapped orthogonal transform (LOT) [69] and complex lapped transform (CLT) were proposed [86]. They still require searching over a larger search area and thus result in a very high computational burden. Moreover, motion estimation using CLT-ME is accurate on moving sharp edges but not on blur edges. In addition to block-based approaches, pel-based estimation methods such as pel-recursive algorithm [87], [88] and architecture [89], optical flow approach [90] have been proposed. However, the pel-based estimation methods are very vulnerable to noise by virtue of their involving only local operations and may suffer from the instability problem. In the category of transform-domain motion estimation, three-dimensional FFT has been successfully used to estimate motion in several consecutive frames [91]. But the FFT operating on complex numbers is not used in most video standards [92], and the global routing structure is undesirable in VLSI design. Also, it requires processing of several frames rather than two.

To further improve the compression rate and resolution, motion estimation with subpixel accuracy is essential as movements in a video sequence are not necessarily multiples of the sampling grid distance. Studies have revealed that the temporal prediction error variance is generally decreased by employing subpixel motion compensation. Also, beyond a certain "critical accuracy," the possibility of further improving prediction by using more accurate motion compensation schemes is small [93]. As a result, motion compensation with half-pel accuracy is recommended in MPEG-2 and H.263 standards. Some implementations that employ exhaustive block matching were proposed in [94] and [95]. In [95], the block matching is implemented with a 1-D systolic array. Then a half-pel precision processing unit is introduced to estimate half-pel motion vectors based on integer-pel accuracy using bilinear interpolation method. Interpolation, which is commonly used to perform spatial-domain fractional-pel motion estimation [96], not only increases the complexity and data flow of a coder but also may adversely affect the accuracy of motion estimated from the interpolated images. Therefore, the drawbacks of the implementation in [95] are the loss of accuracy and increased computational complexity.

1) Low-Power and High-Performance Motion-Estimator Design: Based on the concept of pseudophase and the sinusoidal orthogonal principles [46], [47], we design a novel low-power, low-complexity, and high-throughput CORDIC [48] architecture (CORDIC-HDXT-ME) for half-pel motion estimation. Techniques such as look-ahead [39], [40], [42], multirate [43], pipelining [2], and folding [97] have been combined and used in the design. This proposed multiplier-free and fully pipelined parallel architecture works solely at DCT domain without interpolation of input images to meet the needs of portable, high-quality, high-bit-rate picture transmission. To obtain an estimation of half-pel accuracy, we can have better flexibility and scalability by first computing the integer-pel motion vectors [98] and then considering only those around them to determine

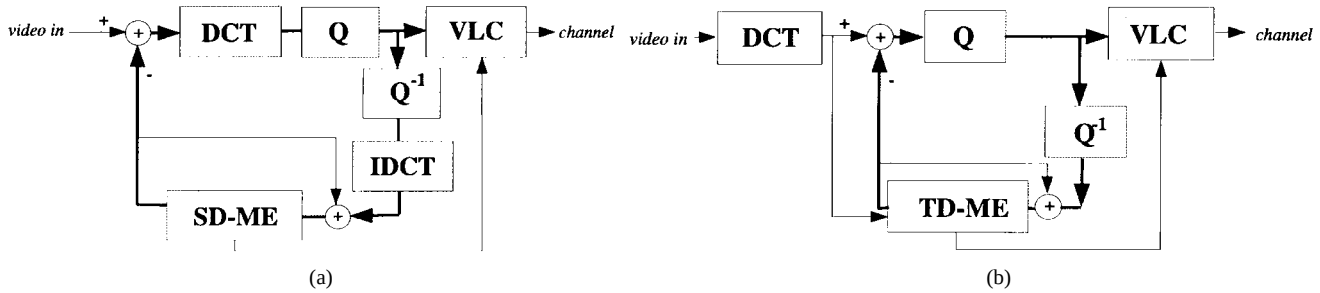


Fig. 19. Coder structures, where SD-ME stands for spatial-domain motion estimation and TD-ME stands for transform-domain motion estimation. (a) Conventional hybrid coder structure. (b) DCT-based coder structure.

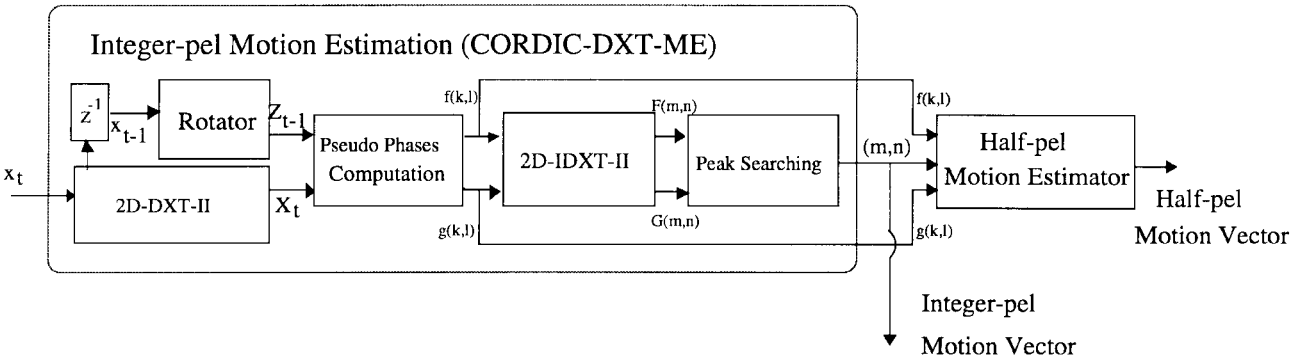


Fig. 20. The assembled block diagram of CORDIC-HDXT-ME.

the half-pel motion vectors [99]. This architecture's low computational complexity—$O(N^2)$ as compared with $O(N^4)$ for commonly used HBKM-ME [94], [95]—makes it attractive in real-time multimedia applications. Unlike the low-power video codec design in [100], our low-power design is achieved at the algorithmic and architectural levels. Also, unlike those fast block search integer-pel motion-estimation methods (such as three-step search [101], logarithmic search, etc.), which pick several displacement candidates out of all possible displacement values in terms of minimum MAD values of a reduced number of pixels, the DCT pseudophase technique employs the sinusoidal orthogonal principles to directly extract displacement information from the discrete sinusoidal DXT (DCT/DST) transform coefficients of images. With this direct extraction, we also avoid the operation of interpolation in finding half-pel motion vectors.

As explained in [46], the hybrid DCT motion-compensated video coder structure in Fig. 19(a) used in MPEG-2, H.261, and H.263 is not an efficient coder structure because the throughput of the coder is limited by the processing speed of the feedback loop. This becomes the major bottleneck of the entire digital video system for high-throughput, real-time applications. On the other hand, the DCT-based subpixel motion estimation [spatial-domain (SD)XT-ME] algorithm works solely in the DCT transform domain so that we can move DCT/IDCT out of the loop, as shown in Fig. 19(b).

Now the performance-critical feedback loop of the DCT-based coder contains only a transform-domain motion estimation (TD-ME) unit instead of three major components (DCT, IDCT, SD-ME), as in the conventional hybrid DCT motion-compensated video coder. This not only reduces the complexity of the coder but also achieves higher system throughput. In addition to its low complexity, its ability to estimate motion completely in DCT domain enables us to replace all multiply-accumulate (MAC) operations in plane rotation with CORDIC processors [48] and to efficiently combine the two major processing components—the DCT and motion estimation—into one single component of relatively low complexity. Furthermore, from the implementation point of view, we can get rid of the IDCT module used for half-pel motion estimation by interleaving it with the DCT module. Those major features, along with the regular, modular, and only local connected properties of the proposed CORDIC-HDXT-ME architecture, make the MPEG-2, H.263 compatible real-time video codec design on a single dedicated chip feasible without sacrificing the performance.

According to the SDXT-ME algorithm in [46] and [47], we derive the assembled block diagram of the CORDIC-HDXT-ME architecture outlined in Fig. 20. It has four major processing stages/phases.

1) The $N \times N$ block of pixels at the current frame x_t are fed into a *type II DCT/DST* (2D-DXT-II) [73], [102] coder, which computes four coefficients $X_t^{cc}(k,l)$,

$X_t^{cs}(k,l)$, $X_t^{sc}(k,l)$, and $X_t^{ss}(k,l)$ such as

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

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

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

2) The pseudophases computation module utilizes all the previous parameters and takes $O(N)$ times to produce two pseudophase functions $f(k,l)$ and $g(k,l)$.
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 $G(m,n)$.
4) Last, we search the peak values based on sinusoidal orthogonal principles among $F(m,n)$ and $G(m,n)$ to determine the integer-pel motion vector (m,n).

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

In what follows, we describe the detailed design of each block in Fig. 20.

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

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

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

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

where

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

Here, the time index t in $X_t^c(k)$ and $X_t^s(k)$ denotes that the transform starts from $x(t)$. Unlike the commonly used two-dimensional (2-D) DCT structures (such as matrix factorization [103], systolic structure implementation [104], etc.), the time-recursive DCT architecture derived in [44] is able to generate DCT and DST outputs simultaneously, which can be used in computing pseudophases later. The time-recursive updating of 1D-DXT-II is given by

$$\begin{bmatrix} X_{t+1}^c(k) \\ X_{t+1}^s(k) \end{bmatrix} = \begin{bmatrix} \cos\frac{k\pi}{N} & \sin\frac{k\pi}{N} \\ -\sin\frac{k\pi}{N} & \cos\frac{k\pi}{N} \end{bmatrix} \cdot \left[\begin{bmatrix} X_t^c(k) \\ X_t^s(k) \end{bmatrix} + \begin{bmatrix} \cos\frac{k\pi}{2N} & \sin\frac{k\pi}{2N} \\ -\sin\frac{k\pi}{2N} & \cos\frac{k\pi}{2N} \end{bmatrix} \cdot \begin{bmatrix} \frac{2}{N}(-x(t)+(-1)^k x(t+N)) \\ 0 \end{bmatrix}\right]. \quad (49)$$

Note that the orthogonal rotation of the lattice structure in (49) is numerically stable so that the roundoff errors will not be accumulated. This nice numerical property is very useful in finite-precision implementation.

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

$$\begin{bmatrix} x_{t+1}^c(n) \\ x_{t+1}^{as}(n) \end{bmatrix} = \begin{bmatrix} \cos\left(\frac{2n+1}{2N}\right)\pi & \sin\left(\frac{2n+1}{2N}\right)\pi \\ -\sin\left(\frac{2n+1}{2N}\right)\pi & \cos\left(\frac{2n+1}{2N}\right)\pi \end{bmatrix}\left[\begin{bmatrix} x_t^c(n) \\ x_t^{as}(n) \end{bmatrix} + \begin{bmatrix} -\frac{1}{\sqrt{2}}X(t) \\ (-1)^n X(t+N) \end{bmatrix}\right] + \begin{bmatrix} \left(\frac{1}{\sqrt{2}}-1\right)X(t+1) \\ 0 \end{bmatrix}. \quad (50)$$

The auxiliary variable $x_t^{as}(n)$ is related to $x_t^s(n)$ by$x_{t+1}^s(n) = x_t^{as}(n) + (-1)^n \frac{1}{\sqrt{2}}X(t+N)$. From the above discussion, we observe that both 1D-DXT-II and its inverse 1D-IDXT-II share a common computational module with only some minor differences in the data inputs and the rotation angles. Therefore, we can integrate both DCT and IDCT by interleaving for different functionalities.

Because the time-recursive 1D-DXT/IDXT-II module has an inherent feedback loop, we can use the look-ahead method to achieve low-power design. The two-stage look-ahead relation for DCT/DST is given by

$$\begin{bmatrix} X_{t+2}^c(k) \\ X_{t+2}^s(k) \end{bmatrix} = \begin{bmatrix} \cos\frac{2k\pi}{N} & \sin\frac{2k\pi}{N} \\ -\sin\frac{2k\pi}{N} & \cos\frac{2k\pi}{N} \end{bmatrix} \cdot \left[\begin{bmatrix} X_t^c(k) \\ X_t^s(k) \end{bmatrix} + \begin{bmatrix} \cos\frac{k\pi}{2N} & \cos\frac{3k\pi}{2N} \\ \sin\frac{k\pi}{2N} & \sin\frac{3k\pi}{2N} \end{bmatrix} \cdot \begin{bmatrix} -x(t)+(-1)^k x(t+N) \\ -x(t+1)+(-1)^k x(t+N+1) \end{bmatrix}\right]. \quad (51)$$

Furthermore, the two-stage look-ahead 1D-DXT-II can be interleaved with 1D-IDXT-II into one unified structure that contains three CORDIC processors. Clearly, the look-ahead system can be clocked at a two-times faster frequency than that of the original system. By following our arguments in

Section I-C, this system can also be used to obtain a low-power implementation. Since the capacitances along the critical path remain the same after look-ahead transformation, we get $V_{\text{look}} = 3.08$ V. Provided that the capacitances due to the CORDIC's are dominant in the circuit and are roughly proportional to the number of CORDIC's, $P_{\text{look}} = C_{\text{look}} V_{\text{look}}^2 f_{\text{look}} = (\frac{3}{2} C_{\text{dxt}})(3.08V)^2(\frac{1}{2} f_{\text{dxt}}) = 0.28 P_{\text{dxt}}$, where P_{dxt} denotes the power consumption of the original DCT/IDCT.

In the third stage of Fig. 20, two pseudophase functions, $f(k,l)$ and $g(k,l)$, pass through 2D-IDXT-II module (*type II inverse IDCST* and *IDSCT*) to generate $F(m,n)$ and $G(m,n)$ in view of the orthogonal property

$$\begin{aligned} F(m,n) &= \textit{IDCSTII}(f(k,l)) \\ &= \frac{4}{N^2} \sum_{k=0}^{N-1} \sum_{l=1}^{N} C(k)C(l) f(k,l) \\ &\quad \cdot \cos\frac{k\pi}{N}\left(m+\frac{1}{2}\right) \sin\frac{l\pi}{N}\left(n+\frac{1}{2}\right) \\ &= [\delta(m-m_u) + \delta(m+m_u+1)] \\ &\quad \cdot [\delta(n-m_v) - \delta(n+m_v+1)] \qquad (52) \\ G(m,n) &= \textit{IDSCTII}(g(k,l)) \\ &= \frac{4}{N^2} \sum_{k=1}^{N} \sum_{l=0}^{N-1} C(k)C(l) g(k,l) \\ &\quad \cdot \sin\frac{k\pi}{N}\left(m+\frac{1}{2}\right) \cos\frac{l\pi}{N}\left(n+\frac{1}{2}\right) \\ &= [\delta(m-m_u) - \delta(m+m_u+1)] \\ &\quad \cdot [\delta(n-m_v) + \delta(n+m_v+1)] \qquad (53) \end{aligned}$$

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

To interleave the DCT module used in the first stage with the IDCT module in the third stage of Fig. 20, we extend and modify the 2-D DCT structure in [12] to simultaneously generate *type II* DCCT, DCST, DSCT and DSST coefficients when the pixels of the current frame x_t are selected, or $F(m,n)$ and $G(m,n)$ when $f(k,l)$ and $g(k,l)$ are in turn selected, as shown in Fig. 21. To dually generate $F(m,n)$ and $G(m,n)$, we use a multiplexer to control the input data flow by choosing N of $f(k,l)$ $l \in \{1,\cdots,N\}$ followed by $g(k,l)$ $l \in \{0,\cdots,N-1\}$ alternately for different k. As a result, the module generates 1-D *IDST*$(f(k,l))$, which enters the lower circular shift matrix in Fig. 21, and *IDCT*$(g(k,l))$, which enters the upper matrix in turn. The 2-D *IDCST*$(f(k,l))$ and *IDSCT*$(g(k,l))$ can be obtained independently afterwards. In Fig. 21, the 2D-DXT-II/2D-IDXT-II module is composed of three 1D-DXT-II/1D-IDXT-II arrays; thus, the coder needs a total of $9N$ CORDIC's and $27N+12$ adders for 2D-DXT-II and its inverse 2D-IDXT-II.

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 pseudophase computation in the second stage of Fig. 20 can actually be obtained by the plane rotation of the *type II* 2D-DXT-II kernels, as shown in Fig. 22(a). By inserting the latches across any feed-forward cut-set shown in Fig. 22(b), we obtain a pipelined architecture at the expense of an increase in the output latency. Now the pipelined structure, which still requires four CORDIC's, can be run at a two-times faster clock rate than that of the original design without pipelining. By reducing the supply voltage, however, we can increase the propagation delay of the pipelined system until it equals that of the origin system. Because the capacitances along the critical path get halved, by following the similar derivations in Section I-C, we get $V_{\text{pipe}} = 3.08$ V and $P_{\text{pipe}} = C_{\text{pipe}} V_{\text{pipe}}^2 f_{\text{pipe}} = (\frac{4}{4} C_{\text{conv}})(3.08V)^2(\frac{1}{2} f_{\text{conv}}) = 0.19 P_{\text{conv}}$, where P_{conv} denotes the power consumption of the original conversion module. The rotation module in Fig. 20 consists of N such units, as shown in Fig. 22(b), for parallel rotations of different channels. It requires a total of $4N$ CORDIC's and takes $O(N)$ time to perform transformation from 2D-DXT-II to 2D-DXT-I.

E. Pseudophase Computation and Peak Searching

As mentioned in [46], the pseudophase 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*

$$\underbrace{\begin{bmatrix} Z_{t-1}^{cc}(k,l) & -Z_{t-1}^{cs}(k,l) & -Z_{t-1}^{sc}(k,l) & Z_{t-1}^{ss}(k,l) \\ Z_{t-1}^{cs}(k,l) & Z_{t-1}^{cc}(k,l) & -Z_{t-1}^{ss}(k,l) & -Z_{t-1}^{sc}(k,l) \\ Z_{t-1}^{sc}(k,l) & -Z_{t-1}^{ss}(k,l) & Z_{t-1}^{cc}(k,l) & -Z_{t-1}^{cs}(k,l) \\ Z_{t-1}^{ss}(k,l) & Z_{t-1}^{sc}(k,l) & Z_{t-1}^{cs}(k,l) & Z_{t-1}^{cc}(k,l) \end{bmatrix}}_{\mathbf{Z}_{t-1}(k,l)} \cdot \underbrace{\begin{bmatrix} g^{CC}(k,l) \\ g^{CS}(k,l) \\ g^{SC}(k,l) \\ g^{SS}(k,l) \end{bmatrix}}_{\vec{\theta}_{m,n}(k,l)} = \underbrace{\begin{bmatrix} X_t^{cc}(k,l) \\ X_t^{cs}(k,l) \\ X_t^{sc}(k,l) \\ X_t^{ss}(k,l) \end{bmatrix}}_{\vec{\mathbf{x}}_t(k,l)}$$
$$\text{for } k,l \in \{1,\cdots,N-1\} \qquad (54)$$

where (m,n) is the displacement, $\vec{\theta}_{m,n}$ is the pseudophase vector, which also equals $[\star, f(k,l), g(k,l), \star]^{\mathbf{T}}$, and $\star$ stands for "don't care."

If we adopt Gaussian elimination or the Givens rotation method to solve for pseudophase $f(k,l)$ and $g(k,l)$ in the system equation, it incurs a heavy computational burden. However, by exploring the structure embedded in the $\mathbf{Z}_{t-1}(k,l)$, we can reduce the 2-D problem to 1-D. Because we are only interested in solving pseudophase $f(k,l)$ and $g(k,l)$, we multiply both sides of the system equation by $\mathbf{Z}_{t-1}^T(k,l)$ and get the reduced 2×2 matrix equation

$$\begin{aligned} &\begin{bmatrix} E & F \\ F & E \end{bmatrix} \begin{bmatrix} f(k,l) \\ g(k,l) \end{bmatrix} \\ &= \underbrace{\begin{bmatrix} Z_{t-1}^{cs}(k,l) & Z_{t-1}^{sc}(k,l) \\ Z_{t-1}^{sc}(k,l) & Z_{t-1}^{cs}(k,l) \end{bmatrix} \begin{bmatrix} -X_t^{cc}(k,l) \\ X_t^{ss}(k,l) \end{bmatrix}}_{\Gamma_1} \\ &\quad + \underbrace{\begin{bmatrix} Z_{t-1}^{cc}(k,l) & -Z_{t-1}^{ss}(k,l) \\ -Z_{t-1}^{ss}(k,l) & Z_{t-1}^{cc}(k,l) \end{bmatrix} \begin{bmatrix} X_t^{cs}(k,l) \\ X_t^{sc}(k,l) \end{bmatrix}}_{\Gamma_2} \triangleq \begin{bmatrix} A \\ B \end{bmatrix} \end{aligned} \qquad (55)$$

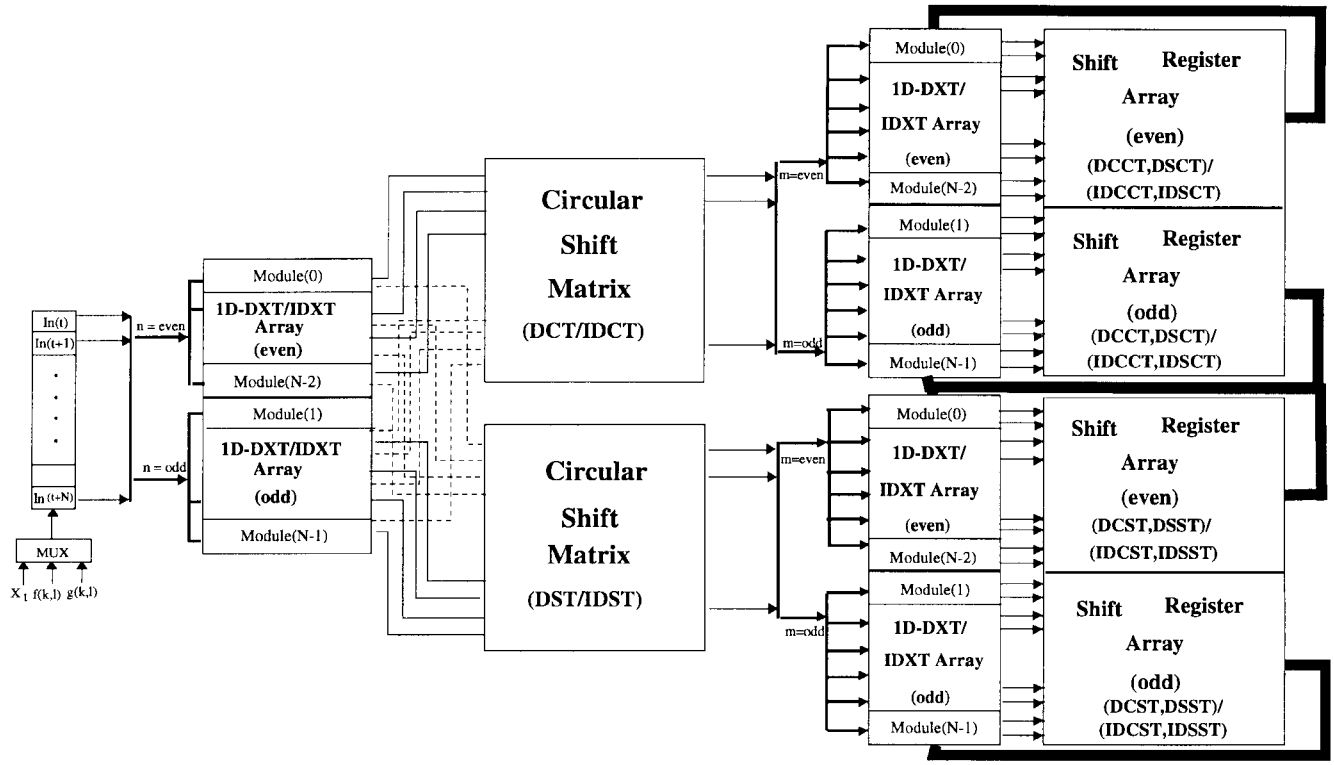


Fig. 21. Time-recursive structure to compute DCCT/DCST/DSCT/DSST and the inverse.

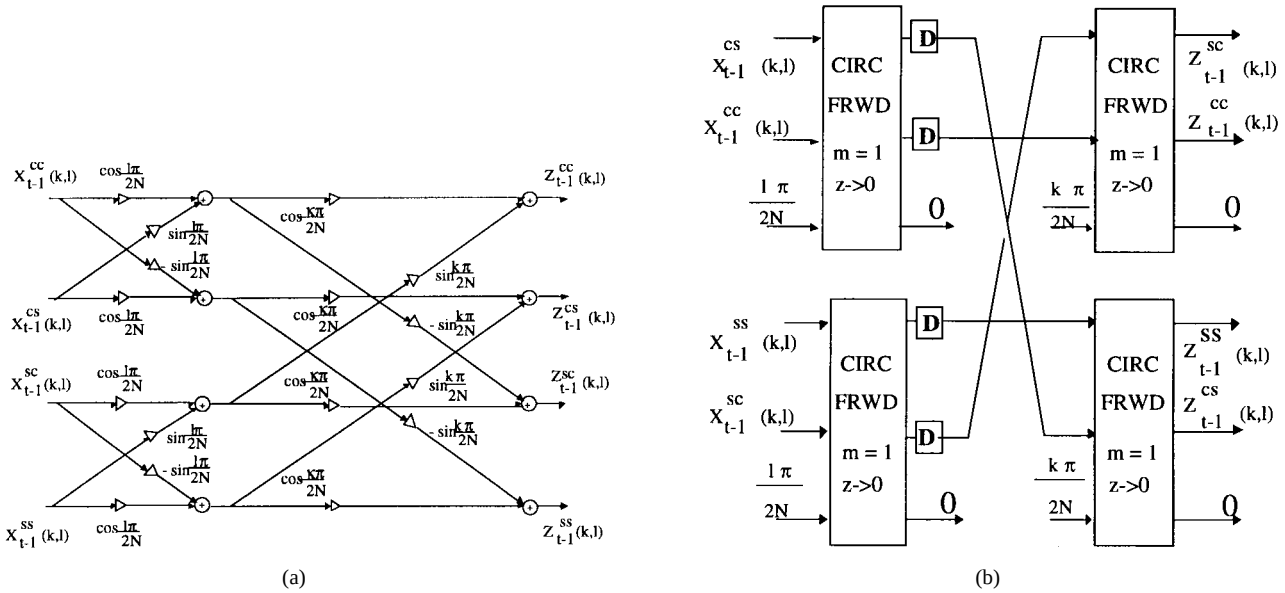


Fig. 22. The architecture for 2D-DXT-II to 2D-DXT-I conversion. (a) Lattice structure. (b) Low-power design.

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

$$\begin{bmatrix} f(k,l) \\ g(k,l) \end{bmatrix} = \frac{1}{\Delta} \begin{bmatrix} E & -F \\ -F & E \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix}, \quad \text{where } \Delta = E^2 - F^2. \tag{56}$$

From the above discussion, we observe that those computations like Γ_1 and Γ_2 in (25) are similar. We thus employ the folding technique [97] to save area. The folded architecture in Fig. 23 requires ten CORDIC's. The hardware complexity is almost halved compared to the unfolded pipeline architecture. This reduction in hardware is achieved at the expense of increasing the latency of the module. The switches' settings in Fig. 23 are for E, Γ_1, and the complementary settings are for F, Γ_2. The

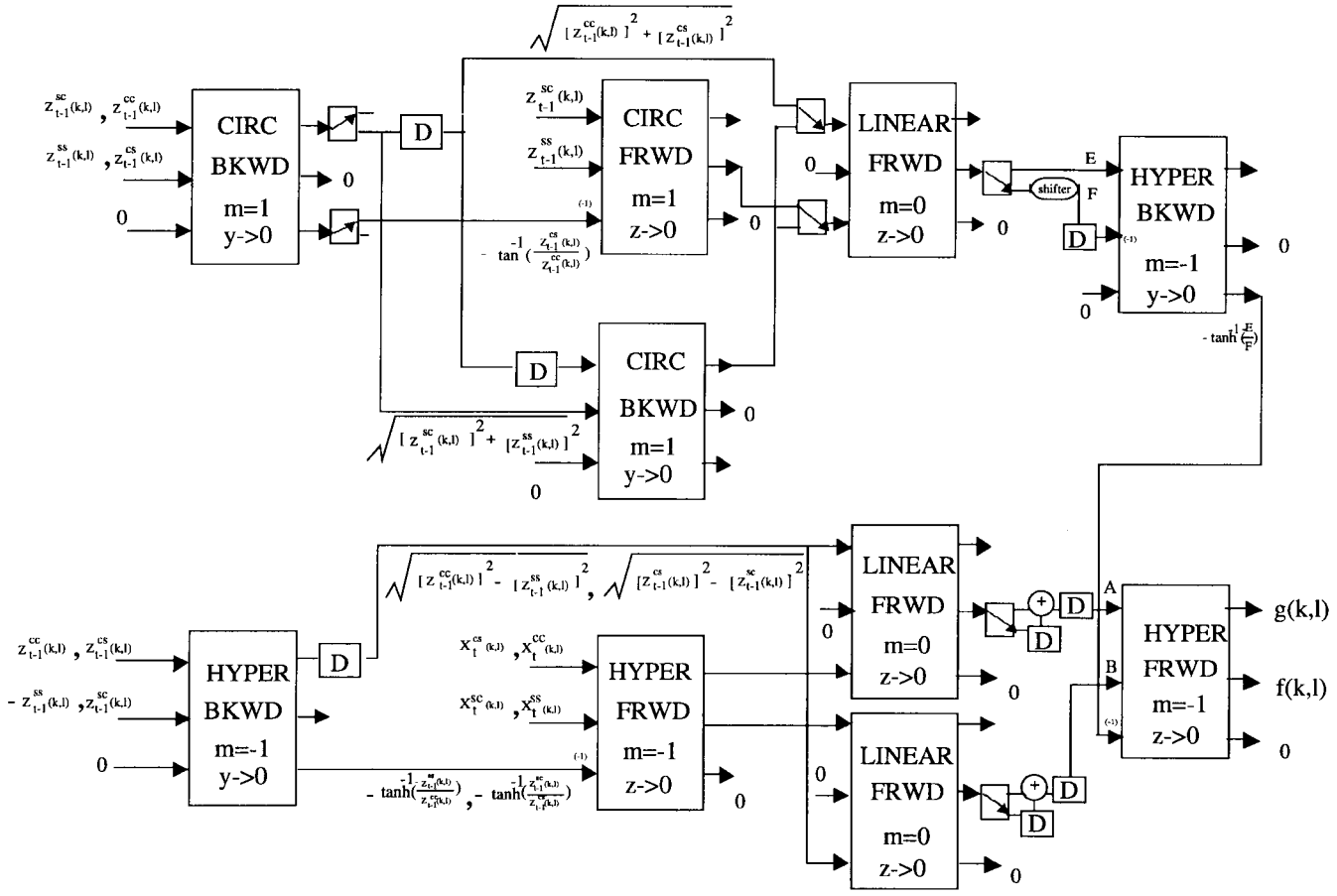


Fig. 23. Folded pipeline structure for pseudophase computation.

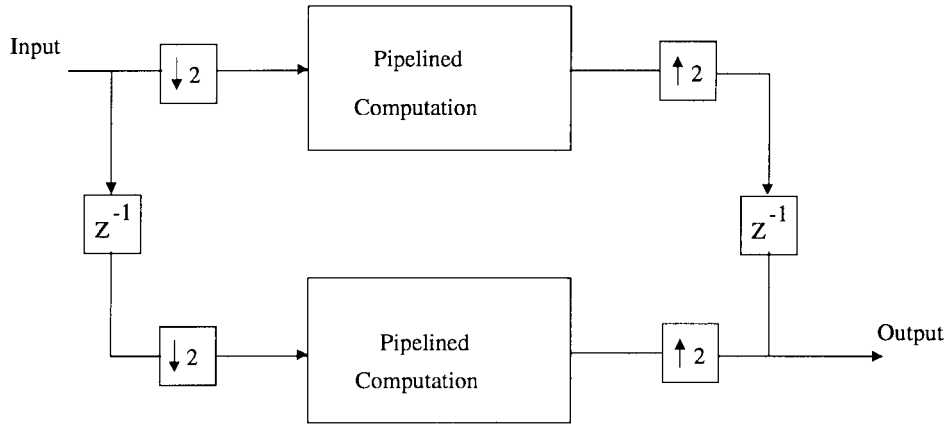


Fig. 24. Low-power design for pseudophase computation.

latches (D's in Fig. 23) driven by two nonoverlap clocks are introduced across the feed-forward cut-set in order to reduce the critical path and to synchronize the pseudophase computation. Here, some CORDIC processors are operated in *hyperbolic rotation mode* [48] to evaluate A, B, 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 pseudophase functions $f(k,l)$ and $g(k,l)$.

By following the multirate design procedure in Section II, we can obtain the low-power multirate structure with $M = 2$ outlined in Fig. 24, where the *pipelined computation* module corresponds to that in Fig. 23. The multirate design needs 20 CORDIC's. Based on the arguments in Section II-B3, it can be shown that the power consumption for the multirate design can be roughly estimated as $P_{\text{multi}} = C_{\text{multi}} V_{\text{multi}}^2 f_{\text{multi}} = (\frac{20}{10} C_{\text{phase}})(3.08V)^2(\frac{1}{2} f_{\text{phase}}) = 0.38 P_{\text{phase}}$, where P_{phase} denotes the power consumption of the original pseudophase computation. In summary, we can parallelize N such multirate modules for different channels to speed up

the pseudophase computation. It takes $O(N)$ time to accomplish the work, and the hardware cost is $20N$ CORDIC's and $2N$ adders.

The 2-D search for the peak value among $F(m,n)$ and $G(m,n)$ can be reduced to the 1-D search by the row-column decomposition search. The decomposition search looks for the peak value of each row, followed by a vertical search of the previous results, as shown in Fig. 25(a) (here, we use $G(m,n)$ as an example, it is also applicable to $F(m,n)$). Since it is fully pipelined, a careful design to optimize the speed or reduce the power consumption of the system is required by inserting latches along the critical path. The latency of peak search is $2N$. The peak search needs $2(N+1)$ process elements (PE's), and the hardware structure of a PE is shown in Fig. 25(b).

F. Time-Recursive Programmable Module for Half-Pel Motion Estimator

To obtain an estimation at half-pel accuracy, we can first compute the integer-pel motion estimate (m,n) and then utilize the half-pel motion estimator block to produce $\overline{DCS}(u,v)$ and $\overline{DSC}(u,v)$ for $u \in \{m-0.5, m, m+0.5\}$ and $v \in \{n-0.5, n, n+0.5\}$, as shown in Fig. 20. The $\overline{DCS}(u,v)$ and $\overline{DSC}(u,v)$ are defined as follows [47]:

$$\begin{aligned}\overline{DCS}(u,v) = &\sum_{k=0}^{N-1}\sum_{l=1}^{N} C(k)C(l)f(k,l) \\ &\cdot \cos\frac{k\pi}{N}\left(m+\mu_u+\frac{1}{2}\right) \\ &\cdot \sin\frac{l\pi}{N}\left(n+\mu_v+\frac{1}{2}\right) \qquad (57)\\ \overline{DSC}(u,v) = &\sum_{k=1}^{N}\sum_{l=0}^{N-1} C(k)C(l)g(k,l) \\ &\cdot \sin\frac{k\pi}{N}\left(m+\mu_u+\frac{1}{2}\right) \\ &\cdot \cos\frac{l\pi}{N}\left(n+\mu_v+\frac{1}{2}\right) \qquad (58)\end{aligned}$$

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

For the computation of the two-dimensional $\overline{DSC}(u,v)$, we can decompose its computation into tree-structured hierarchical *one-dimensional type II inverse IDCT/IDST*, as shown in Fig. 26 (here, we use $\overline{DSC}(u,v)$ as an example; the same approach can be applied to $\overline{DCS}(u,v)$). Note that the computations encircled by dot boxes in Fig. 26 are the same except for the phase differences. Let us define

$$\begin{aligned}G_t^c(l) &= \frac{2}{N}\sum_{l=t}^{t+N-1} C(l-t)g(k,l)\cos\phi \\ G_t^s(l) &= \frac{2}{N}\sum_{l=t+1}^{t+N} C(l-t)g(k,l)\sin\phi \qquad (59)\end{aligned}$$

Table 6 Components of Different Phases in (60) and (61)

	$n+\frac{1}{2}$	n	$n+1$
J	$\mathbf{\Phi}(2)$	$\mathbf{\Phi}(2)\cdot\mathbf{\Theta}^{-1}(2)$	$\mathbf{\Phi}(2)\cdot\mathbf{\Theta}(2)$
H	$\begin{bmatrix}\epsilon\\ \zeta\end{bmatrix}$	$\begin{bmatrix}\epsilon+\zeta\\ 0\end{bmatrix}$	$\begin{bmatrix}\epsilon-\zeta\\ 0\end{bmatrix}$
L	$\mathbf{\Phi}(1)$	$\mathbf{\Phi}(1)\cdot\mathbf{\Theta}^{-1}(1)$	$\mathbf{\Phi}(1)\cdot\mathbf{\Theta}(1)$
M	$\begin{bmatrix}\eta\\ \theta\end{bmatrix}$	$\begin{bmatrix}\eta+\theta\\ 0\end{bmatrix}$	$\begin{bmatrix}\eta-\theta\\ 0\end{bmatrix}$
N	$\begin{bmatrix}\kappa\\ 0\end{bmatrix}$	**0**	**0**

$$\mathbf{\Phi}(m) = \begin{bmatrix} cos\frac{(n+\frac{1}{2})m\pi}{N} & sin\frac{(n+\frac{1}{2})m\pi}{N} \\ -sin\frac{(n+\frac{1}{2})m\pi}{N} & cos\frac{(n+\frac{1}{2})m\pi}{N}\end{bmatrix}$$

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

$$\epsilon = -\tfrac{1}{\sqrt{2}}g(k,t);\ \zeta = (-1)^n g(k,t+N);\ \eta = -g(k,t+1);$$
$$\theta = (-1)^n g(k,t+N+1);\ \kappa = (-1)^n(\tfrac{1}{\sqrt{2}}-1)g(k,t+N+1);$$

where ϕ stands for different phases such as $\frac{l\pi}{N}n$, $\frac{l\pi}{N}(n+\frac{1}{2})$, and $\frac{l\pi}{N}(n+1)$ in the middle level of Fig. 26. Therefore, by using the same time-recursive approach for inverse 1D-IDXT-II described in Section III-C, we can obtain the one-step time-recursive updating similar to (50) [99]. To achieve the low-power design, we need to double the step size to get the two-step look-ahead time-recursive updating for different phases. Those updating equations are similar but with only some minor differences in the data paths and the rotation angles. Therefore, we can combine them to obtain the following unified equation to generate $G_{t+2}^c(l)$ and $G_{t+2}^s(l)$ simultaneously:

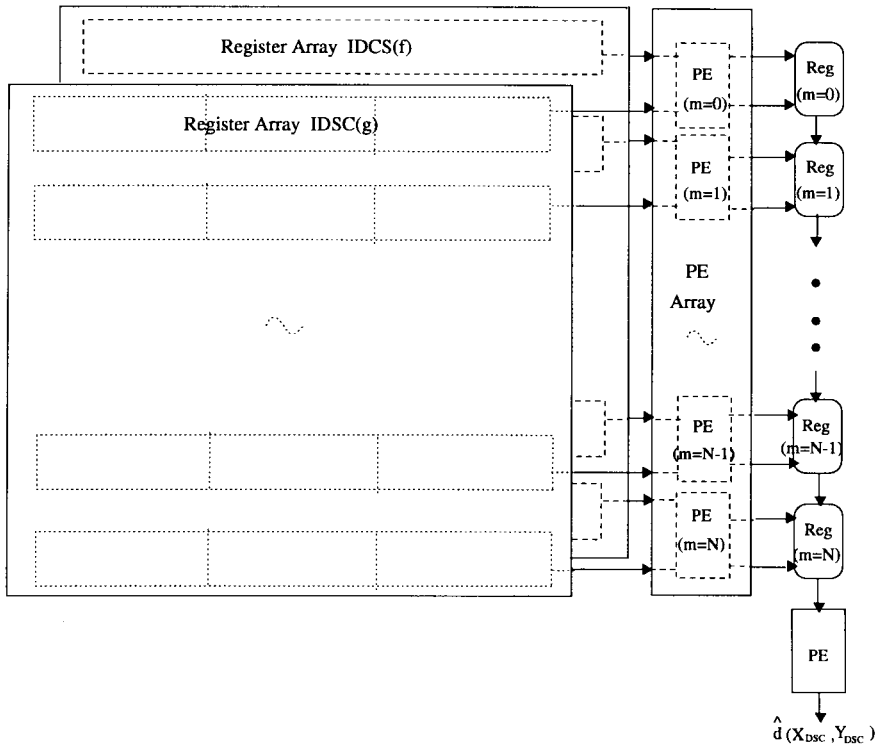
$$\begin{aligned}\begin{bmatrix}G_{t+2}^c(l)\\ G_{t+2}^{as}(l)\end{bmatrix} = &\mathbf{J}\left[\begin{bmatrix}G_t^c(l)\\ G_t^{as}(l)\end{bmatrix}+\mathbf{H}\right] \\ &+\mathbf{L}\cdot\mathbf{M}+\begin{bmatrix}(\frac{1}{\sqrt{2}}-1)g(k,t+2)\\ 0\end{bmatrix}. \qquad (60)\end{aligned}$$

The auxiliary variable $G_{t+2}^{as}(l)$ is related to $G_{t+2}^s(l)$ by

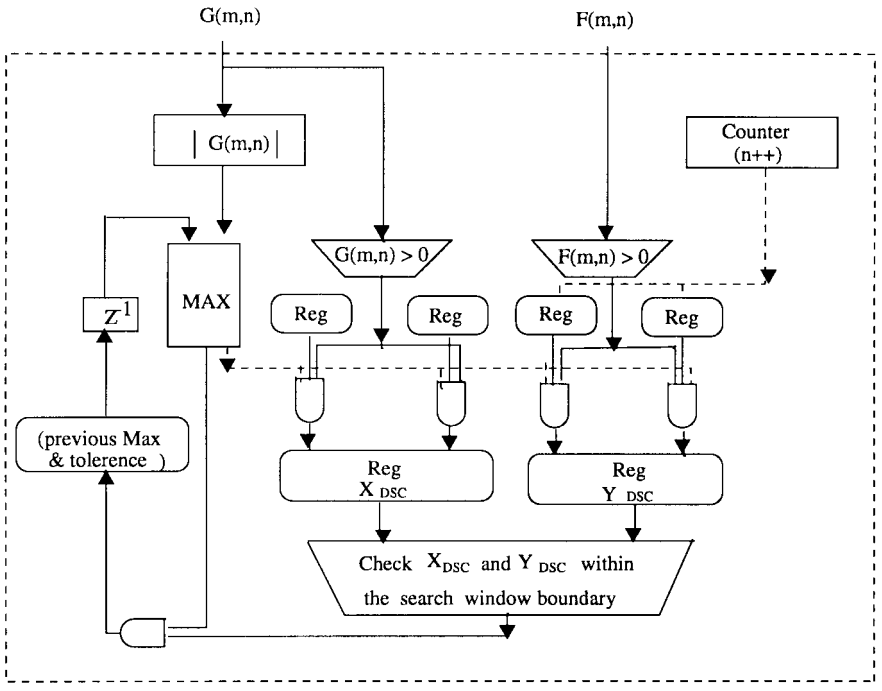
$$\begin{bmatrix}G_{t+2}^s(l)\\ \star\end{bmatrix} = \mathbf{L}\begin{bmatrix}G_{t+2}^{as}(l)\\ G_{t+2}^c(l)+(\frac{1}{\sqrt{2}}-1)g(k,t+2)\end{bmatrix}+\mathbf{N}. \qquad (61)$$

The corresponding parameters **H**, **J**,, **L**, **M**, and **N** in (60) and (61), depending on the different phases, are listed in Table 6. The unified programmable module requires three CORDIC's. Provided that the capacitances due to the CORDIC's are dominant in the circuit and are roughly proportional to the number of CORDIC's, we get $P_{\text{2-stage}} = C_{\text{2-stage}}V_{\text{2-stage}}^2 f_{\text{2-stage}} = (\frac{3}{2}C_{\text{half}})(\frac{3.08V}{5V})^2(\frac{1}{2}f_{\text{half}}) = 0.28P_{\text{half}}$, where P_{half} denotes the power consumption of the original half-pel motion estimator block.

$N+3$ such programmable modules can be used for parallel computing of $\overline{DSC}(u,v)$ for different channels [99]. It takes $O(N)$ time to produce $\overline{DSC}(u,v)$ for $u \in \{m-0.5, m, m+0.5\}$ and $v \in \{n-0.5, n, n+0.5\}$ and takes $O(1)$ time to find the peak position corresponding to the half-pel motion vector among nine possible locations around (m,n).



(a)



(b)

Fig. 25. The two-dimensional peak search structure. (a) Scheme diagram. (b) PE design.

G. Hardware Cost and Simulation Results

Based on the above discussion, the hardware cost and throughput of each stage in Fig. 20 are summarized in Table 7. The structure is a scalable design that uses $33N$ CORDIC processors and $29N + 12$ adders to perform an integer-pel motion estimation and requires an additional $3N+9$ CORDIC's and $7N+33$ adders to perform half-pel motion estimation. In Fig. 27(a), we move the image $X1$ in the direction (3.5,−1.5) corresponding to image $X2$ with additive Gaussian noise at SNR $= 40$ dB. Our simulation of

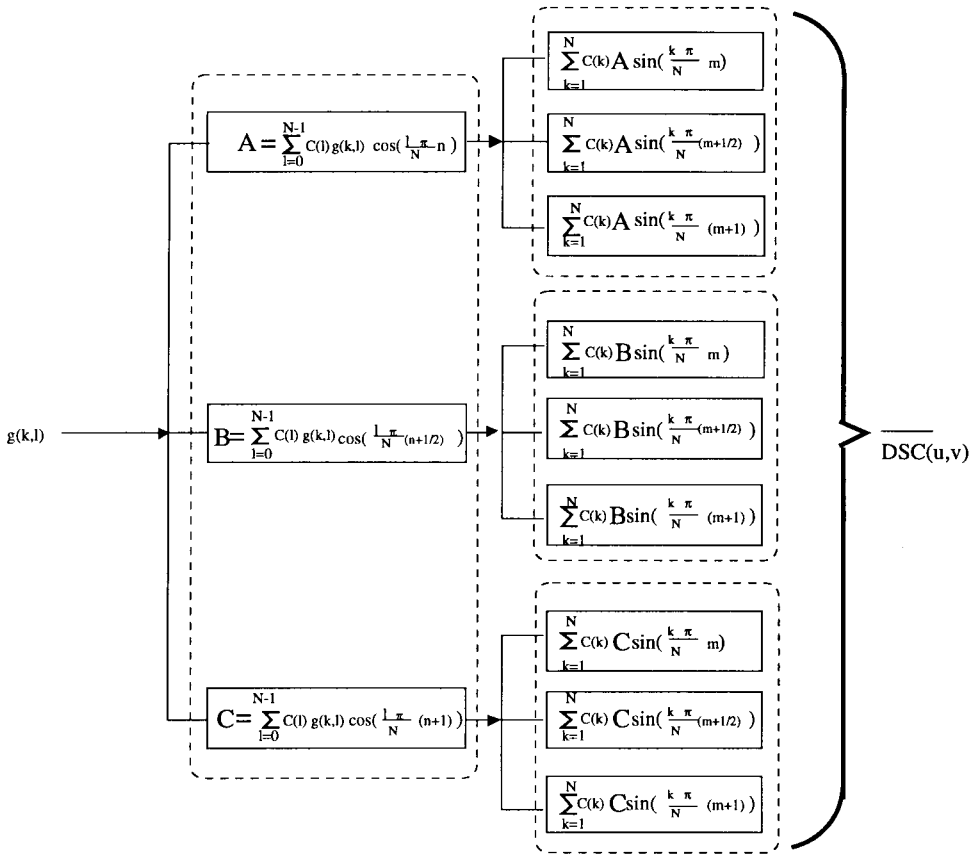


Fig. 26. Schematic diagram for computing $\overline{DSC}(u, v)$.

Table 7 Hardware Cost and Throughput

Component	CORDICs	Adders	latches	Registers	Throughput
2D-DXT/IDXT	9N	27N+12	6N	$N + 6N^2$	$O(N)$
Conversion	4N	0	4N	0	$O(N)$
Pseudo Phase	20N	2N	12N	0	$O(N)$
Peak Searching	0	0	2N+2	$2N^2$	$O(N)$
Half-pel Estimator	3N+9	7N+33	2N+6	$3N + N^2$	$O(N)$
Total	36N+9	36N+45	26N+8	$4N + 9N^2$	$O(N)$

the designed low-power CORDIC-HDXT-ME architecture shows that it can estimate the correct integer-pel motion vector (3,−1) and half-pel motion vector (3.5,−1.5), as shown in Fig. 27(b) and (c), respectively. Simulation is also made on the "Miss America" sequence in quarter-common intermediate format, whose frame size is 176 × 144. The original frames 90 and 91 are shown in Fig. 28(a) and (b), and the reconstructed frame 91 based on the moving vectors obtained from low-power CORDIC-HDXT-ME is shown in Fig. 28(c).

IV. Channel Coding

A. RS Coding

RS codes have come into widespread use for forward error correction in communication and storage systems. RS codes, which are a special case of Bose–Chaudhuri–Hocquenheim codes, are a popular choice to provide data integrity because they can provide good error-correction capability for random and bursty errors alike. In space applications, a concatenated scheme consisting of a convolutional inner code and an RS outer code has been accepted as a standard [105]. In audio compact discs, a cross-interleaved pair of RS codes is used to provide error protection [106]. The U.S. Cellular Digital Packet Data service has adopted the (63,47) RS code [107]. RS codes are being considered to provide forward error protection against impulse noise in digital subscriber line (xDSL) applications over telephone lines [108]. They are also good candidates for use in portable wireless applications as a part of concatenated coding systems with convolutional codes [109].

An (n, k) RS code with $n - k = 2t$ consecutive roots in $GF(q)$—viz, $\alpha^b, \alpha^{b+1}, \cdots, \alpha^{b+2t-1}$, where α is the primitive element of $GF(q)$—has a minimum distance

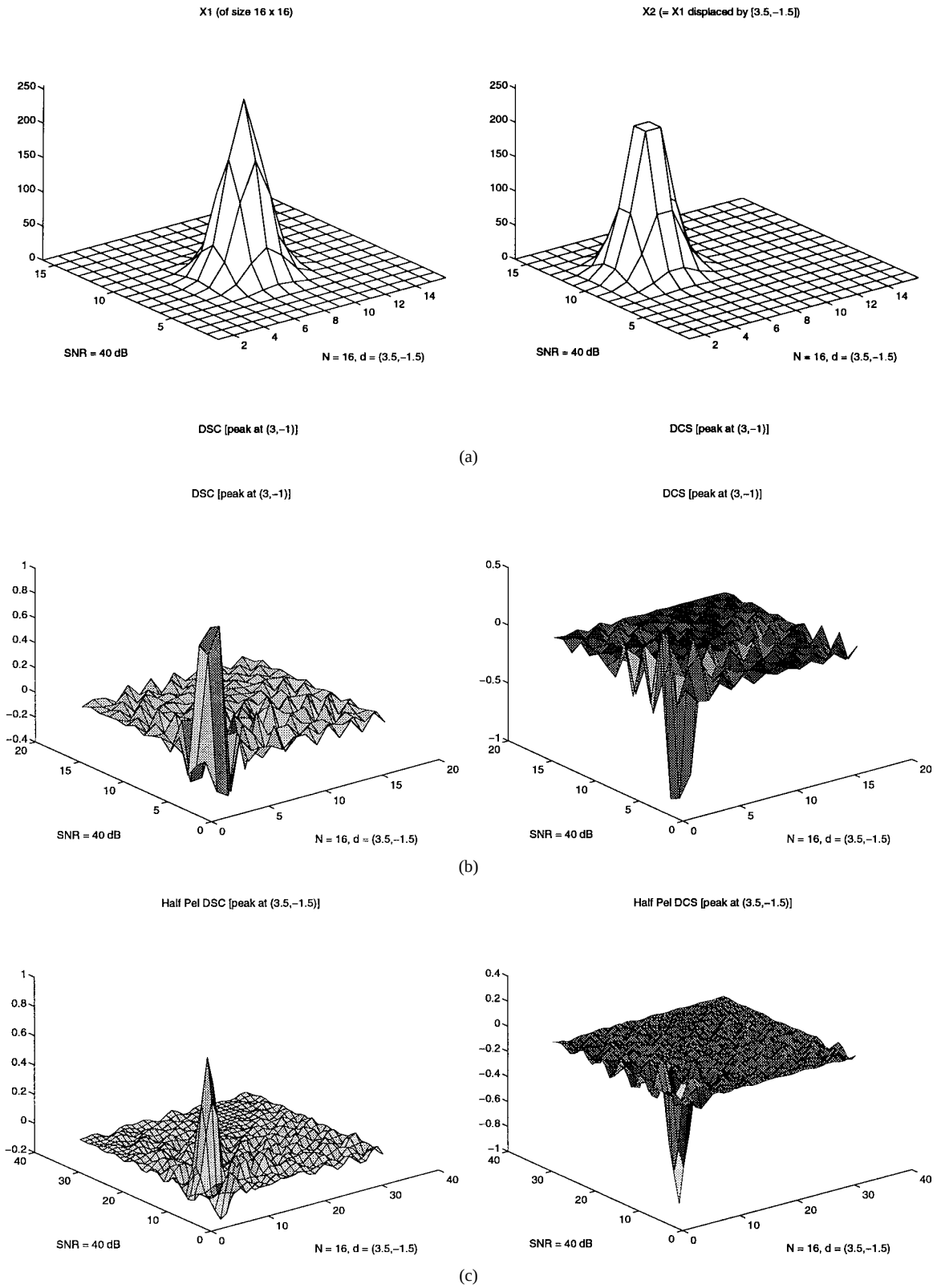


Fig. 27. CORDIC-HDXT-ME estimates a movement $(3.5, -1.5)$ with additive white Gaussian noise at SNR $= 40$ dB. (a) Inputs X1 and X2. (b) Peaks of integer-pel. (c) Peaks of half-pel.

$d_{\min} = 2t + 1$. Let e_l denote the errors introduced by the channel at positions $l = i_1, i_2, \cdots, i_\nu$. The symbols v_i of the corrupted word received from the channel can be written in terms of the code-word symbols c_i and the error symbols e_i as $v_i = c_i + e_i$ for $i = 0, 1, \cdots, (n-1)$. Then the decoding problem is to find the error values e_l and the error locations $l = i_1, i_2, \cdots, i_\nu$. The error-locator polynomial $\Lambda(x)$ is defined as $\Lambda(x) = \prod_{l=1}^{\nu}(1 - x\alpha^{i_l})$. The syndromes can be computed as $S_j = \sum_{i=0}^{n-1} \alpha^{i(b+j)} v_i$ for $j = 0, 1, \cdots, (2t-1)$ and written in polynomial form as

(a) (b) (c)

Fig. 28. Simulation on the "Miss America" sequence. (a) Frame 90. (b) Frame 91. (c) Reconstructed frame.

$S(x) = \sum_{i=0}^{2t-1} S_j x^j$. The *key equation* can then be written as

$$\Lambda(x)S(x) = \Gamma(x) \bmod x^{2t} \tag{62}$$

where $\Gamma(x)$ is the error-magnitude polynomial.

RS decoding algorithms can be classified into time- and frequency-domain approaches. Both approaches start with the computation of the syndromes. In the next step, the key equation is solved to obtain the error-locator polynomial. The key equation can primarily be solved in three different ways—Peterson's algorithm [110], Berlekamp's algorithm [111] (or Berlekamp Massey synthesis [112]) or Euclid's greatest common divisor (GCD) algorithm [113]. While Peterson's algorithm is inefficient for large t ($t > 3$), Berlekamp's algorithm and Euclid's algorithm are more efficient for larger t. Once the error-locator polynomial has been obtained, the frequency-domain method uses recursive extension to obtain the error transform. Then an inverse discrete Fourier transform is performed to obtain the error values. In the time-domain method, the error-locator and error-magnitude polynomials are used to compute the error values directly using Chien search and Forney's algorithm [114].

Blahut [115] proposed another approach that avoided the syndrome computation by transforming the Berlekamp Massey (BM) algorithm into the time domain. This approach (also called transform decoding without transforms) avoided the Fourier transform (i.e., the syndrome computation) at the beginning as well as the inverse Fourier transform at the end of the frequency-domain approach. Through this technique, the design of versatile decoders for any RS code over a specific field [116] is facilitated at the cost of increased algorithm complexity [115].

Many of the early decoder implementations used either Massey's synthesis [117] or Euclid's GCD algorithm [118], [119] to solve the key equation and then computed the errors in the frequency domain. Later designs [105], [120]–[122] use the extended Euclid GCD computation followed by time-domain computation of errors by Forney's method. The VLSI implementation of the GCD computation was based on the systolic array structure of Brent and Kung [123]. Whitaker *et al.* [105] designed a pipelined decoder for HDTV based on polynomial multiplication and division modules, while Berlekamp *et al.* [122] designed a hypersystolic array targeted at extremely high-speed decoding.

1) A Look-Ahead-Like Transformation for the Berlekamp Algorithm: We consider an RS decoder that uses the Berlekamp algorithm to solve the key equation and then calculates the errors in the time domain using Forney's method. We show how to apply an algorithm-level transformation to obtain high-speed/low-power/low-latency operation. The basic idea is to halve the number of iterations in the Berlekamp algorithm while at the same time increasing the concurrency in each iteration. The proposed transformation is closely related to the loop-unrolling transformation and the look-ahead transformation [41].

The original Berlekamp algorithm is shown below

Berlekamp algorithm: Algorithm 1
0) Initialize: $\Lambda^{(0)}(x) = 1$, $B^{(0)}(x) = 1$, $\Gamma^{(0)}(x) = 0$, $A^{(0)}(x) = x^{-1}$, $L_0 = 0$
1) **for** $= r = 1$ **to** $2t$
2) $\Delta_r = \sum_{j=0}^{L_{r-1}} \Lambda_j^{(r-1)} S_{r-1-j}$
3) **if** $\Delta_r \neq 0$ **then** $b1 = 0$ **else** $b1 = 1$
4) **if** $2L_{r-1} \leq (r-1)$ **then** $b2 = 0$ **else** $b2 = 1$
5a) $\begin{bmatrix} \Lambda^{(r)}(x) \\ B^{(r)}(x) \end{bmatrix} = M_r(x) \begin{bmatrix} \Lambda^{(r-1)}(x) \\ B^{(r-1)}(x) \end{bmatrix}$ (refer Table 8)
5b) $\begin{bmatrix} \Gamma^{(r)}(x) \\ A^{(r)}(x) \end{bmatrix} = M_r(x) \begin{bmatrix} \Gamma^{(r-1)}(x) \\ A^{(r-1)}(x) \end{bmatrix}$ (refer to Table 8)
6) $L_r = f_r(L_{r-1})$ (refer to Table 8)
7) **end for loop**
8) Output: $\Lambda(x) = \Lambda^{(2t)}(x)$, $\Gamma(x) = \Gamma^{(2t)}(x)$.

We observe that the degree of $\Lambda(x)$ increases in iteration r only when $\Delta_r \neq 0$ and $2L \leq r-1$. This implies that the degree of $\Lambda(x)$ can increase only once in any two consecutive iterations. We can prove this fact by contradiction. Let us assume that the degree of $\Lambda(x)$ in fact increases in iterations r and $r+1$. This implies that $\Delta_r \neq 0$, $2L_{r-1} \leq r-1$, $L_r = r - L_{r-1}$ and $\Delta_{r+1} \neq 0$, $2L_r \leq (r+1)-1$. Using $L_r = r - L_{r-1}$ in $2L_r \leq (r+1)-1$, we get $2L_{r-1} > r-1$. Since we cannot have $2L_{r-1} > r-1$ and $2L_{r-1} \leq r-1$ at the same time, we have obtained a contradiction that completes our proof.

In addition, we note that $B(x)$ and L need to be updated (row 1 of Table 8) only when the degree of $\Lambda(x)$

Table 8 Updating $(\Lambda(x), B(x))$, $(\Gamma(x), A(x))$, and L in the Berlekamp Algorithm

Condition	$M_r(x)$	$f_r(L_{r-1})$
$\overline{b1}\ \overline{b2}$	$\begin{bmatrix} 1 & \Delta_r x \\ \Delta_r^{-1} & 0 \end{bmatrix}$	$r - L_{r-1}$
$b1 + b2$	$\begin{bmatrix} 1 & \Delta_r x \\ 0 & \mathrm{x} \end{bmatrix}$	L_{r-1}

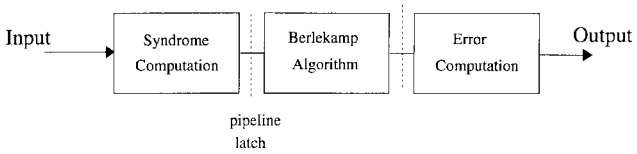


Fig. 29. Normal pipelined RS decoder.

increases. Otherwise, $B(x)$ is just shifted and L remains unchanged (row 2 of Table 8). This suggests that if we apply the idea of look-ahead to update the pair of polynomials from $(\Lambda^{(2k-2)}(x), B^{(2k-2)}(x))$ to $(\Lambda^{(2k)}(x), B^{(2k)}(x))$ without going through the intermediate computation of $(\Lambda^{(2k-1)}(x), B^{(2k-1)}(x))$, then we can halve the number of iterations. Of course, this modification will lead to a more complicated design than that of the original algorithm. In addition, we observe that the pair of polynomials $(\Gamma(x), A(x))$ are updated with the same matrix $M_r(x)$ as the pair $(\Lambda(x), B(x))$. Therefore, any transformation that is applied to the update of $(\Lambda(x), B(x))$ can also be applied to update $(\Gamma(x), A(x))$.

The modified Berlekamp algorithm is derived by considering the update of all variables over two consecutive iterations and then using the properties mentioned above to obtain simplifications (for details, see [124]). The basic idea of the derivation is similar to that used in [115] to derive the BM algorithm

Modified Berlekamp algorithm: Algorithm 2

0) Initiallize $\Lambda^{(0)}(x) = 1$, $B^{(0)}(x) = 1$, $\Gamma^{(0)}(x) = 0$, $A^{(0)}(x) = x^{-1}$, $L_0 = 0$, $\alpha_0 = S_0$
1) **for** $k = 1$ **to** t
2a) $\Delta_{2k-1} = \sum_{j=0}^{L_{k-1}} \Lambda_j^{(2k-2)} S_{2k-2-j}$
2b) $\delta_{2k} = \sum_{j=0}^{L_{k-1}} \Lambda_j^{(2k-2)} S_{2k-1-j}$
2c) $\eta_{2k} = \sum_{j=0}^{L_{k-1}} \Lambda_j^{(2k-2)} S_{2k-j}$
3) $\beta_{2k-1} = \delta_{2k} - \Delta_{2k-1}\alpha_{2k-2}$
4a) **if** $\delta_{2k} \neq 0$ **then** $b0 = 0$ **else** $b0 = 1$ **fi**
4b) **if** $\Delta_{2k-1} \neq 0$ **then** $b1 = 0$ **else** $b1 = 1$ **fi**
5) **if** $2L_{2k-2} \leq (2k-2)$ **then** $b2 = 0$ **else** $b2 = 1$ **fi**
6a) $\begin{bmatrix} \Lambda^{(2k)}(x) \\ B^{(2k)}(x) \end{bmatrix} = M_k(x) \begin{bmatrix} \Lambda^{(2k-2)}(x) \\ B^{(2k-2)}(x) \end{bmatrix}$ (refer to Table 9)
6b) $\begin{bmatrix} \Gamma^{(2k)}(x) \\ A^{(2k)}(x) \end{bmatrix} = M_k(x) \begin{bmatrix} \Gamma^{(2k-2)}(x) \\ A^{(2k-2)}(x) \end{bmatrix}$ (refer to Table 9)
7) $L_{2k} = f_k(L_{2k-2})$ (refer to Table 9)
8) $\alpha_{2k} = g_k(\alpha_{2k-2})$ (refer to Table 9)
9) **end for loop**
10) Output: $\Lambda(x) = \Lambda^{(2t)}(x)$, $\Gamma(x) = \Gamma^{(2t)}(x)$.

In the modified algorithm, the number of iterations has been reduced to t, while the complexity of each iteration has increased. It is important to note that the available concurrency in each iteration has also increased. For example, steps 2a), 2b) and 2c) can be performed in parallel.

Table 9 Updating $\Lambda(x)$, $B(x)$, and L for the BM Algorithm, Where $G(x) = 1 + \Delta_{2k-1}^{-1}\beta_{2k-1}x$ and $G1(x) = \beta_{2k-1}x^2 + \Delta_{2k-1}x$

Condition	$M_k(x)$	$f_k(L_{2k-2})$	$g_k(\alpha_{2k-2})$
$b1\ b0$	$\begin{bmatrix} 1 & 0 \\ 0 & x^2 \end{bmatrix}$	L_{2k-2}	α_{2k-2}
$b1\ \overline{b0}\ \overline{b2}$	$\begin{bmatrix} 1 & \delta_{2k}x^2 \\ \delta_{2k}^{-1} & 0 \end{bmatrix}$	$2k - L_{2k-2}$	$\delta_{2k}^{-1}\eta_{2k}$
$b1\ \overline{b0}\ b2$	$\begin{bmatrix} 1 & \delta_{2k}x^2 \\ 0 & x^2 \end{bmatrix}$	L_{2k-2}	α_{2k-2}
$\overline{b1}\ \overline{b2}$	$\begin{bmatrix} G(x) & \Delta_{2k-1}x \\ x\Delta_{2k-1}^{-1} & 0 \end{bmatrix}$	$2k - 1 - L_{2k-2}$	$\Delta_{2k-1}^{-1}\delta_{2k}$
$\overline{b1}\ b2$	$\begin{bmatrix} 1 & G1(x) \\ 0 & x^2 \end{bmatrix}$	L_{2k-2}	α_{2k-2}

2) Speeding Up the Syndrome and Error Computation: The syndrome and error computations can be transformed more easily to obtain concurrency. The syndrome computation involves the computation of $2t$ components of a discrete Fourier transform over a Galois field

$$S_j = \sum_{i=0}^{n-1} \alpha^{i(b+j)} v_i \quad \text{for } j = 0, 1, \cdots, (2t-1). \tag{63}$$

If Horner's rule is used to compute each of the $2t$ syndromes, then n iterations are required to complete the computation. We can reorganize (63) as

$$S_j = \sum_{l=0}^{\lfloor (n+1)/2 \rfloor} \alpha^{2l(b+j)} v_{2l} + \alpha^{b+j} \sum_{l=0}^{\lceil (n-1)/2 \rceil} \alpha^{2l(b+j)} v_{2l-1} \tag{64}$$

so that the number of iterations required gets halved (for details see [124]).

The error computation can be written as

$$e_i = \begin{cases} 0 & \text{if } \Lambda(\alpha^{-i}) \neq 0 \\ -\frac{\Gamma(\alpha^{-i})}{\alpha^{(b-1)i}\Lambda'(\alpha^{-i})} & \text{if } \Lambda(\alpha^{-i}) = 0 \end{cases} \tag{65}$$

where $\Lambda(x)$ is the error locator and $\Gamma(x)$ is the error evaluator polynomial [115]. If we assume that the errors are computed sequentially, then n iterations are again required. If we compute e_{2i} and e_{2i+1} in parallel, then the number of iterations can be halved at the cost of increased complexity (for details, see [124]).

To illustrate how these transformations can affect the VLSI design, we apply them to the pipelined design in Fig. 29. Each of the pipelined stages is implemented as a finite-state machine. In general, the modified design has a critical path T' that is longer than the critical path T of the normal design. Obviously, we would like to have T' as small as possible. Let us assume that the design in Fig. 29 can work at a maximum clock frequency $f = 1/T$ and the modified design in Fig. 30 at a maximum clock frequency $f' = 1/T'$. We can operate the modified design

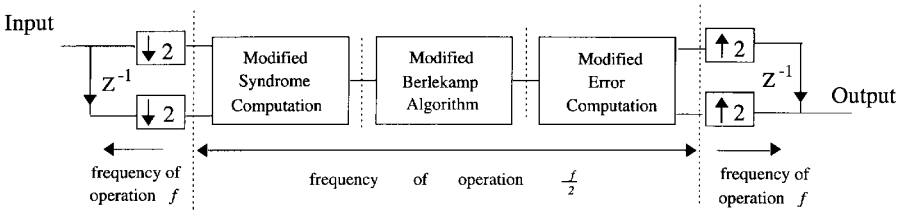


Fig. 30. Modified pipelined RS decoder.

in two modes—high speed and low power. In the high-speed mode, the throughput can be improved by a factor of $2 * f'/f$. In the low-power mode, it is desired that the modified design consumes less power while maintaining the same throughput as the normal design. The modified design can operate at a frequency f'' that is half the clock frequency of the normal design (i.e., $f'' = f/2$). The voltage of the modified design can then be scaled down from V to V' in order to achieve low-power operation. The voltage V' can be chosen [based on the delay model in (2)] so that the critical path delay increases from T' at V to $T'' = 2T$ at V'. Using (1), a simple formula can be written for the ratio of the power consumption in the modified design to the normal design

$$\frac{P_M}{P_N} = \frac{A_M}{A_N}\left(\frac{V'}{V}\right)^2 \frac{f/2}{f}. \tag{66}$$

Note that we have estimated the ratio of effective capacitances as the ratio of active areas (A) of the designs.

Based on a VLSI synthesis [124] of the normal and modified designs from a hardware description language specification in PARTHENON, the ratio A_M/A_N was found to be 2.37. Also, the maximum frequencies of operation for the modified and normal circuits were found from simulation to be $f' = 28.5$ MHz and $f = 33$ MHz, respectively. This gives us a throughput speed-up of 1.73 at 5 V. On the other hand, we can scale the voltage for the modified design from $V = 5$ V to $V' = 3.4$ V based on (2), with $V_t = 0.7$ V to obtain low-power operation. Using (66), we get $P_M = 0.55P_N$—in other words, a power savings of about 45%.

B. Viterbi Decoding

The Viterbi algorithm (VA) has been applied to a variety of decoding and estimation problems in communications and signal processing. One of the most important applications of the VA is in the maximum-likelihood (ML) decoding of convolutional and trellis codes [106]. Various VLSI implementations have targeted high-performance systems such as terrestrial digital HDTV systems [125], code division multiple access systems [126], magnetic recording systems [127], and satellite communication systems [128]. Speed, area, power consumption, or a combination of these criteria were targeted depending on the application.

The VA uses the dynamic programming technique to estimate the transitions of the convolutional encoder based on noisy observations. Given the ML paths for each of the $N = 2^\nu$ nodes at time t, the VA finds the N new optimum paths at time $t+1$. A Viterbi decoder (VD) that implements the VA can be divided into three units [106].

1) *Branch metric unit (BMU):* The BMU computes the Hamming or Euclidean distance [called the branch metric (BM)] between the received noisy symbol and the encoder output for each transition from state i to state j. The branch metric for the branch from state i to state j at time t is denoted as $b_{ij}^{[t]}$.
2) *Add-compare-select unit (ACS):* The ACS recursively computes the path metrics (PM) $P_j^{[t+1]}$ using the old path metrics at time t according to $P_j^{[t+1]} =$ maximum value of $(P_i^t + b_{ij}^{[t]})$ over all possible i. Information about the surviving paths at each state is stored in the survivor memory (SM).
3) *Survivor memory unit (SMU):* The SMU performs the task of finding the decoded symbol from the surviving sequences. It has been shown that the surviving paths merge with high probability when traced backward L $(\geq 5\nu)$ steps [106]. The SMU uses this property to find the decoded symbol after a decoding delay of L steps.

The most important parameters in the design of the VD are the constraint length ν and the rate of the code R. As ν increases, codes with larger d_{free} can be found and, therefore, the probability of error P_e can be expected to decrease dramatically [106]. The number of states $N = 2^\nu$ increases exponentially, causing the complexity of the VD also to increase exponentially. The rate R affects the complexity of the ACS computation. For an $R = k/n$ code, the ACS computation has to add, compare, and select among 2^k paths for each of the N states. This means that the complexity of the ACS unit grows exponentially with k, the size of the input alphabet.

Viterbi decoders can be classified according to the number of ACS units as follows: *state serial decoders* (in which a single ACS unit is used to perform the ACS computation for all states), *fully parallel decoders* (in which each state has an ACS unit assigned to it), and *intermediate decoders* (in which several states share the same ACS unit). In addition, there are cascaded pipelined structures and parallel pipelined cascades, in which each ACS unit is responsible for one of ν stages [129], [130].

The SM can be implemented in two possible ways [106], [131], viz, *exchange register* (ER) and *traceback*. In the ER scheme, the actual symbols of the surviving path are maintained for each state and updated at each time instant.

This implies a great deal of exchange between the registers corresponding to different states. This is a disadvantage in the VLSI implementation of VD's for large constraint length codes because of the large chip area and power required. In the traceback scheme, pointers to the previous state of the surviving path (instead of the actual symbols corresponding to the transition) are maintained. Exchange of data across registers is not required. The simplest approach uses a single read pointer to trace back the surviving path [132]. The sequential process of traceback necessitates the use of fast read operations. The k-pointer algorithm [129], [133] traces back k survivor paths concurrently. This algorithm reduces memory requirements as well as alleviates the requirement of read–write pointers moving at different speeds. Hybrids between ER and traceback, such as pretraceback and trace forward, have been proposed in [131] and utilized in [134] and [135].

1) Design of the ACS Unit: Various area-efficient techniques have been proposed in the literature to implement *intermediate decoders* [129], [130], [134], [136]–[141]. These techniques attempt to reduce area by mapping a group of states onto a single ACS unit. Area-efficient techniques are useful in VD's for large constraint length codes [140], [141]. This is because of the high complexity of *parallel state decoders* for such codes. Area-efficient techniques can also be used to minimize decoder complexity [126] in moderate throughput applications.

The properties of convolutional codes generated by a feed-forward shift register encoder are used to obtain an appropriate grouping of the states. The computations for each group of states are performed by a single ACS unit. The grouping is chosen to minimize the communication requirements between ACS units (to transfer path metrics) as well as between ACS units and the BMU (to transfer branch metrics). One of the important aspects of mapping more than one state to an ACS is that internal parallelism is created. This parallelism allows us to pipeline the ACS units [138], [139].

Other complexity-reduction techniques in VD's include the scarce-state transition (SST) VD [142], [143] and the use of punctured codes [144]. In the SST scheme proposed in [142], the received data are preprocessed so that the zero state becomes more probable when compared to other states in the absence of channel errors. This fact can be used to simplify the VD and thus obtain low-power operation. In [144], high-rate punctured codes are constructed by deleting bits periodically from rate $1/n$ codes. This permits us to use the simple VD's of rate $1/n$ codes to decode higher rate punctured codes.

For high-throughput applications, fully parallel decoders need to be used. Speed-up techniques need to be applied when throughput requirements cannot be directly satisfied by parallel-state decoders. The ACS unit essentially performs a recursive operation that is not easy to speed up. On the other hand, the BMU and the SMU can be sped up by using various techniques for feed-forward computations, such as parallelism and pipelining [41]. The throughput of the decoder is therefore constrained by the speed of the ACS loop. Fettweis and Meyr [49], [145], [146] (and, independently, Thapar and Cioffi [51] and Lin and Messerschmitt [50]) proposed the application of the look-ahead transformation [39] to break the ACS loop.

For the sake of our discussion, let us consider a parallel-state VD for a rate $R = 1/n$ decoder. The M-step look-ahead collapses M consecutive 1-step trellis (normal trellis) stages into a single M-step trellis stage with equivalent branch metrics b'_{ij}. The computation of b'_{ij} for any transition from state i to j can be computed in parallel by N normal one-step VD's on rooted trellises [49]. The M-step ACS units now have to add, compare, and select among 2^M different survivor paths (i.e., the ACS implementation complexity grows exponentially with M). The M-step ACS units can work M times slower while maintaining the same overall throughput. Alternately, if the critical path of the M-step trellis is shorter than M times the critical path of the one-step trellis, then a speed-up can be obtained.

We will illustrate how this look-ahead transformation can be used to obtain low-power operation. We apply a two-step look-ahead to the trellis of a rate $1/n$ code. Each one-step ACS unit compares and selects one of the two possible paths, whereas the two-step ACS unit compares and selects one out of four possible surviving paths (see Figs. 31 and 32). The complexity of a one-step ACS unit is two adders, one comparator, and one multiplexer, whereas the complexity of a two-step ACS unit is four adders, three comparators, and three multiplexers. Let us define C_1 and C_2 as the effective capacitances of the one-step and two-step ACS units, respectively. We neglect the complexity of the multiplexers and consider the complexity of a comparator to be the same as an adder (in reality, it is slightly less than an adder) in our estimates. We can write the ratio of the power consumption of the two-step ACS unit to that of the one-step ACS unit as

$$\frac{P_2}{P_1} = \frac{C_2}{C_1}\left(\frac{V_{\mathrm{dd2}}}{V_{\mathrm{dd1}}}\right)^2 \frac{f/2}{f}. \tag{67}$$

We approximate the ratio C_2/C_1 as the ratio of the adder complexities of the two ACS units, so that $C_2 = 7C_1/3$. If we assume a parallel-state decoder and that the ACS unit is not pipelined, then we can write the critical path of the two-step ACS in terms of the one-step ACS $T_2 = \frac{3T_1}{2}$. To obtain low-power operation, the supply voltage of the two-step ACS can be scaled from $V_{\mathrm{dd1}} = 5$ V to $V_{\mathrm{dd2}} = 4.05$ V. The value for V_{dd2} is chosen based on the delay model in (2), with $V_t = 0.7$ V, so that the critical path of the two-step ACS becomes $T'_2 = 2T_1$ at V_{dd2}. Using these values in (67), we get $P_2 = 0.77P_1$—in other words, a power savings of 23%. If we further assume that we have external parallelism in the form of interleaved independent streams [50], [138], then pipelining can be used in the ACS unit so that $T_2 = T_1$. Based on (2) and (67), we obtain $V_{\mathrm{dd2}} = 3.08$ V and $P_2 = 0.44P_1$—in other words, a power savings of 56%. Note that in these estimates, we have considered the ACS unit only.

V. Adaptive Filtering

Adaptive filtering is a major signal-processing component in a variety of communications systems. Adaptive filtering is used in channel equalization to compensate for the distortion introduced by the channel as well as to suppress noise and interference from other users. For example, in xDSL systems, adaptive filtering is employed in equalization/cancellation schemes to combat intersymbol interference, echoes, and cross talk [147]–[150]. In wireless systems, adaptive equalization is used to combat time dispersion caused by multipath propagation [107], [151].

Linear adaptive filtering structures can be of two types, viz, transversal and lattice. In a transversal filter, the taps of the FIR filter are updated. On the other hand, in a lattice filter, the reflection coefficients are updated. Linear filtering structures do not perform well when they are used to equalize channels that have spectral nulls in their frequency response or have severe amplitude distortion [152]. One of the commonly used nonlinear filter structures is the *decision feedback equalizer* (DFE). The DFE consists of a *feed-forward equalizer* (FFE) that performs FIR filtering on the input data and a *feed-back equalizer* (FBE) that attempts to cancel the effect of the past symbols (postcursor intersymbol interference) on the present symbols. The most common adaptation algorithms are the LMS and the RLS algorithms [153]. Various VLSI implementations of adaptive filters can be found in [154]–[159].

One common complexity reduction technique for long adaptive filters is to perform adaptive filtering in blocks. The block adaptive filters can then be transformed into the frequency domain [160]. The adaptive filter computations can be performed efficiently in the frequency domain using the FFT. In fact, equalization and echo cancellation in the discrete multitone (DMT)-asymmetrical (A)DSL standard are performed in the frequency domain [149], [150]. The cyclic prefix in DMT-ADSL enables simple equalization and echo cancellation in the frequency domain.

The main problem in the high-speed VLSI implementation of adaptive LMS transversal filters is the existence of nonlinear feedback loops, in particular, the weight-update loop and the error-feedback loop [42]. The weight-update loop arises from the fact that the adaptive filter weights need to be updated before the next filter output can be computed. The error-feedback loop occurs because the error must be computed before the weights are adapted and the next filter output computed. The error itself is computed using the output of the filter and the decision device. The speed of DFE's is also constrained by similar recursive loops [42].

The lattice filters (whether they use gradient or RLS adaptation algorithms) have localized recursive loops. The reflection coefficients of a particular stage of the lattice are updated using the prediction error outputs of the previous stage of the lattice at the previous instant [153]. The throughput is again constrained by the feedback loop.

The transversal LMS algorithm can be transformed into the delayed LMS algorithm [52] by using delayed versions of the error and data in the weight vector update. By retiming [161] these delays, we can pipeline the FIR filter (and therefore the error feedback loop). This approach of using delayed versions of the error was generalized into the delay-relaxation technique that can also be applied to the DFE. Note that this pipelining speed-up is obtained at the cost of a small loss in tracking performance [52], [162] as well as increased complexity due to the pipelining latches. The M-step look-ahead transformation [41] has been applied to the weight-update LMS recursion. Look-ahead introduces M delays that can then be retimed to pipeline the feedback loop. The complexity of the look-ahead recursion increases linearly with M. This has been addressed in [42] by applying sum relaxations. The M-step look-ahead transformation has also been applied to the lattice filter reflection coefficient recursion in [42] and [53] to obtain M-fold speed-up.

Once the look-ahead transformation has been applied to the recursion to facilitate pipelining, the speed-up obtained can be traded off for low-power operation by voltage scaling. In fact, Goel and Shanbhag [163] have applied the relaxed look-ahead technique in combination with the strength reduction transformation [41] to obtain a pipelined low-power architecture for the cross-coupled LMS QAM equalizer. The authors exploit the properties of the cross-coupled equalizer at the algorithm level to obtain complexity reduction by strength reduction.

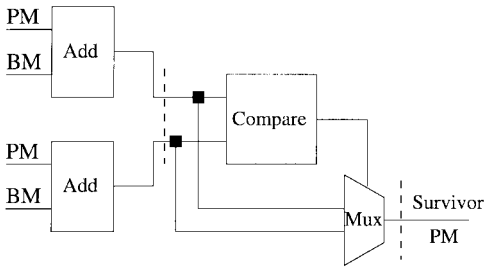


Fig. 31. Normal one-step ACS unit.

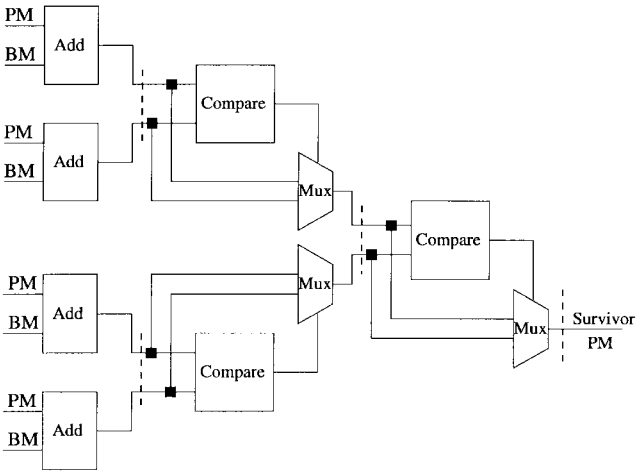


Fig. 32. Two-step look-ahead ACS unit.

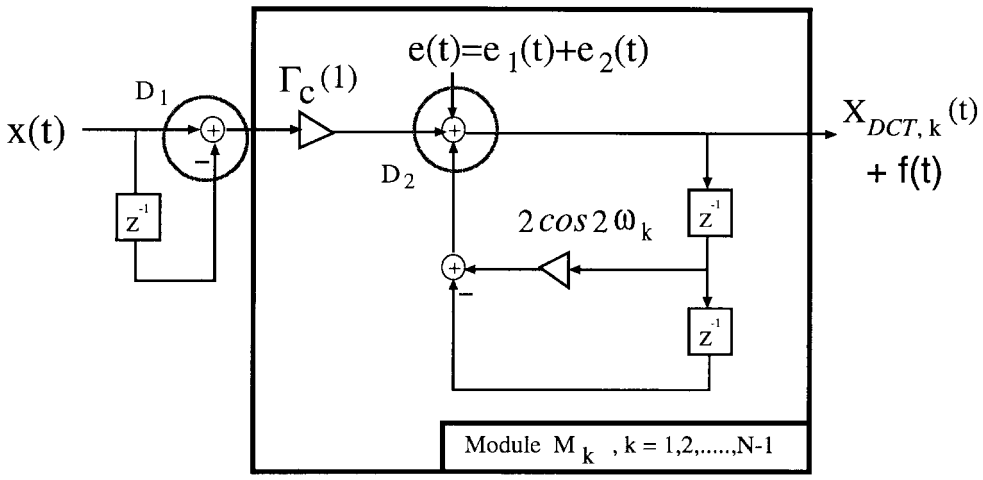


Fig. 33. IIR DCT architecture under fixed-point implementation.

VI. Finite-Precision Analysis of the Multirate Approach

In this section, we will show that if designed carefully, the finite-length numerical property of the multirate approach can be very good. We use the finite-precision analysis of multirate DCT architectures as an example. We start with the DCT architecture under normal operation, then extend the analysis to the low-power design with $M = 2$. The general results for arbitrary M are also presented. Throughout the derivations, the "statistical error model" for fixed-point analysis is used [38, ch. 6].

A. Analysis of the IIR DCT Using Direct Form I Structure

When the IIR DCT architecture of Fig. 14 is implemented in fixed-point operations, we need to perform suitable scaling operations to avoid overflow. Also, rounding errors occur due to the fixed-point operators. The resulting fixed-point IIR structure to compute the kth DCT coefficient is shown in Fig. 33.

1) Rounding Errors: Using the statistical error model, the rounding error of the IIR DCT structure can be modeled as

$$e(t) = e_1(t) + e_2(t) \tag{68}$$

where $e_i(t)$ is the rounding error caused by the ith multiplier in the structure and the mean and variance of the rounding error are given by [38, ch. 6]

$$m_R = E\{e_i(t)\} = 0, \quad \sigma_R^2 = E\{e_i^2(t)\} = \frac{2^{-2B}}{12} \tag{69}$$

respectively. Due to the presence of $e(t)$, the actual output of the DCT module after N iterations can be represented as

$$\hat{X}_{\mathrm{DCT},k}(t) = X_{\mathrm{DCT},k}(t) + f(t) \tag{70}$$

where $f(t)$ is the output error contributed by $e(t)$.

Let $H_{ef}(z)$ denote the transfer function of the system from the node at which $e(t)$ is injected to the output and $h_{ef}(n)$ be the corresponding unit-sample response. From Fig. 33, $H_{ef}(z)$ is given by

$$H_{ef}(z) = \frac{1}{1 - 2\cos 2\omega_k z^{-1} + z^{-2}} \tag{71}$$

and $h_{ef}(n)$ can be derived as

$$h_{ef}(n) = \frac{1}{\sin(2\omega_k)} \sin[(n+1)2\omega_k] u(n) \tag{72}$$

where $u(n)$ denotes the step function. Since only N iterations are performed in the IIR circuit, the mean and variance of $f(t)$ of the kth DCT channel can be computed as

$$\begin{aligned} m_f &= m_e \sum_{n=0}^{N-1} h_{ef}(n) \\ &= m_e \sum_{n=0}^{N-1} \frac{1}{\sin(2\omega_k)} \sin[(n+1)2\omega_k] \end{aligned} \tag{73}$$

$$\begin{aligned} \sigma_f^2 &= \sigma_e^2 \sum_{n=0}^{N-1} |h_{ef}(n)|^2 \\ &= \frac{\sigma_e^2}{\sin^2(2\omega_k)} \sum_{n=0}^{N-1} \sin^2[(n+1)2\omega_k] \\ &= \frac{\sigma_e^2}{\sin^2(2\omega_k)} \left(\frac{N}{2}\right) \end{aligned} \tag{74}$$

where $m_e = E\{e(t)\} = 0$ and $\sigma_e^2 = E\{e^2(t)\} = E\{e_1^2(t)\} + E\{e_2^2(t)\} = (1 + N_s(k))\sigma_R^2$; $N_s(k)$ is the number of the noise sources contributed by the multiplier $M_2 = 2\cos(2\omega_k)$ in the IIR loop

$$N_s(k) = \begin{cases} 4, & \text{if } |2\cos(2\omega_k)| > 1, \\ 1, & \text{if } |2\cos(2\omega_k)| < 1, \\ 0, & \text{if } |2\cos(2\omega_k)| = 1. \end{cases} \tag{75}$$

Now, based on (73)–(75), we can represent the total noise power at the kth DCT channel as

$$P_f = m_f^2 + \sigma_f^2 = \frac{N(N_s(k)+1)}{2\sin^2(2\omega_k)} \left(\frac{2^{-2B}}{12}\right). \tag{76}$$

As we can see, given the system word length B, the rounding error grows linearly with the block size N. This

Table 10 Optimal Word-Length Assignment Under the Constraint SNR = 40 dB, Where B_A Is the Average Word Length. (a) Normal IIR DCT. (b) Low-Power DCT with $M = 2$

DCT channel	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	B_A
B_k ($N = 8$)	12	11	10	9	10	11	12	N/A								10.7
B_k ($N = 16$)	13	12	12	11	11	10	10	10	10	10	11	11	12	12	13	11.2

(a)

DCT channel ($M = 2$)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	B_A
B_k ($N = 8$)	10	9	10	11	10	9	10	N/A								9.9
B_k ($N = 16$)	12	11	10	10	10	11	12	12	12	11	10	10	10	11	12	10.9

(b)

indicates that we will have 3-dB degradation in the SNR as N doubles; however, such degradation can be compensated by adding $1/2$ (in average) bit in the word length. On the other hand, the noise power is inversely proportional to $\sin^2(2\omega_k)$. That is, the effect of the rounding error in each channel of the IIR DCT greatly depends on the pole locations of the IIR transfer function. The closer $2\omega_k$ is to zero or π, the larger the rounding error. As a result, the first and last DCT channels suffer most from the finite-word-length effect, while the middle channels have good SNR in terms of rounding error. This phenomenon is quite different from what we have seen in other DCT algorithms (e.g., [164, Fig. 7]).

2) Dynamic Range: In fixed-point arithmetic, the input sequence $x(t)$ is represented as a fraction and is bounded by $|x(t)| \leq 1$. Hence, the dynamic range of the circled nodes in Fig. 33 can be computed as

$$\begin{aligned} D_1 &= 2, \\ D_2 &= \max\{X_{DCT,k}(t)\} \\ &= C(k) \sum_{n=0}^{N-1} |\cos[(2n+1)\omega_k]| \end{aligned} \tag{77}$$

and the dynamic range of the overall architecture is given by

$$D = \max\{D_1, D_2\}. \tag{78}$$

To prevent overflow in fixed-point implementations, a suitable scaling of the input signal is usually employed according to the dynamic range of the system. In practice, the SNR of the scaled system, SNR$'$, will be degraded by the scaling process; it is given by [38, ch. 6]

$$\text{SNR}' = s^2 \, \text{SNR}_0 \tag{79}$$

where s is the scaling factor and SNR_0 is the signal-to-noise ratio of the original system. Suppose that a one-time scaling scheme is provided at the input end to avoid overflow and is done by shifting the data to the right by K bits. The scaling factor s can be represented as

$$s = 2^{-K}, \quad \text{where } K = \lceil \log_2 D \rceil. \tag{80}$$

3) Optimal Word-Length Assignment: Assume that the input sequence $x(t)$ is uniformly distributed over $(-1, 1)$ with zero mean. From (76), (79), and (80), we have

$$\text{SNR}' = s^2 \frac{E\{(X_{\text{DCT},k}(t))^2\}}{P_f} = \frac{8\sin^2(2\omega_k)}{N(N_s(k)+1)} \cdot 2^{2B-2K} \tag{81}$$

where the fact that [164]

$$E\{(X_{\text{DCT},k}(t))^2\} = E\{x^2(t)\} = 1/3, \quad k = 1, 2, \cdots, N-1 \tag{82}$$

is used. If we want to achieve a performance of 40 dB in SNR for the kth DCT component, the optimal word length B_k for that channel can be computed from (81) as

$$B_k = \left\lceil \frac{4 - \log_{10}\left[\sin^2(2\omega_k) \cdot \frac{8}{N(N_s(k)+1)}\right]}{2 \cdot \log_{10} 2} + K \right\rceil. \tag{83}$$

As an example, the B_k for the case $N = 8$ and 16 under the constraint SNR = 40 dB are listed in Table 10(a), where B_A denotes the average system word length. As we can see, $B_A = 12$ bits is sufficient to meet the accuracy criteria.

B. Analysis of the IIR DCT Using Direct Form II Structure

Given the IIR DCT transfer function in (20), we can also implement it using the direct form II structure, as shown in Fig. 34. Following the derivations in the preceding section, the rounding error and dynamic range of the direct form II structure can be derived as

$$\begin{aligned} P_f &= (N_s(k)+1)\frac{2^{-2B}}{12} \\ D &= \max\left\{ \frac{1}{|\sin(2\omega_k)|} \sum_{n=0}^{N-1} |\sin[(2n+1)\omega_k]|, 2 \right\}. \end{aligned} \tag{84}$$

In contrast to the direct form I structure, the dynamic range of the direct form II structure is affected by the factor $\frac{1}{|\sin(2\omega_k)|}$ in D_1; that is, we will have nonuniform dynamic ranges for different DCT channels. This feature is not desirable in real implementations, even though the SNR results of both structures are comparative to each other (see simulation results in Section VI-D). It not only

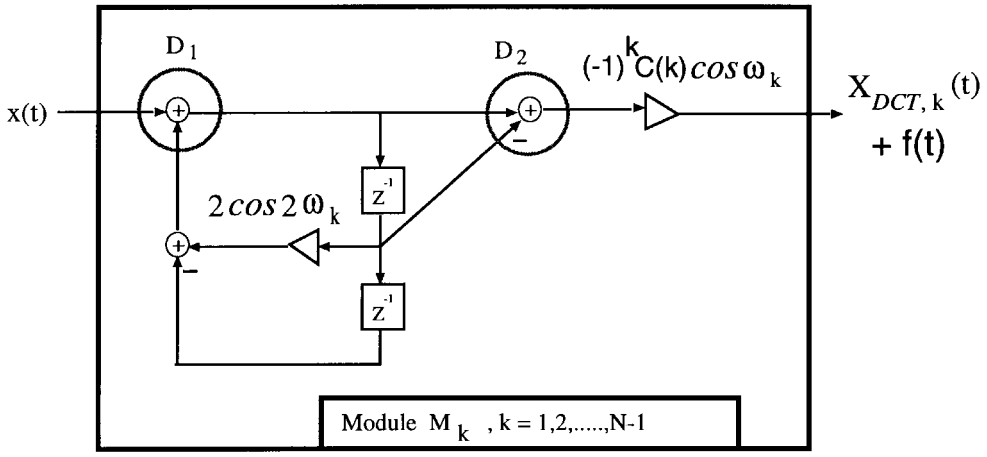


Fig. 34. IIR DCT using the direct form II structure.

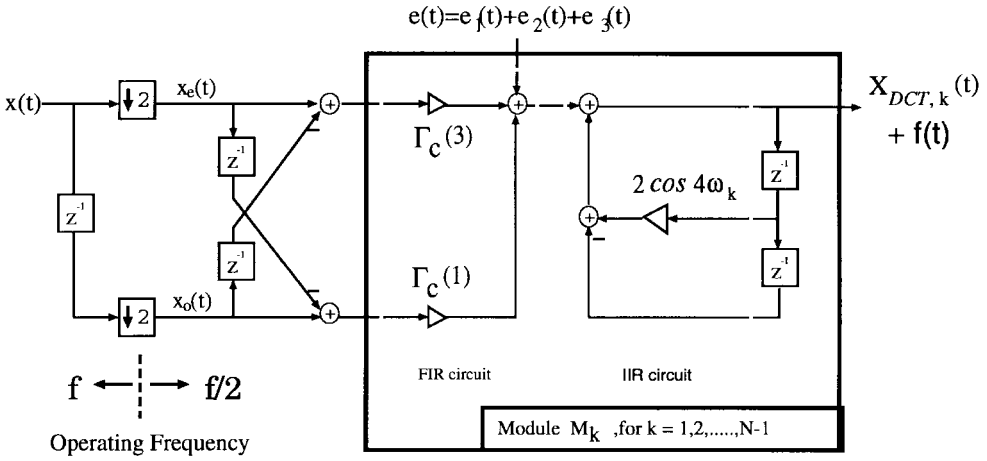


Fig. 35. Low-power IIR DCT architecture with $M = 2$.

requires a different scaling scheme in each DCT channel but also makes the data interface between VLSI modules complicated.

C. Analysis for the Low-Power IIR DCT with $M = 2$

Consider the multirate IIR DCT architecture with $M = 2$ in Fig. 35. In the fixed-point implementation of the multirate DCT architecture, the injected rounding error can be modeled as

$$e(t) = e_1(t) + e_2(t) + e_3(t) \tag{85}$$

where $e_i(t)$, $i = 1,2,3$ denotes the rounding errors contributed by the three multipliers in Fig. 35, and its power is given by

$$\sigma_e^2 = E\{e^2(t)\} = (2 + N_s(k))\sigma_R^2. \tag{86}$$

Note that

$$H_{ef}(z) = \frac{1}{1 - 2\cos 4\omega_k z^{-1} + z^{-2}} \tag{87}$$

and the total iteration is reduced to $N/2$. Thus, the total power of the rounding error at the output becomes

$$\sigma_f^2 = \sigma_e^2 \sum_{n=0}^{N/2-1} |h_{ef}(n)|^2 = \frac{\sigma_e^2}{\sin^2(4\omega_k)}\left(\frac{N}{4}\right) = \frac{(2 + N_s(k))N\sigma_R^2}{4\sin^2(4\omega_k)}. \tag{88}$$

From (88), we observe the following.

1) Although the total number of noise sources increases, the total noise power is compensated by the halved number of iterations.
2) Compared with the factor $1/(\sin(2\omega_k)^2)$ in (76), the factor $1/(\sin(4\omega_k)^2)$ in (88) will have a similar effect on the SNR of each DCT channel but with halved period.

Next, let us consider the dynamic range of the low-power DCT structure with $M = 2$. We apply the technique of "superimposition" to analyze the dynamic range of the system. Namely, we first set $x_o(t)$ to zero while analyzing the dynamic range contributed by $x_e(t)$. Then we perform the same analysis for $x_o(t)$ by setting $x_e(t)$ to zero. The

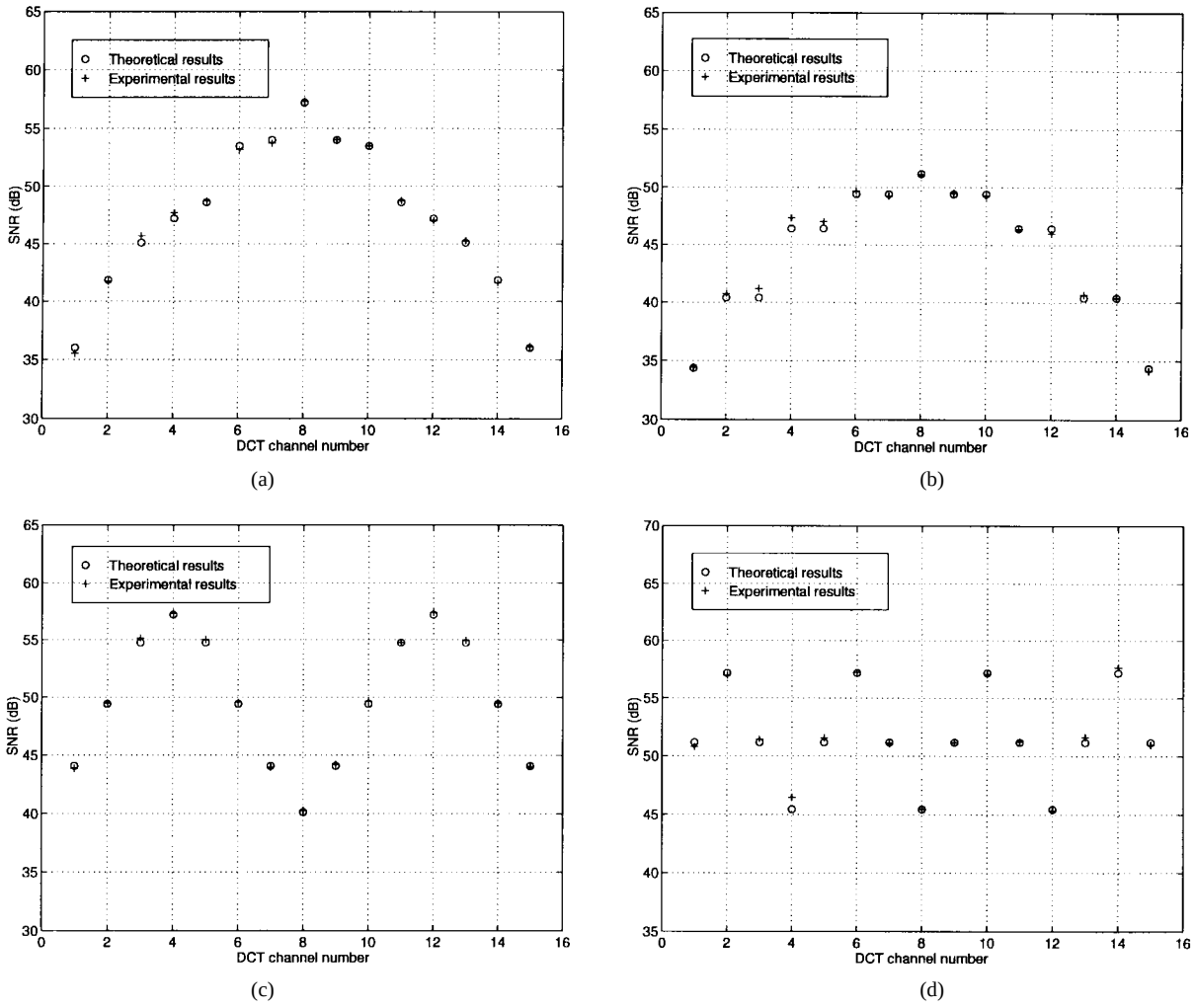


Fig. 36. Average SNR as a function of DCT channel number under fixed-point arithmetic $(N = 16, B = 12)$. (a) Normal IIR DCT using direct form I structure. (b) Normal IIR DCT using direct form II structure. (c) Low-power DCT with $M = 2$. (d) Low-power DCT with $M = 4$.

overall D can be found from the summation of the two dynamic ranges, which is given by

$$\begin{aligned} D_1 &= 2C(k)(|\cos\omega_k| + |\cos 3\omega_k|) \\ D_2 &= C(k)\sum_{n=0}^{N/2-1}(|\cos[(4n+1)\omega_k]| \\ &\quad + |\cos[(4n+3)\omega_k]|) \\ D &= \max\{D_1, D_2\}. \end{aligned} \tag{89}$$

Using the analytical results in (88) and (89), we can also find the optimal word lengths for $N = 8$ and 16 under the 40-dB SNR constraint. The results are listed in Table 10(b). It is interesting to note that the average word lengths of the multirate DCT architectures are even less than those of the normal DCT architectures. This is due to the fact that the number of the iterations in the IIR loop will be reduced to N/M. As M increases, the accumulation of the rounding errors becomes smaller and thus, less word length can be allocated. This indicates that the multirate DCT architecture not only can reduce low-power consumption but its numerical properties also become better as M increases.

The above analyses can be extended to low-power DCT designs with decimation factor equal to M $(M \geq 2, M \in 2^{+Z})$. The results are given by

$$\begin{aligned} P_f &= (M + N_s(k))\left(\frac{N}{2^{m+1}}\right)\frac{\sigma_R^2}{\sin^2(2^{m+1}\omega_k)} \\ D_1 &= M \cdot C(k)\sum_{n=0}^{M-1}|\cos[(2n+1)\omega_k]| \\ D_2 &= C(k)\sum_{n=0}^{\frac{N}{M}-1}\sum_{i=0}^{M-1}|\cos[(2^{m+1}n+2i+1)\omega_k]| \\ D &= \max\{D_1, D_2\} \end{aligned} \tag{90}$$

where $m = \log_2 M$.

D. Simulation Results

To verify our analytical results, computer simulations are carried out by using the aforementioned DCT architectures. Fig. 36 shows the average SNR as a function of the DCT channel number k. As we can see, there is a close agreement between the theoretical and experimental results. Basically,

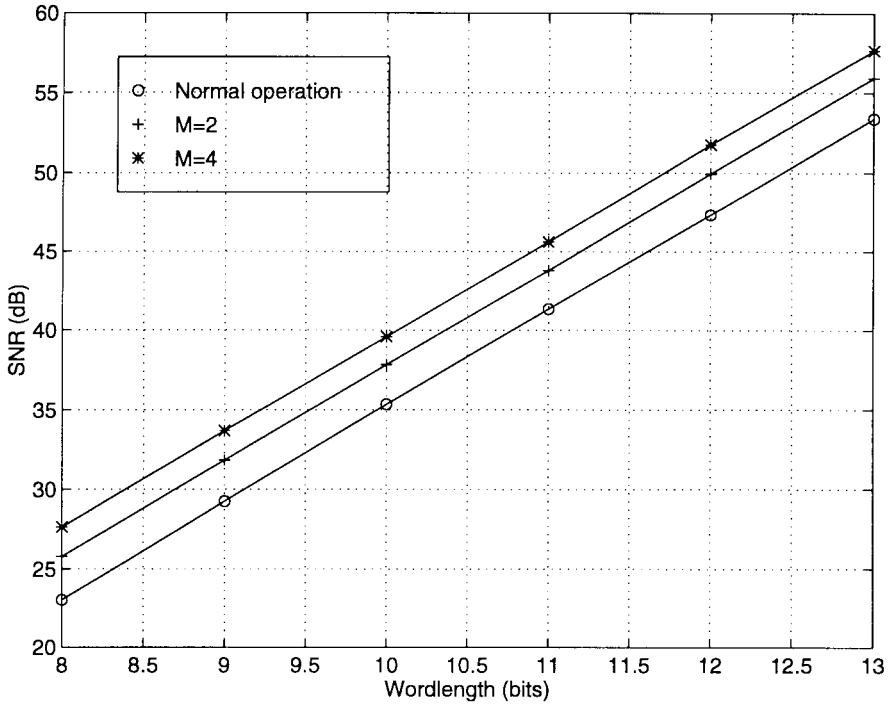


Fig. 37. Average SNR as a function of word length under fixed-point arithmetic ($N = 16$). The multirate low-power architectures have better SNR as M increases.

the SNR distribution is affected by the factor $\sin^2(2^{m+1}\omega_k)$ in (90) so that its period varies with the decimation factor M. It should be noted that although Fig. 36(a) and (b) yield similar SNR results, the uniform dynamic range of the direct form I structure makes it a better choice for VLSI implementations.

Fig. 37 shows the relationship between the average SNR and the word length for three IIR DCT architectures (normal, $M = 2$, and $M = 4$) with $N = 16$. Compared to the simulation results in [164], these IIR DCT architectures give comparative SNR performance to the DCT architectures by Hou [61] and Lee [62] under fixed-point arithmetic. It is worth noting that the multirate DCT architectures have better SNR results than the normal IIR DCT architectures under the same word length; That is, the multirate DCT has better numerical properties under fixed-point arithmetic, which is consistent with what we have seen in Table 10.

VII. A Reconfigurable Multirate DSP Computing Engine

In this section, we present a unified approach to integrate the rotation-based FIR/IIR structure, QMF lattice structure [165], DT architecture [60], [70], and adaptive RLS lattice filter [166] into one reconfigurable parallel processing architecture. It can serve as a computing engine in the audio/video system to perform those front-end computationally intensive DSP functions in multimedia applications. We first extract the common inherent features of each function, then design a programmable rotation-based computational module that can serve as a basic PE in all desired functions. The overall system of the proposed DSP computing engine consists of an array of identical programmable computational modules and one reconfigurable interconnection network. Each computational module can be adaptively changed to the basic PE in each desired DSP function by setting suitable parameters and switches. Next, according to the data paths, the interconnection network is configured to connect/combine the appropriate module outputs to perform the programmed function. When the system is in the execution mode, all PE's are operating in parallel, and the output data can be collected by the host processor in a fully pipelined way. Since the properties of each programmed function such as parallelism and pipelinability have been fully exploited, the maximum processing speed of the proposed design can be as fast as dedicated ASIC designs. Our approach is similar to the CORDIC-based FIR, IIR, and DCT architectures in the literature [48], [167], [168], but the functionality is much more general purpose. It can be classified as the *algorithm-driven* solution [169] since the programmability and ASIC-like processing speed of our design are achieved by fully exploiting the inherent properties of the given DSP algorithms. Besides, the proposed architecture is very suitable for VLSI implementation due to its modularity and regularity.

Next, we will show how to improve the speed performance of the system by using the multirate approach. Thus, we can speed up the system at the algorithmic/architectural level without using expensive high-speed circuits nor advanced VLSI technologies. We will show that we can map the multirate FIR/IIR/DT operations onto our design. As a result, we can double the speed performance of the DSP computing engine on the fly by simply reconfiguring the programmable modules and the interconnection network.

As discussed before, such a speed-up can be used for compensation of low-power operation. In what follows, we will present the design approach as well as the basic operations of the DSP computing engine. For details of this DSP engine design, the readers may refer to [170].

A. Basic Computational Module in the FIR/QMF/IIR/DT

1) Basic Module in FIR and QMF: The FIR filter is widely used in DSP applications. In addition to the MAC implementation of the filtering operation, an alternate realization of the FIR filter is the lattice structure [171, ch. 10]. It consists of N basic lattice sections that are connected in a cascade form. Each lattice module has one rotation-based circuit plus two scaling multipliers. In general, the rotation circuit can be implemented by the CORDIC processor in *hyperbolic mode* [48].

The QMF plays a key role in image compression and subband coding. Recently, the two-channel paraunitary QMF lattice was proposed [165]. It has been shown that every two-channel (real-coefficient, FIR) paraunitary QMF bank can be represented using the QMF lattice. The QMF lattice is very similar to the FIR lattice except that the inputs of the lattice become the decimated sequences of the input signal. Besides, the two scaling multipliers are set to one and the CORDIC processor works in the *circular mode* in the QMF lattice.

2) Basic Module in IIR: Next, we want to consider the basic module for IIR filtering. Due to the opposite data flow and the irregularity of the autoregressive moving average IIR lattice structure (see [171, ch. 10]), it is difficult to incorporate the traditional IIR lattice module into our unified design. Therefore, we are motivated to find an IIR lattice structure that has similar data paths as in the FIR/QMF lattice while retaining the property of modularity.

Fig. 38 shows the lattice structure that can be used to realize a second-order IIR filter [170]. The transfer functions of the two outputs are given by

$$\tilde{H}_0(z) = \frac{Y_0(z)}{X(z)} = \frac{r(k_0\cos\theta + k_1\sin\theta) - r^2 k_0 z^{-1}}{1 - 2r\cos\theta z^{-1} + r^2 z^{-2}} \quad (91)$$

$$\tilde{H}_1(z) = \frac{Y_1(z)}{X(z)} = \frac{r(k_1\cos\theta - k_0\sin\theta) - r^2 k_1 z^{-1}}{1 - 2r\cos\theta z^{-1} + r^2 z^{-2}}. \quad (92)$$

Now, given an even-order real-coefficient IIR (ARMA) filter $H(z)$, we can first rewrite it in cascade form

$$H(z) = \prod_{i=0}^{N/2-1} H_i(z) \quad (93)$$

and each subfilter $H_i(z)$ is of the form

$$\begin{aligned} H_i(z) &= \frac{c_i + d_i z^{-1} + e_i z^{-2}}{1 + a_i z^{-1} + b_i z^{-2}} \\ &= c_i + z^{-1}\frac{d_i' + c_i' z^{-1}}{1 + a_i z^{-1} + b_i z^{-2}} \\ &= c_i + z^{-1} A_i(z) \end{aligned} \quad (94)$$

where $d_i' = d_i - a_i c_i$ and $c_i' = e_i - b_i c_i$. Comparing (91) and (92) with (94), we see that $A_i(z)$ can be realized by either $\tilde{H}_0(z)$ or $\tilde{H}_1(z)$, with appropriate settings of k_0, k_1, r, and θ.

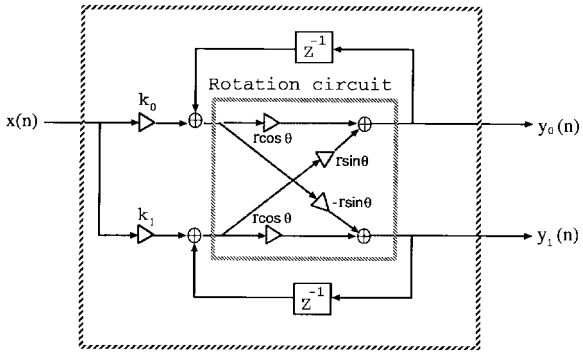


Fig. 38. Second-order IIR lattice architecture.

Now, based on (93) and (94), we can realize $H(z)$ using the structure depicted in Fig. 39. Each stage performs the filtering of $H_i(z)$ for $i = 0, 1, \cdots, N/2-1$, where $A_i(z)$ is realized by the second-order IIR module in Fig. 38. Note that the scaling multiplier c_i is also realized by the IIR module by setting $k_0 = c_i$, $k_1 = 0$, $r = 1$, $\theta = 0$ and disconnecting the feedback data paths. We use a dashed box to symbolize this implementation.

3) Basic Module in Discrete Transforms: In Section III-B, we have presented an IIR-based unified transform-coding architecture that is capable of performing most of the discrete transforms. However, the IIR-based module used in Fig. 17(a) is not applicable to the programmable architecture proposed here. To incorporate the unified DT operations into our design, we need a rotation-based computational module instead of the IIR-based one in Fig. 17(a) as the processing element.

In [60] and [70], a rotation-based module was derived for the dual generation of $X_{C,k}(t)$ and $X_{S,k}(t)$ in (30) and (31) (see Fig. 40)

$$\mathbf{f}_k = \begin{bmatrix} f_{0,k} \\ f_{1,k} \end{bmatrix} = \begin{bmatrix} \beta\cos((2L+1)\omega_k + \eta_k) \\ \beta\sin((2L+1)\omega_k + \eta_k) \end{bmatrix}$$

$$\mathbf{R}_k = \begin{bmatrix} \cos(2\omega_k) & \sin(2\omega_k) \\ -\sin(2\omega_k) & \cos(2\omega_k) \end{bmatrix}. \quad (95)$$

The rotation-based module works in an SIPO way as the IIR-based module in Fig. 17(a). The operation of the unified rotation-based DT architecture is the same as the unified architecture in Fig. 17(b) except that the IIR-based modules are replaced with the rotation-based modules.

4) Unified Module Design: From the basic computational modules discussed above, we observe that all the architectures share a common computational module with only some minor differences in the data paths, the module parameters (multiplier coefficients and rotation angle), and the way the modules are connected. With this observation in mind, we first integrate those basic computational modules into one unified programmable module, as shown in Fig. 41. It consists of six switches, four scaling multipliers, and one rotation circuit. One pipelining stage (the dashed line in Fig. 41) is inserted after the scaling multipliers $f_{0,i}$ and $f_{1,i}$, respectively, to shorten the critical path of the

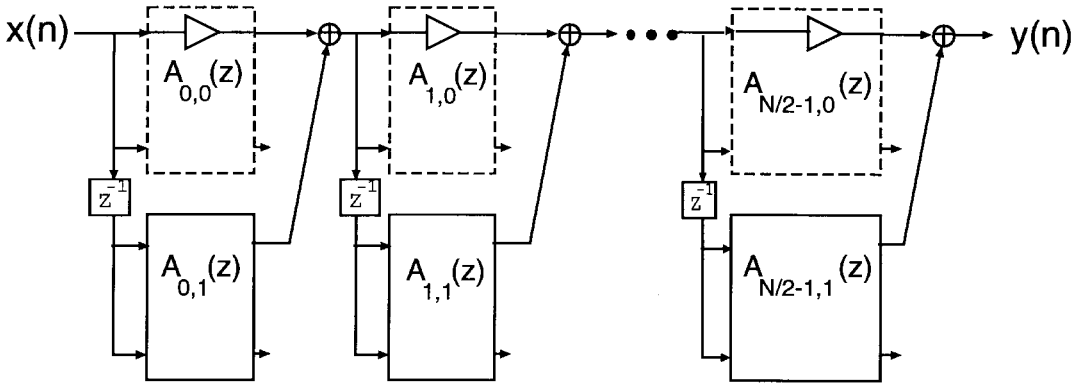


Fig. 39. IIR (ARMA) structure based on the second-order IIR lattice module.

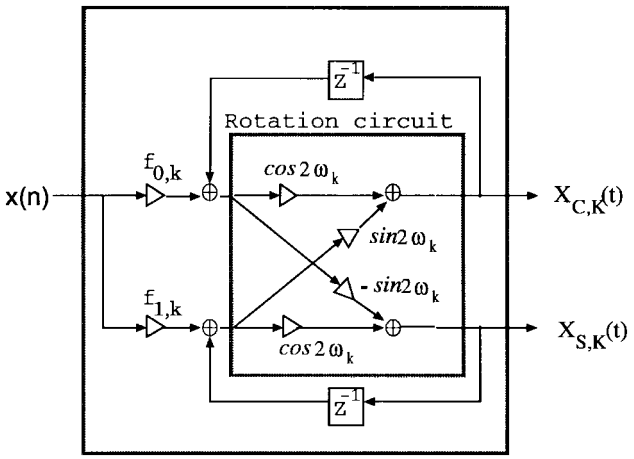


Fig. 40. Rotation-based module for the dual generation of $X_{C,k}(t)$ and $X_{S,k}(t)$.

programmable module. By setting the switches, multiplier coefficients, and rotating angle, the unified programmable module can be programmed to act as the basic PE in FIR/QMF/IIR/DT.

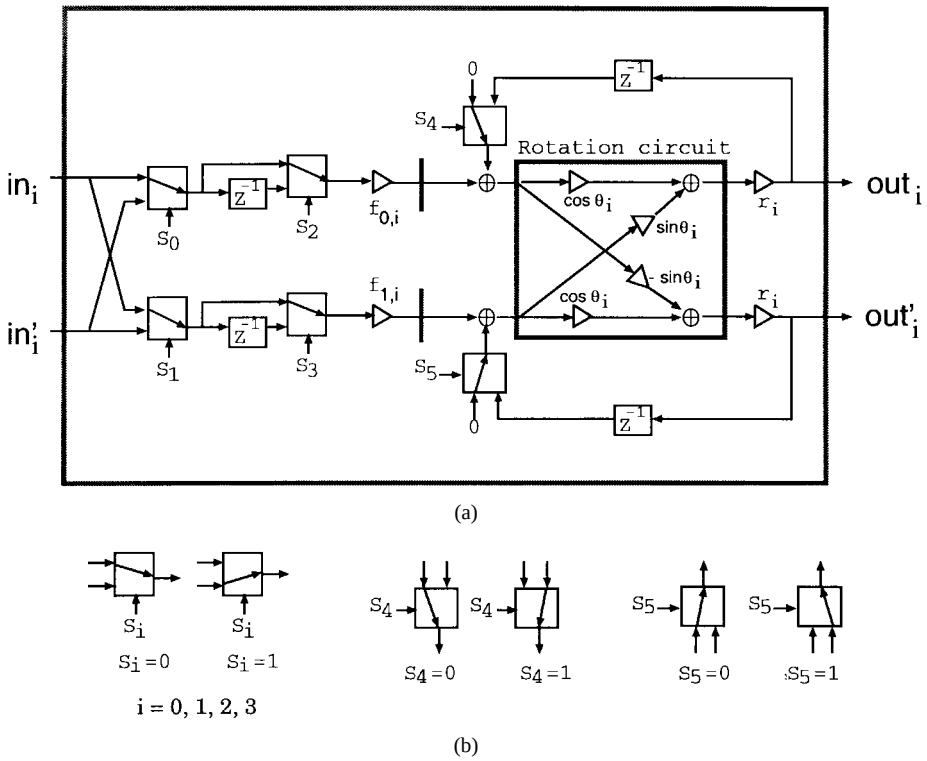
The six switches $[s_0 s_1 s_2 s_3 s_4 s_5]$ control the data paths inside the module. The switch pair s_0 and s_1 select the input from either in_i or in_i'. Switches s_2 and s_3 decide if the delay element is used or not. With the setting $s_2 s_3 = 11$, the delay element in Fig. 39 can be incorporated into the module $A_i(z)$. Therefore, we do not need to implement the delay elements explicitly in the IIR filtering operation. The last switch pair is s_4 and s_5. They control the two feedback paths in the module. When $s_4 s_5 = 11$, the delayed module outputs are added with the current inputs as required in the IIR and DT operations. The two multipliers' r_i at the outputs of the rotation circuit are required only when we want to incorporate IIR function into this unified module design.

In addition to the data paths, we also need to set the values of the scaling multipliers $f_{0,i}$, $f_{1,i}$, and r_i, as well as the rotating angle θ_i. For the FIR/QMF filtering, these parameters can be easily computed using the formula in [43, ch. 6] and [171]. For the IIR/DT operations, we can apply the results in Sections VII-A2 and VII-A3 to compute those parameters. The complete settings of the programmable module for the FIR/QMF/IIR/DT operations can be found in [170].

B. Operation of the DSP Computing Engine

In the previous section, we have derived a unified programmable module that can be used as the basic PE in the operations of FIR/QMF/IIR/DT. Since in each function of the FIR/QMF/IIR/DT the basic PE's are connected in different way, we employ a reconfigurable interconnection network to perform the connection task. Fig. 42 shows the overall architecture of the proposed DSP computing engine under FIR filtering mode. It consists of two parts. One is the *programmable module array* with identical unified programmable modules. The other is the *reconfigurable interconnection network*, which connects those programmable modules according to the data paths. In the FIR/QMF/IIR operations, the data are processed in a serial-input serial-output way. Hence, the programmable modules need to be cascaded for those operations. For example, the FIR modules can be connected by setting the interconnection network as shown in Fig. 42. Similarly, we can also realize the connections of the IIR modules in Fig. 39 by using the network setting as shown in Fig. 43. On the other hand, the DT architecture in Section VII-A3 performs the block transforms in an SIPO way. The interconnection network is configured according to the combination functions defined in Table 5.

The operation of the DSP computing engine is as follows. During the *initialization mode*, the host processor will compute all the necessary parameters $\mathbf{f}_i, r_i, \theta_i$ according to the function type (FIR/QMF/IIR/DT) and the function specification. Next, the host processor needs to configure the interconnection network according to the function type. Once the computing engine has been initialized, it enters the *execution mode*. In the applications of FIR/IIR/QMF, the host processor continuously feeds the data sequence into the computing engine. All PE's are running in parallel, and the host processor can collect the filtering outputs in a fully pipelined way. In the block DT application, the block input data is fed into the computing engine serially. Each PE of the programmable module array updates $X_{C,k}(t)$ and $X_{S,k}(t)$ in (30) and (31), $k = 0, 1, \cdots, N-1$, simultaneously. After the last datum enters the programmable module



(a)

(b)

Fig. 41. (a) Programmable module to perform the FIR/QMF/IIR/DT. (b) Switches used in the module.

array, the evaluation of DT coefficients is completed. Then the interconnection network will combine the module outputs according to the combination function defined in Table 5, and the transform coefficients can be obtained in parallel at the outputs of the network. At the same time, the host processor will reset all internal registers (delay elements) of the programmable modules to zero so that the next block transform can be conducted immediately. The detailed settings of the computing engine as well as some design examples are given in [170].

C. Speed-Up of the DSP Computing Engine Using Multirate Approach

In preceding sections, we have shown that the multirate approach provides a direct and efficient way to achieve very-high-speed data processing at the algorithmic/architectural level. Therefore, if we can find a way to reconfigure the computing engine to perform multirate operations, the speed performance can be further improved. In what follows, we will show how to map the multirate FIR architectures in Fig. 10 to our computing engine design. Since processing elements now operate at only half of input data rate, the speed performance of the proposed DSP engine can be doubled on the fly based on the same programmable modules and interconnection network. The same principle also applies to the low-power operation.

To map the multirate FIR architecture to our design, we first find the three $(N/2)$th-order FIR subfilters $H_0(z)$, $H_1(z)$, and $\hat{H}(z) \triangleq H_0(z) + H_1(z)$ given FIR transfer function. Then we can implement each subfilter using the FIR lattice structure discussed in Section VII-A1. The resulting architecture is depicted in Fig. 44(a), where $\tilde{R}_i$, $\hat{R}_i$, and $\bar{R}_i$ correspond to the ith basic modules used in $H_0(z)$, $\hat{H}(z)$, and $H_1(z)$, respectively. Note that each basic FIR module can be realized by the unified computational module in Fig. 41. Hence, we can map Fig. 44(a) onto our computing engine design with the mapping

$$\tilde{R}_i \longrightarrow M_{3i}, \quad \hat{R}_i \longrightarrow M_{3i+1}, \quad \bar{R}_i \longrightarrow M_{3i+2} \tag{96}$$

for $i = 0, 1, \cdots, N/2 - 1$. Fig. 44(b) illustrates the realization of a multirate sixth-order FIR by using nine programmable modules. The detailed settings are described in [170].

The operation of the DSP computing engine to perform multirate FIR filtering is as follows. Once the programmable modules and interconnection network have been initialized, the host processor sends data at f_s rate to the downsampling circuit in Fig. 10. Then the outputs of the downsampling circuit $x_i(n), i = 0, 1, 2$ will be processed by the three FIR subfilters in parallel at only $f_s/2$ rate. After the subfilter outputs' $y_i(n)$ are generated, the FIR filtering output $y(n)$ is reconstructed through the upsampling circuit in a fully pipelined way, and the data rate of $y(n)$ is back to f_s. Since all PE's are running in the $f_s/2$ region, now the data rate is twice as fast as that in Fig. 42. Namely, we double the speed performance at the architectural level without any specially designed high-speed circuits. Since we need $N/2$ modules for the implementation of each subfilter, a total of $3N/2$ modules will be used to perform an Nth-order multirate FIR filtering operation; i.e., we only need 50% more PE's for the doubled speed performance. This overhead is handled by simply activating more PE's in the programmable module array and reconfiguring the

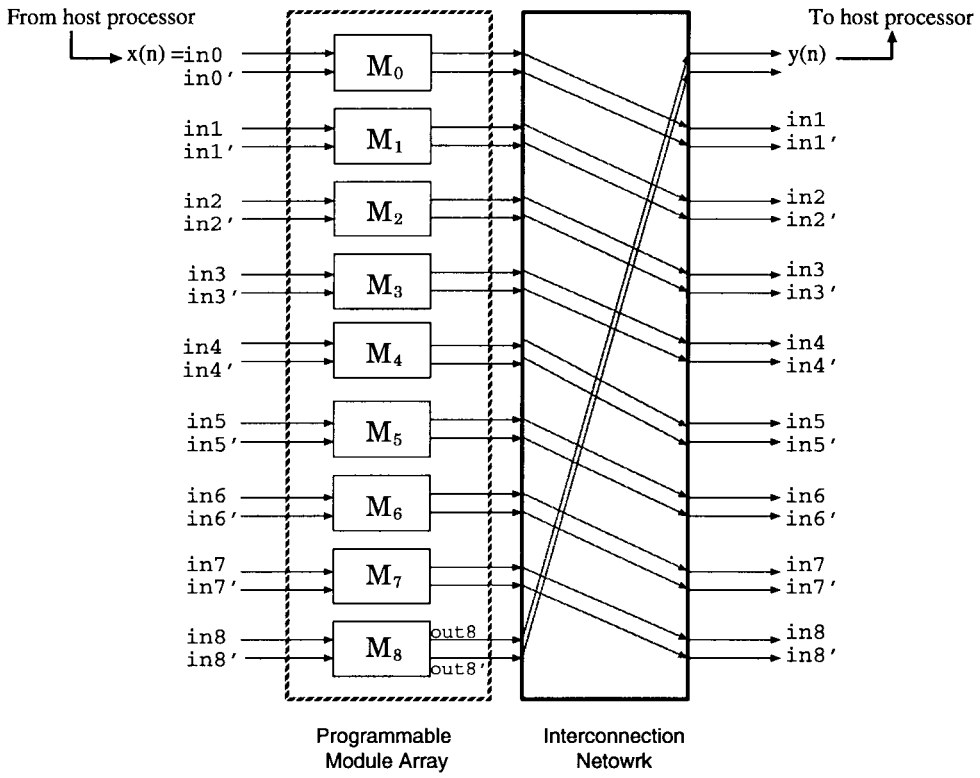


Fig. 42. Overall architecture of the DSP computing engine for FIR filtering.

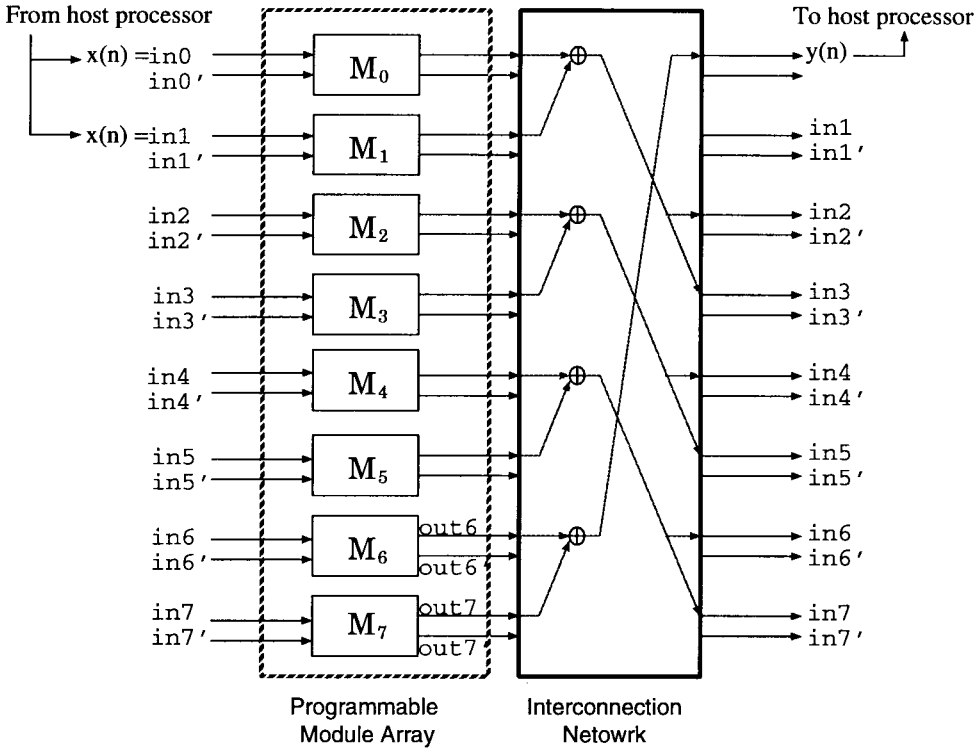


Fig. 43. Overall architecture of the DSP computing engine for IIR (ARMA) filtering.

interconnection network instead of implementing new types of PE's and a new interconnection network explicitly.

Similarly, the multirate IIR filtering operation can be mapped to the unified design by finding the polyphase components of the IIR function and realizing each subfilter with the IIR lattice structure in Section VII-A2. The mapping of the multirate DT architecture, QR decomposition least squares lattice adaptive filtering [166], and DCT-based ME scheme [46] is also achievable. The readers may refer to [170] for details.

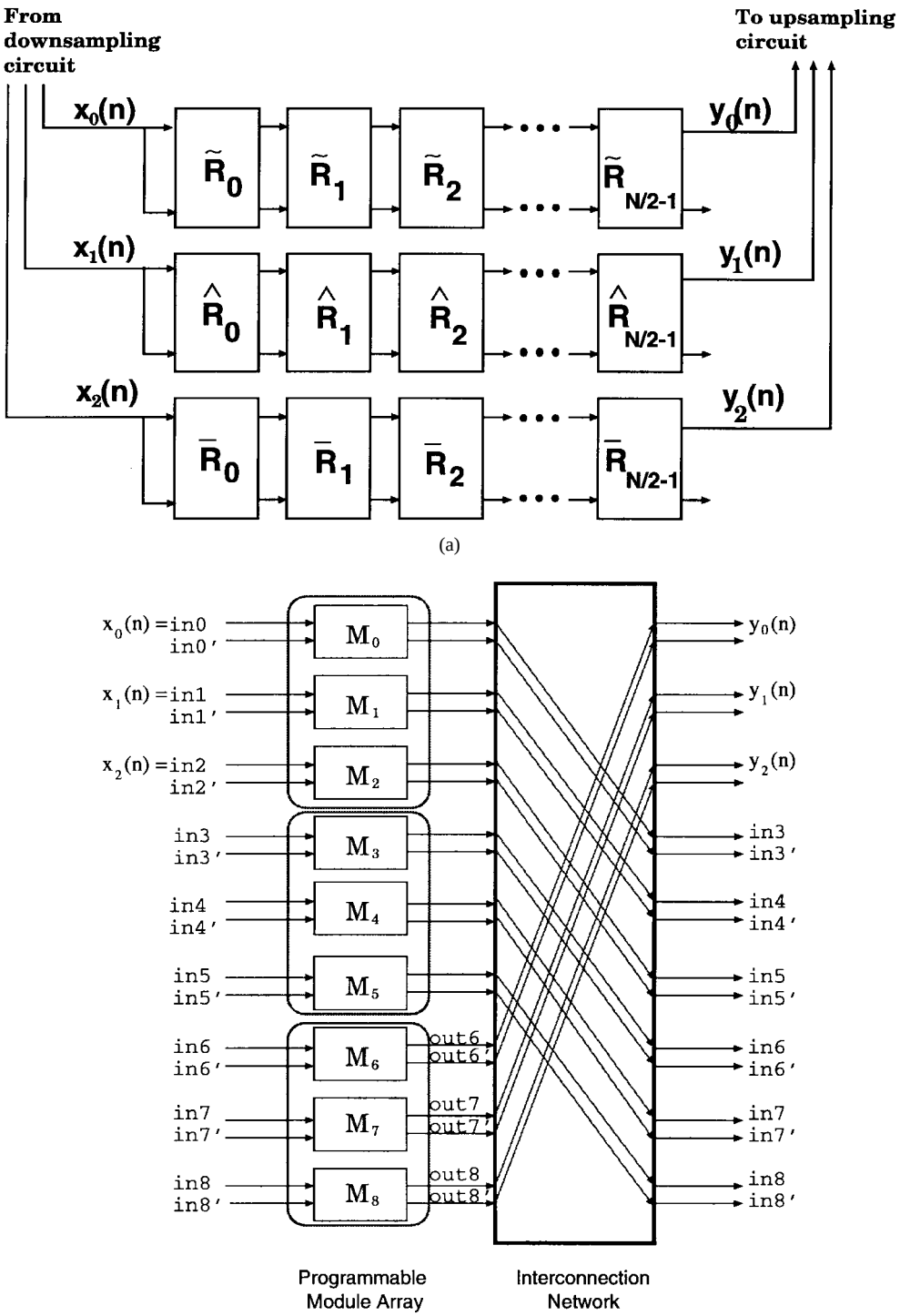


Fig. 44. (a) Multirate FIR filtering based on the FIR lattice structure. (b) Mapping part (a) to the unified computing architecture.

VIII. Conclusion

In this paper, we presented a new algorithm-based design technique, the multirate approach, to achieve low-power and high-performance computations in multimedia applications. We illustrate this new design concept by applying it along with other low-power/high-performance methods such as look-ahead and pipelining to several major problems encountered in multimedia signal processing, including the design of multirate DSP architectures, a unified architecture with speed-up capability, and source/channel-coding architectures. The proposed multirate approach provides VLSI system designers and algorithm developers a new design tool in compensating the speed penalty in addition to the existing techniques such as parallel processing and pipelining. Besides the application to low-power design, the proposed approach can also be readily used for very-high-speed data processing.

REFERENCES

[1] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design—A Systems Perspective*, 2nd ed. Reading, MA: Addison-Wesley, 1993.

[2] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digital design," *IEEE J. Solid-State Circuits*, vol. 27, pp. 473–484, Apr. 1992.

[3] J. M. Rabaey, "Low power digital design," in *Tutorials of the IEEE Int. Symposium on Circuits and Systems*. London: IEEE Press, 1994, ch. 8, pp. 373–386.

[4] G. M. Blair, "Designing low power digital CMOS," *Electron. Commun. Eng. J.*, pp. 229–236, Oct. 1994.

[5] Z. J. Lemnios and K. J. Gabriel, "Low-power electronics," *IEEE Design Test Comput. Mag.*, vol. 11, pp. 14–23, Winter 1994.

[6] A. P. Chandrakasan and R. W. Brodersen, "Minimizing power consumption in digital CMOS circuits," *Proc. IEEE*, vol. 83, pp. 498–523, Apr. 1995.

[7] D. Liu and C. Svensson, "Trading speed for low power by choice of supply and threshold voltages," *IEEE J. Solid-State Circuits*, vol. 28, pp. 10–17, Jan. 1993.

[8] A. Yoshino, K. Kumagai, S. Kurosawa, H. Itoh, and K. Okumura, "Design methodology for low-power, high-speed CMOS devices utilizing SOI technology," in *Proc. IEEE Int. SOI Conf.*, Palm Springs, CA, Oct. 1993, pp. 170–171.

[9] K. Hwang, *Computer Arithmetic: Principles, Architecture, Design*. New York: Wiley, 1979.

[10] S. A. White, "Applications of distributed-arithmetic to digital signal processing: A tutorial review," *IEEE Acoust., Speech, Signal Processing Mag.*, pp. 4–19, July 1989.

[11] M. T. Sun, T. C. Chen, and A. M. Gottlieb, "VLSI implementation of a 16 $\times$ 16 discrete cosine transform," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 610–617, Apr. 1989.

[12] 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.

[13] V. Srinivasan and K. J. R. Liu, "VLSI design of high-speed time-recursive 2-D DCT/IDCT processor for video application," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 87–96, Feb. 1996.

[14] F. J. Taylor, "Residue arithmetic: A tutorial with examples," *IEEE Comput. Mag.*, pp. 50–62, May 1984.

[15] M. A. Bayoumi, G. A. Jullien, and W. C. Miller, "A look-up table VLSI design methodology for RNS structures used in DSP applications," *IEEE Trans. Circuits Syst.*, vol. 34, pp. 604–616, June 1987.

[16] K. M. Elleithy and M. A. Bayoumi, "Fast and flexible architectures for RNS arithmetic decoding," *IEEE Trans. Circuits Syst. II*, vol. 39, pp. 226–235, Apr. 1992.

[17] Y. C. Lim and S. R. Parker, "FIR filter design over a discrete power-of-two coefficient space," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 31, pp. 583–591, June 1983.

[18] Y. C. Lim, "Design of discrete-coefficient-value linear phase FIR filters with optimum normalized peak ripple magnitude," *IEEE Trans. Circuits Syst.*, vol. 37, pp. 1480–1486, Dec. 1990.

[19] C.-K. Chen and J.-H. Lee, "Design of linear-phase quadrature mirror filters with power-of-two coefficients," *IEEE Trans. Circuits Syst. I*, vol. 41, pp. 445–456, July 1994.

[20] J. Candy and G. Temes, *Oversampling Delta-Sigma Data Converters*. New York: IEEE Press, 1991.

[21] W. C. Athas, L. Svensson, J. Koller, N. Tzartzanis, and E. Y.-C Chou, "Low-power digital systems based on adiabatic-switching principles," *IEEE Trans. VLSI Syst.*, vol. 2, pp. 398–406, Dec. 1994.

[22] K. Roy and S. C. Prasad, "Circuit activity based logic synthesis for low power reliable operations," *IEEE Trans. VLSI Syst.*, vol. 1, pp. 503–513, Dec. 1993.

[23] M. Alidina, J. Monteiro, S. Devadas, A. Ghosh, and M. Papaefthymiou, "Precomputation-based sequential logic optimization for low power," *IEEE Trans. VLSI Syst.*, vol. 2, pp. 426–436, Dec. 1994.

[24] C.-L. Su, C.-Y. Tsui, and A. M. Despain, "Saving power in the control path of embedded processors," *IEEE Design Test Comput. Mag.*, vol. 11, pp. 24–30, Winter 1994.

[25] L. Benini, P. Siegel, and G. D. Micheli, "Saving power by synthesizing gated clocks for sequential circuits," *IEEE Design Test Comput. Mag.*, vol. 11, pp. 32–41, Winter 1994.

[26] S. Gary, P. Ippolito, G. Gerosa, C. Dietz, J. Eno, and H. Sanchez, "PowerPC 603, a microprocessor for portable computers," *IEEE Design Test Comput. Mag.*, vol. 11, pp. 14–23, Winter 1994.

[27] S. A. Khan and V. K. Madisetti, "System partitioning of MCM for low power," *IEEE Design Test Comput. Mag.*, vol. 12, pp. 41–52, Spring 1995.

[28] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion-compensated interframe coding for video conferencing," in *Proc. National Telecommun. Conf.*, New Orleans, LA, 1981, pp. G.5.3.1–G.5.3.5.

[29] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Commun.*, vol. COM-29, pp. 1799–1808, Dec. 1981.

[30] A.-Y. Wu and K. J. R. Liu, "Split recursive least-squares: Algorithms, architectures, and applications," *IEEE Trans. Circuits Syst. II*, vol. 43, pp. 645–658, Sept. 1996.

[31] S. H. Nawab, A. Oppenheim, A. P. Chandrakasan, J. M. Winograd, and J. T. Ludwig, "Approximate signal processing," *J. VLSI Signal Processing*, vol. 15, pp. 177–200, Jan. 1997.

[32] L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*. Englewood Cliffs, NJ: Prentice-Hall, 1978.

[33] H. Li, A. Lundmark, and R. Forchheimer, "Image sequence coding at very low bitrates: A review," *IEEE Trans. Image Processing*, vol. 3, pp. 589–609, Sept. 1994.

[34] A. V. Oppenheim and H. Nawab, *Symbolic and Knowledge Based Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1992.

[35] T. H. Meng, R. Broderson, and D. Messerschmitt, "Asynchronous design for programmable digital signal processors," *IEEE Trans. Signal Processing*, vol. 39, pp. 939–952, Apr. 1991.

[36] K. J. R. Liu, S.-F. Hsieh, and K. Yao, "Systolic block householder transformation for RLS algorithm with two-level pipelined implementation," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 40, pp. 946–958, Apr. 1992.

[37] C.-L. Wang, "Bit-serial VLSI implementation of delayed LMS adaptive FIR filter," *IEEE Trans. Signal Processing*, vol. 42, pp. 2169–2175, Aug. 1994.

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

[39] K. K. Parhi and D. G. Messerschmitt, "Pipeline interleaving and parallelism in recursive digital filters—Part I: Pipelining using scattered look-ahead and decomposition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp. 1099–1117, July 1989.

[40] ——, "Pipeline interleaving and parallelism in recursive digital filters—Part II: Pipelined incremental block filtering," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp. 1118–1134, July 1989.

[41] K. K. Parhi, "High-level algorithm and architecture transformations for DSP synthesis," *J. VLSI Signal Processing*, vol. 9, pp. 121–143, Jan. 1995.

[42] N. R. Shanbhag and K. K. Parhi, *Pipelined Adaptive Digital Filter*. Norwell, MA: Kluwer, 1994.

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

[44] K. J. R. Liu and C. T. Chiu, "Unified parallel lattice structures for time-recursive discrete cosine/sine/Hartley transforms," *IEEE Trans. Signal Processing*, vol. 41, pp. 1357–1377, Mar. 1993.

[45] K. J. R. Liu, C. T. Chiu, R. K. Kolagotla, and J. F. Ja' Ja', "Optimal unified architectures for the real-time computation of time-recursive discrete sinusoidal transforms," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, pp. 168–180, Apr. 1994.

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

[47] ——, "DCT-based subpixel motion estimation," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, Atlanta, GA, 1996, pp. 1930–1933.

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

[49] G. Fettweis and H. Meyr, "Parallel Viterbi algorithm implementation: Breaking the ACS-bottleneck," *IEEE Trans. Commun.*, vol. 37, pp. 785–789, Aug. 1989.

[50] H.-D. Lin and D. G. Messerschmitt, "Algorithms and architectures for concurrent Viterbi decoding," in *Proc. IEEE Int. Conf. Communications*, 1989, pp. 836–840.

[51] H. K. Thapar and J. Cioffi, "A block processing method for designing high speed Viterbi decoders," in *Proc. IEEE Communications Conf.*, June 1989, pp. 1096–1100.
[52] G. Long, F. Ling, and J. G. Proakis, "The LMS algorithm with delayed coefficient adaptation," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp. 1397–1405, Sept. 1989.
[53] T. H. Meng and D. G. Messerschmitt, "Arbitrarily high sampling rate adaptive filtering," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, pp. 455–70, Apr. 1987.
[54] A.-Y. Wu, K. J. R. Liu, Z. Zhang, K. Nakajima, A. Raghupathy, and S.-C. Liu, "Algorithm-based low power DSP design: Methodology and verification," in *VLSI Signal Processing VIII*, T. Nishitani and K. K. Parhi, Eds. New York: IEEE Press, 1995, pp. 277–286.
[55] H. K. Kwan and M. T. Tsim, "High speed 1-D FIR digital filtering architectures using polynomial convolution," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, Dallas, TX, Apr. 1987, pp. 1863–1866.
[56] Z.-J. Mou and P. Duhamel, "Short-length FIR filters and their use in fast nonrecursive filtering," *IEEE Trans. Signal Processing*, vol. 39, pp. 1322–1332, June 1991.
[57] J. D. Johnston, "A filter family designed for use in quadrature mirror filter banks," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, 1980, pp. 291–294.
[58] Y. Nakamura, K. Oguri, and A. Nagoya, "Synthesis from pure behavioral descriptions," in *High-Level VLSI Synthesis*, R. Camposano and W. Wolf, Eds.. Norwell, MA: Kluwer, 1991, pp. 205–229.
[59] A.-Y. Wu, K. 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.
[60] E. Frantzeskakis, J. S. Baras, and K. J. R. Liu, "Time-recursive computation and real-time parallel architectures: A framework," *IEEE Trans. Signal Processing*, vol. 43, pp. 2762–2775, Nov. 1995.
[61] H. S. Hou, "A fast recursive algorithm for computing the discrete cosine transform," *IEEE Trans. Acoust. Speech, Signal Processing*, vol. 35, pp. 1455–1461, Oct. 1987.
[62] B. G. Lee, "A new algorithm to compute the discrete cosine transform," *IEEE Trans. Acoust. Speech, Signal Processing*, vol. 32, pp. 1243–1245, Dec. 1984.
[63] A.-Y. Wu and K. J. R. Liu, "Algorithm-based low-power transform coding architectures: The multirate approach," *IEEE Trans. VLSI Syst.*, to be published.
[64] K. Konstantinides, "Fast subband filtering in MPEG audio coding," *IEEE Signal Processing Lett.*, vol. 1, pp. 26–28, Feb. 1994.
[65] A.-Y. Wu and K. J. R. Liu, "A low-power and low-complexity DCT/IDCT VLSI architecture based on backward Chebyshev recursion," in *Proc. IEEE Int. Symp. Circuits and Systems*, London, England, May 1994, pp. IV.155–158.
[66] A. N. Skodras, "Fast discrete cosine transform pruning," *IEEE Trans. Signal Processing*, vol. 42, pp. 1833–1837, July 1994.
[67] H. S. Malvar, "Lapped transforms for efficient transform/subband coding," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 38, pp. 969–978, June 1990.
[68] ——, "Extended lapped transforms: Properties, applications, and fast algorithms," *IEEE Trans. Signal Processing*, vol. 40, pp. 2703–2714, Nov. 1992.
[69] H. S. Malvar and D. H. Staelin, "The LOT: Transform coding without blocking effects," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp. 553–559, Apr. 1989.
[70] E. Frantzeskakis, J. S. Baras, and K. J. R. Liu, "Time-recursive computation and real-time parallel architectures with application on the modulated lapped transform," in *Proc. SPIE Advanced Signal Processing Algorithms, Architectures, and Implementations IV*, San Diego, CA, July 1993, pp. 100–111.
[71] H. S. Malvar, "Fast algorithm for modulated lapped transform," *Electron. Lett.*, vol. 27, pp. 775–776, Apr. 1991.
[72] H. Chiang and J. Liu, "Regressive implementations for the forward and inverse MDCT in MPEG audio coding," *IEEE Signal Processing Lett.*, pp. 116–118, Apr. 1996.
[73] 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.
[74] P. Pirsch, N. Demassieux, and W. Gehrke, "VLSI architectures for video compression—A survey," *Proc. IEEE*, vol. 83, pp. 220–245, Feb. 1995.
[75] E. Petajan and J. Mailhot, "Grand alliance (HDTV) system," in *Proc. SPIE*, San Jose, CA, 1994, pp. 2–15
[76] L. Chiariglione, "The MPEG and multimedia communications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, pp. 5–18, Feb. 1997.
[77] G. Karlsson, "Asynchronous transfer of video," *IEEE Commun. Mag.*, vol. 36, pp. 118–126, Aug. 1996.
[78] R. Srinivasan and K. R. Rao, "Motion compensating coder for video conferencing," *IEEE Trans. Commun.*, vol. COM-33, pp. 888–896, Aug. 1985.
[79] K. M. Yang, M. T. Sun, and L. Wu, "A family of VLSI designs for the motion compensation block-matching algorithm," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 1317–1325, Oct. 1989.
[80] T. Komarek and P. Pirsch, "Array architectures for block-matching algorithm," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 1301–1308, Oct. 1989.
[81] L. D. Vos and M. Stegherr, "Parameterizable VLSI architectures for the full-search block-matching algorithm," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 1309–1316, Oct. 1989.
[82] H.-M. Jong, L.-G. Chen, and T.-D. Chiueh, "Parallel architectures of 3-step search block-matching algorithms for video coding," in *Proc. IEEE Int. Symp. Circuits and Systems*, 1994, pp. 209–212.
[83] G. Gupta and C. Chakrabarti, "VLSI architecture for hierarchical block matching," in *Proc. IEEE Int. Symp. Circuits and Systems*, 1994, pp. 215–218.
[84] S. B. Pan, S. S. Chae, and R. H. Park, "VLSI architectures for block matching algorithms using systolic array," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 67–73, Feb. 1996.
[85] S. Dutta and W. Wolf, "A flexible parallel architecture adapted to block-matching motion-estimation algorithms," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 74–86, Feb. 1996.
[86] R. W. Young and N. G. Kingsbury, "Frequency-domain motion estimation using a complex lapped transform," *IEEE Trans. Image Processing*, vol. 2, pp. 2–17, Jan. 1993.
[87] A. N. Netravali and J. D. Robbins, "Motion compensated television coding—Part 1," *Bell Syst. Tech. J.*, vol. 58, pp. 631–670, Mar. 1979.
[88] J. D. Robbins and A. N. Netravali, "Recursive motion compensation: A review," in *Image Sequence Processing and Dynamic Scene Analysis*, T. S. Huang, Ed. Berlin, Germany: Springer-Verlag, 1983, pp. 76–103.
[89] R. C. Kim and S. U. Lee, "A VLSI architecture for a pel recursive motion estimation algorithm," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 1291–1300, Oct. 1989.
[90] A. Singh, *Optic Flow Computation—A Unified Perspective*. New York: IEEE Computer Society Press, 1991.
[91] A. Kojima, N. Sakurai, and J. Kishigami, "Motion detection using 3D-FFT spectrum," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, Minneapolis, MN, Apr. 1993, pp. V213–V216.
[92] K. R. Rao and J. J. Hwang, *Techniques and Standards for Image, Video, and Audio Coding*. Englewood Cliffs, NJ: Prentice-Hall, 1994.
[93] B. Girod, *Motion Compensation: Visual Aspects, Accuracy, and Fundamental Limits*. Norwell, MA: Kluwer, 1993.
[94] S.-I. Uramoto, A. Takabatake, and M. Yoshimoto, "A half-pel precision motion estimation processor for NTSC-resolution video," *IEICE Trans. Electronics*, vol. 77, p. 1930, Dec. 1994.
[95] K. Ishihara, S. Masuda, S. Hattori, H. Nishikawa, Y. Ajioka, T. Yamada, H. Amishiro, S. Uramoto, M. Yoshimoto, and T. Sumi, "A half-pel precision MPEG2 motion-estimation processor with concurrent three-vector search," *IEEE J. Solid-State Circuits*, vol. 30, pp. 1502–1509, Dec. 1995.
[96] B. Girod, "Motion-compensating prediction with fractional-pel accuracy," *IEEE Trans. Commun.*, vol. 41, p. 604, Apr. 1993.
[97] 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.
[98] J. Chen and K. J. R. Liu, "A complete pipelined parallel CORDIC architecture for motion estimation," in *Proc. IEEE Int. Symp. Circuits and Systems*, Hong Kong, 1997, pp. 2801–2804.
[99] J. Chen and K. J. R. Liu, "A fully pipelined parallel CORDIC architecture for half-pel motion estimation," in *Proc. IEEE Int. Conf. Image Processing*, Santa Barbara, CA, Oct. 1997, vol. 2, pp. 574–577.
[100] K. Hasegawa, K. Ohara, A. Oka, T. Kamada, Y. Nagaoka, K.

Yano, E. Yamauchi, T. Kashiro, and T. Nakagawa, "Low-power video encoder/decoder chip set for digital VCR's," *IEEE J. Solid-State Circuits*, vol. 31, pp. 1780–1788, Nov. 1996.
[101] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, pp. 438–442, Aug. 1994.
[102] K. R. Rao and P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, and Applications*. New York: Academic, 1990.
[103] N. I. Cho and S. U. Lee, "DCT algorithms for VLSI parallel implementations," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 38, pp. 1899–1908, Dec. 1989.
[104] L. W. Chang and M. C. Wu, "A unified systolic array for discrete cosine and sine transforms," *IEEE Trans. Signal Processing*, vol. 39, pp. 192–194, Jan. 1991.
[105] S. Whitaker, J. Canaris, and K. Cameron, "Reed Solomon VLSI codec for advanced television," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 1, pp. 230–236, June 1991.
[106] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*. Englewood Cliffs, NJ: Prentice-Hall, 1995.
[107] T. S. Rappaport, *Wireless Communications*. New York: IEEE Press, 1996.
[108] K. Maxwell, "Asymmetric digital subscriber line," *IEEE Commun. Mag.*, vol. 34, pp. 100–107, Oct. 1996.
[109] J. B. Cain and D. N. McGregor, "A recommended error control architecture for ATM networks with wireless links," *IEEE J. Select. Areas Commun.*, vol. 15, pp. 16–27, Jan. 1997.
[110] I. S. Reed and G. Solomon, "Reed–Solomon codes: A historical overview," in *Reed Solomon Codes and Their Applications*, S. P. Wicker and V. K. Bhargava, Eds. New York: IEEE Press, 1991, pp. 1–16.
[111] E. R. Berlekamp, *Algebraic Coding Theory*. New York: McGraw-Hill, 1968.
[112] J. L. Massey, "Shift register synthesis and BCH coding," *IEEE Trans. Inform. Theory*, vol. IT-15, pp. 122–127, Jan. 1969.
[113] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa, "A method for solving key equation for decoding Goppa codes," *Inform. Control*, vol. 27, pp. 87–99, 1975.
[114] G. D. Forney, Jr., "On decoding BCH codes," *IEEE Trans. Inform. Theory*, vol. IT-11, pp. 549–557, Oct. 1965.
[115] R. E. Blahut, *Algebraic Methods for Signal Processing and Error Control Coding*. Berlin, Germany: Springer-Verlag, 1992.
[116] Y. R. Shayan, T. Le-Ngoc, and V. K. Bhargava, "A versatile time domain Reed–Solomon decoder," *IEEE J. Select. Areas Commun.*, vol. 8, pp. 1535–1542, Oct. 1990.
[117] K. Y. Liu, "Architecture for VLSI design of Reed–Solomon decoders," *IEEE Trans. Comput.*, vol. C-33, pp. 178–189, Feb. 1984.
[118] H. M. Shao, T. K. Truong, L. J. Deutsch, J. H. Yuen, and I. S. Reed, "A VLSI design of a pipeline Reed–Solomon decoder," *IEEE Trans. Comput.*, vol. C-34, pp. 393–402, May 1985.
[119] N. Demassieux, F. Jutand, and M. Muller, "A 10 MHz (255, 233) Reed–Solomon decoder," in *Proc. IEEE Custom Integrated Circuits Conf.*, 1988, pp. 17.6.1–17.6.4.
[120] H. M. Shao and I. S. Reed, "On the VLSI design of a pipeline Reed–Solomon decoder using systolic arrays," *IEEE Trans. Comput.*, vol. 37, pp. 1273–1280, Oct. 1988.
[121] P. Tong, "A 40 MHz encoder-decoder chip generated by a Reed-Solomon code compiler," in *Proc. Custom Integrated Circuits Conf.*, Boston, MA, May 1990, pp. 13.5.1–13.5.4.
[122] E. Berlekamp, G. Seroussi, and P. Tong, "A hypersystolic (Reed–Solomon) decoder," in *Reed Solomon Codes and Their Applications*, S. P. Wicker and V. K. Bhargava, Eds. New York: IEEE Press, 1991, pp. 205–241.
[123] R. P. Brent and H. T. Kung, "Systolic VLSI arrays for polynomial GCD computation," *IEEE Trans. Comput.*, vol. C-33, pp. 731–736, Aug. 1984.
[124] A. Raghupathy and K. J. R. Liu, "Low power/high speed design of a Reed Solomon decoder," in *Proc. IEEE Int. Symp. Circuits and Systems*, Hong Kong, June 1997, pp. 2060–2063.
[125] D.-I. Oh, Y. Kim, and S.-Y. Hwang, "A VLSI architecture of the trellis decoder block for the digital HDTV grand alliance system," *IEEE Trans. Consumer Electron.*, vol. 42, pp. 346–356, Aug. 1996.
[126] I. Kang and A. Willson, Jr., "A low-power state-sequential Viterbi decoder for CDMA digital cellular applications," in *Proc. IEEE Int. Symp. Circuits and Systems*, May 1996, pp. 272–275.
[127] C. B. Shung, P. H. Siegel, H. K. Thapar, and R. Karabed, "A 30 MHz trellis codec chip for partial-response channels," *IEEE J. Solid-State Circuits*, vol. 26, pp. 1981–1987, Dec. 1991.
[128] H. Dawid, G. Fettweis, and H. Meyr, "A CMOS IC for Gb/s Viterbi decoding: System design and VLSI implementation," *IEEE Trans. VLSI Syst.*, vol. 4, pp. 17–31, Mar. 1996.
[129] G. Feygin, P. Chow, P. G. Gulak, J. Chappel, G. Goodes, O. Hall, A. Sayes, S. Singh, M. B. Smith, and S. Wilton, "A VLSI implementation of a cascade Viterbi decoder with traceback," in *Proc. IEEE Int. Symp. Circuits and Systems*, May 1993, pp. 1945–1948.
[130] P. J. Black and T. H. Meng, "A unified approach to the Viterbi algorithm state metric update for shift register processes," in *Proc. IEEE Int. Conf. Acoust. Speech, Signal Processing*, Mar. 1992, pp. V.629–V.632.
[131] ——, "Hybrid survivor path architectures for Viterbi decoders," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, 1993, pp. I.433–I.436.
[132] C. M. Rader, "Memory management in a Viterbi decoder," *IEEE Trans. Commun.*, vol. COM-29, pp. 1399–1401, Sept. 1981.
[133] R. Cypher and C. B. Shung, "Generalized traceback techniques for survivor memory management in the Viterbi algorithm," in *Proc. IEEE Global Telecommunications Conf.*, Dec. 1990, pp. 1318–1322.
[134] M. Boo, F. Arguello, J. D. Bruguera, R. Doallo, and E. L. Zapata, "High-performance VLSI architecture for the Viterbi algorithm," *IEEE Trans. Commun.*, vol. 45, pp. 168–176, Feb. 1997.
[135] P. J. Black and T. H. Meng, "A 140-Mb/s, 32-state, radix-4 Viterbi decoder," *IEEE J. Solid-State Circuits*, vol. 27, pp. 1877–1885, Dec. 1992.
[136] P. G. Gulak and T. Kailath, "Locally connected VLSI architectures for the Viterbi algorithm," *IEEE J. Select. Areas Commun.*, vol. 6, pp. 527–537, Apr. 1988.
[137] H.-D. Lin, B. Shung, and D. G. Messerschmitt, "Viterbi decoders for convolutional codes," in *VLSI Signal Processing IV*. New York: IEEE Press, 1990, pp. 381–391.
[138] C. B. Shung, H. D. Lin, R. Cypher, P. H. Siegel, and H. K. Thapar, "Area-efficient architectures for the Viterbi algorithm—Part I: Theory," *IEEE Trans. Commun.*, vol. 41, pp. 636–643, Apr. 1993.
[139] ——, "Area-efficient architectures for the Viterbi algorithm—Part II: Applications," *IEEE Trans. Commun.*, vol. 41, pp. 802–807, May 1993.
[140] F. Daneshgaran and K. Yao, "The iterative collapse algorithm: A novel approach for the design of long constraint length Viterbi decoders—Part I," *IEEE Trans. Commun.*, vol. 43, pp. 1409–1418, Feb. 1995.
[141] H.-L. Li and C. Chakrabarti, "A new architecture for the Viterbi decoder for code rate k/n," *IEEE Trans. Commun.*, vol. 44, pp. 158–164, Feb. 1996.
[142] T. Ishitani, K. Kansho, N. Miyahara, S. Kubota, and S. Kato, "A scarce-state-transition Viterbi-decoder VLSI for bit error correction," *IEEE J. Solid-State Circuits*, vol. SC-22, pp. 575–581, Aug. 1987.
[143] S. Kubota, S. Kato, and T. Ishitani, "Novel Viterbi decoder VLSI implementation and its performance," *IEEE Trans. Commun.*, vol. 41, pp. 1170–1178, Aug. 1993.
[144] J. B. Cain, G. C. Clark, and J. M. Geist, "Punctured convolutional codes of rate $(n-1)/n$ and simplified maximum likelihood decoding," *IEEE Trans. Inform. Theory*, vol. IT-25, pp. 97–100, Jan. 1979.
[145] G. Fettweis and H. Meyr, "High speed Viterbi processor: A systolic array solution," *IEEE J. Select. Areas Commun.*, vol. 8, pp. 1520–1533, Oct. 1990.
[146] ——, "High-speed parallel Viterbi decoding: Algorithm and VLSI-architecture," *IEEE Commun. Mag.*, pp. 46–55, May 1989.
[147] G.-H. Im, D. B. Harman, G. Huang, A. V. Mandzik, M. H. Nguyen, and J.-J. Werner, "51.84 Mb/s 16-CAP ATM LAN standard," *IEEE J. Select. Areas Commun.*, vol. 13, pp. 620–632, May 1995.
[148] G.-H. Im and J.-J. Werner, "Bandwidth-efficient digital transmission over unshielded twisted-pair wiring," *IEEE J. Select. Areas Commun.*, vol. 13, pp. 1643–1655, Dec. 1995.
[149] J. S. Chow, J. C. Tu, and J. M. Cioffi, "A discrete multitone transceiver system for HDSL applications," *IEEE J. Select. Areas Commun.*, vol. 9, pp. 895–909, Aug. 1991.
[150] M. Ho, J. M. Cioffi, and J. A. C. Bingham, "Discrete mul-

titone echo cancellation," *IEEE Trans. Commun.*, vol. 44, pp. 817–825, July 1996.
[151] J. G. Proakis, "Adaptive equalization for TDMA digital mobile radio," *IEEE Trans. Veh. Technol.*, vol. 40, pp. 333–341, May 1991.
[152] S. U. H. Qureshi, "Adaptive equalization," *Proc. IEEE*, vol. 73, pp. 1349–1387, Sept. 1985.
[153] S. Haykin, *Adaptive Filter Theory*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1991.
[154] C. S. H. Wong, J. C. Rudell, G. T. Uehara, and P. R. Gray, "A 50 MHz eight-tap adaptive equalizer for partial-response channels," *IEEE J. Solid-State Circuits*, vol. 30, pp. 228–233, Mar. 1995.
[155] G. Cherubini, S. Olcer, and G. Ungerboeck, "A quaternary partial-response class-IV transceiver for 125 Mbit/s data transmission over unshielded twisted-pair cables: Principles of operation and VLSI realization," *IEEE J. Select. Areas Commun.*, vol. 13, pp. 1656–1669, Dec. 1995.
[156] R. S. Kajley, P. J. Hurst, and J. E. C. Brown, "A mixed-signal decision-feedback equalizer that uses a look-ahead architecture," *IEEE J. Solid-State Circuits*, vol. 32, pp. 450–459, Mar. 1997.
[157] F. Lu and H. Samueli, "A 60 MBd, 480-Mb/s, 256-QAM decision-feedback equalizer in 1.2 μm CMOS," *IEEE J. Solid-State Circuits*, vol. 28, pp. 330–338, Mar. 1993.
[158] E. D. Man, M. Schulz, R. Schmidmaier, M. Schobinger, and T. G. Noll, "Architecture and circuit design of a 6-GOPS signal processor for QAM demodulator applications," *IEEE J. Solid-State Circuits*, vol. 30, pp. 219–226, Mar. 1995.
[159] H. Samueli, B. Daneshrad, R. B. Joshi, B. C. Wong, and H. T. Nicholas, "A 64-tap CMOS echo canceller/decision feedback equalizer for 2B1Q HDSL transceivers," *IEEE J. Select. Areas Commun.*, vol. 9, pp. 839–847, Aug. 1991.
[160] J. H. Shynk, "Frequency-domain and multirate adaptive filtering," *IEEE Signal Processing Mag.*, pp. 14–37, Jan. 1992.
[161] C. Leiserson and J. Saxe, "Optimizing synchronous systems," *J. VLSI Comput. Syst.*, vol. 1, no. 1, pp. 41–67, 1983.
[162] G. Long, F. Ling, and J. G. Proakis, "Corrections to 'the LMS algorithm with delayed coefficient adaptation,'" *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 40, pp. 230–232, Jan. 1992.
[163] M. Goel and N. R. Shanbhag, "Low-power adaptive filter architectures and their application to 51.84 Mb/s ATM-LAN," *IEEE Trans. Signal Processing*, vol. 45, pp. 1276–1290, May 1997.
[164] I. D. Yun and S. U. Lee, "On the fixed-point-error analysis of several fast DCT algorithms," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 3, pp. 27–41, Feb. 1993.
[165] P. P. Vaidyanathan and P.-Q. Hoang, "Lattice structures for optimal design and robust implementation of two-channel perfect-reconstruction QMF banks," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 36, pp. 81–94, Jan. 1988.
[166] I. K. Proudler, J. G. McWhirter, and T. J. Shepherd, "Computationally efficient QR decomposition approach to least squares adaptive filtering," *Proc. Inst. Elect. Eng.*, vol. 138, pt. F, pp. 341–353, Aug. 1991.
[167] H. M. Ahmed, "Alternative arithmetic unit architectures for VLSI digital signal processors," in *VLSI and Modern Signal Processing*, S. Y. Kung, H. J. Whitehouse, and T. Kailath, Eds. Englewood Cliffs, NJ: Prentice-Hall, 1985, ch. 16, pp. 277–303.
[168] J.-H. Hsiao, L.-G. Chen, T.-D. Chiueh, and C.-T. Chen, "High throughput CORDIC-based systolic array design for the discrete cosine transform," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, pp. 218–225, June 1995.
[169] S. Molloy, B. Schoner, A. Madisetti, R. Jain, and R. Matic, "An algorithm-driven processor design for video compression," in *Proc. IEEE Int. Conf. Image Processing*, Austin, TX, 1994, pp. III.611–615.
[170] A.-Y. Wu, K. J. R. Liu, and A. Raghupathy, "System architecture of an adaptive reconfigurable DSP computing engine," *IEEE Trans. Circuits Syst. Video Technol.*, to be published.
[171] L. B. Jackson, *Digital Filters and Signal Processing*, 2nd ed. Norwell, MA: Kluwer, 1989.

K. J. Ray Liu (Senior Member, IEEE), for a photograph and biography, see p. 753 of the May 1998 issue of this PROCEEDINGS.

An-Yeu Wu (Member, IEEE) received the B.S. degree from National Taiwan University, R.O.C., in 1987 and the M.S. and Ph.D. degrees from the University of Maryland, College Park, in 1992 and 1995, respectively, all in electrical engineering.

During 1987–1989, he was an Army Signal Officer, Taipei, Taiwan, for his mandatory military service. During 1990–1995, he was a graduate Teaching and Research Assistant with the Department of Electrical Engineering and Institute for Systems Research, University of Maryland. From August 1995 to July 1996, he was a Member of Technical Staff at AT&T Bell Laboratories, Murray Hill, NJ, working on high-speed transmission integrated circuit designs. He currently is an Associate Professor with the Electrical Engineering Department of National Central University, Taiwan. His research interests include low-power/high-performance very-large-scale-integration architectures for digital signal-processing applications, adaptive signal processing, and multirate signal processing.

Arun Raghupathy (Student Member, IEEE) received the B.Tech. degree in electronics and communications engineering from the Indian Institute of Technology, Madras, in 1993 and the M.S. degree in electrical engineering from the University of Maryland, College Park, in 1995, where he currently is pursuing the Ph.D. degree.

His research emphasis is on the design of low-power/high-performance very-large-scale-integration signal-processing systems. His other interests include development of systems for digital communications and multimedia applications.

Jie Chen (Student Member, IEEE) 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. He received the M.S. degree in electrical engineering from the University of Maryland, College Park, in 1992, where he currently is pursuing the Ph.D. degree.

During 1992–1995, he was with Hughes Network System research group, Germantown, MD, and participated in E-TMDA and TDMA system performance testing. He was involved in the research and development for fixed wireless telephone/fax systems and patented one design. His current doctoral research is on the low-complexity, low-power design and very-large-scale-integration implementation of video coders. His research interests include multimedia signal processing and digital communication system design.