Algorithm-Based Low-Power Digital Signal Processing System Designs

by A-Y. Wu
Advisor: K.J.R. Liu

Ph.D. 95-6

Algorithm-Based Low-Power Digital Signal Processing System Designs

by

An-Yeu (Andy) Wu

Dissertation submitted to the Faculty of the Graduate School
of The University of Maryland in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
1995

Advisory Committee:

Professor K. J. Ray Liu, Chairman/Advisor
Professor Rama Chellappa
Professor Kazuo Nakajima
Professor Dianne O'Leary
Professor Steve Tretter

ABSTRACT

Title of Dissertation: Algorithm-Based Low-Power Digital Signal Processing System Designs

An-Yeu (Andy) Wu, Doctor of Philosophy, 1995

Dissertation directed by: Professor K. J. Ray Liu
Department of Electrical Engineering

In most low-power VLSI designs, the supply voltage is usually reduced to lower the total power consumption. However, the device speed will be degraded as the supply voltage goes down. In order to meet the low-power/high-throughput constraint, the key issue is to "compensate" the increased delay so that the device can be operated at the slowest possible speed without affecting the system throughput rate.

In this dissertation, new algorithmic-level techniques for compensating the increased delays based on the multirate approach are proposed. Given the digital signal processing (DSP) problems, we apply the multirate approach to reformulate the algorithms so that the desired outputs can be obtained from the decimated input sequences. Since the data rate in the resulting multirate architectures is M-times slower than the original data rate while maintaining the same throughput rate, the speed penalty caused by the low supply voltage is compensated at the algorithmic/architectural level.

This new low-power design technique is applied to several important DSP applications. The first one is a design methodology for the low-power design of FIR/IIR systems. By following the proposed design procedures, users can convert a speed-demanding system function into its equivalent multirate transfer function. This methodology provides a systematic way for VLSI designers to design low-power/high-speed filtering architectures at the algorithmic/architectural level.

The multirate approach is also applied to the low-power transform coding architecture design. The resulting time-recursive multirate transform architectures inherit all advantages of the existing time-recursive transform architectures such as local communication, regularity, modularity, and linear hardware complexity, but the speed for updating the transform coefficients becomes M-times slower.

The last application is a programmable video co-processor system architecture that is capable of performing FIR/IIR filtering, subband filtering, discrete orthogonal transforms (DT) and adaptive filtering for the host processor in video applications. The system can be easily reconfigurated to perform multirate FIR/IIR/DT operations. Hence, we can either double the processing speed on-the-fly based on the same processing elements, or apply this feature to the low-power implementation of this co-processor.

The methodology and the applications presented in this dissertation constitute a design framework for achieving low-power consumption at the algorithmic/architectural level for DSP applications.

Dedication

To my parents, Chang-Tien Wu and Ling-Bo Hsu.

To my dear wife, Ju-Hsing Li

for her patience, support, and love during writing of this work.

Acknowledgements

I would like to express my gratitude to my advisor, Dr. Ray Liu for his guidance, encouragement, and support throughout the course of my graduate studies. His insightful advice always inspired me when I faced difficulties in research and personal life. My time with him has been a rewarding unforgetable experience.

I am grateful to those professors who have taught me in the graduate courses. Special thanks go to Dr. Ahmet Y. Oruc, Dr. Linda Milor, Dr. Rama Chellappa, and Dr. Kazuo Nakajima. They helped me to build most of the necessary background for my research projects.

I would also like to thank my friends in the Digital Signal Processing Laboratory for their assistance and numerous interesting discussions, especially, Dr. Ye Li, Ut-Va Koc, Hong-Yi Wang, Vishnuuss Srinivasan, Arun Raghupathy, and Shang-Chieh Liu. Also, I am would like to acknowledge Zhongying Zhang of VLSI Design Automation Laboratory for providing his low-power chip designs.

Finally, I thank my family and friends for their constant love and support.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Low-power VLSI design has emerged as a major theme in the electronics industry today. One reason is due to the growing markets in portable computing and communication systems. In the past, most research and development efforts focused on increasing the processing speed and reducing the complexity of the chip design. The power consumption of the chip, on the other hand, is given lower priority during the design phase. The scenario has been changed since the advent of personal communications/computing services (PCS). The common feature of the PCS devices is that they demand high-speed data/signal processing, which leads to much higher power consumption than traditional portable applications such as wrist watches and hand-held calculators. Nevertheless, we have only limited power-supply capability of current battery technology. Hence, we are motivated to consider low-power design so as to prolong the operating time of those PCS devices.

The other reason for low-power design arises from the power dissipation problem. As the clock rate and silicon area of IC designs increase, power dissipation of some individual IC components has not only reached the limits of current packaging technology but also affected the reliability/yield in the fabrication process. As an example, the DEC ALPHA 64-bit CPU [1] consumes up to 30 Watts at a clock rate of 200 MHz. The high power dissipation calls for extra cooling systems such as cooling fins and fans, expensive packages, to dissipate the generated heat. As a result, both weight and cost of the system will be increased. This becomes another driving force for the study of low-power VLSI design.

In general, low-power VLSI design can be achieved at all levels of the VLSI system (system, algorithm, architecture, circuit, logic, device, and technology levels). In this dissertation, we focus on developing a new algorithmic/architectural-level low-power design technique based on the **multirate approach**. We apply it to several important digital signal processing (DSP) applications, which leads to new multirate VLSI architectures that can achieve significant power saving compared with the normal design while retaining the same data throughput rate.

The organization of this chapter is as follows. In Section 1.1, current low-power design approaches at different VLSI levels are described. In Section 1.2, we address the motivation of using the multirate approach in lowering the power consumption at the algorithmic/architectural level. In Section 1.3, we give an overview of the results that we obtained by applying this new low-power design technique to the following DSP tasks: general FIR/IIR filtering, transform coding kernel design, and video co-processor design. We conclude with the dissertation organization in Section 1.4.

1.1 Low-Power VLSI Design Approaches

The power dissipation in a well-designed digital CMOS circuit can be modeled as [2]

$$P \approx \alpha \cdot C_{eff} \cdot V_{dd}^2 \cdot f_{clk}, \tag{1.1}$$

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

$$T_D \approx \frac{C_{eff} \times V_{dd}}{I} = \frac{C_{eff} \times V_{dd}}{\mu C_{ox}(W/L)(V_{dd} - V_t)^2}, \tag{1.2}$$

where μ, C_{ox}, W, L are the device parameters and V_t is the threshold voltage of the devices. (1.1) and (1.2) 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, whereas T_D is not sacrificed for those parameter changes [3][4][5][6][7]. The existing low-power design approaches are summarized below.

- **Device/VLSI technology level:** Over the last decade, the CMOS feature size has been reduced from 2 μm to 0.35 μm. The advance in IC fabrication technology also leads the way to low-power design. Smaller transistor size not only improves the device/circuit speed performance, it also reduces the total silicon area and capacitances, hence the total power consumption. Besides, the increased level of integration allows the designer to use the vacated area for extra circuits to compensate for the device speed reduction due to lower supply voltage (as we will discuss it 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 [4][8]. From (1.1), we can see that the delay of the circuit is inversely proportional to $(V_{dd} - V_t)^2$. 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 1.5V (or lower) operation will be needed by the year 2001 for portable product [6]. Since the supply voltage of the conventional scaled CMOS technology will reach its limit at a supply voltage of 1.5 V, an alternate process technology, silicon-on-insulator (SOI), is suggested to replace the CMOS technology [6][9]. The SOI technology allows power supply reduction to 1V or less and also greatly simplifies the fabrication process. Those merits have made SOI the best candidate for future low-power fabrication technology. In general, the cost of the technology/device approach is most expensive among all low-power techniques since it requires the investment of new semiconductor equipment and technology.

- **Circuit approach:** There are numerous options available in choosing the basic circuit approach and topology for implementing the given logic and arithmetic functions. As an example, we can employ several design approaches such as carry-ripple, carry-look-ahead, and carry-select to realize the adder circuit [10]. Each approach renders different trade-off in the performance of power/speed/area. At the CMOS circuit level, various circuit design techniques are available; *e.g.*, dynamic versus static CMOS logic, conventional static versus pass-transistor logic, and synchronous versus asynchronous design [3]. As far as power consumption is concerned, the static CMOS logic and asynchronous design are preferable due to their less number of switching activities. The pass-transistor logic family is also a promising candidate since it uses a less number of transistors than the conventional static CMOS circuits for implementing the same logic function [2, Chap.5].

  Recently, the low-power digital circuits based on adiabatic-switching technique was introduced [11]. 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.

- **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-based low-power design, the major focus is to reduce the frequency of energy consuming transitions for given logic functions, *i.e.*, the activity factor α in (1.1). Existent approaches can be found in

[12][13][14][15]. 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 inputs signals. By doing so, α can be reduced, hence the total power consumption of the circuits. In general, the power saving of the logic approach is in the range of 20%-75%.

- **Architectural/algorithmic approach:** From (1.1), we can see that the reduction of the supply voltage is the leveraged way to reduce the total power consumption due to its quadratic dependence. However, the delay will drastically increase as V_{dd} approaches V_t (see (1.2)). That is, we suffer from a **speed penalty** as V_{dd} goes down. In order 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" the increased delay caused by the lowered supply voltage. Current approaches for compensating the increased delay include the techniques of "parallel processing" and "pipelining" [3][7]. In this dissertation, we propose a new compensation technique based on the multirate approach, which will be discussed in details in the next section.

- **System-level approach:** The system-level low-power VLSI design evolves from the power-saving techniques which are frequently used in lap-top/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 the system power. Recent state-of-the-art CPU designs have employed the same design concept to manage both dynamic and static power of the CPU [16]. The embedded activity management circuit of the CPU provides the capability to shutdown 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 design. 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 to other submodules can be minimized. For ASIC designs that use multichip modules (MCMs) implementation, the partitioning can be done by employing the systematic approach discussed in [17]. 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 [7]. Firstly, the algorithmic/architectural low-power design is achieved by reformulating the algorithms and mapping them to efficient low-power VLSI architectures to compensate the speed penalty caused by low supply voltage. Basically, we only trade more chip area for low power consumption under current technology, without invoking dedicated circuit design, new expensive device materials, 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. Secondly, the power saving of the the algorithmic/architectural approach is in the range of 70%-90% [1] (as we will show it in this dissertation). 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.

1.2 Algorithmic/Architectural-level Low-Power Design Using the Multirate Approach

The architectural-level low-power design was first proposed by Chandrakasan *et al.* [3]. In [3], the techniques of "parallel processing" and "pipelining" were suggested to com-

[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 [6].

pensate 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. However, in most DSP applications, 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 the loss of performance under low-power operations. The main issue here is to reformulate the algorithms so that the desired output can be obtained without hindering the system performance such as data throughput rate. We call such an approach the **algorithm-based low-power design**.

In this dissertation, we propose a new technique–the multirate approach–to compensate the aforementioned speed penalty. To motivate the idea, let us consider the discrete cosine transform (DCT) architecture in Fig. 1.1. For most of the existing serial-input-parallel-output (SIPO) DCT architectures [18][19], the processing rate of the operators must be as fast as the input data rate (see Fig. 1.1(a)). In our low-power design, the DCT is computed from the reformulated circuit using the decimated sequences (Fig. 1.1(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_{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 5 V to 2.9 V [3]. Using the CMOS power dissipation model of (1.1), the overall power consumption of the multirate design can be estimated as

$$(2C_{eff})(\frac{2.9V}{5V})^2(\frac{1}{2}f) \approx 0.34P_0, \tag{1.3}$$

where P_0 denotes the power consumption of the original system. Therefore, the multirate approach provides a direct and efficient way for the low-power design at the algorith-

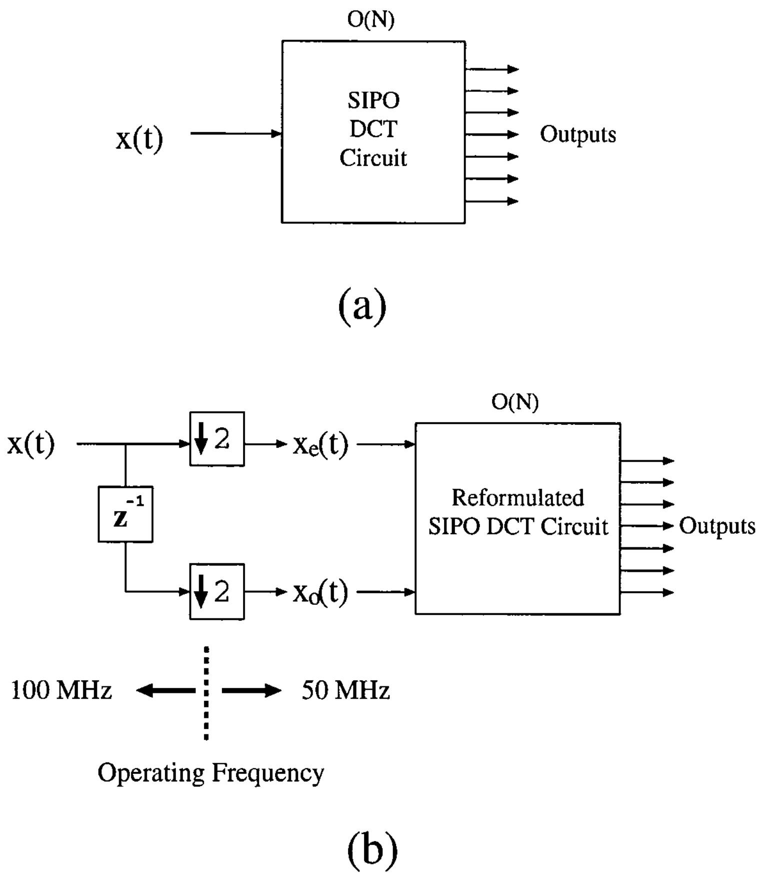


Figure 1.1: (a) Original SIPO DCT circuit. (b) Low-power DCT circuit using the multirate approach.

mic/architectural level.

1.3 Main Contributions

Based on the proposed new algorithmic/architectural-level low-power design technique, several significant results are developed in this dissertation and are summarized as follows:

1. **Design methodology for multirate FIR/IIR filtering architectures:** We present a design methodology for the low-power design of any given FIR/IIR DSP systems. 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 (where M is a positive integer) than the original data rate while maintaining the same throughput rate, we can apply this feature to either the low-power implementation, or the speed-up of the DSP systems. The proposed design methodology provides VLSI designers 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. **Low-power transform coding architecture 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/IDCT 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. We also consider the design of low-power architectures that can lower the power consumption with only $O(\log M)$ increase in hardware complexity. The multirate DCT/IDCT design is 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. Moreover, we perform the finite-precision analysis of the DCT architectures under the normal and multirate operations. Using the analytical results, we can choose the optimal wordlength for each DCT channel under the predetermined signal-to-noise ratio (SNR) constraint. Hence, the total number of switching events and the silicon

area are further reduced, and so is the power consumption of the DCT chip. These design issues constitute a framework of the algorithm-based low-power design with an application to transform coding kernel design.

3. **Video co-processor design:** We present a programmable video co-processor system architecture for numerically intensive front-end video/image data processing. 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), and adaptive filtering for the host processor in video applications. Since the properties of each programmed function such as parallelism and pipelinability have been fully exploited in this design, the computational speed of this co-processor can be as fast as that of ASIC designs which are optimized for individual specific applications. We also show that the system can be easily reconfigurated to perform multirate FIR/IIR/DT operations with negligible hardware overhead. Hence, we can either double the processing speed on-the-fly based on the same processing elements, or apply this feature to the low-power implementation of this co-processor. The programmable/high-speed properties of the proposed design make it very suitable for cost-effective video-rate applications.

In addition to low-power implementation, the other attractive application of 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 applications such as HDTV, the speed constraint will result in the use of expensive high-speed multiplier/adder circuits or full-custom designs. Thus, the manufacturing cost as well as the design cycle will increase drastically. Since the 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. Hence, by employing the multirate parallel architectures discussed in this dissertation, the above-mentioned speed constraint can be resolved at the architectural level under the same design environment and fabrication technology.

1.4 Thesis Organization

In this dissertation, we propose new compensation techniques based on the multirate approach. The effectiveness of this algorithm-based low-power design is demonstrated by applying it to several DSP problems.

In Chapter 2, the design methodology for deriving low-power FIR/IIR filtering architectures is presented. The design and implementation issues of the low-power QMF chips as well as the simulation results are also discussed in this section.

In Chapter 3, the multirate DCT/IDCT VLSI architectures are derived based on two different approaches. One is the Chebyshev polynomial approach and the other is the polyphase approach. The comparison of the proposed multirate architectures with existing DCT architectures is also considered.

In Chapter 4, we present multirate transform coding architectures that can achieve low-power consumption with only $O(\log_2 M)$ hardware overhead. We also extend the multirate DCT/IDCT design to a unified transform coding architecture. It can perform most of the existing orthogonal transforms based on the same processing elements.

In Chapter 5, we perform the finite-wordlength analysis for the proposed multirate DCT architectures. The rounding errors and dynamic range of the VLSI architectures under fixed-point arithmetic are analyzed,

In Chapter 6, we present the system architecture of the video co-processor that is capable of performing various tasks in video applications. The speed-up of the system based on the multirate approach is also considered.

In Chapter 7, we conclude our work and discuss the future research topics that can be extended from this dissertation.

Chapter 2

Design Methodology for Multirate FIR/IIR Systems

FIR/IIR filtering is the most commonly used operation in DSP systems. In general, the direct implementation of the system transfer function $H(z)$ (see Fig. 2.1(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 the speed penalty under low supply voltage. On the other hand, the multirate system in Fig. 2.1(b) requires only low-speed processing elements at one-third of the original clock rate to maintain the same throughput. Therefore, it can be used to compensate the speed penalty at the architectural level. In this chapter, we present a systematic approach for the low-power design of general linear time-invariant (LTI) FIR/IIR systems based on the multirate approach. By following the design methodology presented in this chapter, users can easily convert a speed-demanding FIR/IIR system function into an equivalent multirate transfer function. The resulting multirate filtering architecture is suitable for either the low-power implementation or the processing speed-up of the DSP system.

We also design the low-power Quadrature Mirror Filter (QMF) [20] by using the proposed multirate FIR structure. We consider the architectural design issues for the QMF such as the employment of power-of-two coefficients to reduce the total chip area

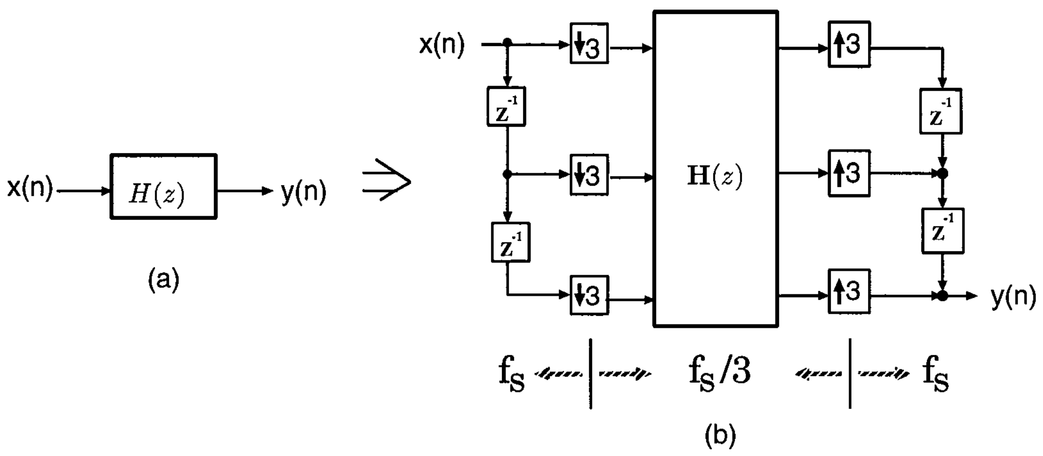


Figure 2.1: (a) An LTI FIR/IIR system. (b) Its equivalent multirate implementation.

and the choice of most suitable system wordlength for the design. The simulation results of the implemented chips confirm the parameter values that we use to estimate the power saving of the multirate design.

The organization of this chapter is as follows: In Section 2.1, we first review some basic multirate operations that are useful for our derivations. In Section 2.2, we present the methodology to derive the equivalent multirate implementation of an arbitrary LTI system. In Section 2.3, the diagonalization approaches to eliminate the global communications of the multirate implementation are discussed. In Section 2.4, the power estimation of the multirate FIR filtering architecture is considered. The design and simulation results of the QMF VLSI chips are discussed in Section 2.5,

2.1 Basic Multirate Operations

Given an FIR transfer function

$$H(z) = \sum_{n=-\infty}^{\infty} h(n)z^{-n} \qquad (2.1)$$

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

$$H(z) = \sum_{l=0}^{M-1} z^{-l} E_l(z^M) \tag{2.2}$$

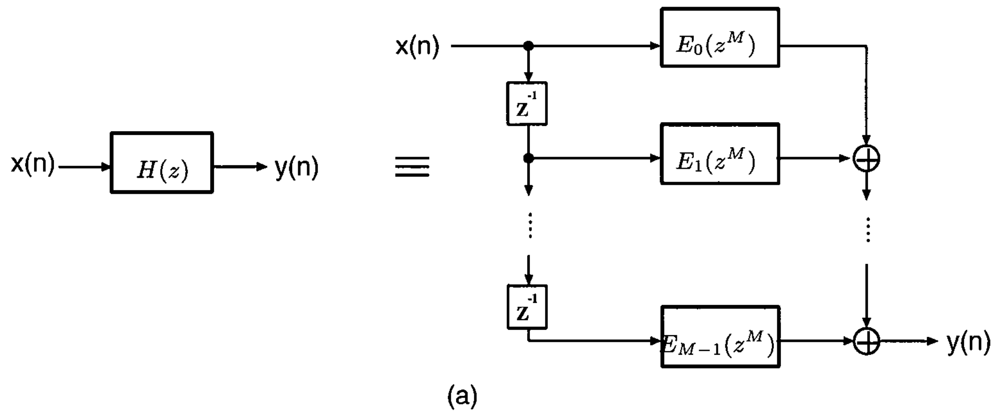
where

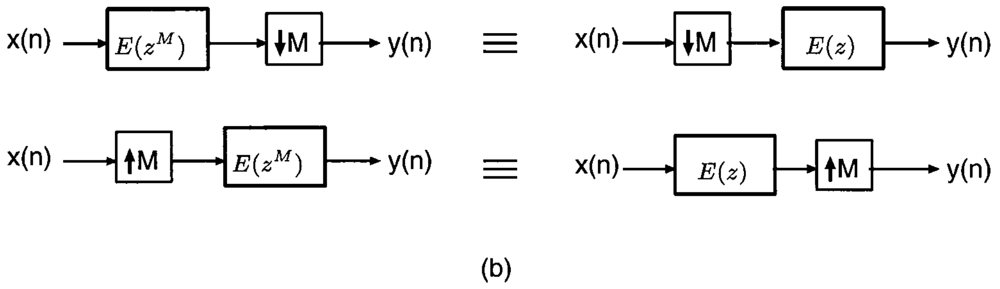
$$E_l(z) = \sum_{n=-\infty}^{\infty} e_l(n) z^{-n} \tag{2.3}$$

with $e_l(n) \triangleq h(Mn + l)$, $0 \leq l \leq M - 1$. (2.2) is referred to as the **Type I polyphase representation** with respect to M, and $E_l(z)'s$, $l = 0, 1, \ldots, M-1$, are the **polyphase components** of $H(z)$ [21]. Figure 2.2(a) shows an implementation of $H(z)$ based on the polyphase representation.

The second commonly used multirate operation is the **noble identities** (Fig. 2.2(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 third property that is useful for our derivation is the equivalent implementation of an $(M - 1)$-delay element as shown in Fig. 2.2(c). The multirate structure at the right is also known as "the delay chain perfect reconstruction system" [21, Chap.5].

he above basic multirate operations provide very efficient tools in the theory and implementation of multirate systems [21]. Take the decimation circuit depicted in Fig. 2.3(a), for example. The decimation filter $H(z)$ is in general a low-pass FIR filter preventing the aliasing effect after the decimation operation. The direct implementation of Fig. 2.3(a) requires N multipliers and N adders. It has the disadvantage that the operating frequency of the processing elements must be as fast as the input data rate. Instead, we can have a more efficient implementation of Fig. 2.3(a) by applying the above-mentioned multirate techniques: Suppose that $M = 3$. We first replace the transfer function $H(z)$ in Fig. 2.3(a) with that in Fig. 2.2(a), which results in Fig. 2.3(b). Then, we move the decimation operation towards the middle of the parallel paths (see Fig. 2.3(c)). After applying the noble identity, we have the polyphase implementation of the decimation circuit as shown in Fig. 2.3(d). Although the total cost is still N





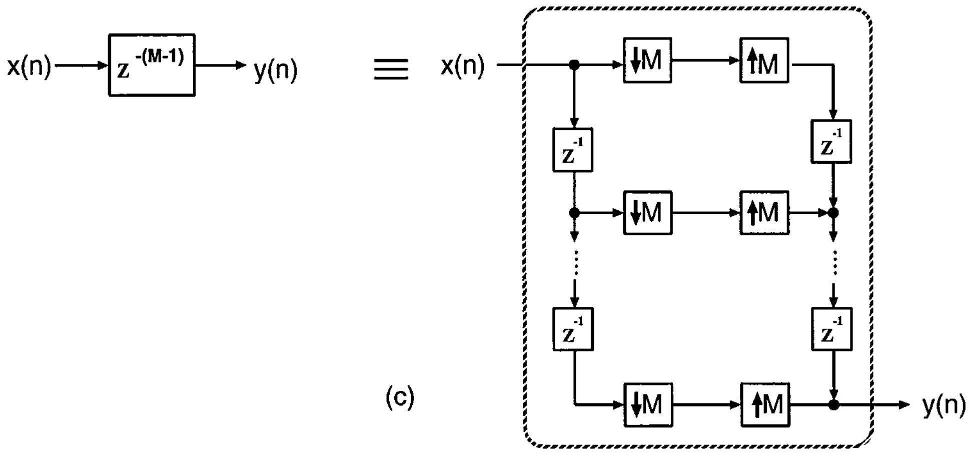


Figure 2.2: Basic multirate operations: (a) Polyphase decompositon. (b) Noble identities. (c) Equivalent multirate implementation of an $(M-1)$ delay element.

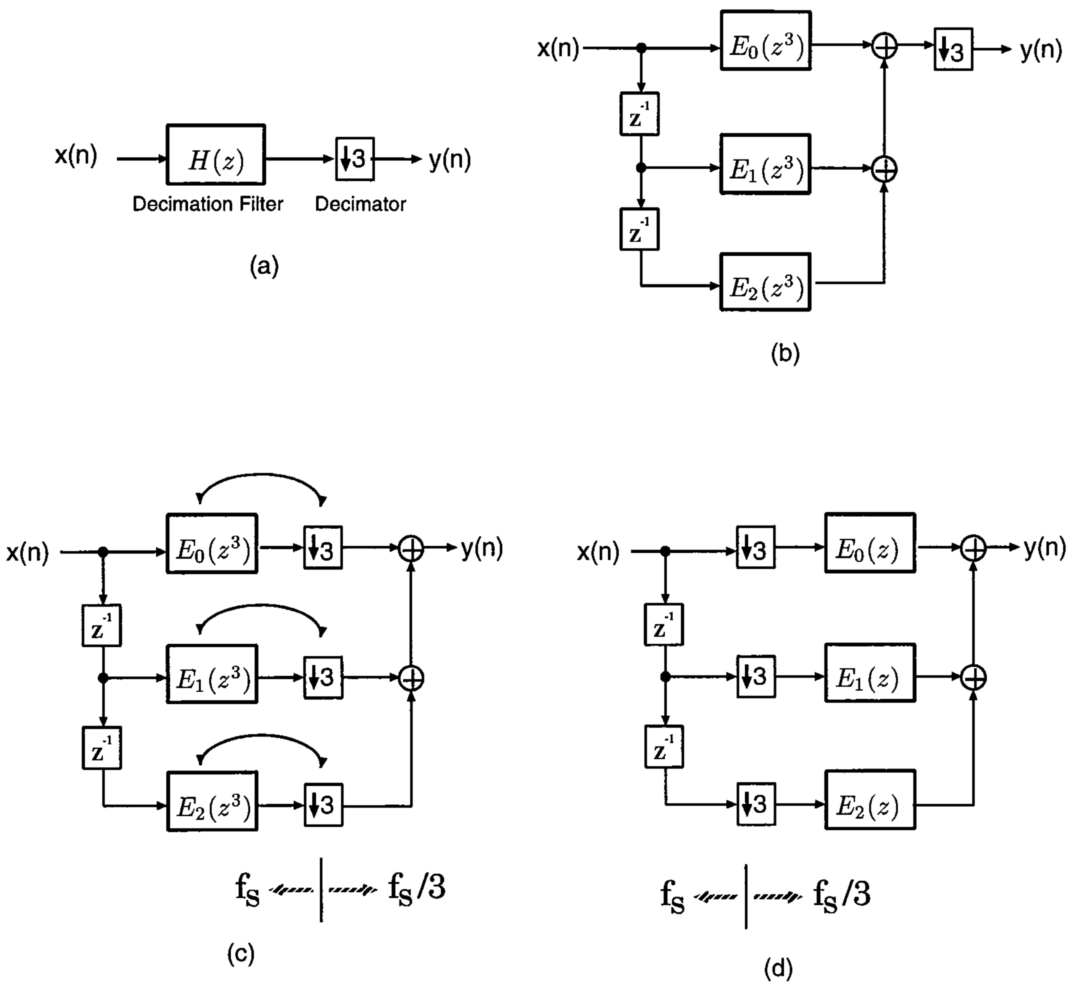


Figure 2.3: Derivation of the polyphase implementation of the decimation circuit, where f_s denotes the data sample rate. (a) The original decimation circuit. (b) Representing the decimation filter using Fig. 2.2(a). (c) Applying the noble indentity. (d) The resulting polyphase implementation.

multipliers and N adders, the operating frequency of the operators has been reduced to only one-third of the original clock rate.

Next let us consider a decimation circuit with transfer function

$$z^{-1}H(z) = h(0)z^{-1} + h(1)z^{-2} + \cdots + h(N-1)z^{-N}. \tag{2.4}$$

We first rewrite (2.4) as

$$z^{-1}H(z) = z^{0}(z^{-3}E_2(z^3)) + z^{-1}E_0(z^3) + z^{-2}E_1(z^3). \tag{2.5}$$

As with Fig. 2.3, we can derive the polyphase implementation of the decimation circuit with transfer function $z^{-1}H(z)$ (see Fig. 2.4). The resulting architecture is similar to that of Fig. 2.3(d) except that the polyphase components $E_i(z)$'s have been rotated and one extra delay element is added to $E_2(z)$. Likewise, given the transfer function $z^{-2}H(z)$, we have

$$z^{-2}H(z) = z^{0}(z^{-3}E_1(z^3)) + z^{-1}(z^{-3}E_2(z^3)) + z^{-2}E_0(z^3) \tag{2.6}$$

and the corresponding decimation circuit is obtained as in Fig. 2.5.

2.2 Multirate Design Methodology

In what follows, we present the design methodology to derive the multirate FIR system of Fig. 2.1. Without loss of generality, we assume that $M = 3$ in our derivation. The results can be easily extended for an arbitrary M.

The Design Procedure

Given an LTI FIR system $H(z)$ with order N and decimation factor M, the design procedure is summarized as follows (see Fig. 2.6).

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

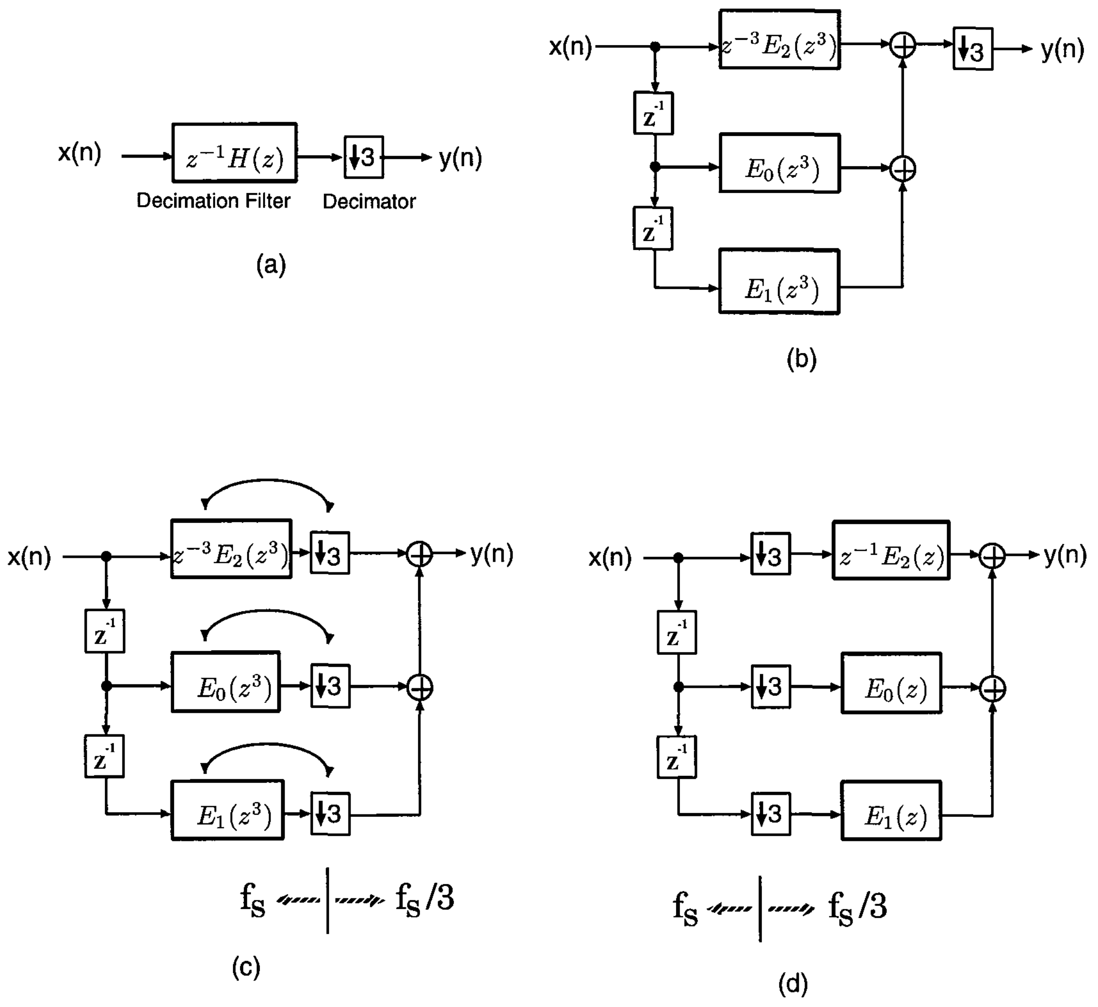


Figure 2.4: Derivation of the polyphase implementation of the decimation circuit with transfer function $z^{-1}H(z)$.

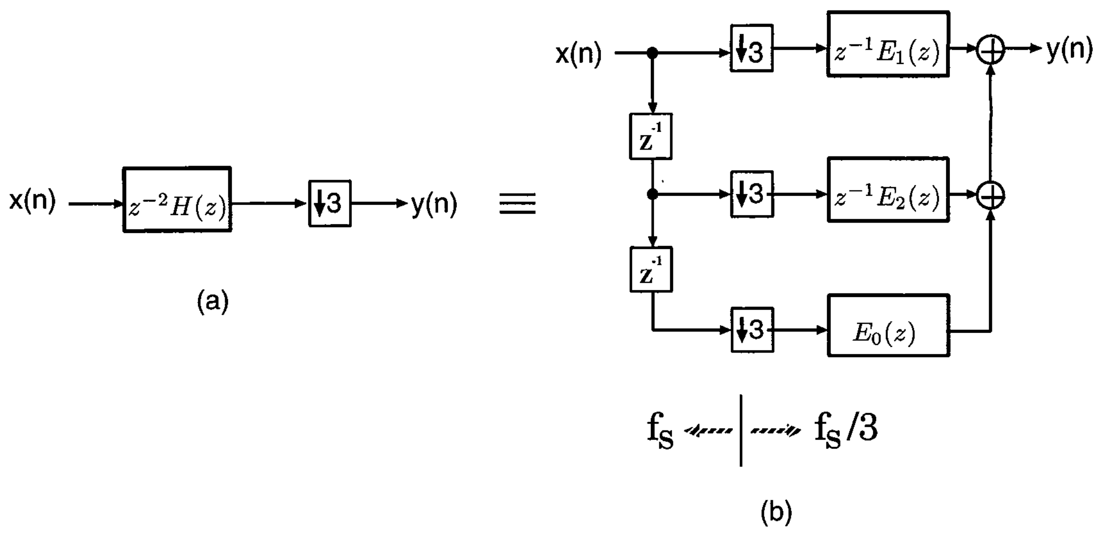


Figure 2.5: The decimation circuit with transfer function $z^{-2}H(z)$.

Step(b): Apply the identity of Fig. 2.2(c) for the inserted delay element, which results in Fig. 2.6(b).

Step(c): Move $H(z)$ to the right till 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): Substitute each circled diagram shown in Fig.2.6(d) with the results of Figs. 2.3-2.5. Then we have Fig. 2.6(e).

Step(f): Note that the data inputs at points designated by **a** are the same, and so are those at points **b** and **c**. After merging the common data paths in Fig. 2.6(e), we obtain Fig. 2.6(f) in which

$$\mathbf{E}(z) \triangleq \begin{bmatrix} E_0(z) & E_1(z) & E_2(z) \\ z^{-1}E_2(z) & E_0(z) & E_1(z) \\ z^{-1}E_1(z) & z^{-1}E_2(z) & E_0(z) \end{bmatrix}. \qquad (2.7)$$

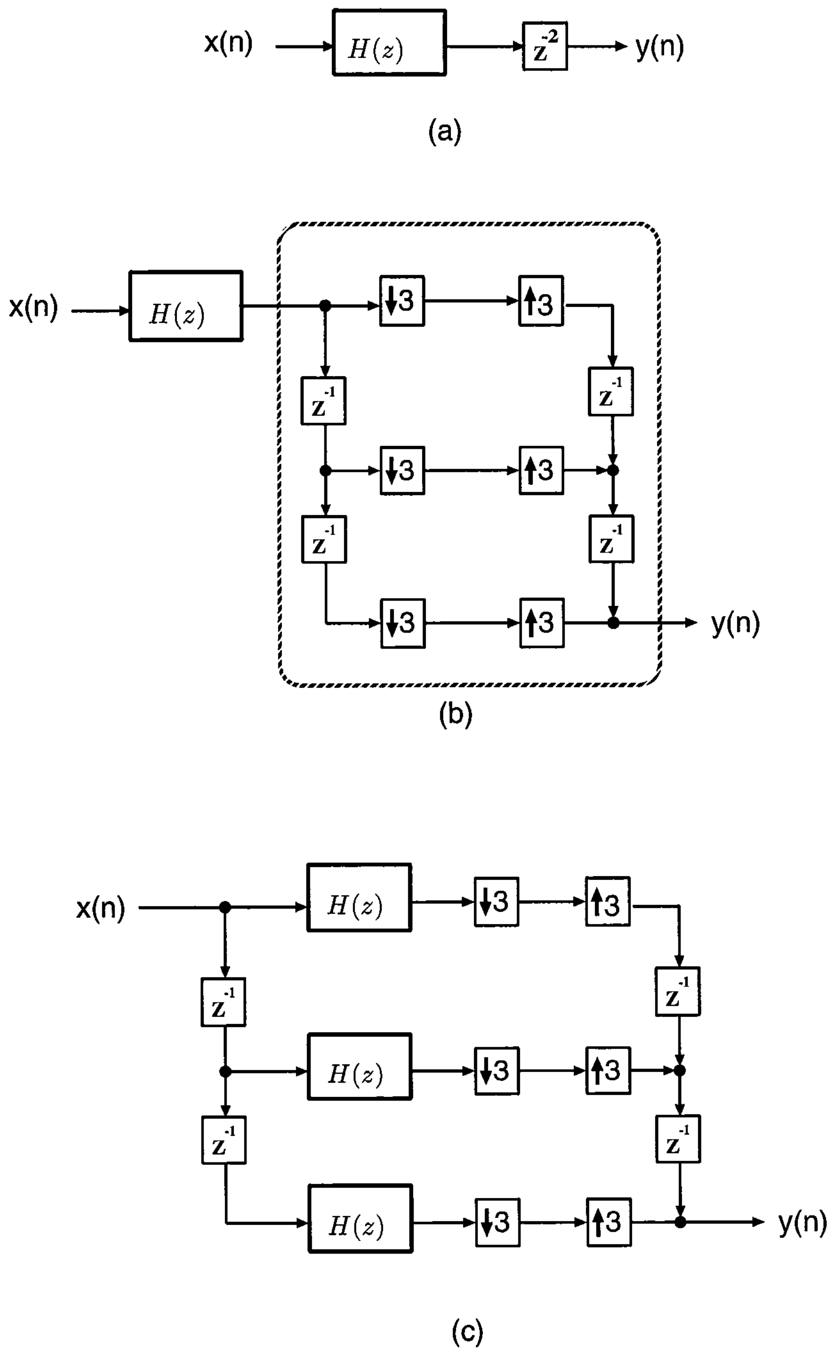
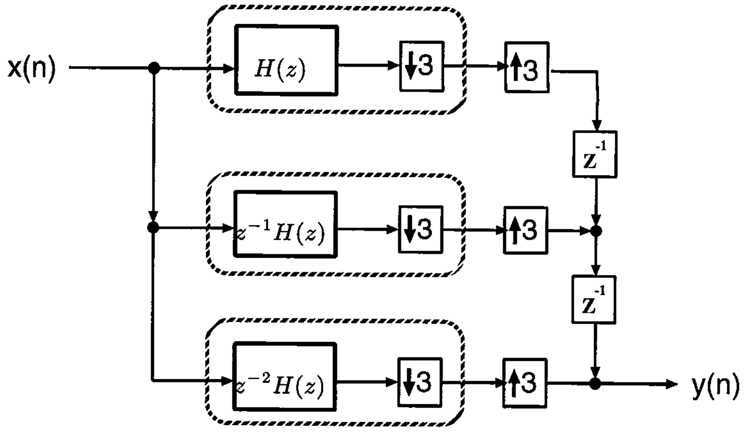
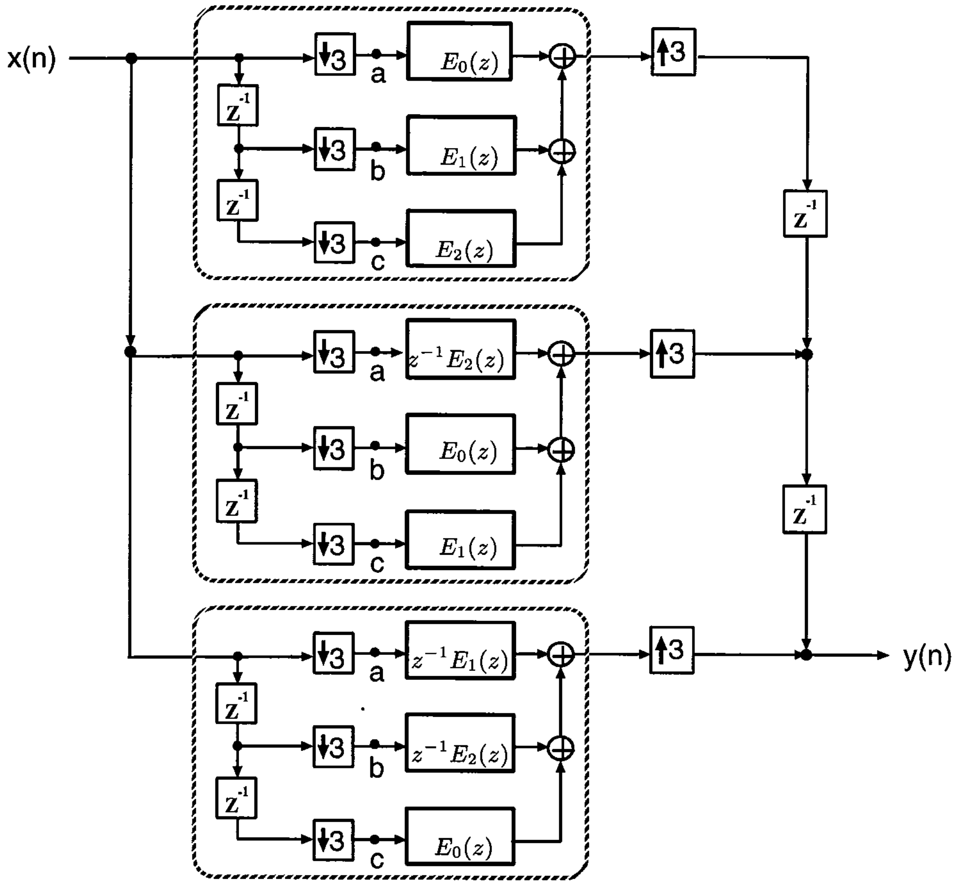


Figure 2.6: Step-by-step derivation of the multirate LTI system.

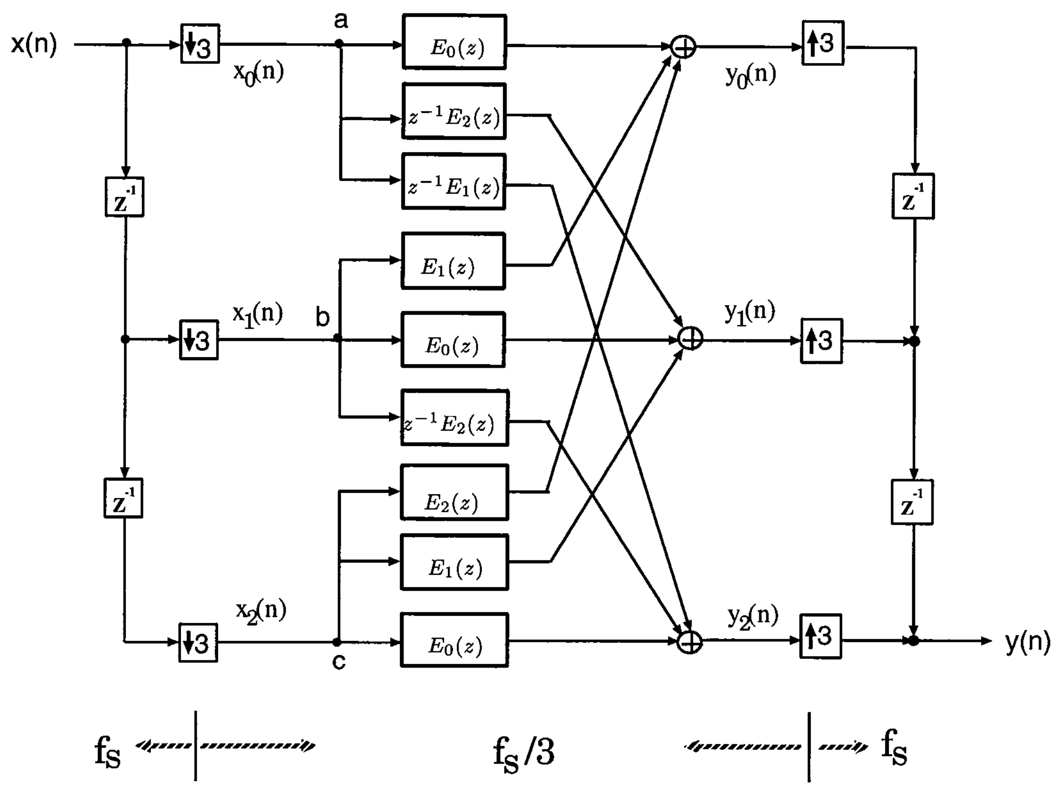


(d)



(e)

Figure 2.6: (cont.)



(f)

Figure 2.6: (cont.)

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

$$\mathbf{E}(z) = \begin{bmatrix} E_0(z) & E_1(z) & \cdots & E_{M-1}(z) \\ z^{-1}E_{M-1}(z) & E_0(z) & \cdots & E_{M-2}(z) \\ \vdots & \vdots & \ddots & \vdots \\ z^{-1}E_1(z) & z^{-1}E_2(z) & \cdots & E_0(z) \end{bmatrix}, \tag{2.8}$$

which is also known as the **pseudocirculant matrix** in the context of alias-free QMF filter banks [22].

The design steps described in Fig. 2.6 provide a systematic way to design a low-power FIR system. Since each $E_i(z)$ in Fig. 2.6(f) represents a subfilter of order N/M, 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.

2.2.1 Low-Power Multirate IIR System

The above methodology can also be applied to multirate IIR system design. We first compute the polyphase components $E_i'(z)'s$, $i = 0, 1, \ldots, M-1$, of the given IIR function

$$H'(z) = \frac{N(z)}{D(z)} = \frac{1 + \sum_{i=1}^{P} p_i z^{-i}}{1 + \sum_{i=1}^{Q} q_i z^{-i}} \tag{2.9}$$

as follows. For each pole of $H'(z)$, we introduce $(M-1)$ extra pole-zero pairs that are of equal angular spacing and have the same radius as that of the original pole. Then $H'(z)$ can be rewritten as [23]

$$H'(z) = \frac{N(z) \prod_{k=1}^{M-1} D(ze^{j2\pi k/M})}{D(z) \prod_{k=0}^{M-1} D(ze^{j2\pi k/M})} = \frac{N'(z)}{D'(z^M)}. \tag{2.10}$$

Let the Type I polyphase representation of $N'(z)$ be

$$N'(z) = \sum_{l=0}^{M-1} z^{-l} G'_l(z^M). \tag{2.11}$$

The polyphase decomposition of $H'(z)$ can be represented as

$$H'(z) = \sum_{l=0}^{M-1} z^{-l} E'_l(z^M) \tag{2.12}$$

with

$$E'_l(z^M) \triangleq \frac{G'_l(z^M)}{D'(z^M)}. \tag{2.13}$$

Note that the multiplication complexity of $D'(z^M)$ is Q, and that of $N'(z)$ is $(P + (M-1)Q)$. Hence, we need a total complexity of $(P+(2M-1)Q)$ multipliers for the polyphase implementation of $H'(z)$.

After replacing each $E_i(z)$ in Fig. 2.6 with its corresponding $E'_i(z)$, for $i = 0, 1, \dots, M-1$, we can use the methodology of Fig. 2.6 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 complexity for the implementation of the multirate IIR filter.

2.3 Diagonalization of the Pseudocirculant Matrix

Although the multirate implementation of Fig. 2.6(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 (2.8) so as to eliminate global communication in the multirate implementation.

One way to diagonalize $\mathbf{E}(z)$ of (2.8) is to use the DFT approach [24]. For example, the multirate system of Fig. 2.6(f) can be diagonalized by using 6-point DFT/IDFT networks as depicted in Fig. 2.7. This DFT requires complex-number operations for the filtering operations of those subfilters $D_i(z)$'s. Besides, it still requires global communication in the DFT/IDFT networks.

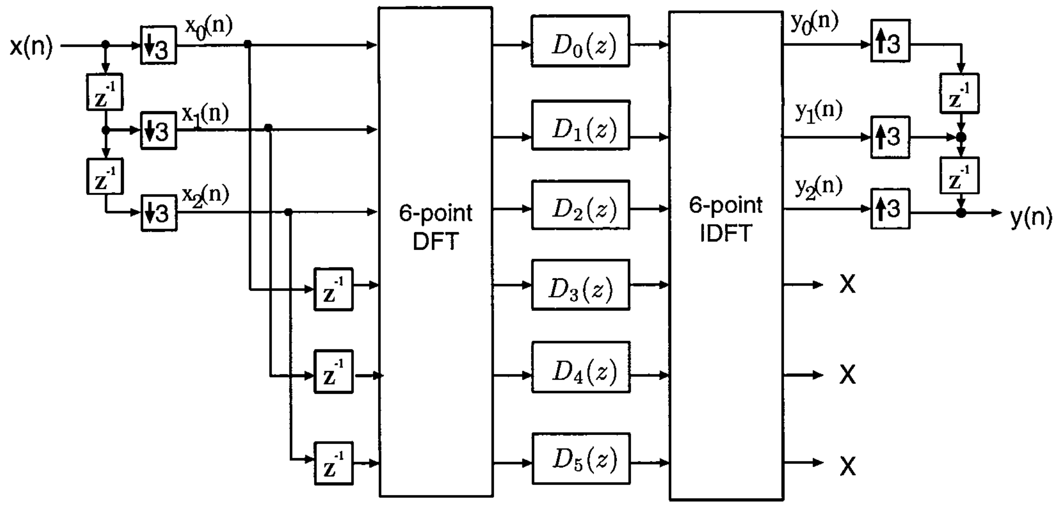


Figure 2.7: Diagonalization of the pseudocirculant matrix using the FFT approach.

Besides the DFT approach, the diagonalization approaches based on polynomial convolution techniques have been studied in [25][26]. As an example, for the case of the decimation factor equal to two, (2.8) can be rewritten as [26]:

$$\begin{bmatrix} E_0(z) & E_1(z) \\ z^{-1}E_1(z) & E_0(z) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}}_{\mathbf{B}} \begin{bmatrix} E_0(z) & 0 & 0 \\ 0 & E_0(z)+E_1(z) & 0 \\ 0 & 0 & E_1(z) \end{bmatrix} \underbrace{\begin{bmatrix} 1 & -1 \\ 0 & 1 \\ z^{-1} & -1 \end{bmatrix}}_{\mathbf{A}}. \tag{2.14}$$

The resulting structure is depicted in Fig. 2.8, where the downsampling circuit and the upsampling circuit are used to realize the pre-processing matrix **A** and the post-processing matrix **B**, respectively, of (2.14). 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) [26]. In the next section, we will use the multirate FIR structure of Fig. 2.8 for our low-power Quadrature Mirror Filter (QMF) chip design.

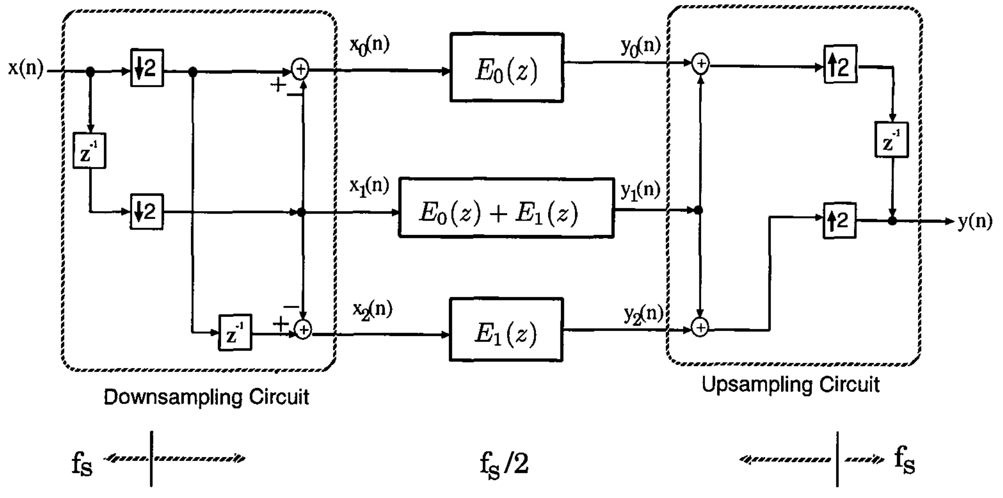


Figure 2.8: Multirate FIR architecture with $M = 2$.

2.4 Power Estimation for the Multirate FIR Architecture

From (1.2), it can be shown that the lowest possible supply voltage V'_{dd} for a device running at an M-times slower clock rate can be approximated by

$$\frac{V'_{dd}}{(V'_{dd}-V_t)^2} = M\frac{V_{dd}}{(V_{dd}-V_t)^2}, \tag{2.15}$$

where V_t is the threshold voltage of the device.

Assume that $V_{dd} = 5V$ and $V_t = 0.7V$ in the original system. For the normal FIR architecture, it requires N multipliers and N adders. For the low-power multirate FIR architecture depicted in Fig. 2.8 (where $M = 2$), $3N/2$ multipliers and $3N/2$ adders are required. From (2.15), it can be shown that V'_{dd} can be as low as $3.1V$ for the case of $M = 2$. Provided that the capacitance 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

$$(\frac{\frac{3N}{2}}{N}C_{eff})(\frac{3.1V}{5V})^2(\frac{1}{2}f) \approx 0.29P_0, \tag{2.16}$$

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. Basically, we trade hardware complexity for low-power consumption.

2.5 Design of the Low-Power QMF Filter

The Quadrature Mirror Filter (QMF) is widely used in the image compression and subband coding [20]. Given the low-pass filter of the QMF banks

$$H_1(z) = \sum_{n=0}^{N-1} h_1(n) z^{-n}, \tag{2.17}$$

its associated high-pass filter $H_2(z)$ is given by

$$H_2(z) = \sum_{n=0}^{N-1} h_2(n) z^{-n} = \sum_{n=0}^{N-1} (-1)^n h_1(n) z^{-n}. \tag{2.18}$$

As depicted in Fig. 2.9(a), the FIR structure of $H_1(z)$ consists of N multiply-and-add (MAA) modules. To reduce the long delay in the summation chain, the technique of pipelining is employed to break down the critical path. Nevertheless, the extra pipeline stages will increase the hardware complexity. In order to have a good compromise between the silicon area and the speed performance, we insert one pipeline stage for every three MAA modules as shown in Fig. 2.9(b). Note that the simple relationship between $H_1(z)$ of (2.17) and $H_2(z)$ of (2.18) enables us to perform $H_2(z)$ by simply complementing the odd-indexed coefficients of the FIR architecture. In our design, we employ the pipelined design of Fig. 2.9(b) to implement the normal FIR filter as well as the three subfilters shown in Fig. 2.8.

One major application of the QMF is the subband coding of speech and image signals. Figure 2.10 shows a four-band two-dimensional (2-D) QMF bank. In the following discussion, we will apply this system to decide the system parameters such as the QMF coefficients and the system wordlength. The performance index is the peak signal-to-

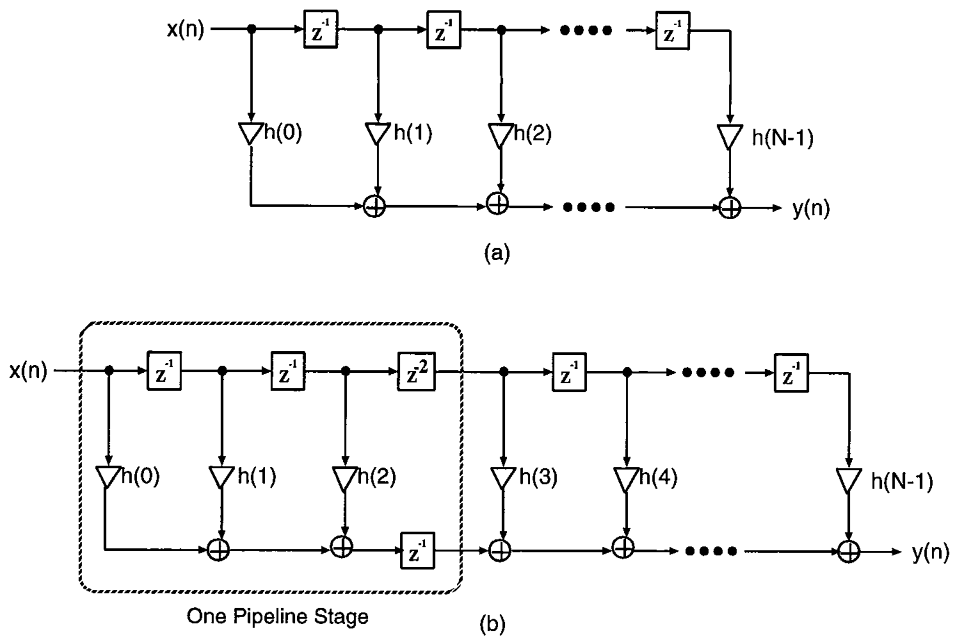


Figure 2.9: (a) FIR structure. (b) FIR structure with pipelined design.

noise ratio (PSNR) defined by

$$PSNR \triangleq \frac{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \max\{x(i,j)\}^2}{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} (x(i,j) - \hat{x}(i,j))^2} \tag{2.19}$$

where $x(i,j)$ and $\hat{x}(i,j)$, $i,j = 0, 1, \ldots, N-1$ denote the pixel values of the original image and the reconstructed image, respectively.

2.5.1 Choice of QMF Coefficients

The QMF coefficient sets in [20] are well-known for their SNR performance in the subband coding applications. However, since those coefficients require floating-point multiplications to have the best performance, the implementation of such a QMF calls for a large silicon area. In order to reduce the chip area while maintaining the SNR performance, we choose the QMF design with power-of-two coefficients [27] for our chip

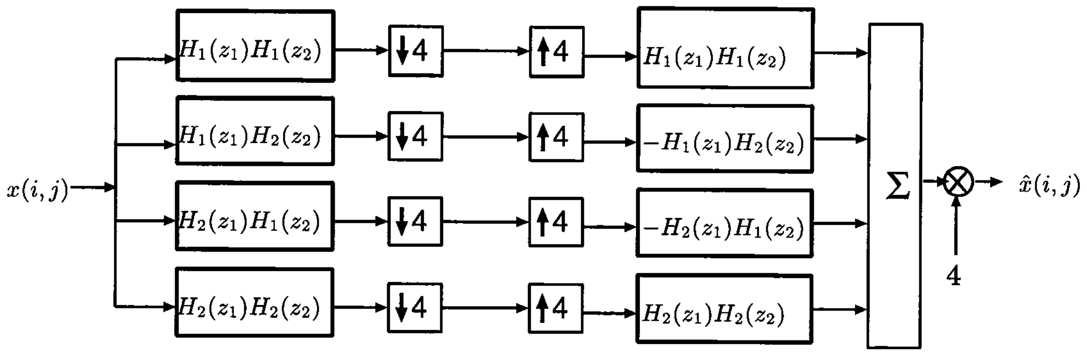


Figure 2.10: The four-band 2-D QMF bank.

implementation. It has been shown [27] that the power-of-two coefficient sets yield comparable PSNR results to those of the floating-point counterpart. In our design, for the purpose of further reducing the hardware complexity (chip area), we modify the QMF design of [27] 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:

$$\begin{aligned}
h(10) &= h(11) = 2^{-2}+2^{-4}+2^{-6}, & h(5) &= h(16) = 2^{-7}+2^{-8},\\
h(9) &= h(12) = 2^{-4}+2^{-5}, & h(4) &= h(17) = -2^{-6}-2^{-7},\\
h(8) &= h(13) = -2^{-4}-2^{-7}, & h(3) &= h(18) = -2^{-8},\\
h(7) &= h(14) = -2^{-5}, & h(2) &= h(19) = 2^{-6},\\
h(6) &= h(15) = 2^{-5}+2^{-7}, & h(1) &= h(20) = 0,\\
& & h(0) &= h(21) = -2^{-7}.
\end{aligned} \tag{2.20}$$

Its frequency response is shown in Fig. 2.11. To verify the performance of this modified QMF, we carried out simulations by passing the LENA image through the subband coding structure depicted in Fig. 2.10. Table 2.1 lists the PSNR results. Compared with the original design of [27], our modified QMF filter has only little degradation in PSNR but with much less hardware complexity.

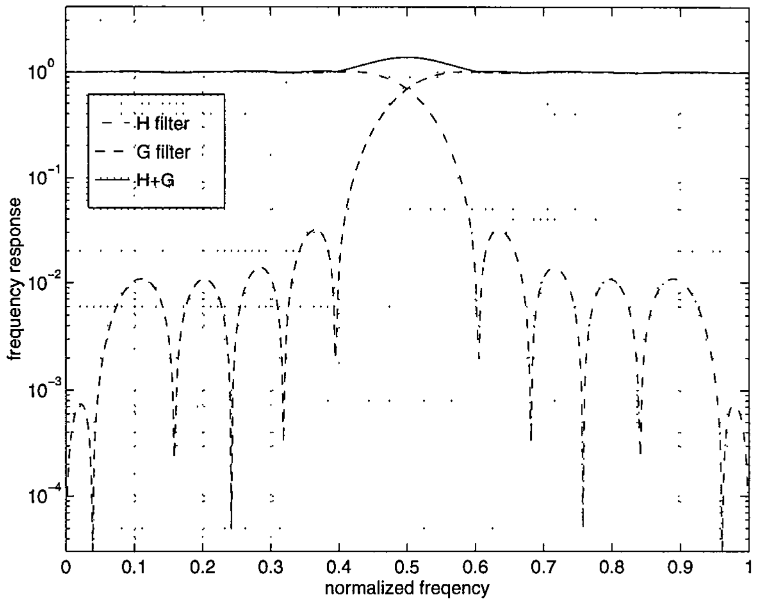


Figure 2.11: Frequency response of the modified QMF filter using power-of-two coefficients.

Filter type	Filter length	PSNR	Coefficient type	Fixed-point adders
Filter (Type 32D) in [20]	32	44.82 dB	floating-point	N/A
Filter in [27]	32	38.79 dB	power-of-two	84
Modified Filter of (2.20)	22	37.01 dB	power-of-two	36

Table 2.1: PSNR results for different QMF's.

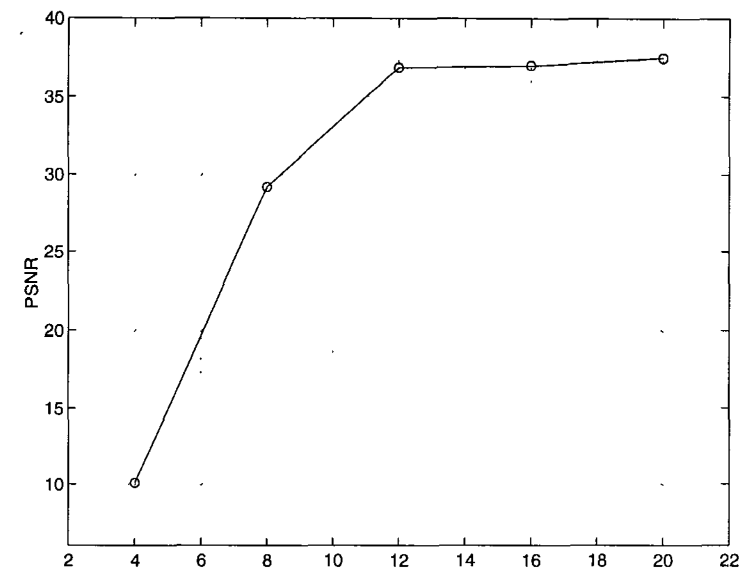


Figure 2.12: PSNR results for the modified QMF as a function of system wordlength B.

2.5.2 Choice of System Wordlength

Next we want to determine the system wordlength to be used in our chip design. Since the wordlength 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 wordlength 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. 2.12. We see that the PSNR curve saturates around $B = 12$. Thus, we use the wordlength of 12 bits in our design. Figure 2.13 shows the original LENA image and the output image under fixed-point operations.

2.5.3 Chip Design and Simulation Results

The modified QMF FIR filter is implemented onto VLSI chips by using two different architectures. One is the normal pipelined design and the other is the multirate design of Fig. 2.8. The resulting chip layouts are shown in Figs. 2.14 and 2.15, respectively.

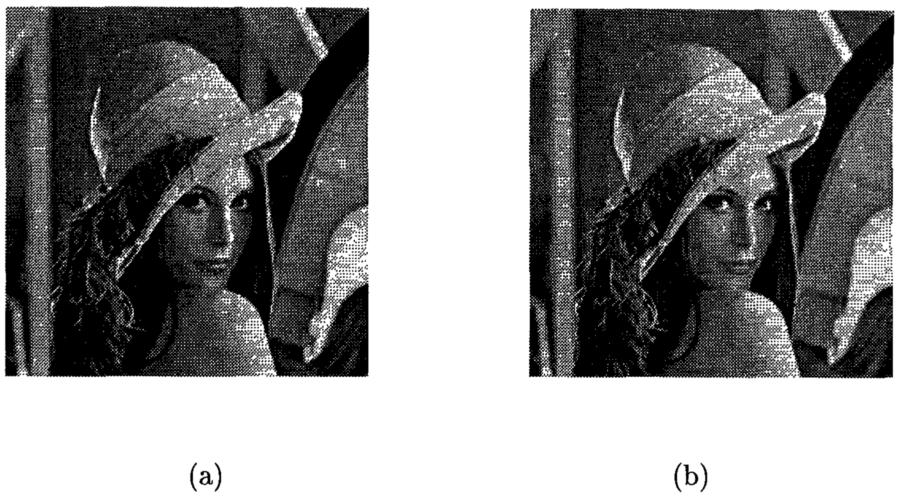


Figure 2.13: (a) The original LENA image. (b) The output image of the subband coding structure based on the modified power-of-two QMF with wordlength $B = 12$ (PSNR=36.9 dB).

There are four modules in the multirate design. The upper right module realizes the upsampling and downsampling circuits of Fig. 2.8. The signal data rate is reduced to $f_s/2$ after this module. 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. 2.8, and the operating frequency of these filters is only $f_s/2$. 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 multirate design is about 50% more than that of the normal design as we expected. Therefore, our estimation of the effective capacitance in (2.16) is very accurate.

In order 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 result depicted in Fig. 2.16 shows that the propagation delay is approximately doubled as the supply voltage reduces from 5V to 3V. This is consistent with the results presented in [3]. 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

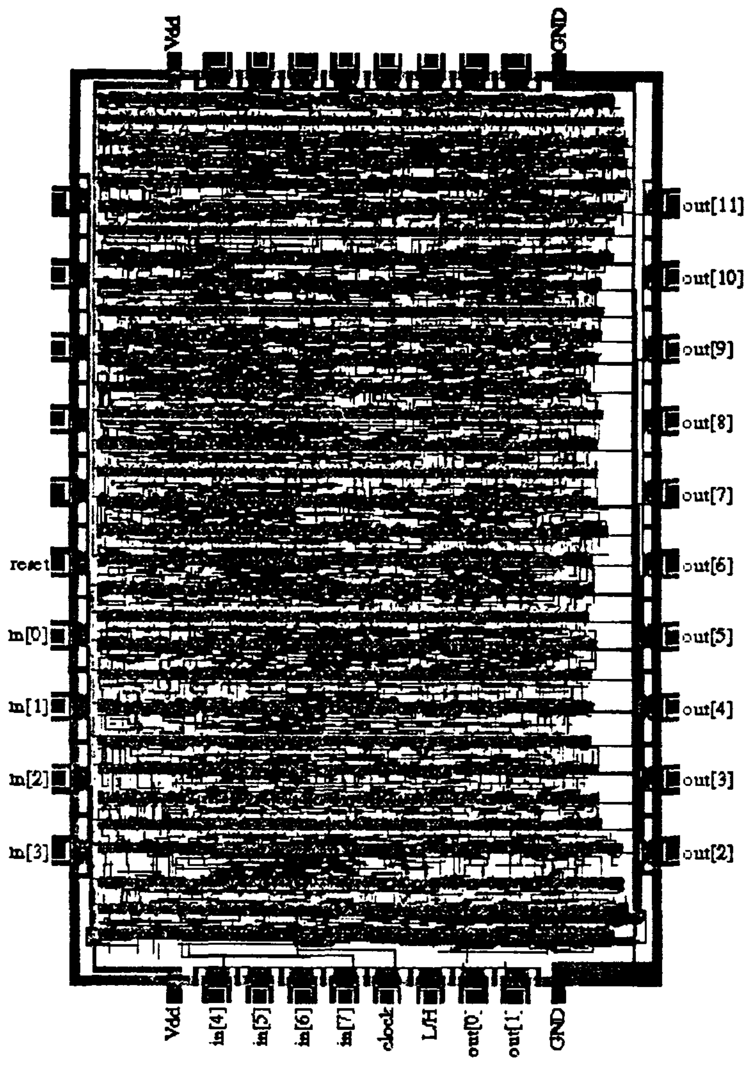


Figure 2.14: Final layout of the normal QMF filter. The chip dimension is 4400 × 6600λ^2 (courtesy of Zhongying Zhang, VLSI Design Automation Labortory, Electrical Engineering Department, University of Maryland at College Park).

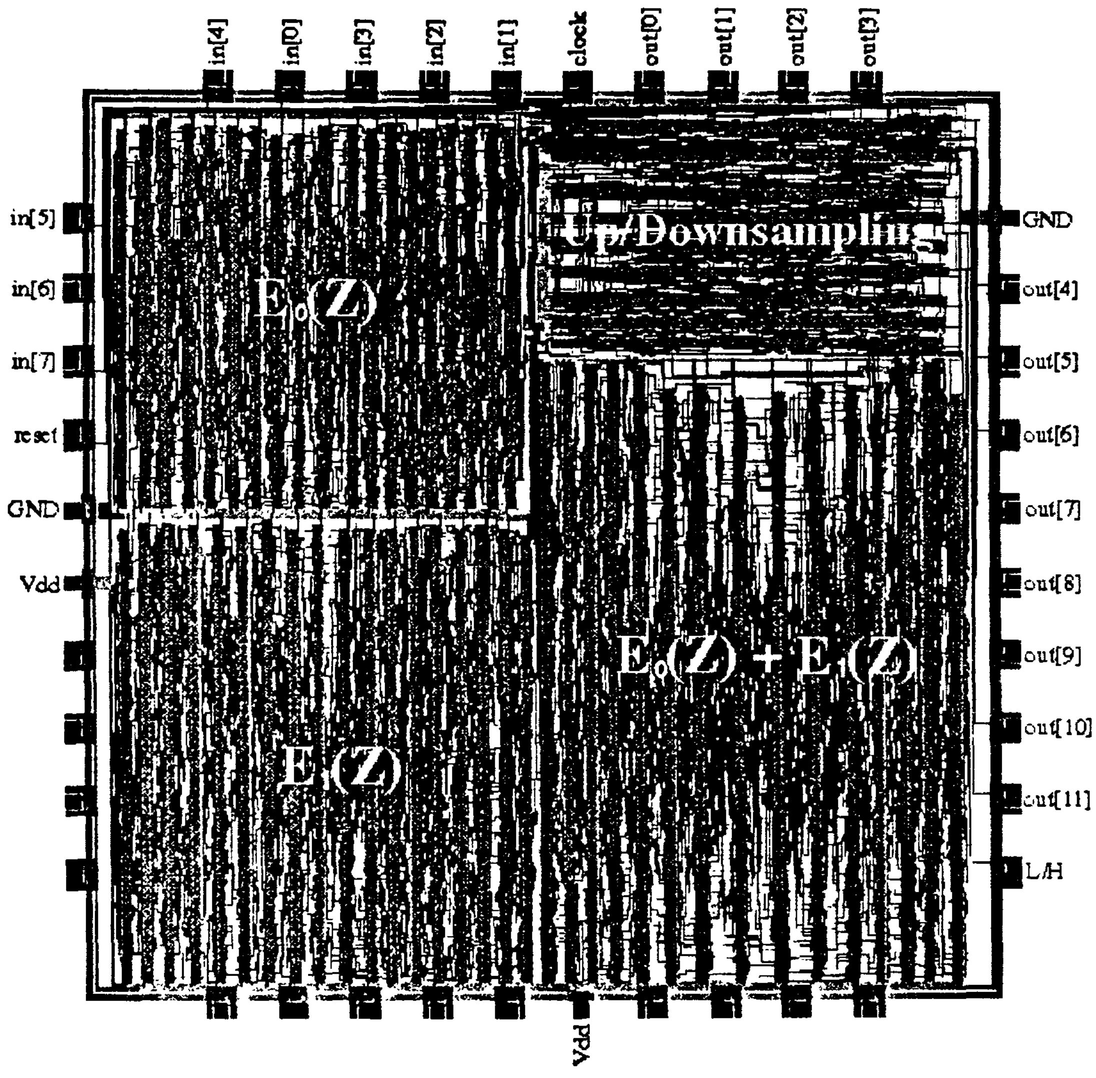


Figure 2.15: Final layout of the multirate QMF filter. The chip dimension is $6500 \times 6600\lambda^2$ (courtesy of Zhongying Zhang, VLSI Design Automation Labortory, Electrical Engineering Department, University of Maryland at College Park).

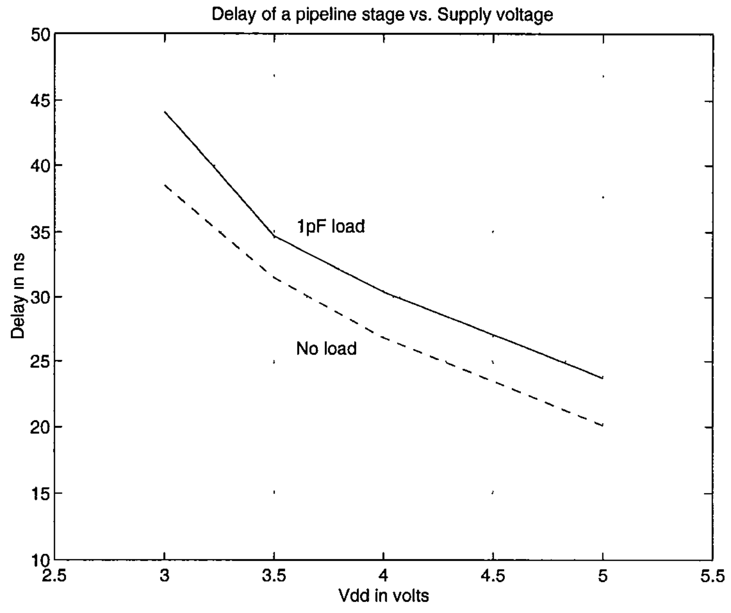


Figure 2.16: Timing analysis of one pipelined stage in the FIR design.

degraded by 50% under the 3V supply voltage. Nevertheless, the data throughput rate of the multirate FIR will not be affected by such a speed penalty since the slowed-down devices are in the $f_s/2$ region (see Fig. 2.8). The I/O data rate will remain at f_s which is the same as the normal FIR design operated at 5V system. The simulation results shown in Figs. 2.14 and 2.16 have confirmed the parameter values of (2.16) that are used to estimate the power consumption of the multirate FIR chip.

Another attractive application of the proposed multirate design is in the very high-speed filtering. If we do not lower down the supply voltage to save chip power consumption, the maximum speed of the multirate design can be M-times faster than the normal design. For example, the multirate FIR structure of Fig. 2.8 can process data at 100 MHz rate while only 50 MHz processing elements are required. This property is also verified by using the IRSIM-a timing analysis CAD tool. The simulation results show that, under the supply voltage of 5V, the maximum speed of the multirate QMF chip can be up to 50 MHz while that of the normal QMF chip is 25 MHz. This again agrees with our argument for the speed performance of the multirate design.

Chapter 3

Multirate DCT/IDCT Architectures

In this chapter, we describe two different approaches to derive multirate low-power VLSI architectures for the discrete cosine transform (DCT) and its inverse transform (IDCT). One is based on the properties of the Chebyshev polynomial. The Chebyshev polynomial derivation of the DCT/IDCT algorithm was first proposed in [28]. However, the architecture in [28] needs global communication and requires $O(N \log N)$ multipliers. Here, we treat the transforms as the evaluation of a Chebyshev series. By exploiting the recurrence property of the Chebyshev polynomial, we can compute the DCT/IDCT through the decimated sequences with a linear increase of hardware complexity; hence the speed penalty can be compensated. The other is based on the polyphase decomposition approach. By applying the polyphase decomposition to the IIR transfer functions of the DCT/IDCT [19], we can also derive multirate DCT/IDCT architectures that are effective in power saving.

The organization of this chapter is as follows. The derivation of the low-power IDCT/DCT algorithms and architectures based on the Chebyshev polynomial is described in Section 3.1. The multirate DCT/IDCT architectures based on the polyphase decomposition approach are derived in Section 3.2. The comparison of both low-power designs with other approaches is discussed in Section 3.3.

3.1 The Chebyshev Polynomial Approach

The nth order Chebyshev polynomial is defined as [29, Chap. 1]

$$T_n(\eta) = \cos(n\omega), \quad \cos\omega = \eta, \quad , -1 \le \eta \le 1, \tag{3.1}$$

which can be generated from the "three-term recurrence" formula

$$T_{n+1}(\eta) = 2\eta T_n(\eta) - T_{n-1}(\eta) \tag{3.2}$$

with the initial conditions $T_0(\eta) = 1$ and $T_1(\eta) = \eta$. Now consider the following Chebyshev series

$$Y_c(\eta) = \frac{1}{2}a_0 + \sum_{k=1}^{N-1} a_k \cos(k\omega) = \frac{1}{2}a_0 + \sum_{k=1}^{N-1} a_k T_k(\eta), \tag{3.3}$$

where $a_k's$, $k = 0, 1, \ldots, N-1$, are constant coefficients. One efficient way to evaluate $Y_c(\eta)$ for a given value η is the **Clenshaw's algorithm** [29, Chap 3] [30, Chap 4], in which a "backward recurrence sequence" is defined as

$$b_k(\eta) = 2\eta\, b_{k+1}(\eta) - b_{k+2}(\eta) + a_k, \quad \text{for } k = N-1, \ldots, 1, 0 \tag{3.4}$$

with the initial conditions $b_N(\eta) = b_{N+1}(\eta) = 0$. After substituting (3.4) into (3.3), and applying the recurrence formula of (3.2), we can simplify the evaluation of $Y_c(\eta)$ as

$$Y_c(\eta) = \sum_{k=0}^{N-1} [b_k(\eta) - 2\eta\, b_{k+1}(\eta) + b_{k+2}(\eta)]T_k(\eta) = \frac{b_0(\eta) - b_2(\eta)}{2}. \tag{3.5}$$

Later in the DCT/IDCT, we will need the evaluation of

$$Y_c'(\eta) = \sum_{k=0}^{N-1} a_k \cos(k\omega) = \sum_{k=0}^{N-1} a_k T_k(\eta). \tag{3.6}$$

It can be seen that by scaling a_0 by 2 (a left shift) beforehand, we can evaluate $Y_c'(\eta)$ through the same steps of (3.4)-(3.5). The corresponding architecture to evaluate $Y_c'(\eta)$ is shown in Fig. 3.1, where a_0 has been pre-scaled by two. Since b_i's are generated in a "backward" manner, the input sequence is in reverse order. The second-order recurrence structure in the middle computes b_i's according to (3.4). After the last input is fed into

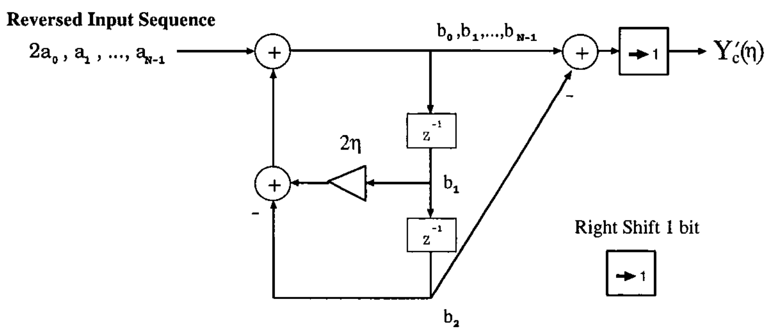


Figure 3.1: Recursive architecture to evaluate $Y_c'(\eta)$.

the system, $b_0(\eta)$ and $b_2(\eta)$ will be available and $Y_c'(\eta)$ can be evaluated from (3.5) with one addition and one right-shift operation.

Other two Chebyshev polynomial properties that will be useful in later derivations are [29, Chap 3]:

1. **Composition property**:

$$T_s(T_r(\eta)) = T_r(T_s(\eta)) = T_{rs}(\eta), \tag{3.7}$$

which allows us to represent a higher-order Chebyshev polynomial using lower-order ones, and vice versa.

2. **Product-sum relationship**:

$$T_s(\eta)T_r(\eta) = \frac{1}{2}(T_{s+r}(\eta) + T_{s-r}(\eta)), \tag{3.8}$$

which shows that the product of two Chebyshev polynomials can be decomposed into the sum of two Chebyshev polynomials, and vice versa.

3.1.1 Chebyshev IDCT Architecture

In order to illustrate the relationship between the Chebyshev polynomial and the transforms, we will begin with the derivation of the IDCT algorithm. Let $X(k)$, $k =$

$0, 1, \cdots, N-1$, be a DCT-domain sequence. The block IDCT to compute the time-domain sequence $x(n)$, $n = 0, 1, \cdots, N-1$, is defined as

$$x(n) = \sum_{k=0}^{N-1} C(k) X(k) \cos[\frac{(2n+1)\pi}{2N} k], \tag{3.9}$$

where

$$C(k) = \begin{cases} \sqrt{\frac{1}{N}}, & \text{if } k = 0 \\ \sqrt{\frac{2}{N}}, & \text{otherwise} \end{cases} \tag{3.10}$$

is the scaling factor used in the DCT/IDCT. If we define

$$\omega_n \triangleq \frac{(2n+1)\pi}{2N} \tag{3.11}$$

and use the definition of the Chebyshev polynomial in (3.1), (3.9) can be written as

$$x(n) = \sum_{k=0}^{N-1} C(k) X(k) \cos(k\omega_n) = \sum_{k=0}^{N-1} X'(k) T_k(\eta_n) \tag{3.12}$$

where

$$\eta_n \triangleq \cos \omega_n \tag{3.13}$$

and $X'(k) = C(k)X(k)$ is the scaled input data. Comparing (3.12) with (3.6), we see that the IDCT with index n can be treated as the evaluation of Chebyshev series at η_n with coefficients $X'(k)'s$, $k = 0, 1, \ldots, N-1$. As a consequence, the recursive architecture in Fig. 3.1 can perform the IDCT at center frequency ω_n if we replace the multiplier coefficient η with η_n.

Figure 3.2 shows the IDCT structure based on the Chebyshev evaluation. It has two parts: the **Reverse Array** (RA) and the IDCT module array. The RA consists of one **serial-input-parallel-output** (SIPO) register array and one **parallel-input-serial-output** (PISO) register array. It provides the capability of reversing the input sequence and scaling $X(0)$ in a fully pipelined way. The IDCT module performs the computation of (3.12) at different index n. Since n varies from 0 to $N-1$, we need N IDCT modules to compute the IDCT in parallel. The whole system works in an SIPO

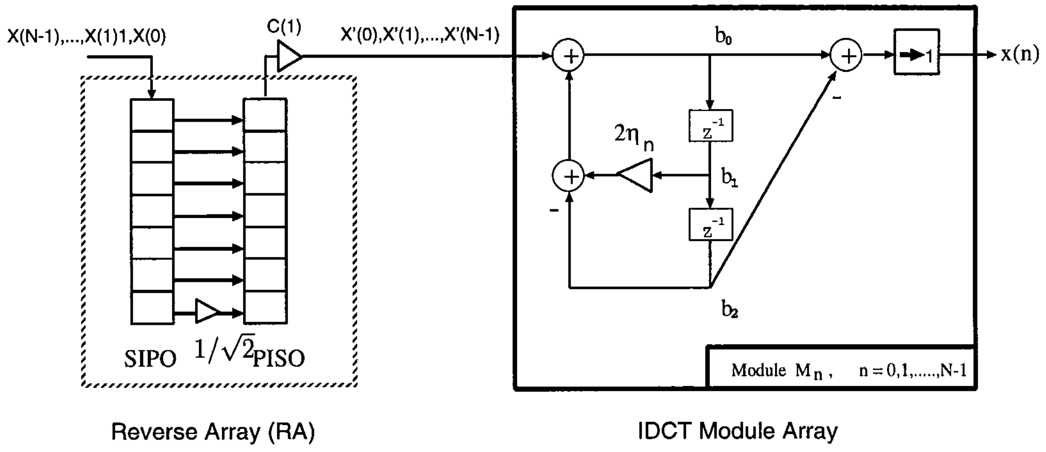


Figure 3.2: Parallel Chebyshev IDCT architecture.

way and requires only $N+1$ multipliers and $3N$ adders including the scaling multiplier in RA. The number of multipliers is almost as low as that in Hou's algorithm [31]. Besides, there is no restriction on the block size N and the regularity of our IDCT architecture is more suitable for VLSI implementation.

3.1.2 Chebyshev DCT Architecture

The DCT of the time-domain block data $x(n)'s$, $n = 0, 1, \ldots, N-1$, is defined as

$$X(k) = C(k) \sum_{n=0}^{N-1} x(n) \cos[(2n+1)\frac{k\pi}{2N}], \quad k = 0, 1, \ldots, N-1. \tag{3.14}$$

As with the derivation of the IDCT algorithm, the DCT can be represented as

$$X(k) = C(k) \sum_{n=0}^{N-1} x(n) \cos[(2n+1)\omega_k] = C(k) \sum_{n=0}^{N-1} x(n) T_{2n+1}(\eta_k) \tag{3.15}$$

where $\omega_k \triangleq \frac{k\pi}{2N}$ and $\eta_k \triangleq \cos\omega_k$. Multiplying $T_1(\eta_k)$ on both sides of (3.15) and using the Chebyshev property in (3.8), we obtain

$$\begin{aligned} T_1(\eta_k)X(k) &= C(k) \sum_{n=0}^{N-1} \frac{x(n)}{2}[(T_{2n}(\eta_k) + T_{2n+2}(\eta_k)] \\ &= \frac{C(k)}{2} \sum_{n=0}^{N} x'(n) T_{2n}(\eta_k) \end{aligned} \tag{3.16}$$

where

$$x'(n) \triangleq x(n-1) + x(n), \quad n = 0, 1, \ldots, N \tag{3.17}$$

with the assumption of $x(-1) = x(N) = 0$. Recall that $T_1(\eta_k) = \eta_k$ and $T_{2n}(\eta_k) = T_n(T_2(\eta_k))$ (from (3.7)). If we define

$$\eta'_k \triangleq T_2(\eta_k) = \cos(2\omega_k) = 2\eta_k^2 - 1, \tag{3.18}$$

$X(k)$ in (3.16) can be computed as

$$X(k) = \frac{C(k)}{2\eta_k} \sum_{n=0}^{N} x'(n) T_n(\eta'_k), \quad k = 0, 1, \ldots, N-1. \tag{3.19}$$

Therefore, the DCT at center frequency ω_k can be obtained by evaluating the Chebyshev series at the value η'_k with coefficients $x'(n)'s, n = 0, 1, \ldots, N$, followed by the scaling of $\frac{C(k)}{2\eta_k}$.

Note that the DCT of the reversed sequence $\tilde{x}(n) = x(N-1-n)$, $n = 0, 1, \ldots, N-1$, is

$$\tilde{X}(k) = C(k) \sum_{n=0}^{N-1} \tilde{x}(n) \cos[(2n+1)\omega_k] = C(k) \sum_{n=0}^{N-1} x(n) \cos[k\pi - \frac{(2n+1)k\pi}{2N}], \tag{3.20}$$

for $k = 0, 1, \ldots, N-1$. We can relate $\tilde{X}(k)$ to $X(k)$ by $\tilde{X}(k) = (-1)^k X(k)$. As a result, the RA, which is used to reverse the input sequence, can be eliminated by complementing the odd-indexed $X(k)$'s while keeping the even-indexed $X(k)$'s unchanged. Figure 3.3 shows the architecture to implement the Chebyshev DCT algorithm. The overall Chebyshev DCT architecture needs a total of $2N-2$ multipliers and $3N-1$ adders. It should be noted that the total number of $x'(n)'s$ is $N+1$. Therefore, an extra zero is appended after $x(N-1)$ for the generation of $x'(n)$, $n = 0, 1, \ldots, N$. After $x'(N)$ is sent to the DCT array, we can obtain the DCT coefficients in parallel at the array outputs.

3.1.3 Low-Power Design for the DCT/IDCT

Consider the Chebyshev series of (3.6) and split it into the even and odd series:

$$Y'_c(\eta) \quad = \quad \sum_{i=0}^{N/2-1} a_{2i} T_{2i}(\eta) + \sum_{i=0}^{N/2-1} a_{2i+1} T_{2i+1}(\eta)$$

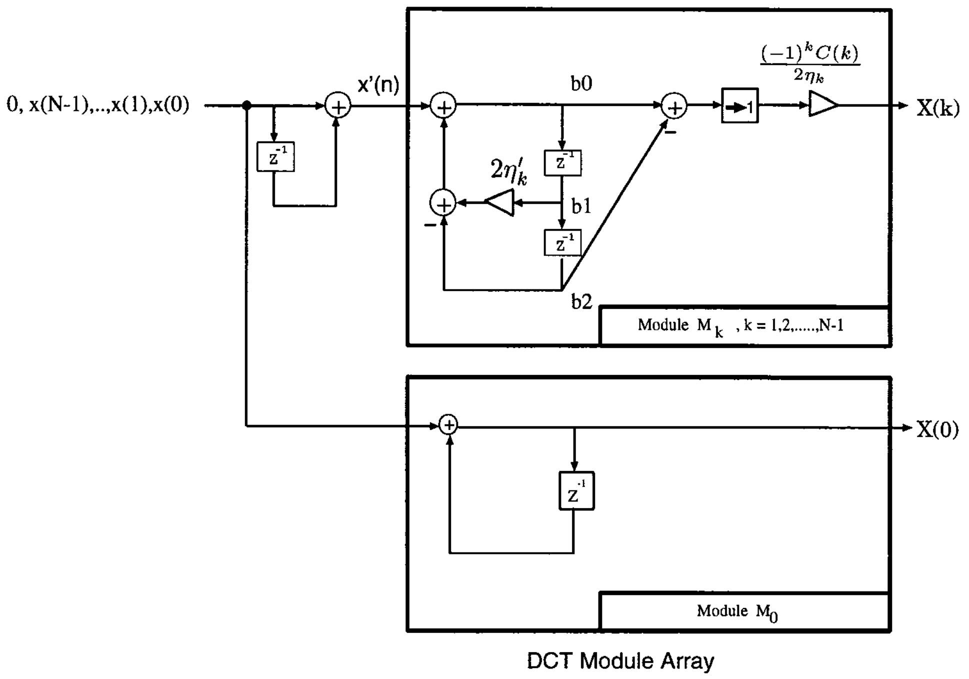


Figure 3.3: Parallel Chebyshev DCT architecture.

$$= \quad Y_e(\eta) + Y_o(\eta) \tag{3.21}$$

where $Y_e(\eta)$ and $Y_o(\eta)$ denote the even and odd series, respectively. By the use of (3.2) and (3.7), $Y_e(\eta)$ can be written as

$$Y_e(\eta) = \sum_{i=0}^{N/2-1} a_{2i}T_i(T_2(\eta)) = \sum_{i=0}^{N/2-1} a_{2i}T_i(\eta') \tag{3.22}$$

with $\eta' = 2\eta^2 - 1$. On the other hand, $Y_o(\eta)$ can be converted into an even series by following the derivations of (3.15)-(3.19):

$$Y_o(\eta) = \sum_{i=0}^{N/2} \kappa(a_{2i-1} + a_{2i+1})T_i(\eta'). \tag{3.23}$$

where $\kappa = \frac{1}{2\eta}$ is a pre-calculated constant coefficient. Now combining (3.22) and (3.23) together, we have

$$Y_c'(\eta) = \sum_{i=0}^{N/2} [a_{2i} + \kappa(a_{2i-1} + a_{2i+1})]T_i(\eta') = \sum_{i=0}^{N/2} d_i T_i(\eta') \tag{3.24}$$

with

$$d_i \triangleq \underbrace{a_{2i}}_{even} + \underbrace{\kappa(a_{2i-1} + a_{2i+1})}_{odd}, \quad i = 0, 1, \ldots, \frac{N}{2}. \tag{3.25}$$

From (3.24) we can see that the evaluation of an N-point Chebyshev series can be reduced to an $(N/2+1)$-point evaluation using the new sequence d_i's which are composed of decimated sequences. This new evaluation method can be easily applied to the computation of the IDCT/DCT as described in Sections 3.1.1 and 3.1.2. The resulting IDCT architecture is depicted in Fig. 3.4, where $\kappa_n = 1/(2\eta_n)$ and $\eta_n' = 2\eta_n^2 - 1$ with η_n defined in (3.13). Firstly, a_i's in (3.24) are replaced with $X(i)$'s in (3.9), for $i = 0, 1, \ldots N-1$, then we use one decimation circuit and one adder to compute the even and odd sequences in (3.25) from the $X(i)$ sequence in a fully-pipelined way (see the left-hand side of Fig. 3.4). After these two decimated sequences are reversed by the RA, they are combined together to generate d_i's of (3.25), and d_i's are sent to the IDCT module to perform the Chebyshev evaluation in (3.24). Once the evaluation is completed, we have

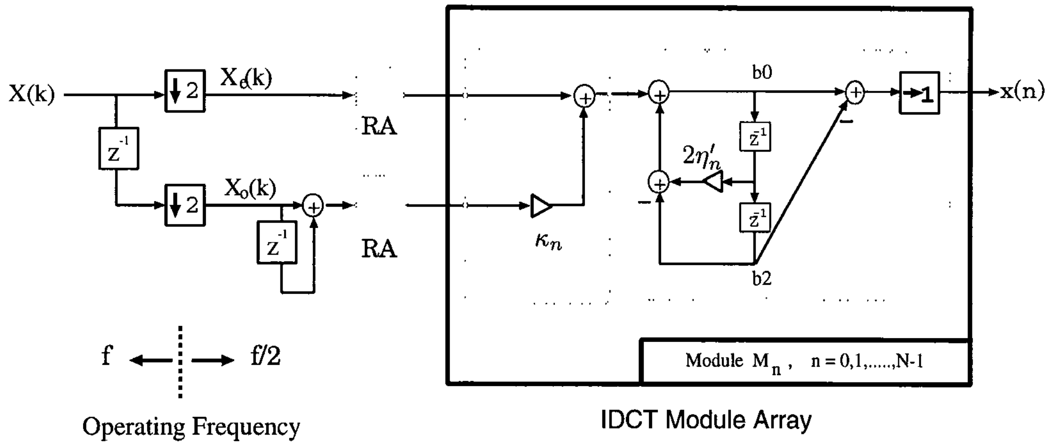


Figure 3.4: Low-power parallel Chebyshev IDCT architecture with decimation factor of two.

the IDCT coefficient with index n at the module output. Since the operating frequency halves after the decimator, we can now use two times slower multipliers and adders in this IDCT module with some hardware overhead. Meanwhile, the throughput rate is still retained. Similarly, the multirate Chebyshev DCT architecture can be derived as shown in Fig. 3.5, where $\kappa'_k = 1/(2\eta'_k)$, $\eta''_k = 2{\eta'_k}^2 - 1$, and η'_k is defined in (3.18).

To achieve downsampling by four, we can recursively compute another new $(N/4+1)$-sequence e_i from d_i, which results in

$$e_i = \kappa'\kappa\left[(a_{4i-3} + a_{4i+1}) + (a_{4i-1} + a_{4i+3})\right] + \kappa'(a_{4i-2} + a_{4i+2}) + \kappa(a_{4i-1} + a_{4i+1}) + a_{4i},$$
$$\text{for} \quad i = 0, 1, \ldots, \tfrac{N}{4}, \tag{3.26}$$

where $\kappa' = \frac{1}{2\eta'}$ is also a pre-computed constant. One possible realization of (3.26) is depicted in Fig. 3.6.

Once the e_i's are computed from the decimated sequences $a'_{4i+k}s$, $k = 0, 1, 2, 3$, the evaluation of $Y'_c(\eta)$ can be computed as

$$Y'_c(\eta) = \sum_{i=0}^{N/4} e_i T_i(\eta'') \tag{3.27}$$

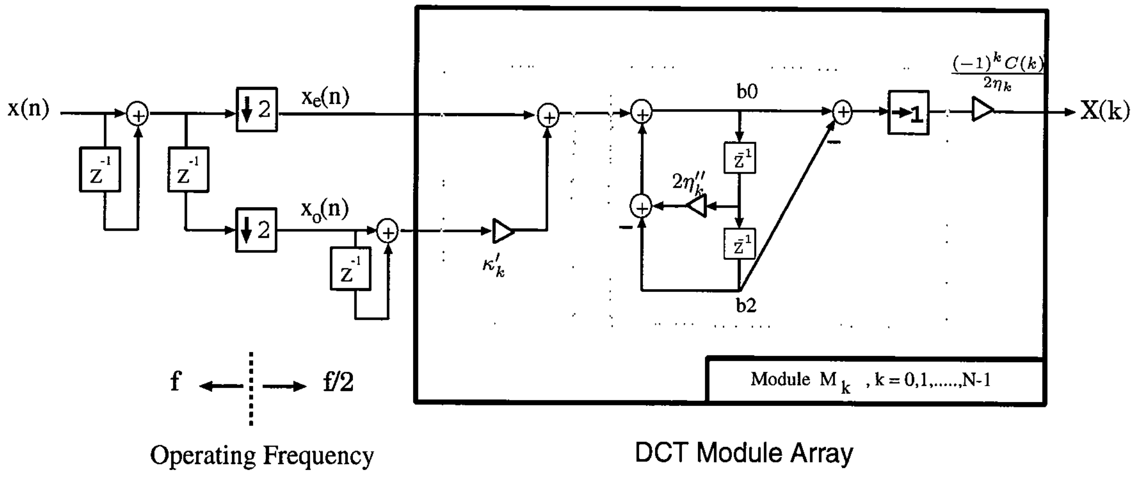


Figure 3.5: Low-power parallel Chebyshev DCT architecture with a decimation factor of two.

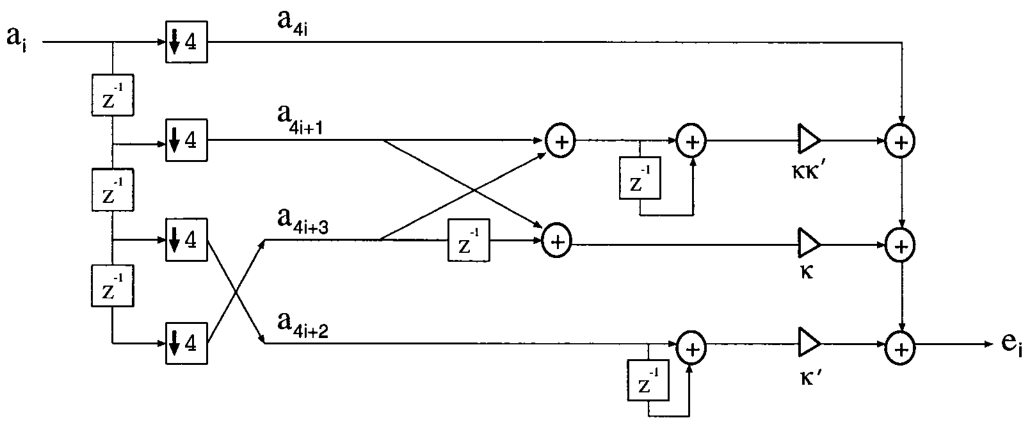


Figure 3.6: Evaluation of e_i using a downsampling circuit.

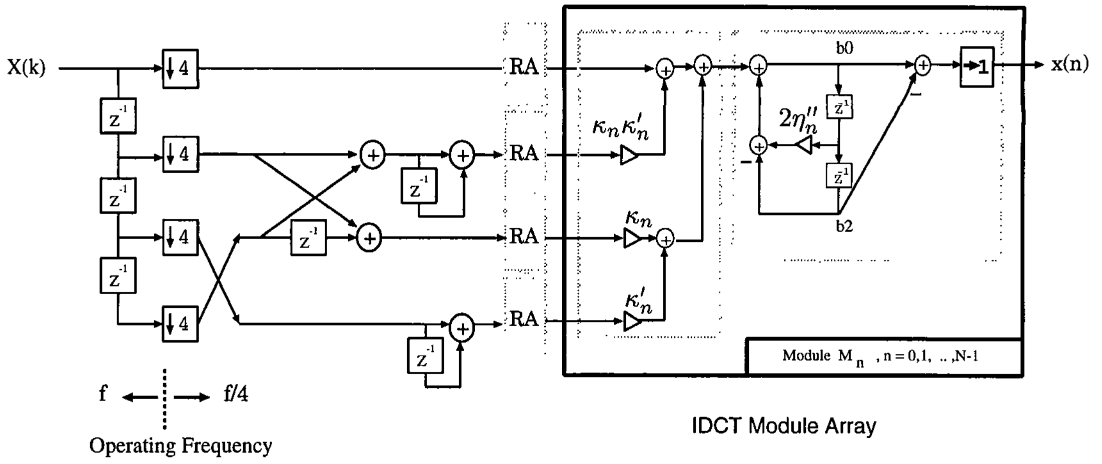


Figure 3.7: Low-power parallel Chebyshev IDCT architecture with a decimation factor of four, where $\eta''_n = 2(\eta'_n)^2 - 1$ and $\kappa'_n = 1/(2\eta'_n)$.

with $\eta'' = 2\eta'^2 - 1$. Likewise, based on (3.26) and (3.27), we can also construct the multirate IDCT and DCT architectures as shown in Figs. 3.7 and 3.8, in which only four times slower operators are required to compute the transform coefficients.

Power Estimation for the Low-Power Design

Now let us consider the power dissipation of the low-power architectures. The 16-point Chebyshev IDCT under normal operation requires 18 multipliers and 48 adders. For the low-power 16-point IDCT with $M = 2$, 34 multipliers and 65 adders are required. Following the arguments in Section 2.4, it can be shown that the power consumption of this design can be estimated as

$$(\frac{34}{18}C_{eff})(\frac{3.1V}{5V})^2(\frac{1}{2}f) \approx 0.36P_0, \tag{3.28}$$

where P_0 denotes the power consumption of the original system. Similarly, for the case of $M = 4$, the 16-point IDCT needs a total of 66 multipliers and 100 adders. Since the lowest possible supply voltage can be $2.1V$ (from (2.15)), the total power can be reduced

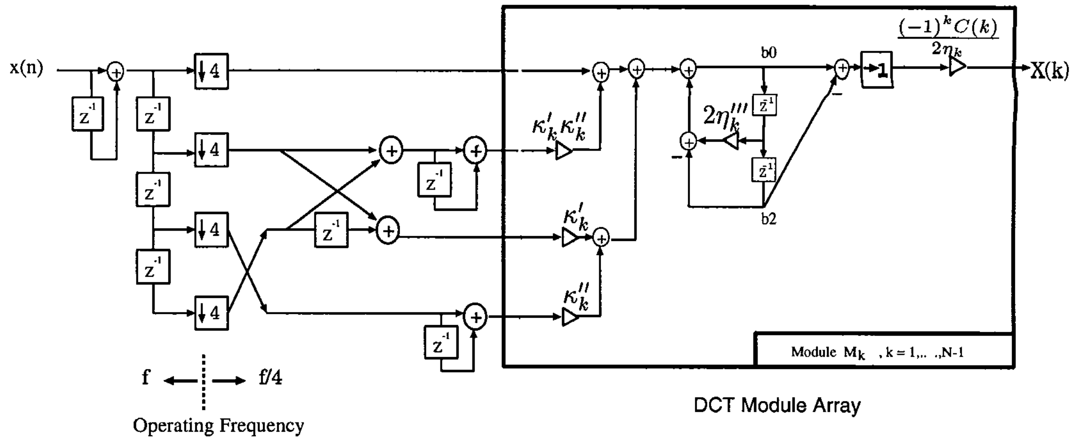


Figure 3.8: Low-power parallel Chebyshev DCT architecture with a decimation factor of four, where $\kappa''_k = 1/\eta''_k$ and $\eta'''_k = 2(\eta''_k)^2 - 1$.

to

$$(\frac{66}{18}C_{eff})(\frac{2.1V}{5V})^2(\frac{1}{4}f) \approx 0.16P_0. \tag{3.29}$$

Therefore, we can achieve low-power consumption at the expense of reasonable complexity overhead. Such a tradeoff will be considered in Section 3.3.

3.2 The Polyphase Decomposition Approach

Performing orthogonal transforms based on the IIR transfer function approach was studied in [19]. By considering the transform operator as a linear shift invariant (LSI) system that maps the serial input data into their transform coefficients, the authors in [19] have shown that most discrete sinusoidal transforms can be realized by using a unified IIR structure. In this section, we will show that, in addition to the Chebyshev approach, we can also derive multirate low-power DCT/IDCT algorithms/architectures by applying the polyphase decomposition to the IIR transfer functions in [19]. We will see later that the polyphase decomposition approach provides a systematic way for architectural low-power design.

3.2.1 The IIR DCT Algorithm

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

$$X_{DCT,k}(t) = C(k) \sum_{n=0}^{N-1} \cos[(2n+1)\frac{k\pi}{2N}]\, x(t+n-N+1), \tag{3.30}$$

for $k = 0, 1, 2, \ldots, N-1$. A second-order IIR transfer function can be derived from (3.30) as [19]

$$H_{DCT,k}(z) = \frac{X_{DCT,k}(z)}{X(z)} = ((-1)^k - z^{-N}) \frac{C(k)\cos\omega_k(1-z^{-1})}{1-2\cos 2\omega_k z^{-1} + z^{-2}} \tag{3.31}$$

where $\omega_k \triangleq \frac{k\pi}{2N}$, and $X_{DCT,k}(z)$ and $X(z)$ denote the z-transforms of $X_{DCT,k}(t)$ and $x(t)$, respectively. For block processing, the z^{-N} in (3.31) can be eliminated because of the reset operation for every N cycles. The corresponding IIR structure to compute the kth frequency component of the DCT is shown in Fig. 3.9, in which

$$\Gamma_c(m) \triangleq (-1)^k C(k) \cos m\omega_k. \tag{3.32}$$

Once the last serial input $x(t)$ is fed into the module, the DCT coefficients can be obtained at the module outputs in parallel. The resulting parallel architecture is regular, modular, and fully-pipelined. Also, the SIPO feature can avoid the input buffers as well as the index mapping operation that are required in most PIPO DCT architectures [31][32]. One disadvantage of the IIR structure in Fig. 3.9 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 the speed constraint can be alleviated.

3.2.2 Low-Power Design of the IIR DCT

Splitting the input data sequence into the **even** sequence

$$x_e(t,n) = x(t+2n-N+1), \quad n = 0, 1, \ldots, N/2-1, \tag{3.33}$$

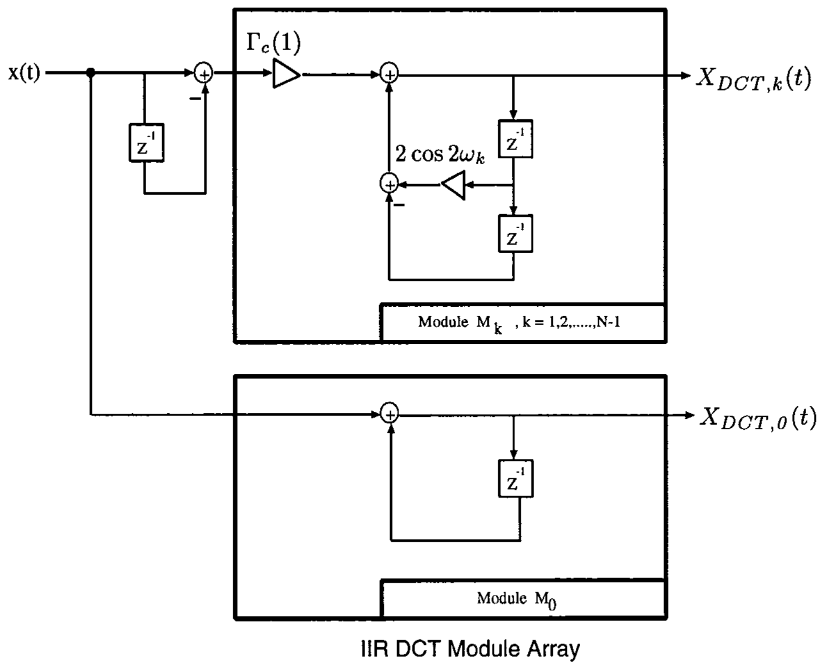


Figure 3.9: IIR DCT architecture.

and the **odd** sequence

$$x_o(t,n) = x(t+2n-N+2), \quad n = 0,1,\ldots,N/2-1, \tag{3.34}$$

(3.30) becomes

$$X_{DCT,k}(t) = C(k)\sum_{n=0}^{N/2-1}\cos[(4n+1)\frac{k\pi}{2N}]\,x_e(t,n) + C(k)\sum_{n=0}^{N/2-1}\cos[(4n+3)\frac{k\pi}{2N}]\,x_o(t,n). \tag{3.35}$$

Taking the z-transform on both sides of (3.35) and rearranging it, we have

$$\begin{aligned} X_{DCT,k}(z) &= \frac{C(k)((-1)^k - z^{-N/2})}{1-2\cos 4\omega_k z^{-1} + z^{-2}} \\ &\times \Big([X_e(z)-X_o(z)z^{-1}]\cos 3\omega_k + [X_o(z)-X_e(z)z^{-1}]\cos\omega_k\Big) \end{aligned} \tag{3.36}$$

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 (3.36) is depicted in Fig. 3.10. The **common circuit** at the left-hand side decimates the input serial data into the even and odd sequences and generates the common inputs for the module array. The numerator and the denominator parts of (3.36) are realized by the FIR structure and IIR structure inside each DCT module at different index k. The overall architecture requires $(3N-3)$ multipliers and $(3N+1)$ adders plus a decimation circuit. Compared with the IIR DCT structure in Fig.3.9, this multirate DCT structure needs only $(N-1)$ extra multipliers and $(N+1)$ extra adders.

To achieve downsampling by the factor of four, we can split the input data sequence into four decimated sequences

$$g_i(t,n) \overset{\triangle}{=} x(t+(4n+i)-N+1), \quad i = 0,1,2,3, \tag{3.37}$$

for $n = 0,1,\ldots,N/4-1$. Following the derivations of (3.35)-(3.36), we can write $X_{DCT,k}(z)$ as

$$X_{DCT,k}(z) = \frac{C(k)((-1)^k - z^{-N/4})}{1-2\cos 8\omega_k z^{-1} + z^{-2}}$$

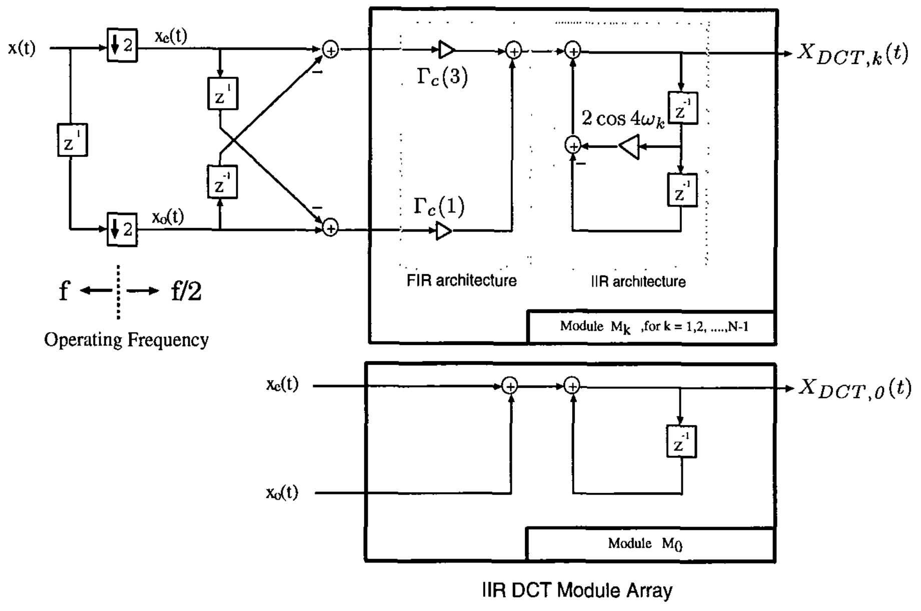


Figure 3.10: Low-power polyphase IIR DCT architecture with $M = 2$.

$$\begin{aligned} \times & \Big([G_0(z) - G_3(z)z^{-1}]\cos 7\omega_k + [G_1(z) - G_2(z)z^{-1}]\cos 5\omega_k \\ + & [G_2(z) - G_1(z)z^{-1}]\cos 3\omega_k + [G_3(z) - G_0(z)z^{-1}]\cos\omega_k\Big) \qquad (3.38) \end{aligned}$$

where $G_i(z)$ is the z-transform of $g_i(t,n)$, $i = 0,1,2,3$. The corresponding multirate architecture is shown in Fig. 3.11.

From Figs. 3.10 and 3.11, we can see that basically the multirate DCT architectures retain all advantages of the original IIR structure in [19] such as modularity, regularity, and local interconnections. It is also interesting to note that the increase in hardware overhead grows only locally rather than globally, and the DCT architecture with $M =$ 4 can be generated by reusing the modules in the $M = 2$ design (*e.g.*, the FIR/IIR structures and the lattice structure in the common circuit). Therefore, neither global routing nor new module design is required in the $M = 4$ case. The characteristics of scalability, modularity, and local interconnections make the multirate structures very suitable for VLSI implementation. Unlike most PIPO DCT algorithms in which the

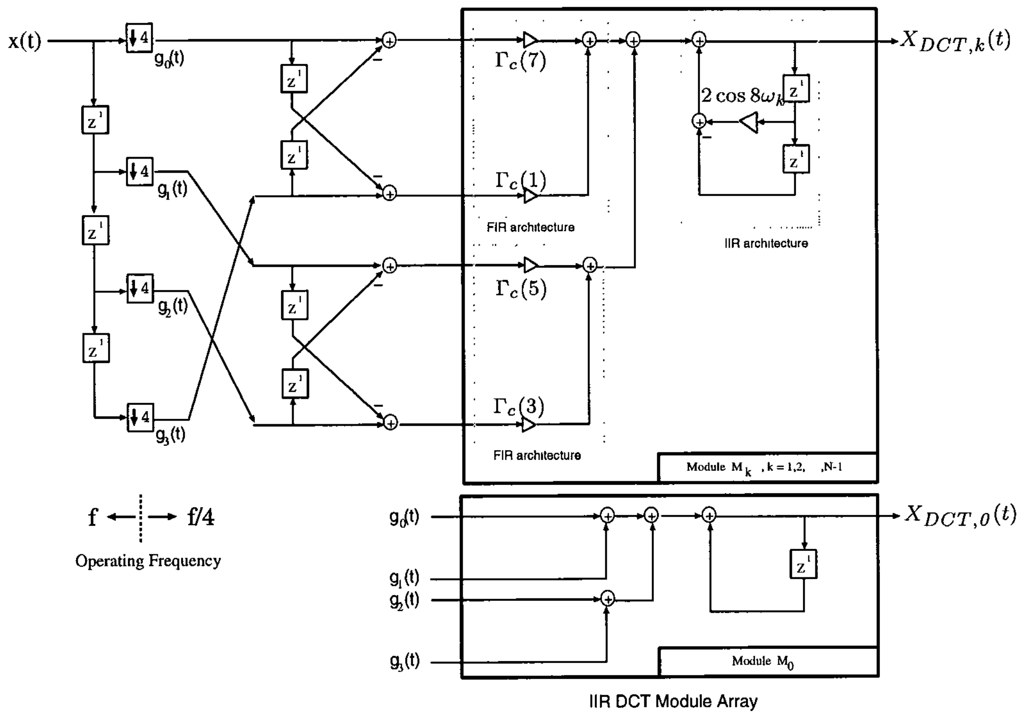


Figure 3.11: Low-power polyphase IIR DCT architecture with $M = 4$.

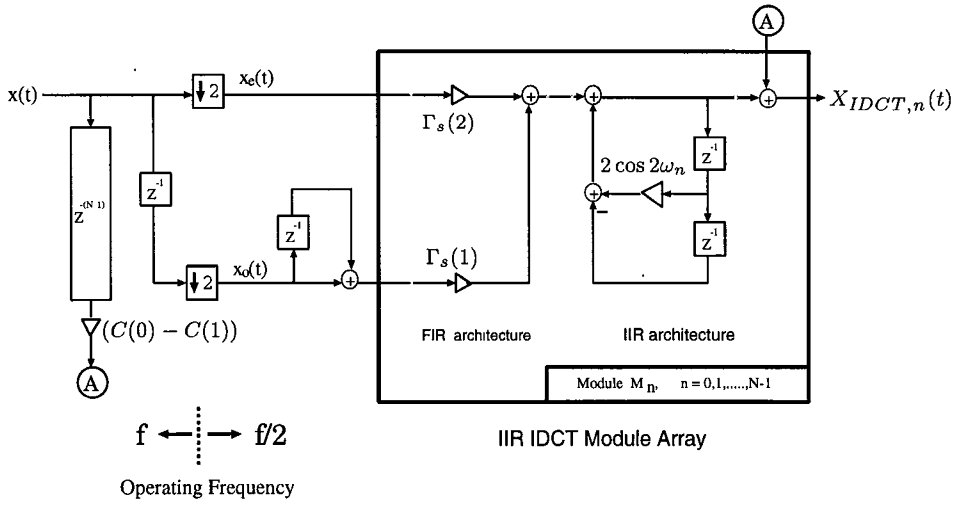


Figure 3.12: Low-power polyphase IIR IDCT architecture with $M = 2$.

interconnections will take up much of the chip area, the feature of local communications of our design can greatly reduce the power dissipation in the routing area. From the discussions in Section 2.3, it can be shown that the total power consumption for the multirate 16-point DCT can be reduced to $0.29P_0$ and $0.11P_0$ for the cases of $M = 2$ and $M = 4$, respectively. The significant power savings for the design with $M = 4$ is achieved only at the cost of $3N - 3$ extra multipliers and $3N + 3$ extra adders.

3.2.3 Low-Power Design of the IIR IDCT

The IIR transfer function for the block IDCT is given by [19]

$$H_{IDCT,n}(z) = \frac{(-1)^n C(1)\sin\omega_n}{1 - 2\cos\omega_n z^{-1} + z^{-2}} + (C(0) - C(1))z^{-(N-1)}, \tag{3.39}$$

where $\omega_n \triangleq \frac{2n+1}{2N}\pi$. As with the derivations of the low-power IIR DCT, the multirate transfer function for the IDCT with $M = 2$ can be derived as

$$\begin{aligned} X_{IDCT,n}(z) &= \frac{(-1)^n C(1)}{1 - 2\cos 2\omega_n z^{-1} + z^{-2}} \left(X_e(z)\sin 2\omega_n + (1 + z^{-1})X_o(z)\sin\omega_n\right) \\ &+ (C(0) - C(1))z^{-(N-1)}X(z). \end{aligned} \tag{3.40}$$

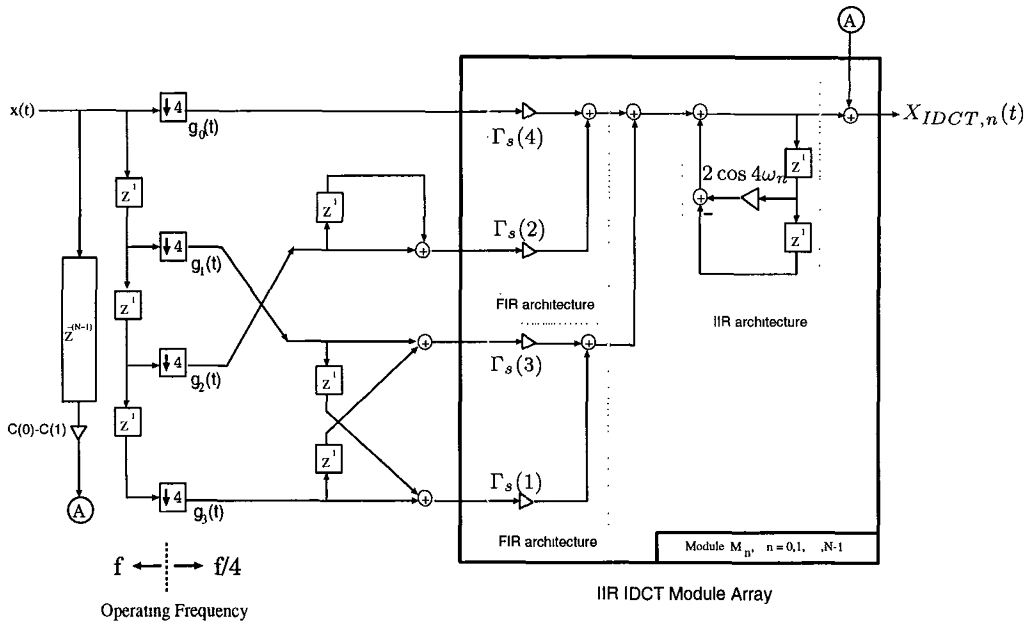


Figure 3.13: Low-power polyphase IIR IDCT architecture with $M = 4$.

Similarly, the transfer function for $M = 4$ is

$$\begin{aligned} X_{IDCT,n}(z) &= \frac{(-1)^n C(1)}{1 - 2\cos 4\omega_n z^{-1} + z^{-2}} \Big(G_0(z)\sin 4\omega_n + [G_1(z) + G_3(z)z^{-1}]\sin 3\omega_n \\ &+ (1 + z^{-1})G_2(z)\sin 2\omega_n + [G_3(z) + G_1(z)z^{-1}]\sin\omega_n\Big) \\ &+ (C(0) - C(1))z^{-(N-1)}X(z). \end{aligned} \tag{3.41}$$

The corresponding low-power IIR IDCT structures based on (3.40) and (3.41) are shown in Figs. 3.12 and 3.13, respectively, where the multiplier coefficient is defined as

$$\Gamma_s(m) \triangleq (-1)^n C(1)\sin m\omega_n. \tag{3.42}$$

As we can see, the low-power IDCT design has similar structures as the low-power DCT except a few differences in the common circuit. Therefore, it is possible to integrate both the forward and backward transforms into one architecture by suitably multiplexing the data path in the common circuit and the coefficients inside the modules.

3.2.4 Polyphase Representation

In the preceding discussions, we have shown how to perform the multirate DCT and IDCT by rearranging the z-transforms of the decimated sequences. Here we will show a systematic way to derive the results by applying the polyphase decomposition to the original IIR transfer function.

Substitute the identity that

$$\frac{1}{1 - 2\cos 2\omega_k z^{-1} + z^{-2}} = \frac{1 + 2\cos 2\omega_k z^{-1} + z^{-2}}{1 - 2\cos 4\omega_k z^{-2} + z^{-4}} \tag{3.43}$$

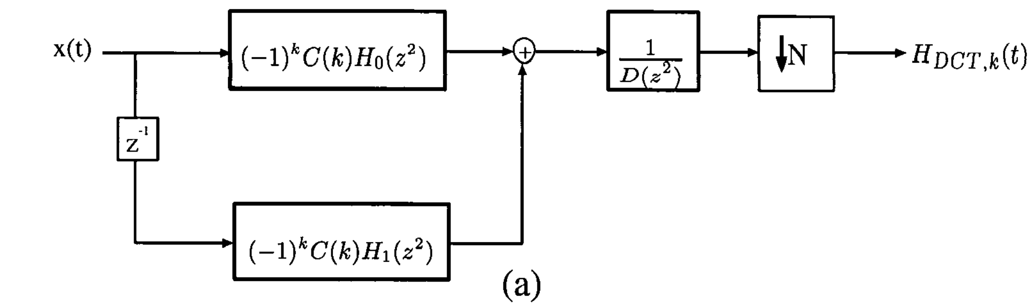
into the IIR DCT transfer function in (3.31). After rearrangement, $H_{DCT,k}(z)$ under block processing can be written as

$$H_{DCT,k}(z) = \frac{(-1)^k C(k)}{D(z^2)} \left[H_0(z^2) + z^{-1} H_1(z^2)\right] \tag{3.44}$$

where

$$\begin{aligned} D(z^2) &= 1 - 2\cos 4\omega_k z^{-2} + z^{-4}, \\ H_0(z^2) &= (\cos\omega_k - \cos 3\omega_k z^{-2}), \text{ and} \\ H_1(z^2) &= (\cos 3\omega_k - \cos\omega_k z^{-2}). \end{aligned} \tag{3.45}$$

(3.44) is the polyphase representation of $H_{DCT,k}(z)$ with $M = 2$, and its corresponding polyphase implementation is shown in Fig. 3.14(a). The downsampling operation $\downarrow N$ at the right end denotes that we pick up the DCT coefficients at the Nth clock cycle and ignore all the previous intermediate results. Given this polyphase implementation, we can use the noble identites to distribute the downsampling operation towards the left and obtain the structure depicted in Fig. 3.14(b), which will lead to the multirate DCT architecture shown in Fig. 3.10. Thus, we can process the input data at a two times slower clock rate. After $N/2$ iterations, the DCT coefficients are available at the output ends. Similarly, the case of $M = 4$ can be achieved by performing another polyphase decomposition on $\frac{1}{D(z^2)}$ in (3.44). After some algebraic simplifications, we can obtain



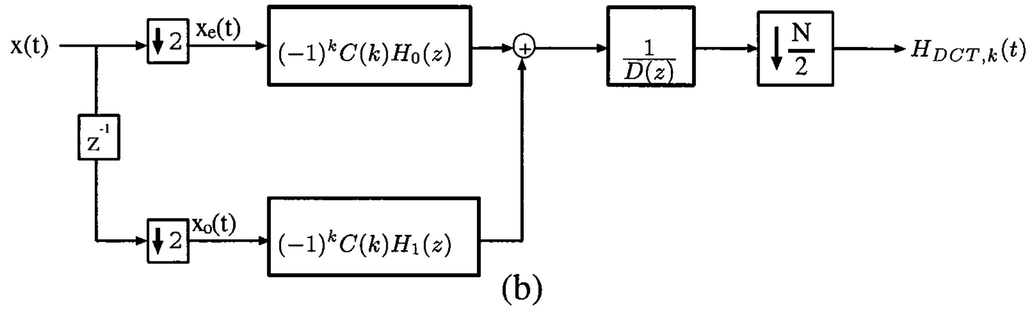


Figure 3.14: (a) Polyphase representation of $H_{DCT,k}(z)$. (b) Polyphase representation of $H_{DCT,k}(z)$ after applying the noble identity.

(3.38) and its corresponding implementation allows us to operate at a four times slower clock rate. The polyphase decomposition can also be used to derive the results for the multirate IDCT. In the next chapter, we will apply this methodology to obtain the low-power architecture of logarithmic complexity as well as the unified transformation module design.

3.3 Comparisons of Architectures

In this section, we will discuss the hardware complexity of the two algorithm-based low-power approaches (the Chebyshev polynomial approach and the polyphase decomposi-

	Normal Operation		Downsampling by 2 ($M = 2$)		Downsampling by 4 ($M = 4$)		Extra iteration
	Multiplier	Adder	Multiplier	Adder	Multiplier	Adder	
Chebyshev DCT	$2N-2$	$3N-1$	$3N-3$	$4N$	$5N-5$	$6N+3$	Yes
Chebyshev IDCT	$N+2$	$3N$	$2N+2$	$4N+1$	$4N+2$	$6N+4$	No
IIR DCT	$2N-2$	$2N$	$3N-3$	$3N+1$	$5N-5$	$5N+3$	No
IIR IDCT	$2N+1$	$3N$	$3N+1$	$4N+1$	$5N+1$	$6N+2$	No

Table 3.1: Comparison of hardware cost for the DCT and IDCT architectures with their low-power designs in terms of 2-input multipliers and 2-input adders.

tion approach) proposed in this chapter. Also, we will compare the proposed multirate SIPO architectures with the existing SIPO and PIPO architectures [19][32]. Table 3.1 summarizes the hardware costs for all the proposed architectures under normal operation and under multirate operation ($M = 2, 4$). As we can see, the hardware overhead of the low-power design is linear complexity increase for the speed compensation. As to the two approaches (Chebyshev and polyphase), the Chebyshev IDCT requires $(N-1)$ less multipliers than the IIR IDCT in both normal and multirate operations. This saving is in particular preferable for the applications which require cost-effective IDCT such as HDTV receivers. On the contrary, the Chebyshev DCT has almost the same complexity as the IIR DCT. Since the Chebyshev DCT needs one more iteration to finish the transform, the polyphase IIR DCT is a better choice for the implementations.

Next, we will compare our low-power DCT architecture with those proposed in [32] and [19]. The architecture in [32], which utilizes the factorization method to perform fast DCT, is a typical representative of the PIPO fast algorithms. The IIR structure proposed in [19], on the other hand, is a good example of the SIPO algorithms. A comparison regarding their inherent properties is listed in Table 3.2. The advantages of the SIPO approach over the PIPO approach in their VLSI implementation, such as local communication and linear hardware complexity increase, have been discussed thoroughly in [18] and [19]. Nevertheless, when the speed compensation capability is of concern, the PIPO is also a good choice since the block PIPO processing with a block size N is equivalent to decimating the input data by a factor of N. However, this advantage is obtained at the price of globally increased hardware and routing paths. Besides, the block size is usually restricted to be power of two due to the "divide-and-conquer" nature of those PIPO fast algorithms. From Table 3.2, 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, *i.e.*,

$$N = Mk, \quad k \in Z^+ \tag{3.46}$$

(M is the decimation factor and Z^+ denotes any positive integer), the choice of N is much more flexible compared with the PIPO algorithms.

The other advantage of the SIPO approaches is in the computation of the pruning DCT [33]. In DCT-based signal compression algorithms, the most useful information of the signal is kept in the low frequency DCT components. Therefore, retaining only $N_0 < N$ coefficients is sufficient for the lossy data compression. Although the pruning DCT can be computed from the PIPO DCT architecture by removing the unnecessary data paths and computational operators [33], the global communication is still the major drawback for its implementation as N increases. On the contrary, the SIPO architecture in [19] and our low-power design can be readily applied to the pruning DCT by simply implementing the first N_0 DCT modules for the computation of the first N_0 DCT coefficients.

	Liu *et. al.* [19]	Proposed multirate IIR DCT with $M = 4$	Lee [32]
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	Direct	Direct	Needs many modifications and global interconnections

Table 3.2: Comparison of different DCT architectures, where f_s denotes the data sample rate, M denotes the programmable downsampling factor, and N is the block size.

Chapter 4

Logarithmic Low-power Design and Unified Low-Power Transform Coding Architecture

In the previous chapter, we showed that the power consumption can be reduced provided that we can perform the DCT/IDCT from the decimated-by-M input sequences at $O(M)$ increase in hardware complexity. In practice, the $O(M)$ overhead may not be desirable when M is large and total chip area is limited. Therefore, the search for compensation scheme with less hardware overhead is desired. In this chapter, we will show a scheme to perform the polyphase decomposition in such a way that only $O(\log M)$ overhead is required to compensate the speed penalty. The resulting structure reduces the operating frequency on a stage-by-stage base: In each stage, the operating frequency is reduced by half. After reaching to the $(\log M)^{th}$ stage, we can operate at M-times slower clock rate of the original data rate. We shall refer to this as **logarithmic low-power design**. This multiple operation frequency environment allows us to perform different speed compensation at each stage; *i.e.*, different low supply voltages can be used to lower the power consumption. In general, the power savings of the logarithmic architecture is between the normal IIR architecture [19] and the full multirate architecture presented in the previous chapter.

Next we extend the low-power DCT/IDCT design to a larger class of orthogonal transforms. We start with the low-power design of the Modulated Lapped Transform (MLT) and Extended Lapped Transform (ELT). The MLT and ELT, which belong to the family of Lapped Orthogonal Transforms (LOT), are very attractive in the applications of transform coding since they can diminish the blocking effect encountered in low bit-rate block transforms [34][35][36]. Recently, Frantzeskakis *et al.* [37] proposed the time-recursive MLT and ELT architectures that are suitable for VLSI implementation due to their modularity and regularity. However, since the updating of the MLT and ELT coefficients should be as fast as the input data rate, those architectures cannot compensate the speed penalty under low supply voltage. In this chapter, we will derive the low-power time-recursive MLT and ELT structures. By applying the polyphase decomposition to their IIR transfer functions, the MLT/ELT coefficients can be updated at an M-times slower rate with linear hardware overhead; hence, the low-power operation is allowed. Later, based on the derivations of the MLT and ELT, we propose a unified low-power transform coding architecture. It can perform most of the existing discrete orthogonal transforms by simply setting the multiplier coefficients of the computational modules as well as the data paths of the module outputs.

The organization of the this chapter is as follows: Section 4.1 presents the low-power DCT architecture of logarithmic complexity. In Section 4.2, we derive the multirate MLT and ELT algorithms and architectures. In Section 4.3, the unified low-power transform coding architecture is described.

4.1 Low-Power Architecture of Logarithmic Complexity

In this section, we will show how to achieve low-power consumption with only logarithmic complexity overhead. The basic principle is to repeat the polyphase decomposition in a certain way instead of fully expanding them. By doing so, the lower-rate operations can be obtained while the complexity will grow slower. The price paid is that the resulting architecture will be operated at multiple low frequencies rather than at the uniform low frequency as discussed in the previous chapter. Nevertheless, the multiple frequency environment enables us to perform different speed compensations at different stages of the design. Therefore, different low supply voltages can be applied according to the given speed constraint, and the total power consumption can be still reduced. In what follows, we will derive the logarithmic low-power DCT architecture. The results can be extended to other low-power transformation designs to be discussed in Section 4.2 and 4.3.

4.1.1 Low-Power DCT Architecture of Logarithmic Complexity

The multirate IIR DCT transfer function with $M = 2$ can be written as

$$H_{DCT,k}(z) = \frac{(-1)^k C(k)}{D(z^2)} \left[H_0(z^2) + z^{-1} H_1(z^2)\right] \tag{4.1}$$

where $C(k)$ is the scaling factor of the DCT and

$$\begin{aligned} D(z^2) &= 1 - 2\cos 4\omega_k z^{-2} + z^{-4}, \\ H_0(z^2) &= (\cos\omega_k - \cos 3\omega_k z^{-2}), \text{ and} \\ H_1(z^2) &= (\cos 3\omega_k - \cos\omega_k z^{-2}). \end{aligned} \tag{4.2}$$

Substituting the polyphase decomposition

$$\frac{1}{D(z^2)} = \frac{H_0'(z^4) + z^{-2} H_1'(z^4)}{D'(z^4)} \tag{4.3}$$

with

$$\begin{aligned} D'(z^4) &= 1 - 2\cos 8\omega_k z^{-4} + z^{-8}, \\ H_0'(z^4) &= 1 + z^{-4}, \text{ and} \\ H_1'(z^4) &= 2\cos 4\omega_k, \end{aligned} \tag{4.4}$$

into (4.1) and rearranging, we can rewrite $H_{DCT,k}(z)$ so that the DCT can be computed at four times slower clock rate. Nevertheless, this multirate design requires $O(M)$

hardware overhead to directly lower the input clock rate by four. In order to save the hardware complexity, we may rewrite (4.1) in a cascade form after the substitution is made, *i.e.*,

$$H_{DCT,k}(z) = (-1)^k C(k) \left[H_0(z^2) + z^{-1}H_1(z^2)\right] \left[H_0'(z^4) + z^{-2}H_1'(z^4)\right] \cdot \frac{1}{D'(z^4)}. \quad (4.5)$$

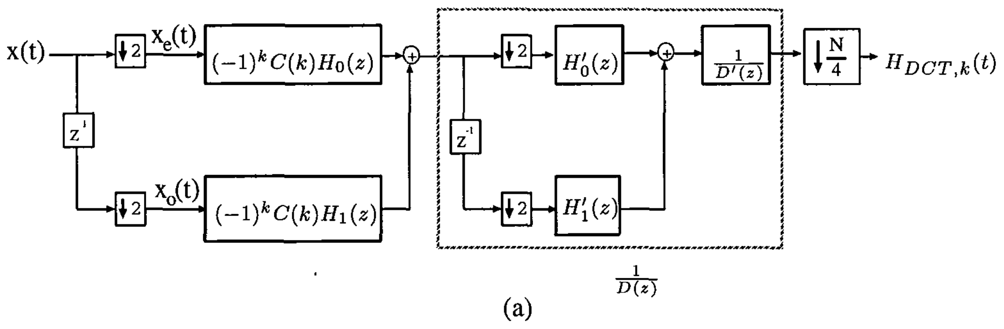
Figure 4.1(a) shows the polyphase implementation of (4.5), which leads to the cascade multirate DCT architecture depicted in Fig. 4.1(b). There are two major blocks. One operates at 50% sample rate and the other at 25% sample rate. Due to the special form of the denominator of the transfer function, we can repeatedly perform the polyphase decomposition on the denominator and retain the cascade form. We then have

$$H_{DCT,k}(z) = (-1)^k C(k) \left[H_0(z^2) + H_1(z^2)\right] \frac{\prod_{i=1}^{\log M-1} \left[(1 + z^{-2^{i+1}}) + 2z^{-2^i} \cos(2^{i+1}\omega_k)\right]}{1 - 2\cos(2M\omega_k)z^{-M} + z^{-2M}} \quad (4.6)$$

for any M, $M \in 2^{Z^+}$. The resulting architecture decimates the operating frequency on a stage-by-stage base: In each stage, the operating frequency is reduced by half. After reaching the $(\log M)^{th}$ stage, we will have M times slower clock rate of the original data rate.

4.1.2 Power Consumption

When low-power implementation is taken into consideration, the feature of multiple operating frequencies in the above architecture implies that different supply voltages will be used according to the slowest allowable operating speed. That is, the operators to realize $H_0(z^2)$ and $H_1(z^2)$ in (4.5) can be operated at $3.1V$ due to the two times slower clock rate, while all other operators to realize $H_0'(z^4)$ and $H_1'(z^4)$ can be operated at $2.1V$ due to the four times slower clock rate (from 2.15). As a consequence, the power consumption of the 16-point low-power DCT architecture in Fig. 4.1(b) can be estimated



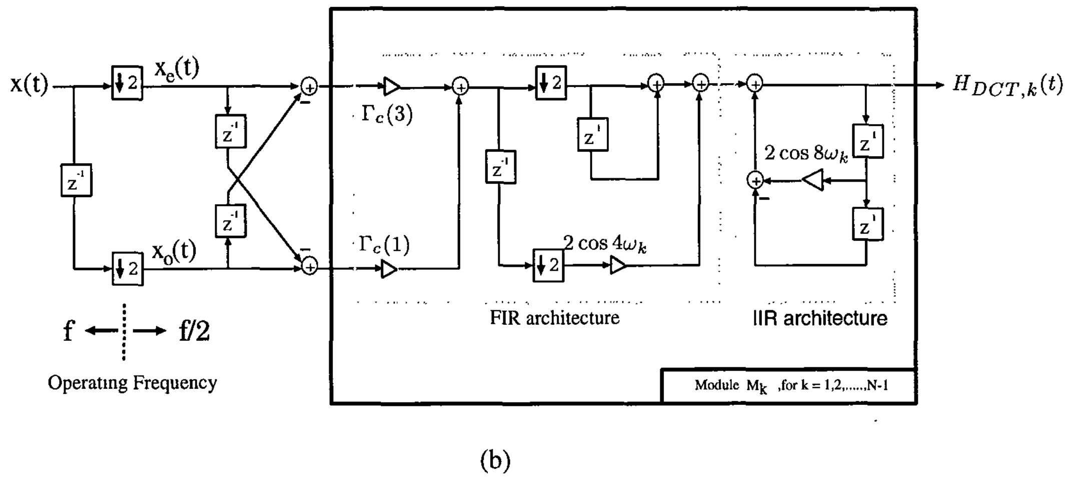


Figure 4.1: (a) Polyphase representation of $H_{DCT,k}(z)$ in cascade form. (b) Multirate DCT architecture with logarithmic complexity.

	Normal DCT architecture in [19]	Logarithmic low-power DCT architecture	Full low-power DCT architecture in Chap. 3
Multipliers	$2N-2$	$(\log M+2)N$ (in order)	$(M+1)N$ (in order)
Adders	$2N$	$(2\log M+1)N$ (in order)	$(M+1)N$ (in order)
Power consumption for 16-point DCT	P_0	$0.24P_0$ $(M=4)$	$0.11P_0$ $(M=4)$

Table 4.1: Comparison of hardware cost and power consumption of the logarithmic low-power DCT architecture with other approaches.

as

$$(\frac{N_2}{N_0}C_{eff})(\frac{3.1V}{5V})^2(\frac{1}{2}f)+(\frac{N_4}{N_0}C_{eff})(\frac{2.1V}{5V})^2(\frac{1}{4}f)\approx 0.24P_0, \tag{4.7}$$

where $N_0=30$ is the total multipliers used in the normal DCT ($M=0$); $N_2=30$ and $N_4=30$ are the number of multipliers in the $M=2$ stage and $M=4$ stage, respectively. From (4.7), we can see that the overall power consumption of the logarithmic low-power design will be in between $M=2$ and $M=4$ of the full multirate DCT systems discussed in the previous chapter.

On the other hand, by examing (4.6), we can see that in order to have M-times slower operating frequency at the final stage, we need a total of $(\log M+2)$ multipliers to realize the multirate transfer function. The comparison of the logarithmic low-power architecture with other approaches is listed in Table 4.1. Although the total power savings of the logarithmic structure is less than that of the full multirate structure given the same decimation factor M, the $O(\log M)$ hardware overhead is preferable when we want to achieve low-power consumption without trading too much chip area.

The multiple-frequency feature of the cascade low-power architecture also allows us to achieve more power and area savings at the arithmetic level. For example, we can use look-ahead adders in the $M=2$ region to match the data throughput rate, whereas we can employ low-speed carry-ripple adders in the $M=4$ region due to the much relaxed speed constraint.

4.2 Low-Power MLT and ELT Architectures

4.2.1 The IIR MLT Algorithm

The 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, $X_{MLT,k}(t)$, $k = 0, 1, \cdots, N-1$, as follows [35]:

$$X_{MLT,k}(t) = S(k)\sqrt{\frac{2}{N}}\sum_{n=0}^{2N-1} \sin\frac{\pi}{2N}(n+\frac{1}{2})\cos[\frac{\pi}{N}(k+\frac{1}{2})(n+\frac{1}{2}+\frac{N}{2})]\,x(t+n-2N+1) \tag{4.8}$$

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 [37]

$$X_{MLT,k}(t) = -S(k)[\,X_{C,k+1}(t) + X_{S,k}(t)\,], \tag{4.9}$$

where

$$X_{C,k}(t) \triangleq \beta_1 \sum_{n=0}^{L-1} \cos[(2n+1)\omega_k + \theta_k]x(t+n-2N+1), \tag{4.10}$$

$$X_{S,k}(t) \triangleq \beta_1 \sum_{n=0}^{L-1} \sin[(2n+1)\omega_k + \theta_k]x(t+n-2N+1), \tag{4.11}$$

with block size $L = 2N$ and

$$\beta_1 \triangleq \frac{1}{\sqrt{2N}}, \quad \omega_k \triangleq \frac{\pi k}{2N}, \quad \text{and} \quad \theta_k \triangleq \frac{\pi}{2}(k+\frac{1}{2}). \tag{4.12}$$

The IIR transfer functions for (4.10) and (4.11) can be computed as

$$H_{C,k}(z) = \beta_1(1-z^{-L})\frac{\cos((2L-1)\omega_k+\theta_k) - \cos((2L+1)\omega_k+\theta_k)z^{-1}}{1-2\cos 2\omega_k z^{-1} + z^{-2}}, \tag{4.13}$$

$$H_{S,k}(z) = \beta_1(1-z^{-L})\frac{\sin((2L-1)\omega_k+\theta_k) - \sin((2L+1)\omega_k+\theta_k)z^{-1}}{1-2\cos 2\omega_k z^{-1} + z^{-2}}. \tag{4.14}$$

The corresponding IIR module for the dual generation of $X_{C,k}(t)$ and $X_{S,k}(t)$ is depicted in Fig. 4.2, where

$$\Gamma_1 \triangleq \beta_1\cos((2L-1)\omega_k+\theta_k), \quad \Gamma_2 \triangleq -\beta_1\cos((2L+1)\omega_k+\theta_k),$$
$$\Gamma_3 \triangleq \beta_1\sin((2L-1)\omega_k+\theta_k), \quad \Gamma_4 \triangleq -\beta_1\sin((2L+1)\omega_k+\theta_k). \tag{4.15}$$

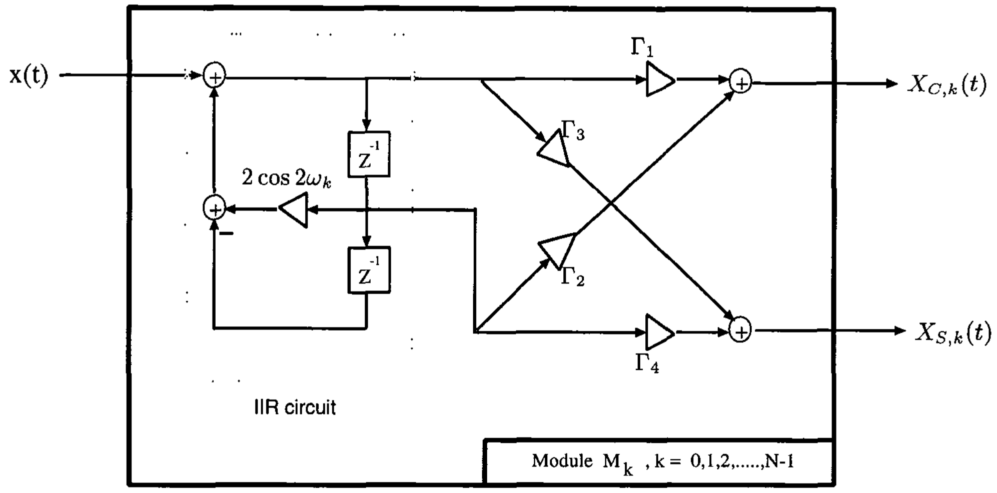


Figure 4.2: IIR MLT module design.

This IIR module can be used as a basic building block to implement MLT according to (4.9). Figure 4.3 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 **combination circuit** which selects and combines the outputs of the IIR array to generate the MLT coefficients.

4.2.2 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 $X_{S,k}(t)$ using the decimated input sequences. After performing the polyphase decomposition on (4.13) and (4.14), we can compute the multirate IIR transfer functions for $H_{C,k}(z)$ and $H_{S,k}(z)$ as

$$\begin{aligned} H_{C,k}(z) &= \frac{\beta_1(1-z^{-L/2})}{1-2\cos(4\omega_k)z^{-1}+z^{-2}} \times \\ &\quad \Big([\cos((2L-3)\omega_k+\theta_k) - \cos((2L+1)\omega_k+\theta_k)z^{-1}]X_e(z) \\ &+ [\cos((2L-1)\omega_k+\theta_k) - \cos((2L+3)\omega_k+\theta_k)z^{-1}]X_o(z)\Big), \qquad (4.16) \end{aligned}$$

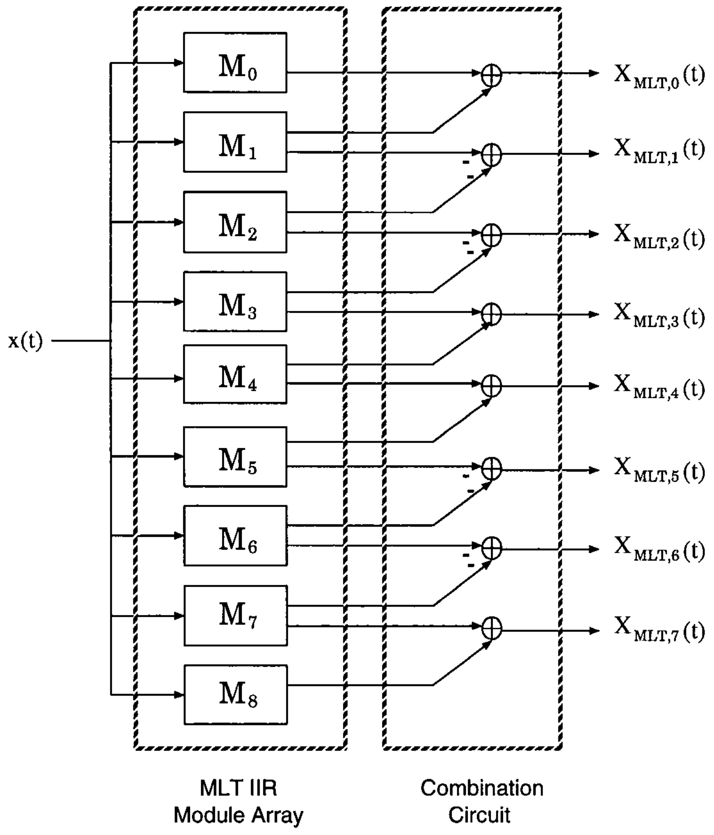


Figure 4.3: The time-recursive MLT architecture.

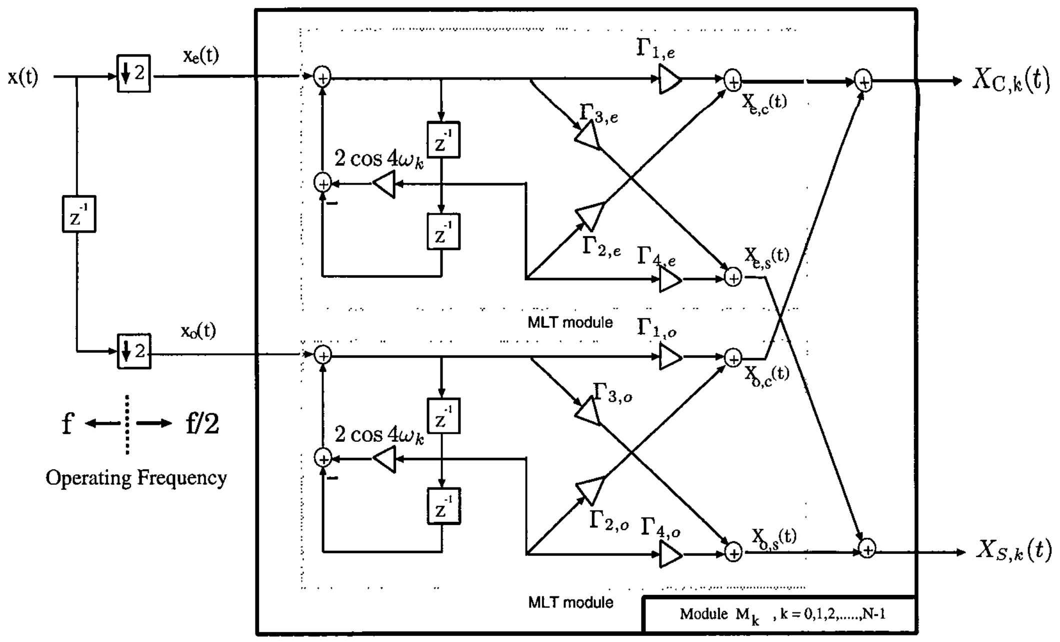


Figure 4.4: Low-power IIR MLT module design.

and

$$\begin{aligned} H_{S,k}(z) &= \frac{\beta_1(1-z^{-L/2})}{1-2\cos(4\omega_k)z^{-1}+z^{-2}} \times \\ & \Big([\sin((2L-3)\omega_k+\theta_k)-\sin((2L+1)\omega_k+\theta_k)z^{-1}]X_e(z) \\ &+ [\sin((2L-1)\omega_k+\theta_k)-\sin((2L+3)\omega_k+\theta_k)z^{-1}]X_o(z)\Big). \end{aligned} \quad (4.17)$$

The parallel architecture for (4.16) and (4.17) is shown in Fig. 4.4, where

$$\begin{aligned} \Gamma_{1,e} &= \beta_1\cos((2L-3)\omega_k+\theta_k), & \Gamma_{2,e} &= -\beta_1\cos((2L+1)\omega_k+\theta_k), \\ \Gamma_{3,e} &= \beta_1\sin((2L-3)\omega_k+\theta_k), & \Gamma_{4,e} &= -\beta_1\sin((2L+1)\omega_k+\theta_k), \\ \Gamma_{1,o} &= \beta_1\cos((2L-1)\omega_k+\theta_k), & \Gamma_{2,o} &= -\beta_1\cos((2L+3)\omega_k+\theta_k), \\ \Gamma_{3,o} &= \beta_1\sin((2L-1)\omega_k+\theta_k), & \Gamma_{4,o} &= -\beta_1\sin((2L+3)\omega_k+\theta_k). \end{aligned} \quad (4.18)$$

It consists of two MLT modules in Fig. 4.2. 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

	Normal Operation		Downsampling by 2		Downsampling by 4	
	Multiplier	Adder	Multiplier	Adder	Multiplier	Adder
IIR MLT	$5N$	$5N$	$10N$	$11N$	$20N$	$23N$
IIR ELT	$6N$	$6N$	$11N$	$12N$	$21N$	$24N$

Table 4.2: Comparison of hardware cost for the MLT and ELT with their low-power designs in terms of 2-input multipliers and 2-input adders.

the even and odd outputs. Through such manipulation, only decimated sequences are processed inside the module. Hence, the MLT module can operate at the half of the original frequency by doubling the hardware complexity. The comparison of hardware cost is shown in Table 4.2. Suppose that P_0 denotes the power consumption of the MLT module in Fig. 4.2. From the CMOS power model, it can be shown that the power consumption for the low-power MLT modules are $0.38P_0$ and $0.17P_0$ for the case of $M = 2$ and $M = 4$, respectively. Basically, this savings is obtained at the expense of linear increase in hardware.

4.2.3 Low-Power Design of the ELT

The ELT with basis length equal to $4N$ operates on data segment of length $4N$, $x(t + n - 4N + 1), n = 0, 1, \cdots, 4N - 1$, and produces N output coefficients, $X_{ELT,k}(t), k = 0, 1, \cdots, N - 1$. One good choice for the ELT is as follows [38][36]:

$$X_{ELT,k}(t) = \sqrt{\frac{2}{N}} \sum_{n=0}^{4N-1} [\frac{1}{2\sqrt{2}} - \frac{1}{2}\cos\frac{\pi}{N}(n+\frac{1}{2})] \cos[\frac{\pi}{N}(k+\frac{1}{2})(n+\frac{1}{2}+\frac{N}{2})]\, x(t+n-4N+1) \tag{4.19}$$

By the use of some trigonometric identities, we can rewrite (4.19) as

$$X_{ELT,k}(t) = -\tilde{X}_{S,k+1}(t) + \sqrt{2}\tilde{X}_{C,k}(t) + \tilde{X}_{S,k-1}(t), \tag{4.20}$$

where

$$\tilde{X}_{C,k}(t) \triangleq \beta_2 \sum_{n=0}^{L-1} \cos[(2n+1)\omega_k' + \theta_k']\, x(t+n-4N+1), \tag{4.21}$$

$$\tilde{X}_{S,k}(t) \triangleq \beta_2 \sum_{n=0}^{L-1} \sin[(2n+1)\omega_k' + \theta_k']\, x(t+n-4N+1), \tag{4.22}$$

with

$$L = 4N, \quad \beta_2 \triangleq \frac{1}{2\sqrt{2N}}, \quad \omega_k' \triangleq \frac{\pi}{2N}(k+\frac{1}{2}), \text{ and } \theta_k' \triangleq \frac{\pi}{2}(k+\frac{1}{2}). \tag{4.23}$$

Define the relationship in (4.9) and (4.20) as the **combination functions**. After comparing (4.9)-(4.12) with (4.20)-(4.23), 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. 4.2, as well as the low-power MLT module in Fig. 4.4, 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. 4.3 except that the combination circuit performs according to (4.20).

Moreover, it can be verified that $X_{S,-1}(t) = -X_{S,0}(t)$ and $X_{S,N}(t) = X_{S,N-1}(t)$. Hence, we can compute the 0^{th} and $(N-1)^{th}$ ELT coefficients from

$$\begin{aligned} X_{ELT,0}(t) &= -X_{S,1}(t) + \sqrt{2}X_{C,0}(t) - X_{S,-1}(t), \\ X_{ELT,N-1}(t) &= -X_{S,N-1}(t) + \sqrt{2}X_{C,N-1}(t) + X_{S,N-2}(t), \end{aligned} \tag{4.24}$$

instead of implementing two extra ELT modules for $X_{S,-1}(t)$ and $X_{S,N}(t)$. The hardware cost for the ELT can be found in Table 4.2. 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.

4.3 Unified Low-Power Transform Coding Archiecture

From the transform functions described in (4.9)-(4.12) and (4.20)-(4.23), we observe that the low-power MLT module in Fig. 4.4 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 (4.10) is equivalent to the DCT by setting

$$L = N, \quad \beta_1 = C(k), \quad \omega_k = \frac{k\pi}{2N}, \text{ and } \theta_k = 0. \tag{4.25}$$

As a result, the multirate MLT module in Fig. 4.4 can compute the DCT with different index k in parallel.

The other example is the discrete Fourier transform (DFT) with real-valued inputs. With the following parameter setting

$$L = N, \quad \beta_1 = \frac{1}{\sqrt{N}}, \quad \omega_k = \frac{-k\pi}{N}, \text{ and } \theta_k = -\omega_k, \tag{4.26}$$

(4.10) and (4.11) become

$$X_{C,k}(t) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} \cos(\frac{-2\pi}{N} kn)\, x(t+n-N+1), \tag{4.27}$$

$$X_{S,k}(t) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} \sin(\frac{-2\pi}{N} kn)\, x(t+n-N+1), \tag{4.28}$$

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 combination circuit in Fig. 4.3 performs as

$$X_{DHT,k}(t) = X_{C,k}(t) + X_{S,k}(t). \tag{4.29}$$

The parameter settings as well as the corresponding combination functions for other orthogonal transforms are summarized in Table 4.3.

The programmable feature of the unified low-power module design makes it very attractive in transform coding applications. Firstly, the unified structure can be implemented as a high-performance programmable co-processor which performs various transforms for the host processor by loading the suitable parameters. Secondly, by hard-wiring the multiplier coefficients of the modules to preset values according to the

	L	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	$C(1)$	$\frac{\pi}{2N}(k+\frac{1}{2})$	$-\omega_k$	$X_{IDCT,k}(t) = X_{C,k}(t) +$ $(C(0) - C(1))x(n-N+1)$
DST-IV in [39]	N	$C(1)$	$\frac{\pi}{2N}(k+\frac{1}{2})$	0	$X_{DST,k}(t) = X_{S,k}(t)$
IDST-IV in [39]	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	$\frac{1}{\sqrt{N}}$	$\frac{-k\pi}{N}$	$-\omega_k$	$X_{DHT,k}(t) = X_{C,k}(t) + X_{S,k}(t)$.

Table 4.3: Parameter settings for the unified low-power IIR transformation architecture, where $Re\{X_{DFT,k}(t)\}$ and $Im\{X_{DFT,k}(t)\}$ denote the the real part and the imaginary part of the DFT, respectively.

transformation type, we can perform any one of the discrete sinusoidal transforms using the same architecture. This can significantly reduce the design cycle as well as the manufacturing cost.

Chapter 5

Finite-Precision Analysis of The Low-Power DCT Architectures

In low-power VLSI implementation, the choice of wordlength is an important issue since it will directly affect the total switching activities inside the operators as well as the total effective capacitance. Besides, an underestimated wordlength will degrade the system performance due to the increased rounding errors. Therefore, we should carefully determine the minimum allowable system wordlength that meets the accuracy criteria for cost-effective and power-saving VLSI implementations.

In this chapter, we perform the finite-wordlength analysis for the proposed low-power DCT architectures. The results can be easily extended to other transform architectures. We start with the DCT architecture under the normal operation, then the analysis is extended to the low-power design with $M = 2$. The general results for arbitrary M is also presented. Throughout the derivations, the "statistical error model" for fixed-point analysis is used [40, Chap.6]:

1. The rounding error is treated as wide-stationary additive white noise with magnitude uniformly distributed over one quantization level.
2. Rounding error occurs only in multiplication.

3. All errors are uncorrelated with the input signal, and are independent of each other.

The organization of this chapter is as follows. In Section 5.1, we describe the two basic issues in finite-precision analysis; *i.e.*, the rounding error and dynamic range. The analyses of the normal IIR DCT architecture using direct form I structure and direct form II structure are discussed in Section 5.2 and 5.3, respectively. The analyses of the multirate IIR DCT architecture are presented in Section 5.4. In Section 5.5, the proposed analytical results are verified by extensive computer simulations.

5.1 Basic Considerations in Finite-Precision Analysis

There are two basic considerations in the fixed-point analysis. One is the **rounding error** behavior. It occurs when we multiply two $(B+1)$-bit numbers together while only $(B+1)$-bit product is kept. The mean and variance of the rounding error are given by [40, Chap.6]

$$m_R = 0, \qquad \sigma_R^2 = \frac{2^{-2B}}{12}, \tag{5.1}$$

respectively. Understanding the rounding error behavior will allow us to minimize the wordlength to achieve a desired output signal-to-noise ratio.

The other is the **dynamic range** issue. In fixed-point implementation, each number in the system is treated as a fraction. The magnitude of each node in the circuit cannot exceed one, otherwise overflow occurs and will result in great distortion in the final output. Therefore, to prevent overflow, a suitable scaling of the input signal is usually employed according to the dynamic range of the system. In practice, the signal-to-noise ratio of the scaled system, SNR', will be degraded by the scaling process and is given by [40, Chap.6]

$$SNR' = s^2 SNR_0, \tag{5.2}$$

where s is the scaling factor and SNR_0 is the signal-to-noise ratio of the original system. This implies that knowing the dynamic range will enable us to perform minimally

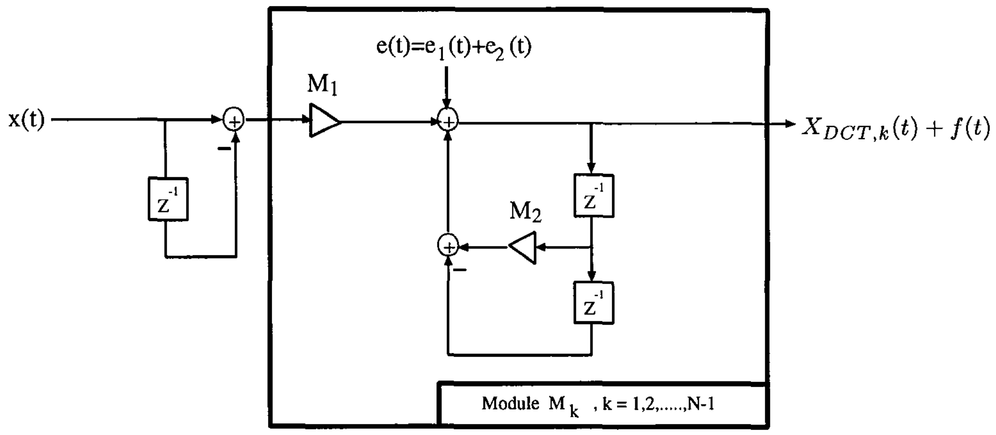


Figure 5.1: Rounding error in the IIR DCT architecture.

necessary scaling to prevent further degradation in SNR.

5.2 IIR DCT Using Direct Form I Structure

5.2.1 Rounding Errors

Using the statistical error model, the rounding error of the IIR DCT structure can be modeled as (see Fig. 5.1)

$$e(t) = e_1(t) + e_2(t) \tag{5.3}$$

where $e_i(t)$, $i = 1, 2$ is the rounding error caused by the i^{th} multiplier in the circuit [1]. Then the actual output of the DCT circuit after N iterations can be represented as

$$\hat{X}_{DCT,k}(t) = X_{DCT,k}(t) + f(t) \tag{5.4}$$

where $f(t)$ is the output error due to the noise error $e(t)$.

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

[1]Since the 0^{th} channel of the DCT is computed by a simple add-and-accumulate operation, we will not consider the finite-wordlength effect of this channel.

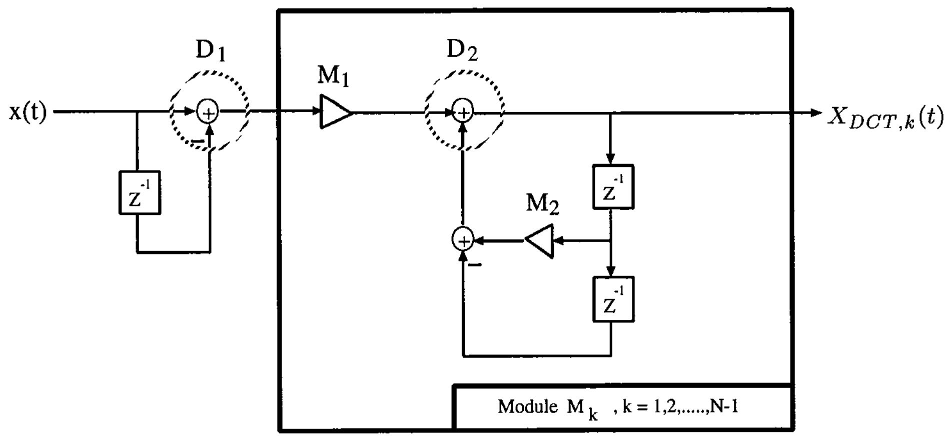


Figure 5.2: Dynamic range of the IIR DCT architecture.

Fig. 5.1, $H_{ef}(z)$ is given by

$$H_{ef}(z) = \frac{1}{1 - 2\cos 2\omega_k z^{-1} + z^{-2}}, \tag{5.5}$$

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

$$h_{ef}(n) = \frac{1}{\sin(2\omega_k)} \sin[(n+1)2\omega_k]u(n) \tag{5.6}$$

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 k^{th} DCT channel can be computed as

$$\begin{aligned} m_f &= m_e \sum_{n=0}^{N-1} h_{ef}(n) = m_e \sum_{n=0}^{N-1} \frac{1}{\sin(2\omega_k)} \sin[(n+1)2\omega_k], & (5.7) \\ \sigma_f^2 &= \sigma_e^2 \sum_{n=0}^{N-1} |h_{ef}(n)|^2 = \frac{\sigma_e^2}{\sin^2(2\omega_k)} \sum_{n=0}^{N-1} \sin^2[(n+1)2\omega_k] = \frac{\sigma_e^2}{\sin^2(2\omega_k)} \left(\frac{N}{2}\right), & (5.8) \end{aligned}$$

where

$$\begin{aligned} m_e &= E\{e(t)\} = 0, & (5.9) \\ \sigma_e^2 &= E\{e^2(t)\} = E\{(e_1(t))^2\} + E\{(e_2(t))^2\} = (1 + N_s(k)) \cdot \sigma_R^2, & (5.10) \end{aligned}$$

and $N_s(k)$ is the number of the noise sources contributed by the multiplier $M_2 =$

$2\cos(2\omega_k)$ in the IIR loop:

$$N_s(k) = \begin{cases} 4, & \text{if } |2\cos(2\omega_k)| > 1, \\ 1, & \text{if } |2\cos(2\omega_k)| < 1, \\ 0, & \text{if } |2\cos(2\omega_k)| = 1. \end{cases} \tag{5.11}$$

When $|2\cos(2\omega_k)| < 1$, a normal multiplication is performed and $E\{(e_2(t))^2\} = \sigma_R^2$. In the case of $|2\cos(2\omega_k)| > 1$, since a left-shift is performed after the multiplication with $\cos(2\omega_k)$, the rounding error is amplified by two and its power becomes $E\{(2e_2(t))^2\} = 4 \cdot \sigma_R^2$. In the case of $|2\cos(2\omega_k)| = 1$, no multiplication is performed, hence $E\{(e_2(t))^2\} = 0$. Now using (5.7)-(5.11), we can represent the total noise power at the k^{th} DCT channel as

$$P_f = m_f^2 + \sigma_f^2 = \frac{N(N_s(k)+1)}{2\sin^2(2\omega_k)} \left(\frac{2^{-2B}}{12}\right). \tag{5.12}$$

As we can see, given the system wordlength B, the rounding error grows linearly with the block size N. This 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 wordlength. 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 0 or π, the larger the rounding error is. As a consequence, the first and last DCT channels suffer most from the finite-wordlength 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 (*cf*, Fig. 7 in [41]).

5.2.2 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. 5.2 can be

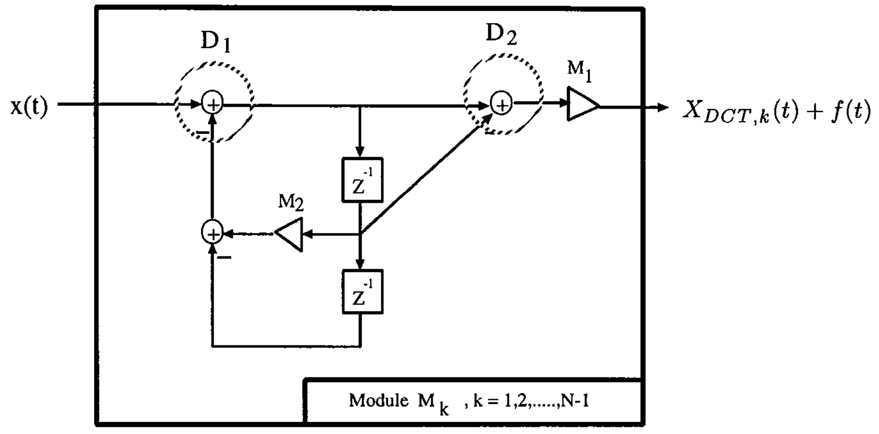


Figure 5.3: IIR DCT using the direct form II structure.

computed as

$$D_1 = 2, \tag{5.13}$$

$$\begin{aligned} D_2 &= \max\{X_{DCT,k}(t)\} = \max\{C(k)\textstyle\sum_{n=0}^{N-1}\cos[(2n+1)\omega_k]x(n)\} \\ &= C(k)\textstyle\sum_{n=0}^{N-1}|\cos[(2n+1)\omega_k]|\cdot\max\{x(n)\} = C(k)\textstyle\sum_{n=0}^{N-1}|\cos[(2n+1)\omega_k]| \end{aligned} \tag{5.14}$$

and the dynamic range of the overall architecture is given by

$$D = \max\{D_1, D_2\}. \tag{5.15}$$

Suppose that a one-time scaling scheme is provided at the input end to avoid overflow, and it is done by shifting the data to the right by K bits. We have

$$K = \lceil \log_2 D \rceil, \tag{5.16}$$

and the scaling factor s is given by

$$s = \frac{1}{2^K}. \tag{5.17}$$

5.2.3 Optimal Wordlength Assignment

Assume that the input sequence $x(t)$ is uniformly distributed over $(-1, 1)$ with zero mean. From (5.2), (5.12), and (5.17), we have

$$SNR' = s^2 \frac{E\{(X_{DCT,k}(t))^2\}}{P_f} = \frac{8\sin^2(2\omega_k)}{N(N_s(k)+1)} \cdot 2^{2B-2K} \tag{5.18}$$

where the fact that [41]

$$E\{(X_{DCT,k}(t))^2\} = E\{x^2(t)\} = 1/3, \quad k = 1, 2, \ldots, N-1, \tag{5.19}$$

is used. If we want to achieve a performance of 40 dB in SNR for the k^{th} DCT component, the optimal wordlength B_k for that channel can be computed from (5.18) as

$$B_k = \left\lceil \frac{4 - \log_{10}[\sin^2(2\omega_k) \cdot \frac{8}{N(N_s(k)+1)}]}{2 \cdot \log_{10} 2} + K \right\rceil . \tag{5.20}$$

As an example, the B_k's for the case $N = 8$ and 16 under the constraint SNR $= 40$ dB are listed in Table 5.1(a), where B_A denotes the average system wordlength. As we can see, $B_A = 12$ bit is sufficient to meet the accuracy criteria. Compared with the DCT implemenation in [42], in which B_A was chosen to be 16 bit based on the experimental simulation results, our system wordlength is much shorter. Suppose that the silicon area of the multiplier is dominant in the chip and the size of the multipliers is proportional to $(B_A)^2$. Using the optimal wordlengths in Table 5.1, we can reduce the total chip area to 56% of the original design without degrading the SNR performance. This shows that our analysis approach provides more insights to determine the architectural specifications than the experimental approach. Moreover, in the applications of transform coding, we can shorten the wordlengths for the high-frequency channels since the human vision system is less sensitive to these components. Thus, the total wordlength can be further reduced.

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)

Table 5.1: Optimal wordlength assignment under the constraint SNR = 40dB, where B_A is the average wordlength. (a) Normal IIR DCT. (b) Low-power DCT with $M = 2$.

5.3 IIR DCT Using Direct Form II Structure

Given the IIR DCT transfer function, we can also implement it using the direct form II structure as shown in Fig. 5.3. Following the above derivations for the direct form I structure, the fixed-point analytical results can be derived as:

1. Rounding error:
$$P_f = (N_s(k)+1)\frac{2^{-2B}}{12}. \tag{5.21}$$

2. The dynamic range:
$$\begin{aligned} D_1 &= \frac{1}{|\sin(2\omega_k)|}\sum_{n=0}^{N-1}|\sin[(2n+1)\omega_k]|, \\ D_2 &= 2, \\ D &= \max\{D_1, D_2\}. \end{aligned} \tag{5.22}$$

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 non-uniform 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 4.5)–It not only requires different scaling scheme in each DCT channel, but also makes the data interface between VLSI modules complicated (*e.g.*, 2-D DCT in which two DCT modules are connected [43].). Therefore, the direct form I is a better choice for the VLSI implementation of the IIR DCT structures.

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

In the low-power IIR DCT architecture with $M = 2$, the injected rounding error can be modeled as (see Fig. 5.4)
$$e(t) = e_1(t) + e_2(t) + e_3(t) \tag{5.23}$$

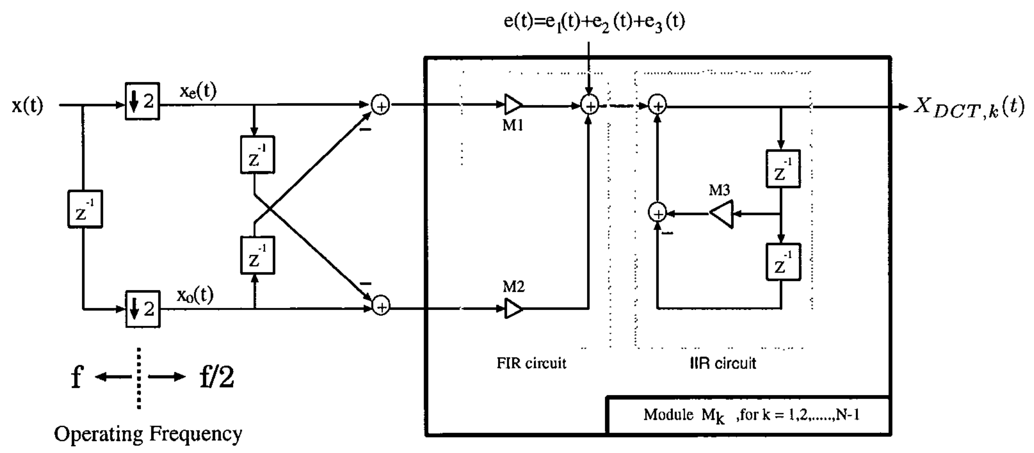


Figure 5.4: Rounding noise in the low-power IIR DCT architecture with $M = 2$.

and its power is given by

$$\sigma_e^2 = E\{e^2(t)\} = (2 + N_s(k))\sigma_R^2. \tag{5.24}$$

Note that

$$H_{ef}(z) = \frac{1}{1 - 2\cos 4\omega_k z^{-1} + z^{-2}}, \tag{5.25}$$

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

$$\sigma_f^2 = \sigma_e^2 \sum_{n=0}^{N/2-1} |h_{ef}(n)|^2 = \frac{\sigma_e^2}{\sin^2(4\omega_k)}\left(\frac{N}{4}\right) = \frac{(2 + N_s(k))N\sigma_R^2}{4\sin^2(4\omega_k)}. \tag{5.26}$$

From (5.26), we observe that

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 $\frac{1}{\sin(2\omega_k)^2}$ in (5.12), the factor $\frac{1}{\sin(4\omega_k)^2}$ in (5.26) will have similar effect on the SNR of each DCT channel but with halved period.

Now let us consider the dynamic range of the low-power DCT structure with $M = 2$. Given the assumption that the input sequence $x(t)$ is an i.i.d. sequence, the decimated inputs $x_e(t)$ and $x_o(t)$ are also i.i.d. sequences and are uncorrelated with each other. Thus, we can apply the technique of "superposition" to analyze the dynamic range of the system as follows.

Setting $x_o(t)$ to zero, Fig. 5.4 is reduced to the IIR structure depicted in Fig. 5.5, where $w_i(t)$, $i = 1, 2$, are the nodes that may have overflow. It is easy to see that

$$D_{1,e} = \max\{w_1(t)\} = C(k)\left(|\cos\omega_k| + |\cos 3\omega_k|\right). \tag{5.27}$$

From the transfer function of $w_2(t)$

$$H_2(z) = \frac{W_2(z)}{X_e(z)} = C(k)\frac{\cos 3\omega_k - \cos\omega_k z^{-1}}{1 - 2\cos 4\omega_k z^{-1} + z^{-2}}, \tag{5.28}$$

we can derive the unit-sample response as

$$h_2(n) = C(k)\cos[(4n+1)\omega_k]u(n). \tag{5.29}$$

Thus,

$$\begin{aligned} D_{2,e} &= \max\{w_2(t)\} \\ &= C(k)\sum_{n=0}^{N/2-1} |\cos[(4n+1)\omega_k]| \cdot \max\{x(n)\} \\ &= C(k)\sum_{n=0}^{N/2-1} |\cos[(4n+1)\omega_k]|. \end{aligned} \tag{5.30}$$

Similarly, by setting $x_e(t) = 0$, we can derive the dynamic ranges of the two circled nodes, $D_{1,o}$ and $D_{2,o}$, as

$$\begin{aligned} D_{1,o} &= C(k)\left(|\cos\omega_k| + |\cos 3\omega_k|\right), \\ D_{2,o} &= C(k)\textstyle\sum_{n=0}^{N/2-1} |\cos[(4n+3)\omega_k]|. \end{aligned} \tag{5.31}$$

Combining (5.30) and (5.31) together, we can write the overall dynamic range of the multirate DCT architecture as

$$\begin{aligned} D_1 &= D_{1,e} + D_{1,o} = 2C(k)\left(|\cos\omega_k| + |\cos 3\omega_k|\right), \\ D_2 &= D_{2,e} + D_{2,o} = C(k)\sum_{n=0}^{N/2-1} \left(|\cos[(4n+1)\omega_k]| + |\cos[(4n+3)\omega_k]|\right), \\ D &= \max\{D_1, D_2\}. \end{aligned} \tag{5.32}$$

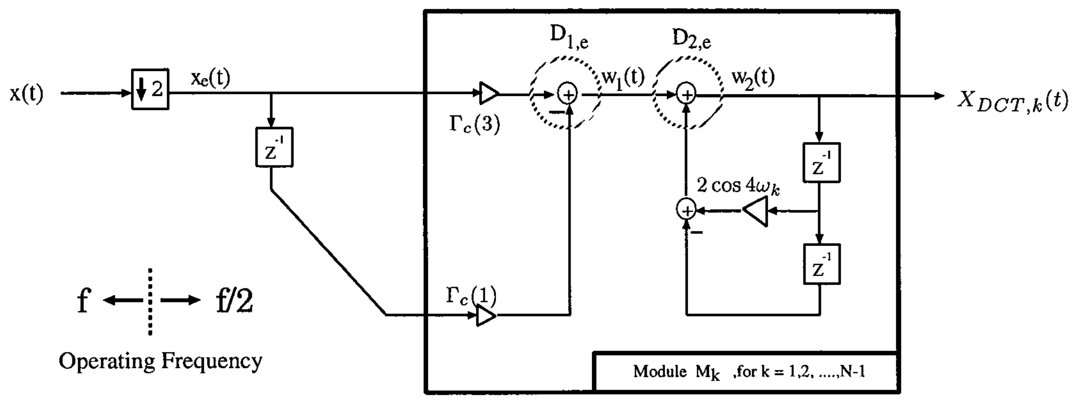


Figure 5.5: Reduced IIR DCT architecture with $M = 2$.

Using the analytical results in (5.26) and (5.32), we can also find the optimal wordlengths for $N = 8$ and 16 under the 40dB SNR constraint. The results are listed in Table 5.1(b). It is interesting to note that the average wordlengths 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 wordlength can be allocated. This indicates that the multirate DCT architecture can not only reduce low-power consumption, its numerical properties also become better as M increases.

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

$$P_f = (M + N_s(k))(\frac{N}{2^{m+1}})\frac{\sigma_R^2}{\sin^2(2^{m+1}\omega_k)}, \tag{5.33}$$

and

$$\begin{aligned} D_1 &= M \cdot C(k) \sum_{n=0}^{M-1} |\cos[(2n+1)\omega_k]|, \\ D_2 &= C(k) \sum_{n=0}^{\frac{N}{M}-1} \sum_{i=0}^{M-1} \left|\cos[(2^{m+1}n + 2i + 1)\omega_k]\right|, \\ D &= \max\{D_1, D_2\}. \end{aligned} \tag{5.34}$$

with $m = \log_2 M$.

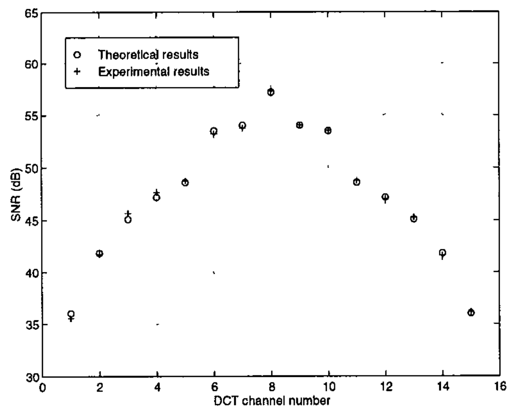
5.5 Simulation Results

To verify our analytical results, computer simulations are carried out by using the aforementioned DCT architectures. The input sequence is a random sequence with uniform probability distribution over the interval (-1,1). All the results are based on the average of 1000 independent DCT computations.

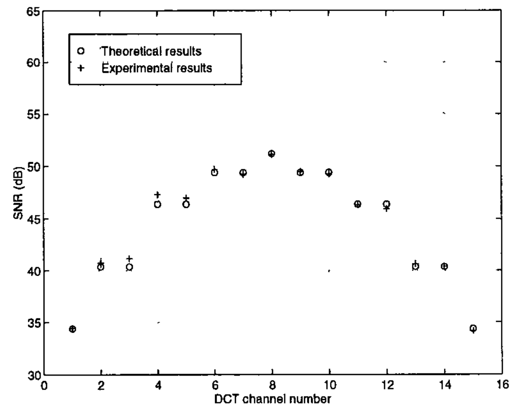
Fig. 5.6 shows the average SNR as a function of the DCT channel number k. As we can see, there is a close agreement between the theoretical and experimental results. Basically, the SNR distribution is affected by the factor $\sin^2(2^{m+1}\omega_k)$ in (5.33) so that its period varies with the decimation factor M. It should be noted that although Fig. 5.6 (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. 5.7 shows the relationship between the average SNR and the wordlength for $N = 16$. Compared to the simulation results in [41], the three IIR DCT architectures give comparative SNR performance to the DCT architectures by Hou [31] and Lee [32] under fixed-point arithmetic. It is worth noting that the multirate DCT architectures have better SNR results than the normal IIR DCT architectures; *i.e.*, the multirate DCT has better numerical properties under fixed-point arithmetic, which is consistent with what we have seen in Table 5.1.

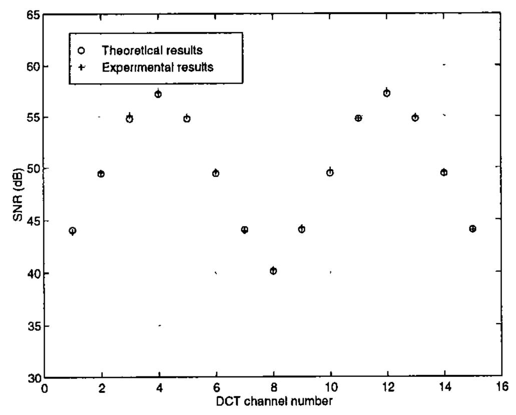
In summary, the analytical results presented in this section can be used as a good index for future applications as N and/or M changes. Furthermore, we can assign the optimal wordlength for each individual DCT channel given the SNR criteria, while this is not the case in the fast-algorithm based PIPO DCT structures [31][32]. Due to the characteristics of global interconnections in the PIPO DCT structure, each operator at each stage will affect part or all of the outputs. Therefore, it is not easy to find optimal wordlength for each channel in the PIPO structure.



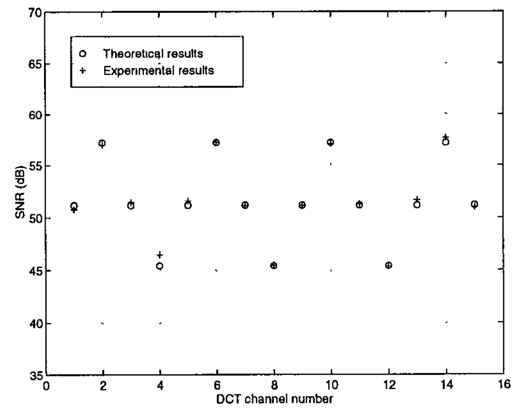
(a)



(b)



(c)



(d)

Figure 5.6: 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 $M = 4$.

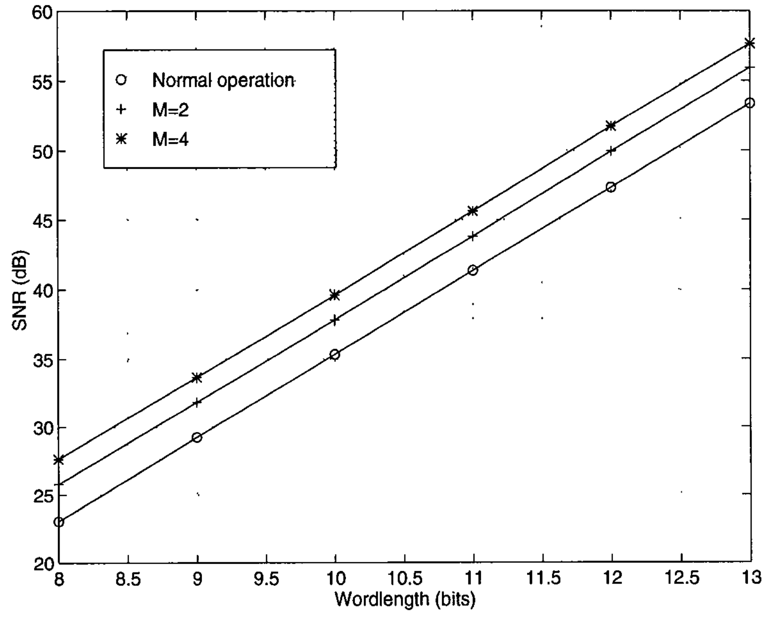


Figure 5.7: Average SNR as a function of wordlength under fixed-point arithmetic (N=16). The multirate low-power architectures have better SNR as M increases.

Chapter 6

Video Co-Processor Design

Modern communication services such as high-definition TV (HDTV), video-on-demand services (VOD), and PC-based multimedia applications call for computationally intensive data processing at video data rate which includes low-level tasks like DCT and filtering/convolution operations as well as medium-level tasks like motion estimation (ME), variable length coding (VLC), and vector quantization (VQ). All these tasks require millions of additions/multiplications per second to ensure the real-time performance of those video applications. As a consequence, the traditional general-purpose programmable DSP processor is not sufficient enough under such a speed constraint. Although the performance of the DSP processor can be improved by utilizing advanced VLSI technology and special arithmetic kernels [44][45], the manufacturing cost as well as the complexity of the design will be enormously increased. On the other hand, dedicated VLSI application-specific integrated circuit (ASIC) chips, which are optimally designed for some given functions, can handle the demanding computational tasks. However, since a collection of ASIC chips are required to perform various different tasks, both manufacturing cost and system complexity will be increased. Therefore, we are motivated to design a programmable video processor with the flexibility of a general DSP processor while meeting the stringent speed requirement in the ASIC designs.

In this chapter, we present a universal programmable architecture which integrates the rotation-based FIR/IIR architecture, Quadrature Mirror Filter (QMF) lattice structure [22], discrete transform (DT) architecture [46][37], adaptive recursive least square (RLS) lattice structure [47]. It can serve as a co-processor in the video system to perform those front-end computationally intensive functions for the host processor. We will first examine the inherent properties of each function, then design a programmable rotation-based computational module that can serve as a basic processing elements (PE) in all functions. The resulting system consists of an array of identical programmable modules and one programmable interconnection network. Each programmable module acts as a basic PE in each programmed function by setting suitable parameters and switches. The interconnection network is used to connect the modules and to combine the appropriate module outputs according to the data paths. Since the properties of each programmed function such as parallelism and pipelinability have been fully exploited, the processing speed of the proposed co-processor design can be as fast as dedicated ASIC designs. 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. In video signal processing, the major constraint is the processing speed of the video processor. Such speed constraint will result in the use of expensive high-speed multiplier/adder circuits or full-custom designs. Thus, the cost and the design cycle will increase drastically. We will show that we can map the multirate FIR/IIR/DT operations onto our video co-processor design. As a result, we can double the speed performance of the co-processor on-the-fly by simply reconfigurating the programmable modules and interconnection network. This feature can also be applied to the low-power implementation of this co-processor

In the last part, we will show how to incorporate the feature of adaptive filtering into our co-processor design. The recursive least-squares (RLS) filter, which is widely used in channel equalization, system identification, and image restoration, has become another important computationally intensive component in portable video equipment since wireless communication requires fast adaptation to highly non-stationary mobile channels. We will show that, with little modification to the programmable module design, the proposed co-processor can also perform the QR-decomposition based RLS lattice algorithm (QRD-LSL) [47] in a fully pipelined way.

The organization of the this chapter is as follows: Section 6.1 presents the programmable co-processor design for the FIR, IIR, QMF filtering, and discrete transforms. In Section 6.2, the speed up of the co-processor design based on the multirate approach is discussed. Later the incorporation of the QRD-LSL array into our co-processor design is presented in Section 6.3. The comparison of our co-processor with other existent approaches is discussed in Section 6.4.

6.1 Video Co-processor Design for the FIR/QMF/IIR/DT

In this section, the design of the video co-processor under normal operation (without speed-up) is discussed. We first examine the basic operations of the FIR filtering, QMF bank, IIR filtering, and discrete orthogonal transforms (DT), to integrate their basic computational modules. Later, a universal programmable module which integrates those basic computational modules in FIR/QMF/IIR/DT is derived. We will show that, by setting appropriate parameters to the modules and connecting them via a programmable interconnection network, we are able to perform all functions in the FIR/QMF/IIR/DT in a fully-pipelined way.

6.1.1 Basic Module in FIR

The finite impulse response (FIR) filter is widely used in DSP applications. In addition to the multiply-and-accumulate (MAC) implementation of the filtering operation, an alternative realization of the FIR filter is the lattice structure as shown in Fig. 6.1

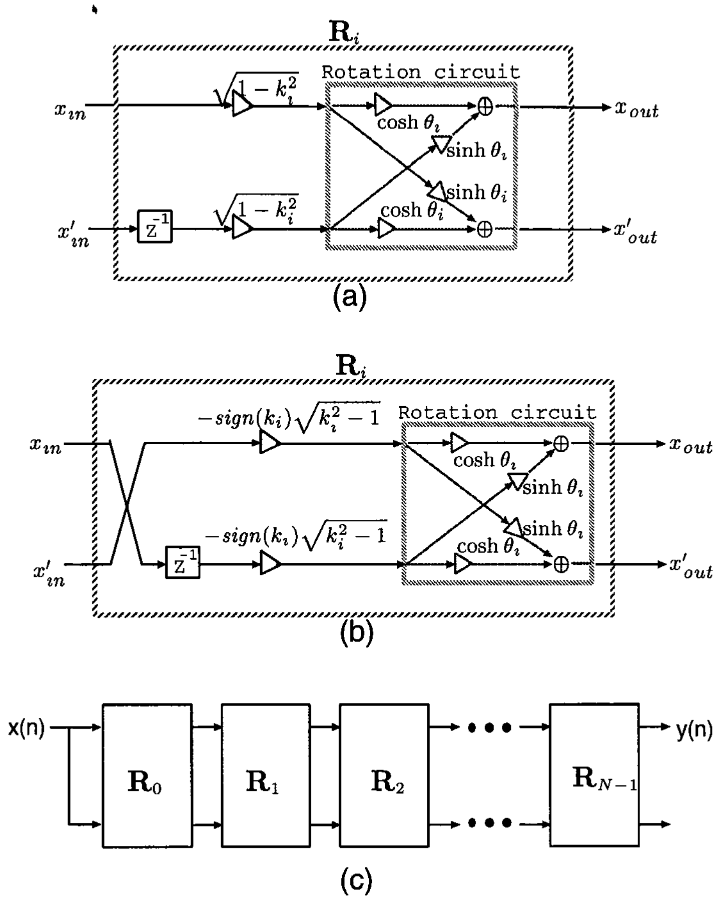


Figure 6.1: (a) Basic lattice filter section with $|k_i| < 1$. (b) Basic lattice filter section with $|k_i| > 1$. (c) Lattice FIR structure.

[48]. It consists of N basic lattice sections that are connected in a cascade form. The advantages of the lattice structure over the MAC implementation is its robustness to the coefficient quantization effect and the smaller dynamic range due to the orthogonal operation used in each lattice.

Given a Nth-order FIR transfer function

$$H(z) = 1 - \sum_{m=0}^{N-1} a_m z^{-(m+1)}, \tag{6.1}$$

the FIR lattice parameters can be computed as follows [40]:

1. Initialization: $a_m^{(N-1)} = a_m,\ m = 0, 1, \ldots, N-1.$

2. For $i = N-1, N-2\ldots,0$

$$\begin{aligned} k_i &= a_i^{(i)}, \\ a_m^{(i-1)} &= \frac{a_m^{(i)} + k_i a_{i-m}^{(i)}}{1-k_i^2}, \quad m = 0,1,\ldots,(i-1). \end{aligned} \tag{6.2}$$

end

where the parameter k_i's, $i = 0,1,\ldots,N-1$, are known as the **reflection coefficients**, or the **PARtial CORrelation** coefficients (PARCOR) in the theory of linear prediction [49]. After k_i's are computed, the lattice section of the FIR filter can be described by

$$\begin{bmatrix} x_{out} \\ x'_{out} \end{bmatrix} = \begin{bmatrix} 1 & -k_i \\ -k_i & 1 \end{bmatrix} \begin{bmatrix} x_{in} \\ x'_{in} \end{bmatrix}, \tag{6.3}$$

or equivalently

1. For $|k_i| < 1$,

$$\begin{aligned} \begin{bmatrix} x_{out} \\ x'_{out} \end{bmatrix} &= \begin{bmatrix} \frac{1}{\sqrt{1-k_i^2}} & \frac{-k_i}{\sqrt{1-k_i^2}} \\ \frac{-k_i}{\sqrt{1-k_i^2}} & \frac{1}{\sqrt{1-k_i^2}} \end{bmatrix} \begin{bmatrix} \sqrt{1-k_i^2} & 0 \\ 0 & \sqrt{1-k_i^2} \end{bmatrix} \begin{bmatrix} x_{in} \\ x'_{in} \end{bmatrix} \\ &= \begin{bmatrix} \cosh\theta_i & \sinh\theta_i \\ \sinh\theta_i & \cosh\theta_i \end{bmatrix} \begin{bmatrix} \sqrt{1-k_i^2} & 0 \\ 0 & \sqrt{1-k_i^2} \end{bmatrix} \begin{bmatrix} x_{in} \\ x'_{in} \end{bmatrix} \end{aligned} \tag{6.4}$$

with

$$\theta_i = \tanh^{-1}(-k_i). \tag{6.5}$$

2. For $|k_i| > 1$,

$$\begin{aligned} \begin{bmatrix} x_{out} \\ x'_{out} \end{bmatrix} &= \begin{bmatrix} \frac{|k_i|}{\sqrt{k_i^2-1}} & \frac{-sign(k_i)}{\sqrt{k_i^2-1}} \\ \frac{-sign(k_i)}{\sqrt{k_i^2-1}} & \frac{|k_i|}{\sqrt{k_i^2-1}} \end{bmatrix} \begin{bmatrix} -sign(k_i)\sqrt{k_i^2-1} & 0 \\ 0 & -sign(k_i)\sqrt{k_i^2-1} \end{bmatrix} \begin{bmatrix} x'_{in} \\ x_{in} \end{bmatrix} \\ &= \begin{bmatrix} \cosh\theta_i & \sinh\theta_i \\ \sinh\theta_i & \cosh\theta_i \end{bmatrix} \begin{bmatrix} -sign(k_i)\sqrt{k_i^2-1} & 0 \\ 0 & -sign(k_i)\sqrt{k_i^2-1} \end{bmatrix} \begin{bmatrix} x'_{in} \\ x_{in} \end{bmatrix}, \end{aligned} \tag{6.6}$$

where $sign(k_i)$ denotes the sign of k_i, and θ is defined by

$$\theta_i = \tanh^{-1}(-1/k_i). \tag{6.7}$$

The lattice modules to realize (6.4) and (6.6) are depicted in Fig. 6.1(a) and (b), respectively. Each module is composed of two scaling multipliers and one rotation circuit. In general, the rotation circuit can be implemented by using either general-purpose multipliers/adders or the CORDIC processor in **hyperbolic mode** [50]. Note that we need to swap the two inputs for the case of $|k_i| > 1$ since the input vector is inverted in (6.6). These two basic modules constitute the FIR lattice structure as shown in Fig. 6.1(c).

6.1.2 Basic Module in QMF

The Quadrature Mirror Filter (QMF) plays a key role in image compression and subband coding [51][52]. Recently, the two-channel paraunitary QMF lattice was proposed [22]. It possesses all the advantages of the lattice structure such as robustness to coefficient quantization, smaller dynamic range, and modularity. Such properties are preferred to as the MAC-based realization when the filter bank is implemented using fixed-point arithmetic. Fig. 6.2 shows the QMF lattice structure, where part (c) is the analysis bank and part (d) is the synthesis bank. We can see that 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 and two scaling multipliers are set to one. If CORDIC processor is employed to realize the rotation circuit in the QMF lattice, it works in the **circular mode** to perform the necessary rotations.

It has been shown in [22] that every two-channel (real-coefficient, FIR) paraunitary QMF bank can be represented using the QMF lattice. Given a pre-designed power-symmetric FIR analysis filter $H_0(z)$ with unit sample response $h_0(n), n = 0, 1, \ldots, N$, we can first find the unit sample response of the other analysis filter $H_1(z)$ by

$$h_1(n) = (-1)^n h_0(N-n). \tag{6.8}$$

Then the rotation angle θ_i in each QMF module can be computed by [21, Chap.6]:

1. <u>Initialization</u>: $H_0^{(J)}(z) = H_0(z)$ and $H_1^{(J)}(z) = H_1(z)$ with $N = 2J + 1$.

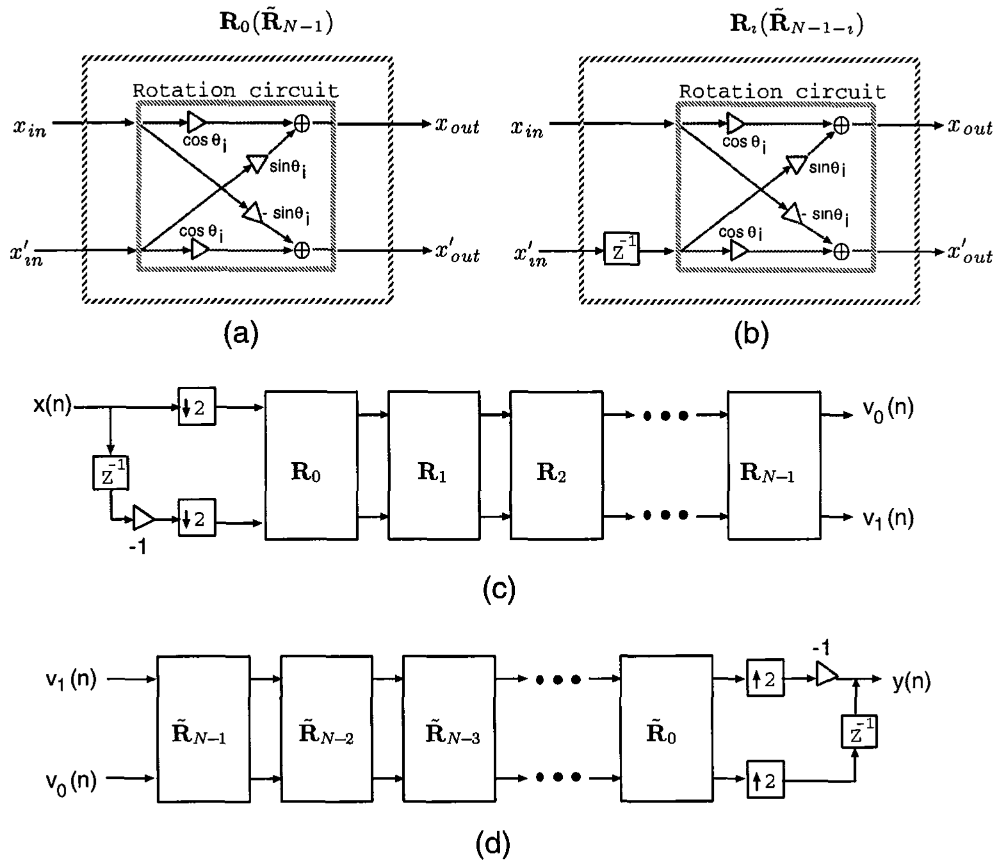


Figure 6.2: The two-channel paraunitary QMF lattice: (a),(b) Basic lattice sections. (c) The analysis bank. (d) The synthesis bank.

2. For $i = J, J-1, \ldots, 0$

$$\begin{aligned} (1+\alpha_i^2)H_0^{(i-1)}(z) &= H_0^{(i)}(z) - \alpha_i H_1^{(i)}(z), \\ (1+\alpha_i^2)z^{-2}H_1^{(i-1)}(z) &= \alpha_i H_0^{(i)}(z) + H_1^{(i)}(z), \\ \theta_i &= \tan^{-1}\alpha_i, \end{aligned} \tag{6.9}$$

end

The coefficient α_i is computed by setting the highest power of z^{-1} in $H_0^{(i)}(z) - \alpha_i H_1^{(i)}(z)$ equal to zero.

6.1.3 Basic Module in IIR

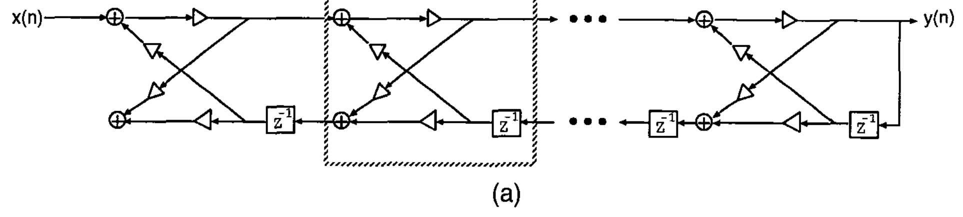
Next we want to consider the basic module in infinite impulse response (IIR) filtering. The lattice structure for an IIR system (all-pole and ARMA) [48] is shown in Fig. 6.3. Although the basic lattice module in IIR filters is similar to the one in FIR lattice, the opposite data flow in the IIR module makes it difficult to be incorporated into our unified design. Besides, the modularity of the lattice structure no longer exists if we want to implement a general IIR (ARMA) filter (see Fig. 6.3(b)). 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.

Second-order IIR Lattice Structure

Fig. 6.4 shows the lattice structure that can be used to realize a second-order IIR filter. It is also known as the "couple-form" of the second-order IIR filter which is robust to the coefficient quantization error under fixed-point arithmetic [40, Chap.6]. It can be shown that the transfer functions of the two outputs are given by

$$\tilde{H}_0(z) = \frac{Y_0(z)}{X(z)} = \frac{r(k_0\cos\theta + k_1\sin\theta) - r^2k_0z^{-1}}{1 - 2r\cos\theta z^{-1} + r^2z^{-2}}, \tag{6.10}$$

$$\tilde{H}_1(z) = \frac{Y_1(z)}{X(z)} = \frac{r(k_1\cos\theta - k_0\sin\theta) - r^2k_1z^{-1}}{1 - 2r\cos\theta z^{-1} + r^2z^{-2}}. \tag{6.11}$$



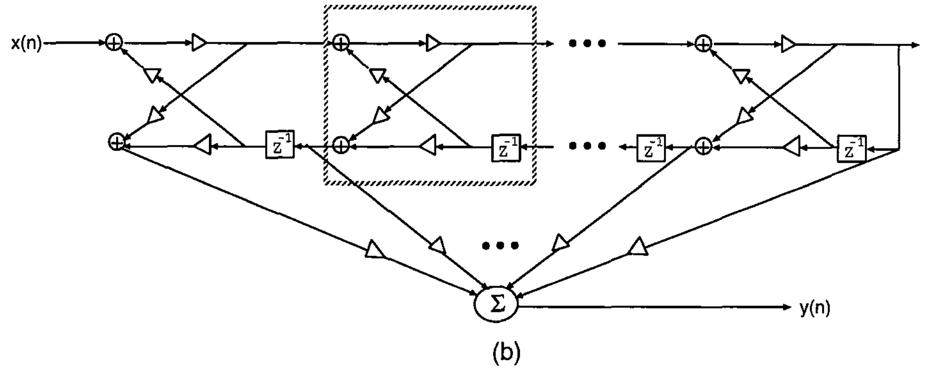


Figure 6.3: (a) All-pole IIR lattice. (b) General IIR (ARMA) lattice.

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

$$H(z) = K \prod_{i=0}^{N/2-1} H_i(z), \tag{6.12}$$

where K is a scaling constant [1] and each subfilter $H_i(z)$ is of the form

$$\begin{aligned} H_i(z) &= \frac{1 + c_i z^{-1} + d_i z^{-2}}{1 + a_i z^{-1} + b_i z^{-2}} \\ &= \frac{1}{1 + a_i z^{-1} + b_i z^{-2}} + z^{-1} \frac{c_i + d_i z^{-1}}{1 + a_i z^{-1} + b_i z^{-2}} \\ &= A_{i,0}(z) + z^{-1} A_{i,1}(z). \end{aligned} \tag{6.13}$$

Comparing (6.10) and (6.11) with (6.13), we see that $A_{i,0}(z)$ and $A_{i,1}(z)$ can be realized by either $\tilde{H}_0(z)$ or $\tilde{H}_1(z)$ with appropriate settings of the parameters k_0, k_1, r, and θ. The conversion of the parameters is derived as follows, where $\tilde{H}_0(z)$ is chosen for the realization.

[1]Here we assume that $K = 1$. This simple scaling operation can be done by the host processor after it collects the outputs from the video co-processor.

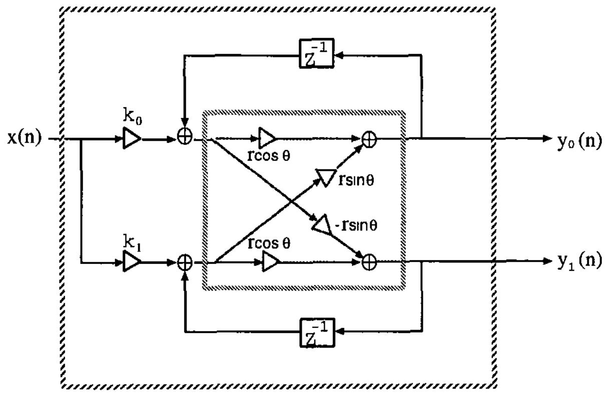


Figure 6.4: Second-order IIR lattice architecture.

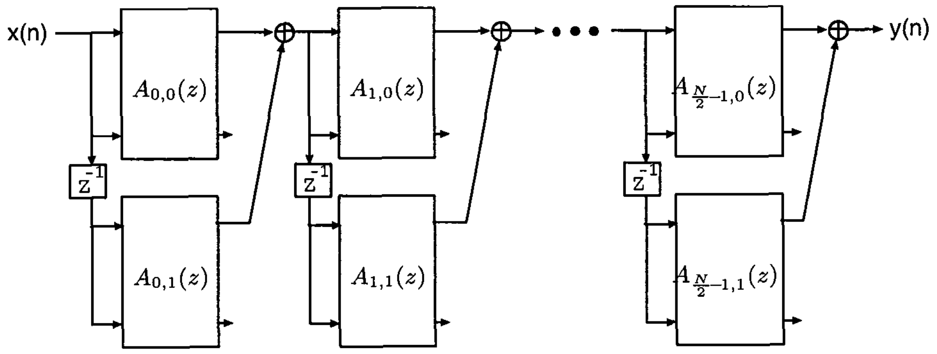


Figure 6.5: IIR (ARMA) structure based on the second-order IIR lattice module.

For $i = 0, 1, \ldots, N/2 - 1$,

1. Find the poles of $H_i(z)$:

$$p_{0,i} = \frac{-a_i + \sqrt{a_i^2 - 4b_i}}{2}, \quad p_{1,i} = \frac{-a_i - \sqrt{a_i^2 - 4b_i}}{2}. \tag{6.14}$$

2. (a) For the case $\sqrt{a_i^2 - 4b_i} < 0$ (complex conjugate poles at $r_i e^{\pm\theta_i}$), compute the radius r_i and phase θ_i of the poles:

$$r_i = \text{mag}(p_{0,i}), \quad \theta_i = \arg(p_{0,i}). \tag{6.15}$$

   (b) For the case $\sqrt{a_i^2 - 4b_i} > 0$ (two real poles at $p_{0,i}$ and $p_{1,i}$), compute r_i and θ_i by equating

$$\begin{cases} 2r_i \cos\theta_i = p_{0,i} + p_{1,i} \\ r_i^2 = p_{0,i} \cdot p_{1,i}, \end{cases} \tag{6.16}$$

   which yields

$$\begin{cases} r_i = \sqrt{p_{0,i} \cdot p_{1,i}} \\ \theta_i = \cos^{-1} \frac{p_{0,i}+p_{1,i}}{2\sqrt{p_{0,i} \cdot p_{1,i}}}. \end{cases} \tag{6.17}$$

3. Solve k_0 and k_1 used in $A_{i,0}(z)$ by setting

$$\begin{cases} r_i(k_0 \cos\theta_i + k_1 \sin\theta_i) = 1 \\ -r_i^2 k_0 = 0 \end{cases} \tag{6.18}$$

   which yields

$$\begin{cases} k_0 = 0 \\ k_1 = 1/(r_i \sin\theta_i). \end{cases} \tag{6.19}$$

4. Solve k_0 and k_1 used in $A_{i,1}(z)$ by setting

$$\begin{cases} r_i(k_0 \cos\theta_i + k_1 \sin\theta_i) = c_i \\ -r_i^2 k_0 = d_i \end{cases} \tag{6.20}$$

   which yields

$$\begin{cases} k_0 = -d_i/r_i^2 \\ k_1 = (c_i/r_i - k_0 \cos\theta_i)/\sin\theta_i. \end{cases} \tag{6.21}$$

end.

Note that all r_i's should be less than one to ensure the stability of the IIR filtering. There are some limitations in this realization: Firstly, the order of the ARMA filter is restricted to be even to facilitate the decomposition in (6.12). Secondly, we cannot realize the second-order IIR which has two multiple real poles or two real poles with opposite signs (r_i in (6.16) cannot be solved). In some cases, this situation can be avoided by arranging the real poles with the same sign as a pair or imposing such constraints in the design phase of the filter.

Now based on (6.12) and (6.13), we can realize $H(z)$ using the structure depicted in Fig. 6.5, in which each stage performs the filtering for $H_i(z)$, and $A_{i,0}$, $A_{i,1}$, $i = 0, 1, \ldots, N/2-1$, are realized by the second-order IIR module in Fig. 6.4. To perform an N^{th}-order (N is even) $H(z)$, we need a total of N second-order IIR modules.

6.1.4 Basic Module in Discrete Transforms

In Section 4.3, we have presented a unified tranform coding architecture that is capable of performing most of the discrete transforms (DTs). However, the IIR-based module used in Fig. 4.3 is not applicable to the programmable architecture proposed here. In order to incorporate the unified DT operations into our co-processor design, we need a rotation-based computational module for the processing element.

In [46][37], a rotation-based module was derived for the dual generation of $X_{C,k}(t)$ and $X_{S,k}(t)$ in (4.10) and (4.11) (see Fig. 6.6), where the scaling multipliers and the rotation operation are given by

$$\mathbf{f}_k = \begin{bmatrix} f_{0,k} \\ f_{1,k} \end{bmatrix} = \begin{bmatrix} \beta\cos((2L+1)\omega_k + \eta_k) \\ \beta\sin((2L+1)\omega_k + \eta_k) \end{bmatrix}. \tag{6.22}$$

and

$$\mathbf{R}_k = \begin{bmatrix} \cos(2\omega_k) & \sin(2\omega_k) \\ -\sin(2\omega_k) & \cos(2\omega_k) \end{bmatrix}, \tag{6.23}$$

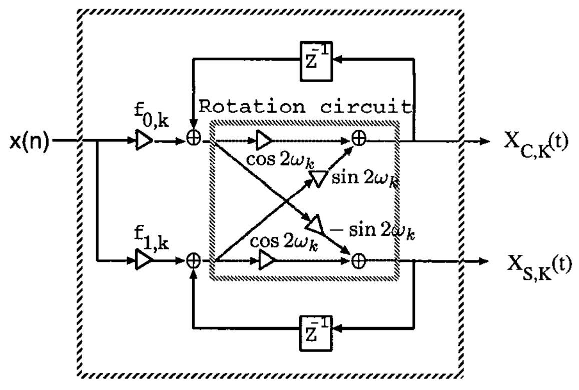


Figure 6.6: Rotation-based module for the dual generation of $X_{C,k}(t)$ and $X_{S,k}(t)$.

respectively. The rotation-based module works in an SIPO way as the IIR-based Module in Fig. 4.2. The operation of the unified rotation-based DT architecture is the same as the unified architecture in Fig. 4.3 except that the IIR-based modules are replaced with the rotation-based modules.

6.1.5 Unified Module Design

From Fig. 6.1, Fig. 6.2, Fig. 6.4, and Fig. 6.6, 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. We thus can integrate those basic modules into one universal programmable module as shown in Fig. 6.7. It consists of six switches, four scaling multipliers and one rotation circuit. The switch set $\mathbf{S} \triangleq [s_0 s_1 s_2 s_3 s_4 s_5]$ controls the data paths inside the module. The switch pair s_0 and s_1 select the input from either in_i or in_i': With $s_0 s_1 = 00$, in_i becomes the common input of the lattice which is required in the first stage of FIR and in the IIR module. Using $s_0 s_1 = 10$, we can swap the inputs for the FIR lattice when $|k_i| > 1$. Switches s_2 and s_3 decide if the delay element is used or not: With $s_2 s_3 = 01$, the lower-left delay element is included in the data path, which is required in the FIR/QMF lattice (except the first stage in QMF banks). With the

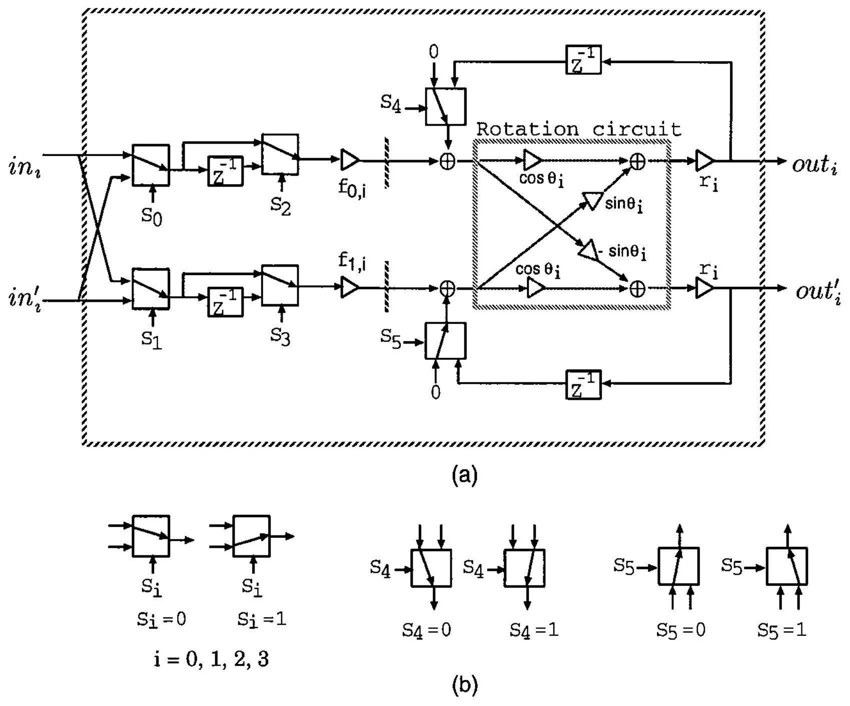


Figure 6.7: (a) Programmable module for the FIR/QMF/IIR/DT. (b) Switches used in the module.

setting $s_2s_3 = 11$, the delay element in Fig. 6.5 can be incorporated into the module $A_{i,1}$. Therefore, we do not need to implement it 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 $s_4s_5 = 11$, the delayed module outputs will be added with the current inputs as in the IIR and DT case. The setting $s_4s_5 = 00$ will disconnect the feedback paths. The two multipliers r_i's at the outputs of the rotation circuit are required only when we want to incorporate IIR function into this universal design. The parameters $\mathbf{f}_i$, r_i, and θ_i can be determined from our discussions in Section 6.1.1-6.1.4. The complete settings of the programmable module for the FIR/QMF/IIR/DT are listed in Table 6.1. Moreover, two pipelining stages (the dash lines in Fig. 6.7) are inserted after the scaling multipliers $f_{0,i}$ and $f_{1,i}$ to shorten the critical path of the programmable module.

6.1.6 Video Co-processor Design

Based on the above programmable module, we are ready to design the video co-processor that is capable of performing parallel implementation for any function in the FIR/QMF/IIR/DT. Fig. 6.8 shows the video co-processor architecture under the FIR mode. It consists of two parts: One is the **programmable module array** with P identical programmable modules. The other is the **programmable interconnection network** which connects those programmable modules according to the data paths. In the FIR/QMF/IIR, the data are processed in a serial-input-serial-output (SISO) 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. 6.8. The connections of IIR modules can also be achieved using the network setting in Fig. 6.9. On the other hand, the DT architecture in Section 6.1.4 performs the block transforms in an SIPO way. The interconnection network will be configurated according to the combination functions defined in Table 4.3. The detailed settings of the interconnection network used in the co-processor design (Type I-IX) are described in Table 6.2.

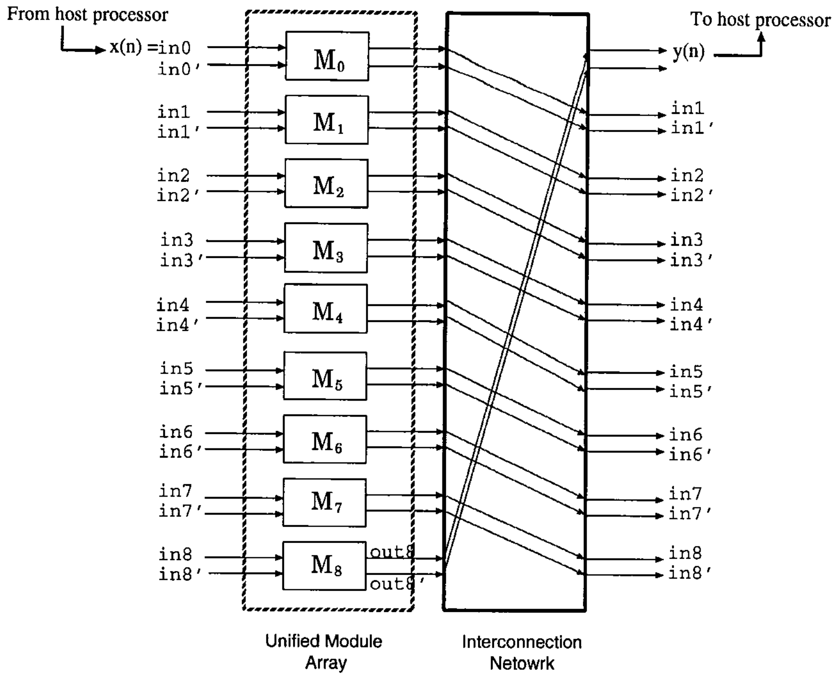


Figure 6.8: Overall architecture for FIR filtering.

The operation of the co-processor is as follows: In the **initialization mode**, the host processor will compute all the necessary parameters $\mathbf{f}_i, r_i, \theta_i$ according to the function types (FIR/QMF/IIR/DT) and the function to be performed. In general, the functions to be performed are determined beforehand. All the parameters can be computed in advance so that the host processor can find the necessary parameters through table-look-up to reduce the set-up time in this mode. Next, the host processor needs to reconfigurate the interconnection network according to the function type.

Once the video co-processor is initialized, it enters the **execution mode**. In the applications of FIR/IIR/QMF, the host processor continuously feeds the data sequence into the co-processor. After the first output data is ready, the processor can collect the filtering outputs in a fully pipelined way. In the block DT application, the block input

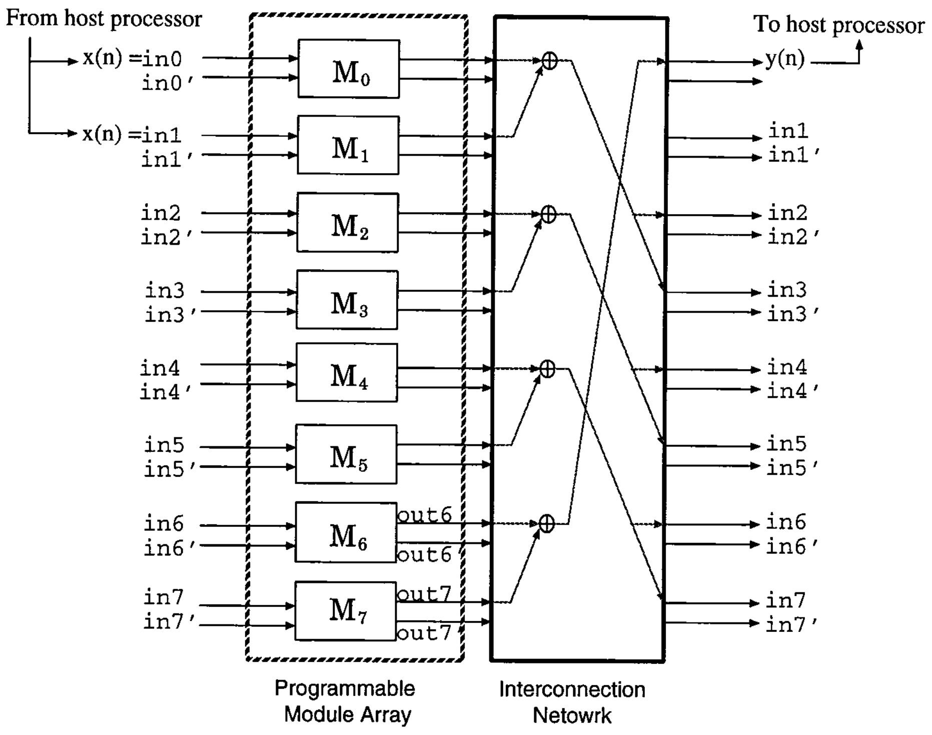


Figure 6.9: Overall architecture for IIR (ARMA) filtering.

	$\mathbf{S} = [s_0 s_1 s_2 s_3 s_4 s_5 s_6]$	$\mathbf{f}_i = [f_{0,i}, f_{1,i}]^T$	$\mathbf{R}_i$	Network Type	$N_{\max}$
FIR ($\lvert k_i \rvert < 1$)	$\begin{cases} 000100\ (M_0) \\ 010100\ (M_i,\ i \neq 0) \end{cases}$	$\begin{bmatrix} \sqrt{1-k_i^2} \\ \sqrt{1-k_i^2} \end{bmatrix}$ with k_i defined in (6 2)	$\begin{bmatrix} \cosh\theta_i & \sinh\theta_i \\ \sinh\theta_i & \cosh\theta_i \end{bmatrix}$, with θ_i defined in (6 5)	I	P
FIR ($\lvert k_i \rvert > 1$)	$\begin{cases} 000100\ (M_0) \\ 100100\ (M_i,\ i \neq 0) \end{cases}$	$\begin{bmatrix} -sign(k_i)\sqrt{k_i^2-1} \\ -sign(k_i)\sqrt{k_i^2-1} \end{bmatrix}$ with k_i defined in (6.2)	$\begin{bmatrix} \cosh\theta_i & \sinh\theta_i \\ \sinh\theta_i & \cosh\theta_i \end{bmatrix}$, with θ_i defined in (6 7)	I	P
QMF (analysis)	$\begin{cases} 010000\ (M_0) \\ 010100\ (M_i,\ i \neq 0) \end{cases}$	$\begin{bmatrix} 1 \\ 1 \end{bmatrix}$	$\begin{bmatrix} \cos\theta_i & \sin\theta_i \\ -\sin\theta_i & \cos\theta_i \end{bmatrix}$, with θ_i defined in (6 9)	I	$2P-1$
QMF (synthesis)	$\begin{cases} 010000\ (\bar{M}_{N-1}) \\ 010100\ (\bar{M}_i,\ i \neq N-1) \end{cases}$	$\begin{bmatrix} 1 \\ 1 \end{bmatrix}$	$\begin{bmatrix} \cos\bar{\theta}_i & \sin\bar{\theta}_i \\ -\sin\bar{\theta}_i & \cos\bar{\theta}_i \end{bmatrix}$ with$\bar{\theta}_i = \theta_{N-1-i}$, where θ_{N-1-i} is defined in (6 9)	I	$2P-1$
IIR ($A_{i,0}$)	000011	$\begin{bmatrix} k_0 \\ k_1 \end{bmatrix}$ defined in (6 19)	$\begin{bmatrix} r_i\cos\theta_i & r_i\sin\theta_i \\ -r_i\sin\theta_i & r_i\cos\theta_i \end{bmatrix}$, with r_i, θ_i defined in (6 15) or (6 17)	III	P
IIR ($A_{i,1}$)	001111	$\begin{bmatrix} k_0 \\ k_1 \end{bmatrix}$ defined in (6 21)	$\begin{bmatrix} r_i\cos\theta_i & r_i\sin\theta_i \\ -r_i\sin\theta_i & r_i\cos\theta_i \end{bmatrix}$, with r_i, θ_i defined in (6 15) or (6 17)	III	P
DT	000011	$\begin{bmatrix} \beta\cos((2L+1)\omega_k+\eta_k) \\ \beta\sin((2L+1)\omega_k+\eta_k) \end{bmatrix}$ with L, β, ω_k, η_k defined in Table 4 3	$\begin{bmatrix} \cos 2\omega_k & \sin 2\omega_k \\ -\sin 2\omega_k & \cos 2\omega_k \end{bmatrix}$, with ω_k defined in Table 4 3	V-VIII	P
Multirate FIR ($\lvert k_i \rvert < 1$)	$\begin{cases} 000100\ (M_0, M_1, M_2) \\ 010100\ (M_i,\ i > 2) \end{cases}$	$\begin{bmatrix} \sqrt{1-k_i^2} \\ \sqrt{1-k_i^2} \end{bmatrix}$ with k_i defined in (6 2)	$\begin{bmatrix} \cosh\theta_i & \sinh\theta_i \\ \sinh\theta_i & \cosh\theta_i \end{bmatrix}$, with θ_i defined in (6 5)	II	$\frac{2P}{3}$
Multirate FIR ($\lvert k_i \rvert > 1$)	$\begin{cases} 000100\ (M_0, M_1, M_2) \\ 100100\ (M_i,\ i \neq 2) \end{cases}$	$\begin{bmatrix} -sign(k_i)\sqrt{k_i^2-1} \\ -sign(k_i)\sqrt{k_i^2-1} \end{bmatrix}$ with k_i defined in (6.2)	$\begin{bmatrix} \cosh\theta_i & \sinh\theta_i \\ \sinh\theta_i & \cosh\theta_i \end{bmatrix}$, with θ_i defined in (6.7)	II	$\frac{2P}{3}$
Multirate IIR ($A_{i,0}$)	000011	$\begin{bmatrix} k_0 \\ k_1 \end{bmatrix}$ defined in (6 19)	$\begin{bmatrix} r_i\cos\theta_i & r_i\sin\theta_i \\ -r_i\sin\theta_i & r_i\cos\theta_i \end{bmatrix}$, with r_i, θ_i defined in (6 15) or (6 17)	IV	$\frac{P}{3}$
Multirate IIR ($A_{i,1}$)	001111	$\begin{bmatrix} k_0 \\ k_1 \end{bmatrix}$ defined in (6.21)	$\begin{bmatrix} r_i\cos\theta_i & r_i\sin\theta_i \\ -r_i\sin\theta_i & r_i\cos\theta_i \end{bmatrix}$, with r_i, θ_i defined in (6.15) or (6 17)	IV	$\frac{P}{3}$
Multirate DT	000011	$\begin{bmatrix} \beta\cos((2L+2l+1)\omega_k+\eta_k) \\ \beta\cos((2L+2l+1)\omega_k+\eta_k) \end{bmatrix}$ with $l = k \bmod 2$, and L, β, ω_k, η_k defined in Table 4 3	$\begin{bmatrix} \cos 4\omega_k & \sin 4\omega_k \\ -\sin 4\omega_k & \cos 4\omega_k \end{bmatrix}$, with ω_k defined in Table 4.3	V-VIII	$\frac{P}{2}$
QRD-LSL	$\begin{cases} 0100101\ (M_{4k}) \\ 0101101\ (M_{4k+1}) \\ 0100100\ M_{4k+2}, \\ 0100100\ M_{4k+3}) \end{cases}$	$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$	$\begin{cases} R_A(\theta) = M_{4k}, M_{4k+1} \\ R_R(\theta) = M_{4k+2}, M_{4k+3} \end{cases}$	IX	$\frac{P}{4}$

Table 6.1: Setting for the programmable module, where $N_{\max}$ is the maximum order (or block size in DT) that can be realized by a P-module array, and switch s_6 is only used in the QRD-LSL operation.

Type I (FIR, QMF)	Type II (Multirate FIR)	Type III (IIR)
1. $in_0 = x(n)$. 2. <u>For</u> $m = 0, 1, \ldots, (N-2)$ $in_{m+1} = out_m$, $in'_{m+1} = out'_m$. <u>end</u> 3. $y(n) = out_{N-1}$.	1. $in_i = x_i(n)$, $i = 0, 1, 2$. 2. <u>For</u> $m = 0, 1, \ldots, (\frac{3N}{2} - 4)$ $in_{m+3} = out_m$, $in'_{m+3} = out'_m$. <u>end</u> 3. $y_i(n) = out_{(\frac{3N}{2}-3)+i}$, $i = 0, 1, 2$.	1. $in_i = x(n)$, $i = 0, 1$. 2. <u>For</u> $m = 0, 1, \ldots, (N-3)$ $in_{m+2} = out_{2\lfloor \frac{m}{2} \rfloor} + out_{2\lfloor \frac{m}{2} \rfloor + 1}$. <u>end</u> 3. $y(n) = out_{N-1} + out_{N-2}$.
Type IV (Multirate IIR)	**Type V (DCT)**	**Type VI (MLT)**
1. $in_i = x_{\lfloor \frac{i}{2} \rfloor}(n)$, $i = 0, 1, \ldots, 5$. 2. <u>For</u> $m = 0, 1, \ldots, (3N-7)$ $in_{m+6} = out_{2\lfloor \frac{m}{2} \rfloor} + out_{2\lfloor \frac{m}{2} \rfloor + 1}$. <u>end</u> 3. $y_i(n) = out_{(3N-5)+2i} + out_{(3N-6)+2i}$, $i = 0, 1, 2$.	1. $in_i = x(n)$, $i = 0, 1, \ldots, N-1$. 2. $X_{DCT}(i) = out_i$, $i = 0, 1, \ldots, N-1$.	1. $in_i = x(n)$, $i = 0, 1, \ldots, N-1$. 2. $X_{MLT}(i) = -s_i(out_{i+1} + out'_i)$, $i = 0, 1, \ldots, N-1$.
Type VII (DFT)	**Type VIII (DHT)**	**Type IX (QRD-LSL)**
1. $in_i = x(n)$, $i = 0, 1, \ldots, N-1$. 2. $X_{DFT}(i) = out_i + j * out'_i$, $i = 0, 1, \ldots, N-1$.	1. $in_i = x(n)$, $i = 0, 1, \ldots, N-1$. 2. $X_{DHT}(i) = out_i + out'_i$, $i = 0, 1, \ldots, N-1$.	1. $in'_i = x(n)$, $i = 0, 1$. 2. <u>For</u> $m = 0, 1, \ldots, (4N-3)$ $in'_{m+2} = out'_m$, if $(m \bmod 4 = 0)$ then $\mu_{in}(m+3) = \mu_{out}(m)$, $\mu_{in}(m+2) = \mu_{out}(m+1)$. <u>end</u> 3. $f(n) = out'_{4N-2}$, $b(n) = out'_{4N-1}$.

Table 6.2: Switch settings for the interconnection network, where N denotes the order of the FIR/QMF/IIR/QRD-LSL or the block size of the DT.

data is fed into the co-processor serially. After the last datum enters the unified module array, the evaluations of $X_{C,k}(t)$ and $X_{S,k}(t)$ in (4.10) and (4.11) are complete. Then the interconnection network will combine the module outputs according to the combination function defined in Table 4.3, 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.

6.1.7 Design Examples

In what follows, we will use some design examples to demonstrate how to convert a given system specification to the parameters used in the programmable modules. The orders of the numerator and the denominator in the IIR ARMA filter are restricted to be even so that we can perform all the necessary decompositions. Here, a 10-module co-processor is used to carry out the given function. As a result, the maximum order of the FIR/IIR/QMF is 10 and the transform size of the DT is also limited to 10. For the DT, we will use an 8-point DCT as an example due to its prevalence in the application of transform coding.

FIR Filtering

Given the FIR transfer function

$$\begin{aligned} H(z) &= 1 - 0.8843z^{-1} - 0.1327z^{-2} - 1.1219z^{-3} + 0.5328z^{-4} - 0.8882z^{-5} \\ &+ 0.1038z^{-6} - 0.3786z^{-7} + 0.2195z^{-8} - 0.1094z^{-9}, \end{aligned} \tag{6.24}$$

we first apply (6.1)-(6.2) to compute all PARCOR coefficients:

$k_0 = -0.4472$, $k_1 = -0.6917$, $k_2 = -0.5865$, $k_3 = -4.1573$, $k_4 = 1.1595$,
$k_5 = 0.2655$, $k_6 = 0.2942$, $k_7 = -0.1243$, $k_8 = 0.1094$.

Then all parameters of each module, such as $\mathbf{f}_i$ and θ_i, can be found by using (6.4)-(6.7). The complete settings are listed in Table 6.3(a).

QMF Filtering

Suppose that the pre-designed power-symmetric low-pass filter described in [21, Example 5.3.2] is used for the QMF filtering. We have the analysis filter

$$H_0(z) = \sum_{n=0}^{N-1} h_0(n) z^{-n} \tag{6.25}$$

with

$$\begin{aligned}
h_0(0) &= 0.1605, & h_0(1) &= 0.4156, & h_0(2) &= 0.4592, & h_0(3) &= 0.1487,\\
h_0(4) &= -0.1642, & h_0(5) &= -0.1245, & h_0(6) &= 0.0825, & h_0(7) &= 0.0888,\\
h_0(8) &= -0.0508, & h_0(9) &= -0.0608, & h_0(10) &= 0.0352, & h_0(11) &= 0.0399,\\
h_0(12) &= -0.0256, & h_0(13) &= -0.0244, & h_0(14) &= 0.0186, & h_0(15) &= 0.0135,\\
h_0(16) &= -0.0131, & h_0(17) &= -0.0074, & h_0(18) &= 0.0129, & h_0(19) &= -0.0050.
\end{aligned}$$

We can go through (6.8)-(6.9) to find all θ_i's in the modules and the results are shown in Table 6.3(b).

IIR (ARMA) Filtering

Given the IIR (ARMA) filter

$$H(z) = \frac{1 + \sum_{i=1}^{M} p_i z^{-i}}{1 + \sum_{i=1}^{N} q_i z^{-i}} \tag{6.26}$$

with $M = 4, N = 10$, and

$$\begin{aligned}
p_1 &= -1.7314, & p_2 &= 1.6788, & p_3 &= -0.7913, & p_4 &= 0.2304, & &\\
q_1 &= 0.4036, & q_2 &= 1.3227, & q_3 &= 0.2376, & q_4 &= 1.1558, & q_5 &= 0.0047,\\
q_6 &= 0.6950, & q_7 &= -0.0733, & q_8 &= 0.2735, & q_9 &= -0.0542, & q_{10} &= 0.0788,
\end{aligned}$$

we first rewrite it in cascade form:

$$\begin{aligned}
H(z) &= \frac{1 - 1.1314z^{-1} + 0.6400z^{-2}}{1 - 0.9192z^{-1} + 0.4225z^{-2}} \times \frac{1 - 0.6000z^{-1} + 0.3600z^{-2}}{1 - 0.7500z^{-1} + 0.5625z^{-2}}\\
&\times \frac{1}{1 + 0.8000z^{-1} + 0.6400z^{-2}} \times \frac{1}{1 + 1.2728z^{-1} + 0.8100z^{-2}}\\
&\times \frac{1}{1 + 0.6400z^{-2}}.
\end{aligned} \tag{6.27}$$

Following the steps described in (6.14)-(6.21), we can find the parameters used in each 2nd-order subfilter. The corresponding settings are in Table 6.3(c).

Block DCT

For the block 8-point DCT, we can calculate $f_{0,i}$, $f_{1,i}$ and θ_i of each programmable module using Table 4.3. The settings are listed in Table 6.3(d).

6.2 Speed-Up of the Video Co-processor Architecture

In video signal processing, the fundamental bottleneck is the processing speed of the processing elements. Although the above mentioned co-processor architecture has fully exploited the parallelism and pipelinability for each programmed function, the input data rate is still limited by the speed of the adders and multipliers inside the programmable module. In the video-rate applications such as HDTV, this speed constraint will result in the use of expensive high-speed multiplier/adder circuits or full-custom design. Thus, the cost as well as the design cycle will increase drastically.

In this section, we will show how to map the multirate FIR/IIR/DT architectures with speed-up of two to our video co-processor design. Since processing elements operate at only half of input data rate, the processing speed of the co-processor is doubled based on the same programmable modules and interconnection network.

6.2.1 Mapping of the Multirate FIR Architecture

In Section 2.3, we presented the multirate FIR filtering architecture with $M = 2$ (see Fig. 2.8). Since the multirate architecture can process data at a rate two times faster than the maximum speed of the operators, it can be readily applied to the speed-up of the filtering operations at the architectural level.

To map this multirate FIR architecture to our co-processor design, we first find the three $(\frac{N}{2})$th-order FIR subfilters $E_0(z)$, $E_1(z)$, and $\hat{E}(z) \triangleq E_0(z) + E_1(z)$ of the given FIR transfer function. Then we implement each subfilter using the FIR lattice structure discussed in Section 6.1.1. The resulting architecture is depicted in Fig. 6.10(a), where

	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
S	000100	010100	010100	100100	100100	010100	010100	010100	010100
$f_{0,i}$	0.8944	0.7222	0.8100	4.0352	-0.5870	0.9641	0.9557	0.9922	0.9940
$f_{1,i}$	0.8944	0.7222	0.8100	4.0352	-0.5870	0.9641	0.9557	0.9922	0.9940
r_i	1	1	1	1	1	1	1	1	1
θ_i	0.4812	0.8512	0.6723	0.2454	-1.3027	-0.2720	-0.3032	0.1249	-0.1098
Interconnection	Type I								

(a)

	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9
S	010000	010100	010100	010100	010100	010100	010100	010100	010100	010100
$f_{0,i}$	1	1	1	1	1	1	1	1	1	1
$f_{1,i}$	1	1	1	1	1	1	1	1	1	1
r_i	1	1	1	1	1	1	1	1	1	1
θ_i	-1.2022	0.6993	-0.4465	0.3051	-0.2146	0.1511	-0.1043	0.0690	-0.0426	0.0311
Interconnection	Type I									

(b)

	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9
S	000011	001111	000011	001111	000011	001111	000011	001111	000011	001111
$f_{0,i}$	0	-1.5148	0	-0.6400	0	0	0	0	0	0
$f_{1,i}$	-2.1757	0.9467	-1.5396	0.5543	-1.4434	0	-1.5713	0	-1.2500	0
r_i	0.6500	0.6500	0.7500	0.7500	0.8000	0	0.9000	0	0.8000	0
θ_i	-0.7854	-0.7854	-1.0472	-1.0472	-2.0944	0	-2.3562	0	-1.5708	0
Interconnection	Type III									

(c)

	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7
S	000011	000011	000011	000011	000011	000011	000011	000011
$f_{0,i}$	0.3536	-0.4904	0.4619	-0.4157	0.3536	-0.2778	0.1913	-0.0975
$f_{1,i}$	0	-0.0975	0.1913	-0.2778	0.3536	-0.4157	0.4619	-0.4904
r_i	1	1	1	1	1	1	1	1
θ_i	0	0.3927	0.7854	1.1781	1.5708	1.9635	2.3562	2.7489
Interconnection	Type V							

(d)

Table 6.3: Settings for the (a) FIR filter, (b) QMF filter, (c) IIR (ARMA) filter, and (d) 8-point DCT.

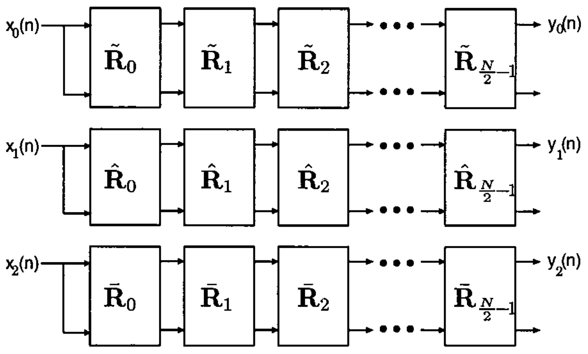
$\tilde{R}_i$, $\hat{R}_i$, and $\bar{R}_i$ correspond to the ith basic modules used in $E_0(z)$, $\hat{E}(z)$, and $E_1(z)$, respectively. Next, we can map Fig. 6.10(a) onto our video co-processor with the mapping

$$\tilde{R}_i \longrightarrow M_{3i}, \quad \hat{R}_i \longrightarrow M_{3i+1}, \quad \bar{R}_\imath \longrightarrow M_{3i+2}, \tag{6.28}$$

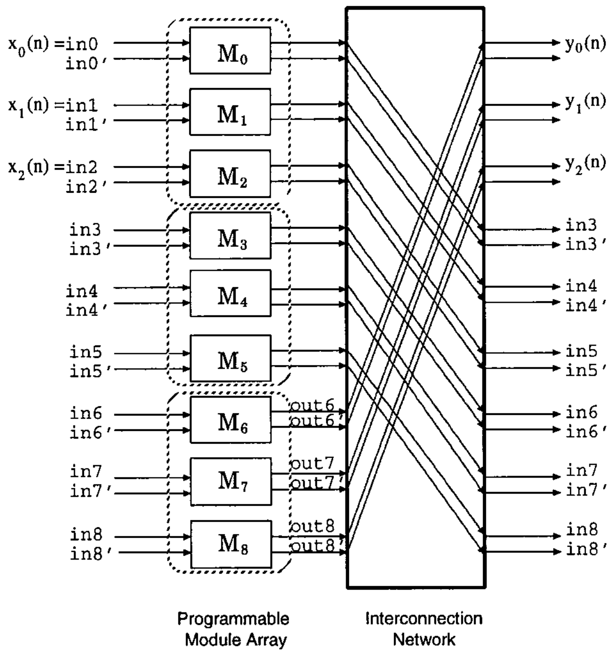
for $i = 0, 1, \ldots, N/2 - 1$. Besides, the interconnection network is set to Type II for the connections. Fig. 6.10(b) illustrates the realization of a $6th$-order FIR by the use of 9 programmable modules. The detailed setting of the video co-processor is described in Table 6.1 and 6.2.

Once the video co-processor has been initialized, the host processor can send data at f_s rate to the downsampling circuit in Fig. 2.8. 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 $y_i(n)$'s are generated, the FIR filtering output $y(n)$ is reconstructed through the upsampling circuit in a fully-pipelined way and the data rate for $y(n)$ is back to f_s.

As we can see, the only hardware overhead for this multirate operation is the downsampling and upsampling circuits in Fig. 2.8 for the pre- and post-processing of the data. Since we need $N/2$ modules for the implementation of each subfilter, a total of $3N/2$ modules will be used for a Nth-order FIR filter. That is, the speed performance is doubled at the expense of 50% hardware overhead. Nevertheless, this overhead is handled by simply activating more modules in the array and reconfigurating the interconnection network rather than implementing it explicitly.



(a)



(b)

Figure 6.10: (a) Multirate FIR lattice structure. (b) Mapping part (a) to the co-processor architecture.

6.2.2 Mapping of the Multirate IIR Architecture

Our co-processor design can also be reconfigurated to perform the multirate IIR filtering. Given an IIR system

$$H'(z) = \frac{1 + \sum_{i=1}^{M} p_i z^{-i}}{1 + \sum_{i=1}^{N} q_i z^{-i}} \tag{6.29}$$

($M \leq N$, M, N are even numbers), we can follow the derivations in Section 2.2.1 to find the polyphase components of $H(z)$, $E_0(z)$ and $E_1(z)$, as well as $\hat{E}(z)$. The mapping of the IIR multirate filtering is similar to the FIR case: We first implement each of the subfilters, $E_0(z)$, $E_1(z)$, and $\hat{E}(z)$, using the cascade IIR structure discussed in Section 6.1.3. The corresponding parallel architecture is shown in Fig. 6.11(a), where $\{\ \tilde{A}_{i,0}(z), \tilde{A}_{i,1}(z)\ \}$, $\{\ \hat{A}_{i,0}(z), \hat{A}_{i,1}(z)\ \}$, and $\{\ \bar{A}_{i,0}(z), \bar{A}_{i,1}(z)\ \}$, $i = 0, 1, \ldots, N/2 - 1$, are the subfilters of $E_0(z)$, $\hat{E}(z)$, $E_1(z)$, respectively. Then it can be mapped to our co-processor architecture by

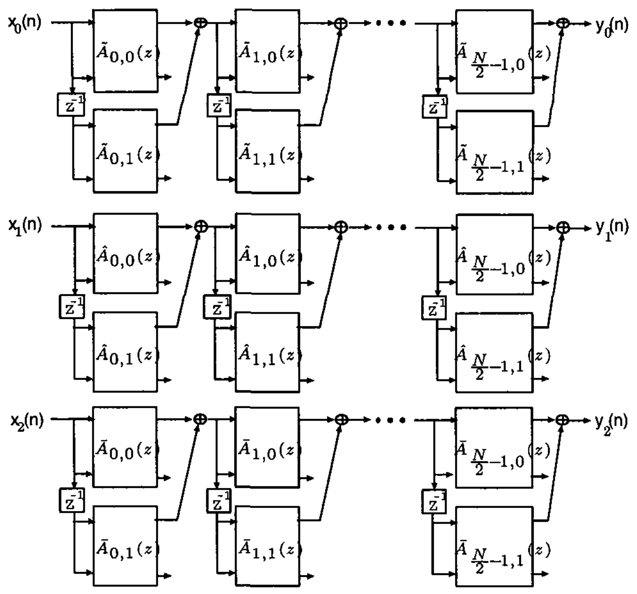
$$\begin{array}{lll} \tilde{A}_{i,0}(z) \longrightarrow M_{3i}, & \hat{A}_{i,0}(z) \longrightarrow M_{3i+2}, & \bar{A}_{i,0}(z) \longrightarrow M_{3i+4}, \\ \tilde{A}_{i,1}(z) \longrightarrow M_{3i+1}, & \hat{A}_{i,1}(z) \longrightarrow M_{3i+3}, & \bar{A}_{i,1}(z) \longrightarrow M_{3i+5}, \end{array} \tag{6.30}$$

for $i = 0, 1, \ldots, N/2 - 1$.

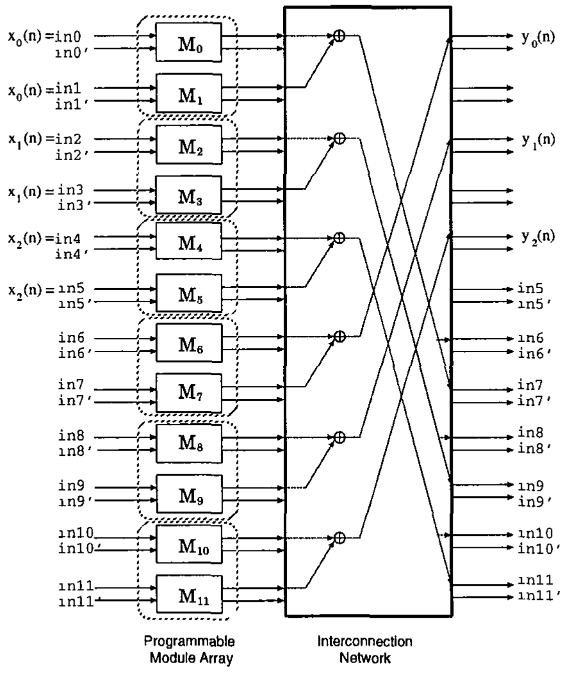
Figure 6.11(b) demonstrates the multirate $4th$-order IIR architecture using 12 programmable modules. The detailed settings of the modules and interconnection network can be found in Table 6.1 and 6.2. Note that the maximum order of $E_0(z)$ and $E_1(z)$ is still N; *i.e.*, the orders of the subfilters do not decrease after the decomposition. This indicates that the use of Fig. 2.8 will triple the hardware cost. Therefore, we will need $3N$ modules to realize a Nth-order multirate IIR filter.

6.2.3 Multirate Discrete Transform Architecture

In addition to multirate FIR/IIR filtering, our video co-processor can also be reconfigurated to perform multirate DT operations as discussed in Section 4.3. Note that the



(a)



(b)

Figure 6.11: (a) Multirate IIR lattice structure. (b) Mapping part (a) to the co-processor architecture.

multirate DT operations in Section 4.3. are performed using the IIR-based computational modules. Here we derive the rotation-based multirate DT architecture so that it can be mapped to our

Splitting the input data sequence, $x(n), i = 0, 1, \ldots, L$, into the *even* sequence

$$x_e(n) = x(2n), \quad n = 0, 1, \ldots, L/2 - 1, \tag{6.31}$$

and the *odd* sequence

$$x_o(n) = x(2n+1), \quad n = 0, 1, \ldots, L/2 - 1, \tag{6.32}$$

(4.10) and (4.11) can be rewritten as

$$\begin{aligned} X_C(k) &= \beta \sum_{n=0}^{L/2-1} \cos[(4n+1)\omega_k + \eta_k]\, x_e(n) \; + \; \beta \sum_{n=0}^{L/2-1} \cos[(4n+3)\omega_k + \eta_k]\, x_o(n) \\ &= X_{C,e}(k) + X_{C,o}(k), \qquad (6.33) \\ X_S(k) &= \beta \sum_{n=0}^{L/2-1} \sin[(4n+1)\omega_k + \eta_k]\, x_e(n) \; + \; \beta \sum_{n=0}^{L/2-1} \sin[(4n+3)\omega_k + \eta_k]\, x_o(n) \\ &= X_{S,e}(k) + X_{S,o}(k), \qquad (6.34) \end{aligned}$$

for $k = 0, 1, \ldots, N-1$. Following the derivations in [46][37], it can be shown that we can use the rotation-based module in Fig. 6.6(a) for the dual generation of $X_{C,e}(k)$ and $X_{S,e}(k)$ by setting

$$(t)\mathbf{f}_{k,e} = \begin{bmatrix} \hat{f}_{0,k} \\ \hat{f}_{1,k} \end{bmatrix} = \begin{bmatrix} \beta\cos((2L+1)\omega_k + \eta_k) \\ \beta\sin((2L+1)\omega_k + \eta_k) \end{bmatrix}, \quad \mathbf{R}_{k,e} = \begin{bmatrix} \cos(4\omega_k) & \sin(4\omega_k) \\ -\sin(4\omega_k) & \cos(4\omega_k) \end{bmatrix}. \tag{6.35}$$

Similarly, the same module can be used to obtain $X_{C,o}(k)$ and $X_{S,o}(k)$ with the setting

$$\mathbf{f}_{k,o} = \begin{bmatrix} \tilde{f}_{0,k} \\ \tilde{f}_{1,k} \end{bmatrix} = \begin{bmatrix} \beta\cos((2L+3)\omega_k + \eta_k) \\ \beta\sin((2L+3)\omega_k + \eta_k) \end{bmatrix}, \quad \mathbf{R}_{k,o} = \begin{bmatrix} \cos(4\omega_k) & \sin(4\omega_k) \\ -\sin(4\omega_k) & \cos(4\omega_k) \end{bmatrix}. \tag{6.36}$$

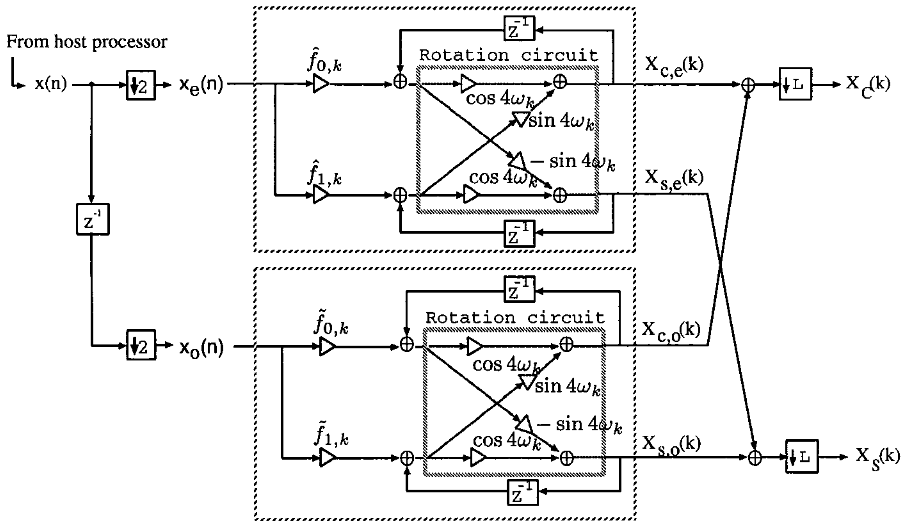
The parallel architecture to realize (6.33)-(6.36) is depicted in Fig. 6.12(a). The input data sequence $x(n)$ is first decimated into $x_e(n)$ and $x_o(n)$ through the decimator. Then

$X_{C,e}(k)$, $X_{S,e}(k)$, $X_{C,o}(k)$, and $X_{S,o}(k)$ are generated by the two modules in parallel and the outputs are summed up to obtain $X_C(k)$ and $X_S(k)$.

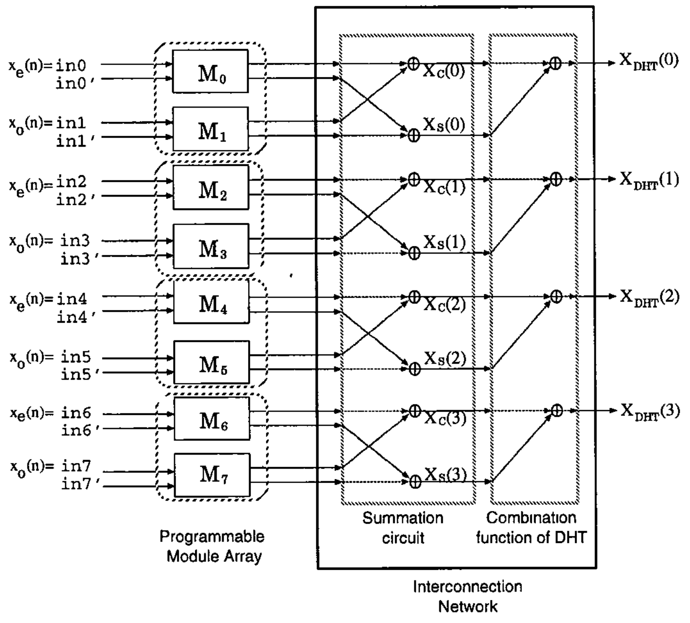
The multirate DT architecture can be mapped to the co-processor design by setting the parameters of module M_{2k} to $\mathbf{f}_{k,e}, \mathbf{R}_{k,e}$ and M_{2k+1} to $\mathbf{f}_{k,o}, \mathbf{R}_{k,o}$, respectively, for $k = 0, 1, \ldots, N/2 - 1$. Figure 6.12 shows the multirate 4-point DHT architecture based on 8 programmable modules. There are two parts inside the interconnection network: One is the summation circuit to combine the even and odd outputs of the array. The other is the circuit to perform the combination function defined in Table 4.3. In real implementation, we can either add one summation circuit so that the switch settings for the DT in Table 6.2 can still be used, or we can define new switch settings by merging these two circuits together. The hardware overhead to perform the multirate DT is the doubled complexity.

6.2.4 Application to Low-Power Design

In addition to speeding up the system, the feature of multirate data processing can also be applied to the low-power implementation of the proposed co-processor design. The co-processor can have a switch for the supply voltage. Under normal operation, the supply voltage is 5V. When the job is not computationally demanding, the supply voltage is switched to 3.1V and the co-processor switched to multirate mode. The system will still maintain the processing speed even though each processing element inside the module has been slowed down by the reduced voltage.



(a)



(b)

Figure 6.12: (a) Multirate architecture for the dual generation of $X_C(k)$ and $X_S(k)$. (b) Multirate 4-point DHT architecture based on 8 programmable modules.

6.2.5 Design Examples Using Multirate Operations

Multirate FIR Filtering

Given the FIR transfer function in (6.24), we first perform the polyphase decomposition which yields

$$\begin{aligned}
H_0(z) &= 1 - 0.1327z^{-1} + 0.5328z^{-2} + 0.1038z^{-3} + 0.2195z^{-4}, \\
H_1(z) &= -0.8843 - 1.1219z^{-1} - 0.8882z^{-2} - 0.3786z^{-3} - 0.1094z^{-4}, \\
\hat{H}(z) &= 0.1157 - 1.2546z^{-1} - 0.3554z^{-2} - 0.2748z^{-3} + 0.1101z^{-4}. \qquad (6.37)
\end{aligned}$$

Then we can follow the steps in (6.1)-(6.2) and (6.4)-(6.7) to find the parameters for each filter in (6.37). The results are listed in Table 6.4(a).

Multirate IIR (ARMA) Filtering

Consider the IIR (ARMA) filter shown below

$$H(z) = \frac{1 - 0.4000z^{-1} + 0.1600z^{-2}}{1 - 1.8192z^{-1} + 2.0598z^{-2} - 1.1248z^{-3} + 0.3422z^{-4}}. \qquad (6.38)$$

We can find its polyphase components from Appendix 2.2.1 as

$$\begin{aligned}
H_0(z) &= \frac{1 + 1.4921z^{-1} + 0.2219z^{-2} + 0.0548z^{-3}}{1 + 0.8100z^{-1} + 0.8346z^{-2} + 0.1446z^{-3} + 0.1171z^{-4}}, \\
H_1(z) &= \frac{1.4192 + 0.5920z^{-1} + 0.0431z^{-2}}{1 + 0.8100z^{-1} + 0.8346z^{-2} + 0.1446z^{-3} + 0.1171z^{-4}}, \\
\hat{H}(z) &= \frac{2.4192 + 2.0841z^{-1} + 0.2650z^{-2} + 0.0548z^{-3}}{1 + 0.8100z^{-1} + 0.8346z^{-2} + 0.1446z^{-3} + 0.1171z^{-4}}. \qquad (6.39)
\end{aligned}$$

Then the necessary parameters of each AMRA filter in (6.39) can be computed from (6.14)-(6.21) in Appendix. The parameter settings for the programmable module are in Table 6.4(b).

Multirate 8-point DCT

The rotation parameters θ_i's and the scaling factors $f_{0,i}$'s, $f_{1,i}$'s for the modules operating on the even and odd subsequences can be found by using (6.35) and (6.36), respectively.

The settings are given in Table 6.4(c).

6.3 Incorporation of the QRD-LSL Adaptive Filtering

In this section, we will incorporate the feature of adaptive filtering into the co-processor design. We will show that, with little modification of the programmable module design, the proposed co-processor can also perform the QR-decomposition based recursive least-squares lattice (QRD-LSL) algorithm [47].

6.3.1 CORDIC Operation and QRD-LSL Architecture

The CORDIC is capable of evaluating various rotation functions based on the shift-and-add operations [50]. There are two operation modes in the CORDIC processor: one is the **vector rotation** mode (see Fig. 6.13(a)) which will rotate the 2-input vector for a given angle θ. Let W be the total iteration number in CORDIC algorithm. In real implementation, the rotation is performed by feeding a sequence of ± 1, μ_i, $i = 0, 1, \ldots, W$, to the CORDIC processor. Suppose that the rotation circuit inside our programmable module is implemented using the CORDIC processor. The values of μ_{in} sequence can be calculated in advance and will be loaded to the module array during the initialization mode. On the other hand, the CORDIC in **angle accumulation** mode (see Fig. 6.13(b)) is to rotate the input vector until one of input components is annihilated; Meanwhile, the μ_{out} sequence that reflect the performed rotation are generated. In the applications of adaptive filtering, both modes are used for the updating of RLS parameters.

The QRD-LSL algorithm is one of the most promising candidate for the implementation of the recursive least-square (RLS) adaptive filtering. Fig. 6.14(a) shows the overall architecture to perform the linear prediction. The readers may refer to [47][53, Chap.18] for detailed operations. The QRD-LSL can be implemented using the CORDIC

	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}
S	000100	000100	000100	010100	100100	010100	010100	100100	010100	010100	010100	010100
$f_{0,i}$	0.9878	0.1154	-0.6574	0.8833	-0.4092	0.8005	0.9902	84.0896	0.9613	0.9756	0.3073	0.9923
$f_{1,i}$	0.9878	0.1154	-0.6574	0.8833	-0.4092	0.8005	0.9902	84.0896	0.9613	0.9756	0.3073	0.9923
r_i	1	1	1	1	1	1	1	1	1	1	1	1
θ_i	-0.1571	0.0731	0.8086	0.5086	-1.6262	0.6920	0.1406	0.0119	0.2827	0.2231	1.8484	0.1244
Intercon-nection	Type II											

(a)

	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}
S	000011	001111	000011	001111	000011	001111	000011	001111	000011	001111	000011	001111
$f_{0,i}$	0	-0.0614	0	-0.1104	0	-0.0657	-7.6102	0	-4.2364	0	0	0
$f_{1,i}$	1.4256	0.1551	3.4488	0.2992	2.0232	0.8060	2.3669	0	2.3669	0	2.3669	0
r_i	0.8100	0.8100	0.8100	0.8100	0.8100	0.8100	0.4225	0.4225	0.4225	0.4225	0.4225	0.4225
θ_i	2.0944	2.0944	2.0944	2.0944	2.0944	2.0944	1.5708	1.5708	1.5708	1.5708	1.5708	1.5708
Intercon-nection	Type IV											

(b)

	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7
S	000011	000011	000011	000011	000011	000011	000011	000011
$f_{0,i}$	0.5000	0.5000	-0.6533	-0.2706	0.5000	-0.5000	-0.2706	0.6533
$f_{1,i}$	0	0	-0.2706	-0.6533	0.5000	0.5000	-0.6533	0.2706
r_i	1	1	1	1	1	1	1	1
θ_i	0	0	0.3927	1.1781	0.7854	2.3562	1.1781	3.5343
Interconnection	Type V							

(c)

Table 6.4: Settings for the (a) FIR filter, (b) IIR (ARMA) filter, and (c) 8-point DCT under multirate operation.

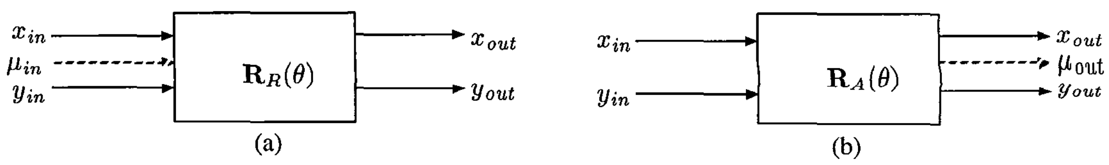


Figure 6.13: (a) CORDIC in vector rotation mode. (b) CORDIC in angle accumulation mode.

processors by replacing the angle computer with CORDIC in angle accumulation mode ($R_A(\theta)$), and the rotator is replaced with CORDIC in vector rotation mode ($R_R(\theta)$). The resulting system is shown in Fig. 6.14(b), where the dashed lines denote the data paths for the $\mu_{\mathbf{in}}$ and $\mu_{\mathbf{out}}$ sequences. The μ_{out} sequences are first computed by the $R_A(\theta)$'s using the forward and backward signals at each stage. Later the generated μ_{out} sequences will be sent to $R_R(\theta)$'s to rotate the signals at each stage.

6.3.2 Mapping QRD-LSL to the Programmable Video Co-processor

From Fig. 6.14, we observe that the basic modules used in QRD-LSL are very similar to our programmable module. Also, the connections can be easily handled by the interconnection network. We thus modify the programmable module by adding one direct path and one more switch for selecting this new direct path. On the other hand, one input port for μ_{in} and one output port for μ_{out} are also added for the propagation of the rotation parameters (see Fig. 6.15). Now based on the new programmable module, we can implement the QRD-LSL in a very straightforward way. The detailed settings of the module array as well as the connection type can be found in Table 6.1 and 6.2. Fig. 6.16 shows the implementation of a 4th-order QRD-LSL based on our programmable co-processor, where the adaptive filtering is performed in a fully-pipelined way without any feedback path.

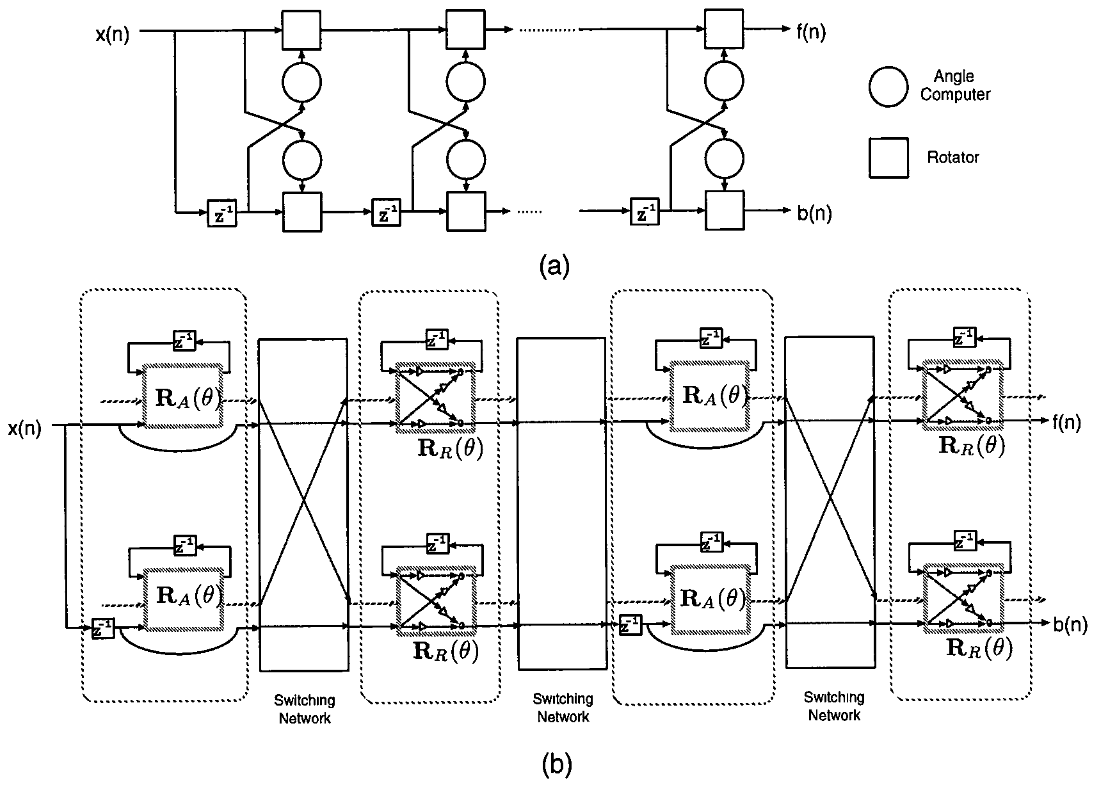


Figure 6.14: (a) QRD-LSL structure. (b) Realizing the QRD-LSL using the CORDIC processor.

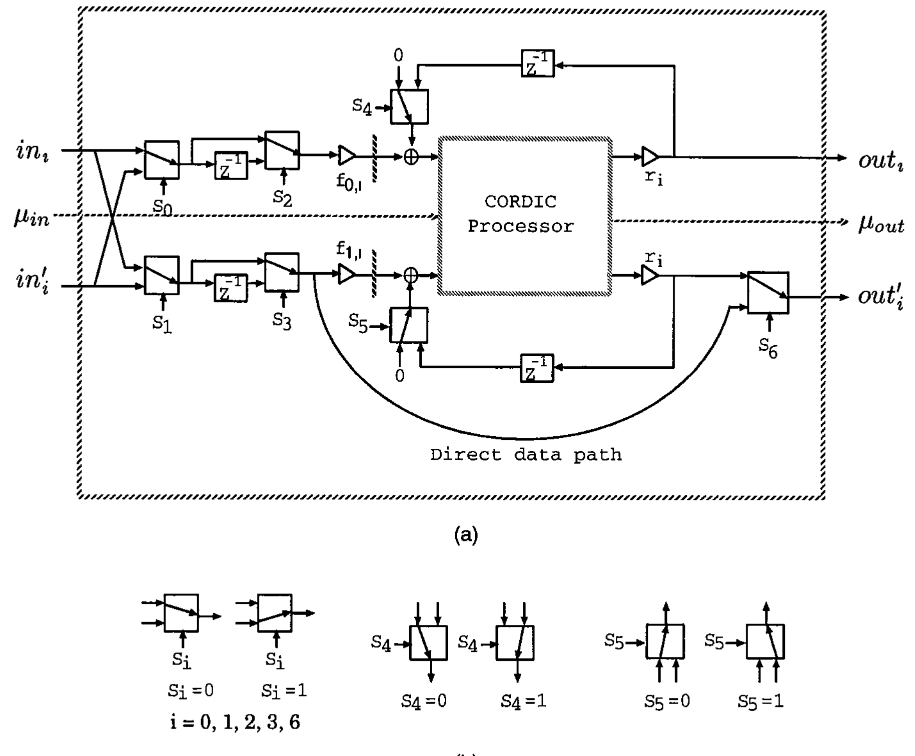


Figure 6.15: (a) New programmable module with the QRD-LSL feature. (b) Switches used in the module.

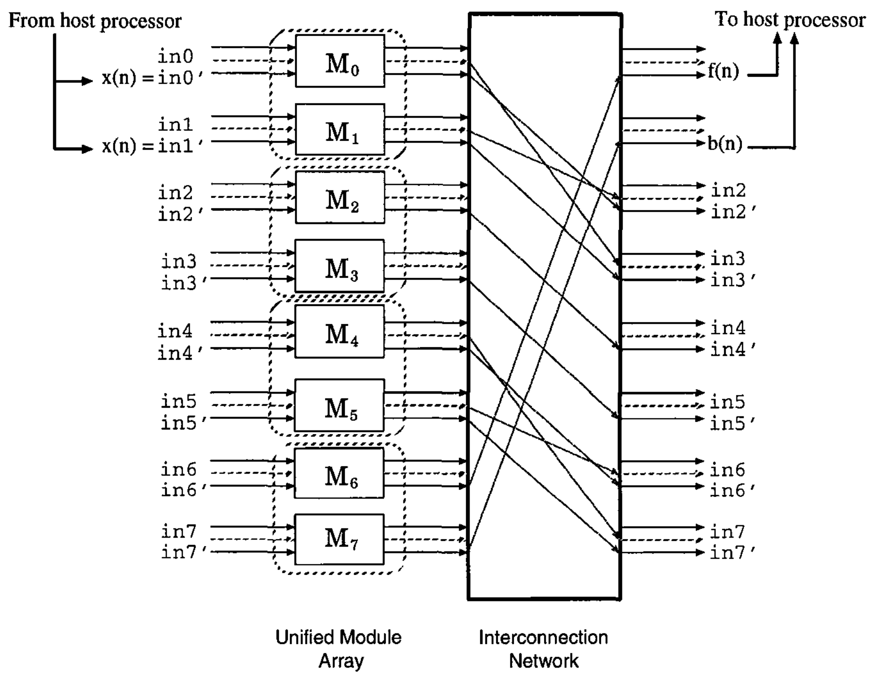


Figure 6.16: Realizing the QRD-LSL using the programmable video co-processor.

6.4 Performance and Comparison

Consider the programmable modules in Fig. 6.7 and Fig. 6.15. We can use either the general-purpose multipliers or the CORDIC processor [50] to implement the rotation circuit. In the former case, the critical path will be the path along two multipliers and two adders in the middle of the programmable module. Hence, the data throughput rate is approximately $\frac{1}{2 \cdot T_{MAC}}$, where T_{MAC} denotes the processing time for an MAC operation. On the other hand, if we use the CORDIC processor as the processing kernel of the programmable module, the data throughput rate will be limited by the CORDIC processor and the scaling multipliers, and can be approximated by $\frac{1}{T_{CORDIC}+T_{MAC}}$, where T_{CORDIC} denotes the total processing time to finish the CORDIC operation. Note that if the users need to perform adaptive filtering for their video applications, we must use CORDIC processor to realize the rotation circuit. The reason is that the operations of square-root and division are required to compute the rotation angles in the QRD-LSL filtering. The programmable module with general-purpose multipliers is not applicable in this case. The CORDIC processor, on the other hand, can perform all operations in QRD-LSL including angle computations and rotations by using μ_{in} and μ_{out} sequences.

The speed performance of the proposed co-processor design can be judged by the following examples. For the N^{th}-order FIR filtering, the general-purpose programmable DSP requires $N \cdot T_{MAC}$ processing time for each incoming serial data. As a result, the data processing rate will be degraded as N increases. On the contrary, our video co-processor can perform the FIR filtering at a fixed data rate of $\frac{1}{2 \cdot T_{MAC}}$ (general multiplier implementation) or $\frac{1}{T_{MAC}+T_{CORDIC}}$ (CORDIC implementation) for any $N < P$, where P is the total number of programmable modules used in the co-processor. Besides, the processing rate can be doubled in the multirate mode with the maximum order of the FIR equal to $\frac{2P}{3}$. Another example is the computation of the DCT. Suppose that the fast DCT algorithm in [32] is employed to realize the transform function using programmable DSP. It takes approximately $(\frac{N}{2}) \log_2 N$ MAC operations to compute N-point DCT; *i.e.*, the average processing time for each serial input data is $\frac{\log_2 N}{2} \cdot T_{MAC}$, which is proportional to the block size N. However, our video co-processor performs the time-recursive DCT at a fixed data rate as in the FIR case. We can also perform the multirate DCT to double the data processing rate by using doubled programmable modules.

In addition to the ASIC-like speed performance, our co-processor design also has good flexibility in modifying the programmed functions in the FIR/IIR/QMF/DT/QRD-RLS. As an example, we can easily increase the order/block-size of those programmed functions at the expense of invoking more programmable modules while retaining the same system throughput rate. Although the programmable DSP also has the flexibility of changing the order/block-size of the programmed function, the system throughput rate will be degraded as a result of increased MAC operations per data sample. Besides, we can easily change function specifications (*e.g.*, FIR tap coefficients) by reprogramming the parameters of the programmable modules. As to ASIC designs, the aforementioned modifications are in general not applicable.

The real-time processing speed as well as the programmable feature of this design makes it very attractive for video-rate applications. The programmable feature can significantly reduce the hardware cost compared to the ASIC-based implementations. Meanwhile, we do not trade the processing speed for this flexibility since all operations are performed in a parallel and fully-pipelined way as in dedicated ASIC designs.

Chapter 7

Conclusions and Future Research

In this dissertation, we presented a new algorithmic/architectural-level low-power design technique, the multirate approach, to compensate the speed penalty caused by lowered supply voltage of the DSP chips. We illustrate this new design concept by applying it to several major DSP problems such as the design of multirate FIR/IIR filtering architectures, low-power transform coding architectures, and programmable video co-processor with speed-up capability.

The major contribution of this dissertation is to link the well-known multirate signal processing techniques with algorithmic/architectural-level low-power VLSI architecture designs. This provides VLSI system designers and algorithm developers a new design tool in compensating the speed penalty in addition to the existing techniques of "parallel processing" and "pipelining". Beside the application of low-power design, the proposed multirate VLSI architectures can be readily used for very high-speed data processing. For example, if we want to perform DCT for serial data at 200 MHz, we may use the parallel architecture of Fig. 3.11, in which only 50MHz adders and multipliers are required. Therefore, we can perform very high-speed DCT by using only low-cost and low-speed operators. In the following, we will summarize our contributions and suggest further research directions.

We first proposed a design methodology for the low-power design of the FIR/IIR DSP systems. By applying the design methodology, we can perform FIR/IIR filtering using a multirate architecture that is running at M-times slower operating frequency than the input data rate. We showed that for the case of $M = 2$, the multirate FIR structure can save up to 71% of the power compared with the normal pipelined FIR structure. The predicted power saving is verified by the simulation results of the implemented QMF chips.

Next, we presented the algorithm-based low-power design of the transform coding architectures based on the multirate approach. Two different approaches are proposed. One is the Chebyshev polynomial approach and the other is the polyphase decomposition approach. We showed that by applying these approaches, we can reformulate the DCT/IDCT algorithms so as to reduce the operating frequency of the transforms at the architectural level without degrading the system throughput rate. Such a compensation capability will lead to drastic savings in the total power consumption. We also considered some new aspects of the multirate low-power design; namely, the logarithmic-complexity low-power architecture, unified IIR-based low-power transform coding architecture, and finite-wordlength analysis of the multirate VLSI architectures. We have shown that logarithmic low-power architecture is a good choice for VLSI implementation when both low-power dissipation and chip area are taken into consideration. The unified transform coding architecture allows us to perform various sinusoidal transforms using the same dedicated VLSI architecture. The real-time operations and the programmability of this design make it a promising candidate to be incorporated into the design of video co-processor. Finally, the finite-wordlength analysis gives us a tool to achieve a desired SNR by choosing a minimal wordlength. It not only reduces the total switching events (hence the power dissipation), but also provides a good control over the total chip area under the SNR constraint.

We also presented a system architecture of a massively parallel programmable video co-processor for numerically intensive front-end video/image communications. It can integrate and perform various functions (FIR/QMF/IIR/DT/QRD-LSL) for the host processor by simply loading the suitable parameters and reconfigurating the interconnection network. The proposed parallel architecture retains the advantage of ASIC designs but is much more flexible. Moreover, the architecture is regular and modular. As a consequence, they are very suitable for VLSI implementation. We also showed that we can reconfigurate the video co-processor to perform the multirate FIR/IIR/DT operations. The significance of the feature is twofold: Firstly, we can speed up the performance of the co-processor since the processing elements can now process data that are twice as fast as its processing speed. Secondly, the multirate feature can be applied to the low-power implementation of the co-processor. Furthermore, since the DCT-based motion estimation (ME) scheme in [54] employs DCT/DST as a basic processing kernel, and the computations are inherently local operations, we can also map the DCT-based ME scheme to our co-processor design.

Based on the research results presented in this dissertation, several new research topics can be further investigated:

- **Multirate Computer Arithmetic:** The bit-serial approach is a good way to save silicon area for the implementations of adders and multipliers [55]. However, the partial sum as well as the partial product need to be updated at every bit clock cycle. The high processing rate will result in an increase in the total switching events, hence the total power consumption of the chip. If we can also decimate the input bit stream into even and odd bit streams and obtain the same summation and/or product from the reformulated multirate bit-serial arithmetic operations, we can achieve low-power consumption at the bit level. The research issues include the study of different computer arithmetic architectures at bit level for a given operation. For example, carry-look-ahead adder, carry-ripple adder and carry-selection adder will have different properties (power/speed/area) when we try to perform the addition operation using the decimated bit streams. Similar studies can be applied to the multiplication operation with different arithmetic architectures.

- **Multirate Linear Algebraic Operations:** The matrix-vector and matrix-matrix multiplications are the basic operations in modern signal processing such as direction-of-arrival (DOA) and recursive-least-squares (RLS) estimation [53]. Their systolic implementations can be found in [56]. We can consider reformulated systolic architectures that perform those linear algebraic operations using the multirate approach; *i.e.*, the inputs are first decimated into low-speed short-length sub-sequences and/or small-size sub-matrices. Then we try to compute the partial matrix-vector/matrix-matrix products at a lower speed. Once the computations are complete, we try to recombine those partial products to obtain the desired product.

- **Applications to Specific DSP Algorithms:** We have applied the multirate approach to the design of transform coding kernels, which results in very efficient low-power/high-speed architectures. The advantage is obtained by fully exploiting the inherent properties of the time-recursive DT operations. In the future work, we can consider how to extend the proposed low-power design technique to other DSP applications.

  One possible application is in the adaptive filtering that has a wide range of applications such as adaptive channel equalization, noise cancellation, and system identification [53]. Currently, most adaptive filtering algorithms need to update the system parameters for each incoming input data. When they are implemented in hardware, the processing elements must finish all the updating operations before next data comes in. This implies that the speed compensation is not easy to achieve for these implementations. We can consider multirate adaptive filtering algorithms in which the updating of parameters can be performed using the decimated input data. Therefore, the speed constraint can be relaxed.

Furthermore, we can consider the multirate implementations of some well-known digital communication algorithms such as the Viterbi algorithm and the Reed-Solomon codec [57]. Due to the prevalence of personal communicators, low-power but high-speed codec chips become the essential parts in the communication devices. The multirate feature can help reduce the power of those codec chips while maintaining the required throughput rate.

Bibliography

[1] D. Dobberpuhl *et al*, "A 200MHz microprocessor implemented in CMOS technology," *IEEE J. Solid-State Circuits*, pp. 1585–1598, Nov. 1992.

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

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

[4] J. M. Rabaey, "Low power digital design," in *Tutorials of IEEE Int'l Symp. Circuits and Systems*, ch. 8, London: IEEE Press, May 1994, pp. 373–386.

[5] G. M. Blair, "Designing low power digital CMOS," *Electronics and Communication Engineering Journal*, pp. 229–236, Oct. 1994.

[6] Z. J. Lemnios and K. J. Gabriel, "Low-power electronics," *IEEE Design & Test of Computers*, vol. 11, pp. 14–23, Winter 1994.

[7] A. P. Chandrakasan and R. W. Brodersen, "Minimizing power consumption in digital CMOS circuits," *Proceedings of the IEEE*, vol. 83, pp. 498–523, April 1995.

[8] 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.

[9] A. Yoshino *et al.*, "Design methodology for low-power, high-speed CMOS devices utilizing SOI technology," in *Proc. IEEE International SOI Conference*, (Palm Springs), Oct. 1993, pp. 170–171.

[10] K. Hwang, *Computer arithmetic : principles, architecture, and design.* Wiley, 1979.

[11] W. C. Athas *et al.*, "Low-power digital systems based on adiabatic-switching principles," *IEEE Trans. VLSI Syst.*, vol. 2, pp. 398–406, Dec. 1994.

[12] 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.

[13] M. Alidina *et al.*, "Precomputation-based sequential logic optimization for low power," *IEEE Trans. VLSI Syst.*, vol. 2, pp. 426–436, Dec. 1994.

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

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

[16] S. Gary *et al.*, "PowerPC 603, a microprocessor for portable computers," *IEEE Design & Test of Computers*, vol. 11, pp. 14–23, Winter 1994.

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

[18] 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, March 1993.

[19] K. J. R. Liu, C. T. Chiu, R. K. Kolagotla, and J. F. J. 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, April 1994.

[20] J. D. Johnston, "A filter family designed for use in quadrature mirror filter banks," *Proc. IEEE Int'l Conf. Acoust. Speech, Signal Processing*, 1980, pp. 291–294.

[21] P. P. Vaidyanathan, *Multirate systems and filter banks.* Englewood Cliffs, NJ: Prentice Hall, 1993.

[22] 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.

[23] 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.

[24] H. K. Kwan and M. T. Tsim, "High speed 1-D FIR digital filtering architectures using polynomial convolution," in *Proc. IEEE Int'l Conf. Acoust. Speech, Signal Processing*, (Dallas, TX), April 1987, pp. 1863–1866.

[25] Z. J. Mou and D. Duhamel, "Fast FIR filtering: Algorithm and implementations," *Signal Processing*, vol. 13, pp. 377–384, 1987.

[26] Z.-J. Mou and P. Duhamel, "Short-length FIR filters and their use in fast non-recursive filtering," *IEEE Trans. Signal Processing*, vol. 39, pp. 1322–1332, June 1991.

[27] 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.

[28] G. Steidl, "Chebyshev polynomial derivation of composite-length DCT algorithms," *Signal Processing*, vol. 29, pp. 17–27, Oct. 1992.

[29] L. Fox and I. B. Parker, *Chebyshev polynomials in numerical analysis.* London: Oxford University Press, 1968.

[30] H. R. Schwarz, *Numerical analysis : A comprehensive introduction.* John Wiley & Sons, 1989.

[31] 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.

[32] 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.

[33] A. N. Skodras, "Fast discrete cosine transform pruning," *IEEE Trans. Signal Processing*, vol. 42, pp. 1833–1837, July 1994.

[34] H. S. Malvar and D. H. Staelin, "The LOT: Tansform coding without blocking effects," *IEEE Trans. Acoust. Speech, Signal Processing*, vol. 37, pp. 553–559, April 1989.

[35] H. S. Malvar, "Lapped transforms for efficient transform/subband coding," *IEEE Trans. Acoust. Speech, Signal Processing*, vol. 38, pp. 969–978, June 1990.

[36] H. S. Malvar, "Extended Lapped Transforms: Properties, applications, and fast algorithms," *IEEE Trans. Signal Processing*, vol. 40, pp. 2703–2714, Nov. 1992.

[37] E. Frantzeskakis, J. 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), July 1993, pp. 100–111.

[38] H. S. Malvar, "Fast algorithm for modulated lapped transform," *Electron. Lett.*, vol. 27, pp. 775–776, Apr. 1991.

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

[40] A. V. Oppenheim and R. W. Schafer, *Discrete-time Signal Processing.* Prentice Hall, 1989.

[41] 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.

[42] V. Srinivasan and K. J. R. Liu, "Full custom VLSI inplementation of high-speed 2-D DCT/IDCT chip," in *Proc. IEEE Int. Conf. Image Processing*, (Austin, Texas), 1994, pp. III.606–610.

[43] C.-T. Chiu and K. J. R. Liu, "Real-time parallel and fully pipelined two-dimensional DCT lattice structures with application to HDTV systems," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, pp. 25–37, March 1992.

[44] K. Aono *et al.*, "A video digital signal processor with a vector-pipeline architecture," *IEEE J. Solid-State Circuits*, vol. 27, pp. 1886–1893, Dec. 1992.

[45] K. Herrmann *et al.*, "Architecture and VLSI implementation of a RISC core for a monolithic video signal processor," in *VLSI Signal Processing VII* (J. Rabaey, P. M. Chau, and J. Eldon, eds.), IEEE Press, 1994, pp. 368–377.

[46] E. Frantzeskakis, J. Baras, and K. J. R. Liu, "Time-recursive architectures and wavelet transforms," in *Proc. IEEE Int. Conf. Acoust. Speech, Signal Processing*, (Minneapolis), April 1993, pp. I.445–448.

[47] I. K. Proudler, J. G. McWhirter, and T. J. Shepherd, "Computationally efficient QR decomposition approach to least squares adaptive filtering," *IEE Proceedings-F*, vol. 138, pp. 341–353, Aug. 1991.

[48] L. B. Jackson, *Digital Filters and Signal Processing.* Kluwer Academic Publishers, 2nd ed., 1989.

[49] J. Makhoul, "Linear Prediction: A tutorial review," *Proc. IEEE*, vol. 63, pp. 561–580, April 1975.

[50] Y. H. Hu, "CORDIC-based VLSI architectures for digital signal processing," *IEEE Signal Processing Magazine*, vol. 9, pp. 16–35, July 1992.

[51] K. N. Ngan and W. L. Chooi, "Very low bit rate video coding using 3D subband approach," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, pp. 309–316, June 1994.

[52] D. Taubman and A. Zakhor, "Multirate 3-D subband coding of video," *IEEE Trans. Image Processing*, vol. 3, pp. 572–588, Sept. 1994.

[53] S. Haykin, *Adaptive Filter Theory.* Prentice-Hall, Englewood Cliffs, N.J., 2nd ed., 1991.

[54] U. V. Koc and K. J. R. Liu, "Discrete-cosine/sine-transform based motion estimation," in *Proceedings of IEEE International Conference on Image Processing (ICIP-94)*, vol. 3, (Austin, Texas), Nov. 1994, pp. 771–775.

[55] P. Denyer and D. Renshaw, *VLSI Signal Processing: A Bit-Serial Approach.* Addison-Wesley, 1985.

[56] S. Y. Kung, *VLSI Array Processors.* Prentice-Hall, 1988.

[57] J. G. Proakis, *Digital Communications.* McGraw-Hill, 2nd ed., 1989.