REALIZATION AND IMPLEMENTATION OF STATE-SPACE IIR DIGITAL FILTERS

by
AYMAN El-SAYED TAWFIK
B.Sc., 1983 and M.Sc., 1989
Ain Shams University, Cairo, EGYPT.

A DISSERTATION SUBMITTED IN PARTIAL FULFILMENT
OF THE REQUIRMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

in the Department of
Electrical and Computer Engineering

We accept this dissertation as conforming
to the required standard

Dr. F. El-Guibaly, Co-Supervisor, Dept. of Elec. and Comp. Eng.

Dr. P. Agathoklis, Co-Supervisor, Dept. of Elec. and Comp. Eng.

Dr. W. -S. Lu, Departmental Member, Dept. of Elec. and Comp. Eng.

Dr. M. Nahon, Outside Member, Dept. of Mechanical Eng.

Dr. V. Sreeram, External Examiner, University of Western Australia

Supervisors: Dr. F. El-Guibaly and Dr. P. Agathoklis

ABSTRACT

Digital filters form an important part in communications and information processing systems, and are increasingly incorporated in consumer electronics products.

In this thesis, the realization and the implementation of IIR digital filters are considered. New techniques for obtaining realizations of IIR filters that yield acceptable performance under finite wordlength constraints and are suitable for efficient hardware implementation are presented. Two main classes of efficient realizations have been obtained based on the state-space description. The first class of realizations consists of cascade or parallel connection of second-order sections having coefficients which are power-of-two and/or sum-of-two power-of-two. It is shown that these realizations provide low output roundoff noise, freedom of limit cycles and low computational complexity. The second class of realizations proposed is based on using simple residue-feedback schemes. It is shown that these realizations involving residue-feedback provide low output roundoff noise with in some cases significant reduction in the number of arithmetic operations required. Both classes of realizations provide lower output roundoff noise than many other existing low-noise realizations.

The implementations of some of the proposed realizations using DSP and/or ASIC VLSI have been also considered. Experimental results from the DSP implementation of some of the realizations proposed confirm their usefulness. Further, three efficient VLSI array-processor implementations of the IIR digital filters are also presented. The first implementation is developed from realization with residue feedback and is guaranteed to provide higher input sampling rate than existing direct implementation for narrow-band filters. The second implementation is developed from the general state-space realizations

with full system matrix and provides a good compromise between hardware area and speed. The third implementation is developed from block-state description of IIR digital filter and provides high input sampling rate which is not limited by the speed of the processing elements involved. A new fixed-point inner-product processor is also developed to enhance the performance of some of the proposed implementations. These proposed implementations give the designer the possibility to choose the one best suited to the requirements (speed and/or area) of a particular application.

The results presented in this thesis indicate that the state-space description is a useful tool in obtaining realizations of narrow-band IIR digital filters which are not only efficient in terms of finite wordlength performances but they are also suitable for efficient hardware implementations.

Examiners

Dr. F. El-Guibaly, Co-Supervisor

Dr. P. Agathoklis, Co-Supervisor

Dr. W. -S. Lu, Departmental Member

Dr. M. Nahon, Outside Member

Dr. V. Sreeram, External Examiner

Table of Contents

List of Tables

List of Figures

Abbreviations

ACM	Accumulator Multiplier
A/D	Analog-To-Digital
AM	Array Multiplier
ASIC	Application Specific Integrated Circuit
AT	area $\times$ time
AT^2	area $\times$ time^2
CSA	Carry Save Adder
CSIPP	Carry Save Inner Product Processor
CPA	Carry Propagate Adder
DG	Dependance Graph
DSP	Digital Signal Processor
FIR	Finite Impulse Response
FFT	Fast Fourier Transform
FPGA	Field Programmable Gate Array
FRF	Full Residue Feedback
FWL	Finite Word Length
IIR	Infinite Impulse Response

I/O	Input/Output
IPSP	Inner Product Step Processor
LSB	Least Significant Byte
MVM	Matrix Vector Multiplication
MRH	Mullis, Roberts & Hwang
MSB	Most Significant Byte
NG	Noise Gain
PE	Processor Element
PIPSP	Pipelined Inner Product Step Processor
RDG	Reduced Dependence Graph
RF	Residue Feedback
RPSD	Relative Power Spectral Density
SB	Smith, Bomar
SFG	Signal Flow Graph
SISO	Single Input Single Output
S/N	Signal-To-Noise
SRF	Suboptimal Residue Feedback
SUN	State Update Network
VLSI	Very Large Scale Integration

Acknowledgments

I would like to express my deepest gratitude to my supervisors, Dr. F. El-Guibaly and Dr. P. Agathoklis for their continuous support and encouragement and for giving lots of their precious time to supervise this research work and the process of writing this manuscript.

Financial assistance received from Dr. F. El-Guibaly and Dr. P. Agathoklis (through the Natural Sciences and Engineering research Council of Canada and the Micronet, National Centers of Excellence Program) is gratefully acknowledged.

Finally, I express my full gratitude to all members of my family, especially my mother, for being with me with their hearts all the time.

Dedication

To my Mother,

To my Family

To my land; Egypt.

Chapter 1

Introduction

1.1 Digital Filters

A digital filter is a digital system that can be used to filter discrete-time signals. Digital filters form an important part in communication, information processing systems, and are increasingly incorporated in many consumer electronic products (e.g., digital hi-fi audio systems). The choice of filter has a significant impact on the performance and viability of all these systems. In recent years, advances in VLSI technology have further increased the usage of digital filters. It is now practical to implement real-time digital filters using high-speed microprocessor [1], [2], new special digital signal processors (DSP) chips [3], [4] or dedicated digital filter chips [5], [6].

The input-output relation of a causal, linear, time-invariant discrete-time system can be expressed in the form

$$y(n) = \sum_{i=0}^{N} \alpha_i u(n-i) - \sum_{i=1}^{N} \beta_i y(n-i) \qquad (1.1)$$

where $u(n)$ and $y(n)$ denote the input and output signals of the system, α_i and β_i are known as the coefficients of the system and N is referred to as the order of the system. If all β_i are zeros, then the linear system is called a non-recursive system; otherwise it is a recursive system. Non-recursive filters are referred as FIR (Finite Impulse Response) fil-

ters, and the recursive filters as IIR (Infinite Impulse Response) filters. The transfer function of the digital filter $H(z)$ can be easily derived by the application of the z-transform on (1.1) which results in

$$H(z) = \frac{\sum_{i=0}^{N} \alpha_i z^{-i}}{1 + \sum_{i=1}^{N} \beta_i z^{-i}} \tag{1.2}$$

The design of digital filters comprises three general steps, as follows:

1. Approximation
2. Realization and study of finite wordlength (FWL) effects
3. Implementation

The approximation step involves determination of the order and the coefficients to satisfy the given design specifications. Extensive discussion on existing design methods for digital filters can be found in many standard books on this subject, such as [7]. The realization (synthesis) step of a digital filter is the process of converting the transfer function of the filter into a network. The network obtained is said to be the realization of the transfer function and there are an infinite number of realizations for the same transfer function. The performance of these various realizations of the same digital filter transfer function, however, is different in a practical implementation. In practice, digital filter hardware implementation has a finite precision which depends on the length of registers used to store numbers, the type of binary number system used (e.g., signed magnitude, two's complement), the type of arithmetic used (e.g., fixed-point, floating-point), etc. Usually, the realization step is accompanied with the study of those FWL effects. The implementation step can take two forms: software or hardware. Software involves the implementa-

tion of the filter network on a general-purpose digital computer or DSP chip. Hardware involves the mapping of the filter network onto dedicated hardware. The choice of implementation depends on the application at hand.

In this dissertation, the realization and the implementation of IIR digital filters are considered. The filter order and the coefficients α_i and β_i are assumed known. Thus, the transfer function of the IIR digital filter in (1.2) is assumed to be already available.

Equation (1.1) or (1.2) indicates that a typical filtering operation involves two basic arithmetic operations, namely, multiplication and addition. The computational complexity of a filter implementation is usually defined by the number of multiplication and addition operations and has direct impact on the hardware cost and processing speed. Further, the type of arithmetic used, fixed-point or floating-point, has an impact on cost and speed. Compared with fixed-point arithmetic operations, floating-point multiplication and addition require more hardware area and result in slower computational speed. For nonreal-time applications on general digital computers, floating-point arithmetic is always preferred since neither the cost of hardware nor the processing speed is a significant factor. For real-time applications, fixed-point arithmetic is preferred since the area of the dedicated hardware and computational speed are always of prime concern [8], [9]. In this dissertation, the realization and the implementation of IIR digital filters using fixed-point arithmetic is considered.

1.2 Finite Wordlength Effects

In digital filter implementations, the representation of signals must have finite-precision due to using registers of finite wordlength. There are three primary finite wordlength effects in fixed-point IIR digital filters implementations. These effects are:

1. Changes in the input/output description of the filter due to representing the filter coefficients in finite wordlength registers.

2. Roundoff noise and quantization limit cycles caused by the quantization of the signals within the realization.
3. Limit cycles due to overflow which may occur when the internal overflowed variables is modified to be within the representable range.

These finite wordlength effects (FWL) will be briefly described in the following subsections

1.2.1 Coefficient Quantization

The quantization of the filter coefficients is manifested by a deterministic change in the input/output (I/O) characteristic of the filter transfer function $H(z)$. As this effect is deterministic, it is easier to analyze than other FWL effects. One popular method of measuring the effects of coefficient quantization is to examine the movement of poles and zeros caused by coefficient quantization [10]. The main conclusion of this analysis is that if the poles are close together, as in the case of narrow-band filters, a small change in the denominator coefficients of $H(z)$ can cause a large change in the location of the poles (a similar argument can be made for the zeros of the filter). Thus, it is recommended to separate the poles (and zeros) of a filter [11]. This is usually done by using cascade or parallel connections of first- and second-order subfilters. One can use these subfilters to isolate closely packed poles and zeros into separate realizations.

1.2.2 Signal Quantization

Signal quantization occurs when a variable wordlength exceeds the wordlength of the available hardware. In an IIR digital filter implementation, a wordlength reduction is necessary to prevent the wordlengths of the signals from increasing indefinitely. This reduction of the wordlength is commonly called "signal quantization process". Let us assume that the signal wordlength after quantization is b fractional bits. So, the quantization step

size between quantization levels is $q = 2^{-b}$. The output of a finite wordlength multiplier (i.e., multiplier followed by a quantizer) can be expressed as

$$Q[cu(n)] = cu(n) - e(n) \tag{1.3}$$

where $cu(n)$ and $e(n)$ are the exact product and quantization error, respectively. $Q[\ \cdot\ ]$ represents the quantization process which can be rounding or truncation and *e(n)* will be referred by the generic term *roundoff noise*. The finite wordlength multiplier can thus be represented by the model depicted in Fig. 1.1.

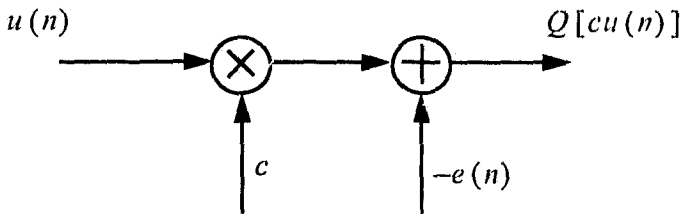


Figure 1.1: The model for product quantization.

Theoretical studies and numerical simulations have shown [12], [13] that $e(n)$ can be approximated by a white noise sequence, uncorrelated with $u(n)$ and uniformly distributed. This model is valid under the assumption that the signal levels throughout the filter are much larger than the quantizer step size q and the input is spectrally active. More discussion about the validity of this model can be found in [14]. The noise power associated with $e(n)$ depends on the binary system used (e.g., two's complement or signed magnitude) and the quantization process (rounding or truncation). The most common type of the product quantization is the rounding of two's complement numbers to the nearest quantization level; i.e., $-q/2 \le e(n) \le q/2$ with equal probability. This two's complement rounding will be assumed throughout the thesis, unless explicitly stated. In two's complement rounding, the mean is zero and the noise variance σ_r^2 can be calculated as

$$\sigma_r^2 = q^2/12 \tag{1.4}$$

If the input to the digital filter is low-level, internal rounding errors are highly correlated and the white noise model for *e(n)* in Fig. 1.1 is not valid in this case. An important example is the case of zero or constant input signal. Ideally, the output of a stable discrete-time filter would asymptotically approach zero or constant for zero-input or constant input, respectively, but because of quantization, small limit cycle oscillations can occur. These limit cycles usually have small amplitude and are usually called quantization (or granularity) limit cycles [15].

1.2.3 Overflow

Overflow in fixed-point digital filters can occurr due to the involved addition operations. Transforming the overflowed variable to be within the representable range results in a highly nonlinear effect. Overflow can be avoided by using scaling of the variables. Scaling implies that the numerical values of internal filter variables (the inputs to the internal registers) remain in a range appropriate to the available hardware. The conservative scaling can eliminate any possibility of overflow, but may result in increasing the output roundoff noise relative to the input signal and thus reducing the output signal-to-noise (S/N) ratio. A popular approach is to make the probability of overflow acceptably small by employing a mild scaling constraint such as l_2-scaling [16]. This l_2-scaling is considered less conservative than many other scaling techniques and in the same time it leads to simple mathematical formulations [16]-[17].

l_2-scaling technique does not exclude the possibility of an overflow and therefore the performance of the filter if overflow happens is critical. The filter performance during overflow depends on many factors such as the filter realization, the nature of the input, the binary system used and the overflow function used in the filter. One possible consequence of the overflow nonlinearity is that the output of the filter after an internal overflow can be independent of the input sequence. This condition is called an overflow

oscillation. The two common overflow functions are the two's complement function and saturation function shown in Fig. 1.2. The two's complement overflow function has the advantage that it is not required to explicitly detect overflow and correct it while saturation overflow function requires special arrangements to detect the overflow and to saturate the result. On the other hand, it has been shown that saturation overflow function can preclude overflow oscillations in second-order sections [18].

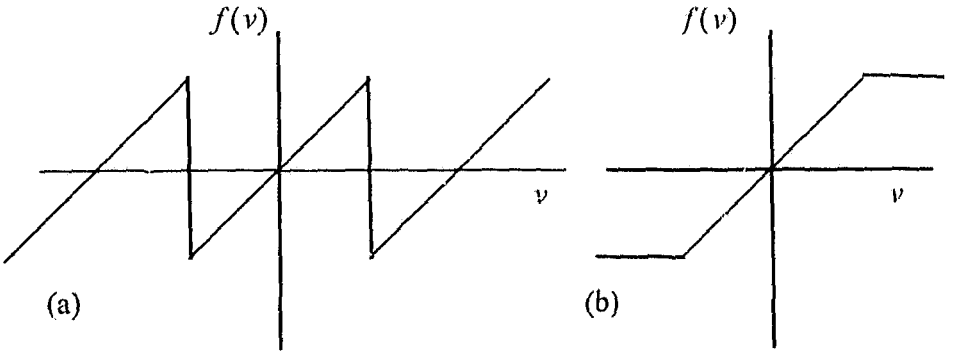


Figure 1.2: Overflow function characteristics. (a) two's complement. (b) saturation.

1.3 Study of FWL Effects in IIR Filter Realizations

Analysis of the FWL effects is usually done considering any of the three FWL effects discussed in Section 1.2 independently [17]. There are many approaches to study these FWL effects. In this section, the study of these FWL effects in direct structure and in state-space structures will be briefly summarized.

1.3.1 Direct Realization of IIR Filters

The realizations of the IIR digital filter by using the coefficients α_i and β_i in (1.2) are called direct realizations. These direct realizations have the advantage of the lowest com-

putational complexity (they require only $2N+1$ multiplications). However, these direct realizations perform poorly under FWL constraints. They have high output roundoff noise, high coefficient sensitivity and they are susceptible to overflow and quantization limit cycles [17], [18]. These bad performance becomes more apparent and more severe when narrow-band IIR filters are considered. Cascade (or parallel) connection of second-order sections, where each second-order section can be realized as direct form, improves the FWL performance [19] to some extent. However, their FWL performance is still not satisfactory especially for narrow-band IIR filters [20], [21]. There are two approaches to handle this problem. The first approach is to try to improve the FWL performance of the direct realization [20-27]. This approach generally trades the improvement in the FWL performance with increased computational complexity. The second approach is to find other realizations which perform much better under the FWL constraints [28-29]. A powerful approach to investigate new realizations is by using the internal description of the linear systems which is known as the state-space description. The investigation of different realizations for the same transfer function can be carried out by applying similarity transformations.

1.3.2 State-Space Description of IIR Filters

The state-space description $\{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},d\}$ of the Nth-order stable minimal SISO transfer function in (1.2) is given by

$$\boldsymbol{x}(n+1) = \boldsymbol{A}\boldsymbol{x}(n) + \boldsymbol{B}u(n) \qquad \boldsymbol{x}(n) \in \Re^N \tag{1.5}$$

$$y(n) = \boldsymbol{C}\boldsymbol{x}(n) + du(n) \qquad u(n), y(n), d \in \Re \tag{1.6}$$

where $\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}$ are matrices of dimension $N \times N$, $N \times 1$, and $1 \times N$, respectively, and d is a scalar. The block diagram of the state-space description is shown in Fig. 1.3 and the transfer function is given by

$$H(z) = d + \boldsymbol{C}(z\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B} \tag{1.7}$$

Under the assumption of infinite precision and exact representation of $\{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},d\}$, all the realizations of (1.5) and (1.6) which satisfy (1.2) would have the exact performance since they represent the same transfer function $H(z)$. However under the FWL constraint, these realizations perform differently.

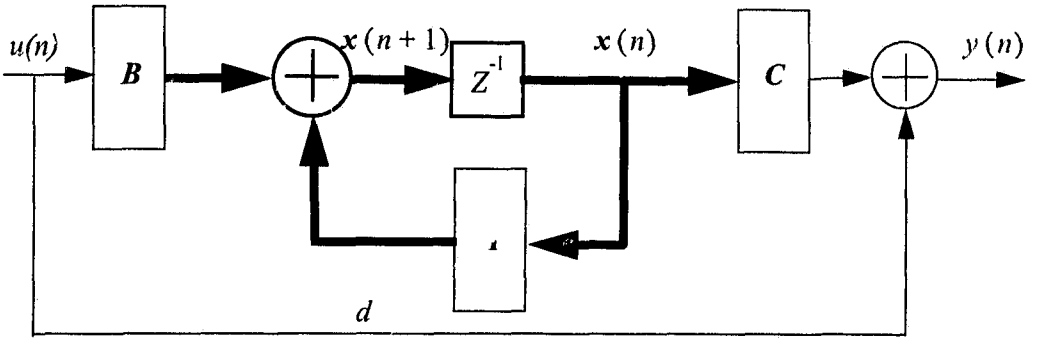


Figure 1.3: Block diagram of the state-space description of a linear system.

In order to calculate the output roundoff noise, the following FWL implementation of (1.5) and (1.6) will be adopted

$$\hat{\boldsymbol{x}}(n+1) = \boldsymbol{A}Q[\hat{\boldsymbol{x}}(n)] + \boldsymbol{B}u(n) \tag{1.8}$$

$$\hat{y}(n) = \boldsymbol{C}Q[\hat{\boldsymbol{x}}(n)] + du(n) \tag{1.9}$$

The state-space matrices $\{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},d\}$ and the input $u(n)$ are assumed to have exact representation, i.e., coefficient and input quantizations are neglected for the present analysis. In (1.8) and (1.9), the addition operations are assumed to be executed in double-precision

accumulators. The quantization process $Q[\ \cdot\ ]$ can be expressed as

$$Q[\hat{\boldsymbol{x}}(n)] = \hat{\boldsymbol{x}}(n) - \boldsymbol{e}(n) \tag{1.10}$$

where $\boldsymbol{e}(n)$ is the roundoff residue vector with each element having a variance of σ_r^2 defined in (1.4).

l_2-scaling which guarantees equal probability of overflow for all states is equivalent to imposing the following constraint [16]

$$k_{ii} = 1 \qquad \forall i \tag{1.11}$$

where k_{ii} is the ith diagonal element of the covariance matrix $\boldsymbol{K}$ which can be calculated from [16]

$$\boldsymbol{K} = \boldsymbol{A}\boldsymbol{K}\boldsymbol{A}^t + \boldsymbol{B}\boldsymbol{B}^t \tag{1.12}$$

where "t" denotes the transpose of a matrix.

By using the noise model in Fig. 1.1 and assuming double-precision additions, the output roundoff noise can be calculated as

$$\sigma^2 = \sigma_r^2 [1 + \text{trace}(\boldsymbol{W})] \tag{1.13}$$

where $\boldsymbol{W}$ is the noise matrix which can be calculated from [16]

$$\boldsymbol{W} = \boldsymbol{A}^t \boldsymbol{W} \boldsymbol{A} + \boldsymbol{C}^t \boldsymbol{C} \tag{1.14}$$

The first term on the RHS of (1.13) is due to rounding at the output node while the second term is due to the propagation of the roundoff errors of the internal states.

Under any similarity transformation $\boldsymbol{T}$, any realization $\{\boldsymbol{A}_0, \boldsymbol{B}_0, \boldsymbol{C}_0, d\}$ of $H(z)$ can be transformed to another realization which in general performs differently under the FWL effects. The new realization matrices $\{\boldsymbol{A}_T, \boldsymbol{B}_T, \boldsymbol{C}_T, d\}$, covariance matrix $\boldsymbol{K}_T$, and the

noise matrix W_T satisfy

$$A_T = T^{-1}A_0T \qquad B_T = T^{-1}B_0 \qquad C_T = C_0T$$

$$K_T = \left(T^{-1}\right)K_0\left(T^{-1}\right)^t \qquad W_T = T^tW_0T \tag{1.15}$$

The problem of finding the state-space structure that minimizes the output roundoff noise under the l_2-scaling has been solved by Mullis and Roberts [28] and Hwang [29]. This structure will be called the MRH structure for the sake of convenience. The MRH structure provides the minimum output roundoff noise, minimum coefficient sensitivity [31] and it is free of zero-input limit cycle oscillations [32], [33]. However, all these merits have been offset by the large number of multiplication and addition operations since $(N+1)^2$ multiplications and $N(N+1)$ additions are required to compute each output sample. One approach to reduce the extensive computations required for Nth-order state-space filter is to implement the filter as a cascaded or parallel combination of first- and second-order minimum-noise sections [34], [35]. Although this solution significantly reduces the computations, the computation for each second-order section is still higher compared to second-order direct form section. Therefore, it is required to reduce the computational complexity of each second order-section. This can be achieved by searching for a second-order realization in which some of the coefficients are zero or power-of-two [36-41]. This is usually achieved at the price of increasing the output roundoff noise. Continuing along this direction, a new second-order realization is proposed in this thesis which has excellent FWL performance with the same number of nontrivial multipliers as a second-order direct form section.

Although the realization of second-order low-complexity state-space sections reduces the computational complexity to an acceptable level, the resulting structure loses some of the desirable features such as minimum output roundoff noise and freedom of zero-input limit cycles. Furthermore, the problem of finding the optimal zero-pole pair-

ing and section ordering is generally difficult to investigate especially for high-order filters. Therefore there is still interest in the realization of the state-space filters as one section. In this thesis, the realization and the implementation of one-section digital filters will be emphasized and investigated.

1.3.3 Residue-Feedback Technique

A general method that has been used to reduce the effect of the error inherent in any quantization operation is error spectral shaping [25], [42] (also called residue-feedback (RF) [43]). It has been effectively used to reduce the output quantization noise and/or to eliminate limit cycles of IIR digital filters in both direct [25-27], [44], [45] and state-space forms [46], [47]. The RF technique is usually implemented by extracting the quantization errors (residues) due to product quantization. These residues are sometimes weighted before they are fed back for use in the next iteration. In general, weighting the residues requires extra multiplication operations [42], [44] which leads to increased computational complexity. Other RF schemes have been proposed in which the residue coefficients are restricted to be either integer or power-of-two [48-50]. One solution which eliminates the need for RF multiplications or shift operations is reported in [51] in which the residue coefficients are restricted to ± 1.

1.4 Filter Implementation

Given a specific IIR digital filter realization, the implementation can assume two forms, namely software and hardware. This classification is somewhat artificial, however, since software and hardware are nowadays highly interchangeable. For nonreal-time applications, speed is not of considerable importance and the implementation might assume the form of a computer program running on a general-purpose computer. General-purpose computers can not offer satisfactory speed for real-time applications due to severe sys-

tem overhead.

Digital signal processors (DSP's) are special-purpose chips designed to efficiently implement most of the digital signal processing algorithms. These DSP chips offer a good tradeoff between hardware speed and cost for real-time applications. Therefore, they offer satisfactory solutions for many applications.

However, for many real-time applications such as real-time digital image and video filtering, speed is of essence and dedicated, highly specialized parallel hardware is the only viable solution. There are many approaches for designing special-purpose hardware (e.g., array-processors, distributed arithmetic, etc.). Very large scale integration (VLSI) array-processor implementation with local communications offers a promising solution for these high-speed applications.

1.5 VLSI Array-Processor Implementation

In many real-time DSP applications, speed is of prime importance. For such applications, parallel processing capabilities in terms of speed and data volume are essential. The availability of low-cost, high-density, high-speed VLSI devices, and the emergence of computer-aided design facilities, presages a major breakthrough in the design of massively parallel processors.

A possibility for the real-time implementation of digital filtering is to use special purpose array-processors, and to maximize the processing concurrency by either pipeline processing, parallel processing or both [52]. A systolic system consists of a set of locally and regularly interconnected processors, each capable of performing some simple operation. Information in a systolic system flows between cells in a pipelined fashion, and communication with the outside world occurs only at the boundary cells [53]. A systolic array is very amenable to VLSI implementation by taking advantage of its regular, and localized data flow [52-56]. It is especially suitable to a special class of compute-bound

algorithms in which the total number of operations is larger than the total number of input and output data elements. A systolic array often represents a direct mapping of computations onto processor arrays. Consequently, the systolic array features the important properties of modularity, local interconnection, as well as a high degree of pipelining and highly synchronized multiprocessing.

Several techniques for mapping algorithms onto processor arrays have been discussed in the literature [52], [55-57]. Kung [52] presented the signal flow graph (SFG) approach which is derived from the dependence graph (DG). The SFG can be mapped directly onto a systolic array by mapping nodes onto processor elements (PEs) and edges onto interconnections. Timing and data movement are derived from a linear timing function applied to the DG nodes.

The VLSI array-processor architecture is suitable for algorithms which mainly contain many parallel computation operations. Block-state space description of IIR digital filters [58], [59] is a highly parallel algorithm where the input data samples are processed in blocks. The speed of the implementation based on the block-state space description of IIR can be increased by increasing the input block length at the expense of increasing the hardware complexity.

The basic elementary operations in state-space IIR filter is a series of multiply-add operations. These operations can be transformed to inner-product operations by mapping the IIR algorithm such that calculations required for each output sample are locally executed. The conventional way to obtain an inner-product is to use a multiplier followed by an accumulator where the result of each multiplication is added to the previous result stored in the accumulator.

1.6 Organization of the Dissertation

The dissertation comprises eight chapters. In this chapter, a brief introduction to the prob-

lems considered in this thesis is presented.

In Chapter 2, new realizations of fixed-point second-order IIR digital filter sections are presented. These realizations have coefficients which are power-of-two or sum of power of two and lower output roundoff noise than many well-known low-roundoff realizations, freedom of limit cycles, and the same number of nontrivial multipliers as the direct structure. Numerical comparisons between the proposed realizations and other low-noise second-order realizations are also presented.

In Chapter 3, the realization of one-section IIR digital filters using residue-feedback is considered. Specifically, the problem of synthesizing fixed-point realizations of Nth-order IIR digital filters, which use diagonal residue feedback matrices to minimize the output roundoff noise subject to l_2-scaling, is considered. Optimal and suboptimal (in terms of output roundoff noise) Nth-order realizations are developed for low-pass, high-pass, band-stop, and band-pass filters using simple residue-feedback schemes. The proposed suboptimal residue feedback structures for narrow-band IIR filters provide near-optimal roundoff noise and have a saving of at least $N(N-2)/2$ multiplications over the optimal structures. Moreover, these suboptimal structures can be chosen to have block-triangular state-update matrices which are more suitable for high-speed hardware implementations. Extensive numerical comparisons between the proposed suboptimal structures and three other low-noise structures show that the proposed suboptimal structures provide excellent performance in terms of output roundoff noise and coefficient sensitivity as well as low computational complexity.

In Chapter 4, two implementations for one of the suboptimal realizations proposed in Chapter 3 are presented. The first is implemented on a a general DSP (Motorola DSP56001). The cost and FWL performance are investigated and analyzed. Experimental and theoretical results are compared and discussed. It is shown that this DSP implementation provides low output roundoff noise, freedom from limit cycles for the

examples considered and speed suitable for most audio applications. The second proposed implementation is a fully-pipelined VLSI array-processor architecture which can be used for high-speed applications. This implementation provides better performance in terms of hardware area and speed compared to existing direct implementation for narrow-band IIR digital filters. This high-speed VLSI implementation obtained by taking advantage of both the parallelism inherent in the residue feedback technique and the special structure of the proposed realization in Chapter 3.

In Chapter 5, a new fixed-point two's complement inner-product processor is developed. The proposed inner-product processor has a high computational speed which is double (for long inner-product operation) the speed of the conventional pipelined inner-product processors. A comparison (in terms of hardware speed and area) between the proposed processor and two conventional inner-product processors is also included.

In Chapter 6, a systolic implementation of a full-matrix state-space IIR digital filter is presented. Judicious mapping choices combined with the features of the new inner-product processor proposed in Chapter 5 are used to obtain this efficient systolic implementation. The new architecture is amenable to VLSI implementation because the number of the required processor elements linearly increases with the filter order. It will be also shown that the proposed implementation provide in many cases better performance in terms of $area \times time$ and $area \times time^2$ compared to existing direct implementation for narrow-band filters.

In Chapter 7, two new array processor implementations are proposed for IIR digital filters with high input sampling rates which are not limited by the speed of the processing elements involved as in the case for the implementation in Chapter 4 and Chapter 6. The first implementation is based on block-state description in which the state-update matrix is full corresponding to implementing the corresponding filter as one section. The other implementation is also based on the block-state description in which the state-

update matrix is block-diagonal corresponding to the case of parallel combination of second-order sections. The proposed array processor architectures are amenable to VLSI implementation because they require a significantly reduced number of processor elements. Performance comparisons (in terms of hardware complexity and speed) of the proposed implementations with other existing implementations are also presented.

In Chapter 8, a summary of the results presented in this thesis is given and suggestions for future work are presented.

Chapter 2

New Realizations of Second-Order IIR Filter Sections

2.1 Introduction

An important task in the implementation of a recursive digital filter is the selection of a realization structure that yields acceptably low roundoff noise at the filter output. State-space description of these filter structures is useful because the noise gain can be minimized by the application of a linear transformation to the state equations.

Recognizing the high number of multiplications required for the implementation of MRH structures proposed in [28], [29] researchers have proposed the realization of the digital filters as parallel or cascade of first- and second-order subfilters where each subfilter employs a minimum-noise structure. Although such designs do not offer the minimum roundoff noise, they represent a good compromise between output noise and computational efficiency [34], [35]. In order to further improve the computational efficiency, a class of second-order digital filter realizations that provides near-minimum roundoff noise performance with significant reduction in the nontrivial multiplies has been presented [36-41]. These structures have some zero coefficients and some power-of-two coefficients and thus the multiplication operations can be eliminated or replaced by shifting operations. They provide an attractive compromise between direct form struc-

tures and minimum-noise structures. Some of these structures were proposed in [40], [41] by Bomar and provide low roundoff noise realizations which are free of zero-input overflow limit cycle oscillations and require the same number of nontrivial multiplies as the direct form structures.

In this chapter, new realizations for second-order sections are presented. These structures have been obtained by extending the technique proposed by Bomar in [40], [41] to allow some coefficients to be sum-of-two power-of-two terms instead of being power-of-two terms only. The proposed structures provide significantly lower output roundoff noise than many other well known structures including the ones in [40], [41] with the same number of nontrivial multiplications.

2.2 Preliminaries

Consider the second-order transfer function $H(z)$ with complex conjugate poles of the form

$$H(z) = \frac{\alpha_1 z + \alpha_2}{z^2 + \beta_1 z + \beta_2} \tag{2.1}$$

$$= \frac{\alpha}{z-\lambda} + \frac{\alpha^*}{z-\lambda^*} \tag{2.2}$$

where λ and λ^* are the complex poles and α_1, α_2, β_1 and β_2 are real parameters. This transfer function can be realized by the following state-space equations

$$\boldsymbol{x}(n+1) = \boldsymbol{A}\boldsymbol{x}(n) + \boldsymbol{B}u(n) \tag{2.3}$$

$$y(n) = \boldsymbol{C}\boldsymbol{x}(n) \tag{2.4}$$

where the state vector $\boldsymbol{x}(n) \in \Re^2$, $y(n)$ and $u(n)$ are scalars, and the matrices $\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}$ are given by

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad \boldsymbol{B} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \quad \boldsymbol{C}^t = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \tag{2.5}$$

The transfer function representing (2.3) and (2.4) can be obtained as:

$$H(z) = \boldsymbol{C}(z\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{B} \tag{2.6}$$

For equality of the transfer functions in (2.1) and (2.6), the elements of ***(A,B,C)*** must satisfy four equations that are given in reference [40]:

$$\begin{aligned} c_1 b_1 + c_2 b_2 &= \alpha_1 \\ c_1 b_2 a_{12} + c_2 b_1 a_{21} - c_1 b_1 a_{22} - c_2 b_2 a_{11} &= \alpha_2 \\ -(a_{11} + a_{22}) &= \beta_1 \\ a_{11} a_{22} - a_{12} a_{21} &= \beta_2 \end{aligned} \tag{2.7}$$

The l_2-scaling condition of (1.11) for a second-order section can be written as

$$k_{ii} = 1 \qquad i = 1, 2 \tag{2.8}$$

where k_{ii} is defined in (1.12)

The scaled realization $(\boldsymbol{A}_T, \boldsymbol{B}_T, \boldsymbol{C}_T)$ can always be obtained from the unscaled one by applying the diagonal scaling transformation [16]

$$\boldsymbol{T} = \text{diag}\{\sqrt{k_{11}}, \sqrt{k_{22}}\} \tag{2.9}$$

The structures are guaranteed to be free of zero-input overflow limit cycles[1] if [60]

$$|a_{11} - a_{22}| + |\lambda|^2 \le 1 \tag{2.10}$$

By adopting the noise model in Fig. 1.1, the filter output noise variance for the second-

1. They are also free of zero-input quantization limit cycles if signed-magnitude is used for quantization process.

order section can be obtained as

$$\sigma_y^2 = (1 + w_{11} + w_{22})\,\sigma_r^2 = (1 + \text{trace}(\boldsymbol{W}))\,\sigma_r^2 = \left(1 + g^2\right)\sigma_r^2 \tag{2.11}$$

where w_{ii} is the ith diagonal element of the noise matrix $\boldsymbol{W}$ defined in (1.14). In (2.11), it is assumed that the rounding operation is performed after the summation at each node. The value of trace($\boldsymbol{W}$), g^2, is often referred to as roundoff noise gain (NG) of the structure.

2.3 Filter Structures Proposed By Bomar

In [40], [41] a technique was proposed to obtain structures comprising second-order filter sections based on sequential selection of three parameters which are invariant under scaling (diagonal) transformations. These design parameters are related to the entries of the state-space matrices and can be defined as [40]

$$\mu = a_{22} \tag{2.12}$$

$$\rho = b_1 c_1 \tag{2.13}$$

$$\gamma = a_{22} b_1 c_2 \tag{2.14}$$

Given values for two of the design parameters, the third parameter can be determined from the following equation [40]

$$\gamma = \frac{\rho(\alpha_1 - \rho)\left(\mu^2 + \beta_1\mu + \beta_2\right)}{\gamma + (\alpha_1 - \rho)(\mu + \beta_1) - \rho\mu - \alpha_2} \tag{2.15}$$

in (2.15) $\gamma \neq 0$. The optimal choice for μ, ρ and γ, which minimizes the output roundoff noise subject to l_2-scaling has been presented in [40]. In order to obtain computational efficient realizations with low roundoff noise, the design parameters should be chosen near their optimal values while at the same time provide four realization coefficients which have zero or power-of-two.

The technique proposed in [41] to obtain these realizations can be summarized in the following steps:

1. Choose values for two of the design parameters that make two of the state-space coefficients in (2.3) zeros or power-of-two terms. The third design parameter can then be determined from (2.15).
2. Define the unscaled state-space coefficients in terms of the three design parameters.
3. Obtain an l_2-scaled structure by applying the diagonal transformation in (2.9).
4. Rescale the scaled structure by using a diagonal scaling matrix making two other coefficients of the state-space matrices equal to a power-of-two [39]. This rescaling reduces the number of the nontrivial multiplications in the implementation and makes the l_2-scaling more conservative i.e., reduces the probability of the overflow at the cost of slightly increasing the output roundoff noise.

Based on this technique, three different structure classes (class 1, 2 and 3) have been proposed [40], [41]. All of them have four nontrivial multiplies. *Class* 1 has two zero multiplies and two power-of-two multiplies. *Class* 2 has one zero multiply and three power-of-two multiplies. *Class* 3 has four power-of-two multiplies. The design parameter values leading to *class* 1 and *class* 2 are listed at Tables II, III in [40]. For *class* 3, ρ and γ should be chosen so that ρ and γ take power-of-two values, μ is real and the structure is free of limit-cycle oscillations. Class 1 and class 2 have more coefficients equal to zero, but it is not always possible to find a structure for a given *H(z)*. On the other hand, class 3 realization do virtually always exists. The appropriate power-of-two values for ρ and γ are determined through a search process described in [41]. The three classes are guar-

anteed to be free of zero-input overflow limit-cycle oscillations (i.e., they satisfy (2.10)).

2.4 The New Proposed Technique

New low-roundoff output noise structures can be obtained by extending the technique of [40], [41]. Two different methods will be presented which result from two different modifications of the technique summarized in the previous section.

Method 1:

This method consists of four steps. The first three steps are identical to the steps in the previous section. The rescaling technique used in step 4 is modified to yield two coefficients of the state-space matrices having sum-of-two power-of-two terms instead of power-of-two terms. A simple algorithm has been developed to find the optimal values for two coefficients with respect to output roundoff noise under the constraint that these two coefficients are sum-of-two power-of-two. All possible pairs of two coefficients which need to be considered can be found in Tables II, III in [40]. A pseudo code version of this algorithm is:

NG_{opt} = LARGEST MACHINE NUMBER
FOR I = 0 TO IMAX
BEGIN
FOR II = 0 TO IIMAX
BEGIN
FOR J = 0 TO JMAX
BEGIN
FOR IJ = 0 TO JJMAX
BEGIN
CHOOSE t_{11}, t_{22} to transform
(|COEF. 1|,|COEF. 2|) to $(2^{-I} + 2^{-II}, 2^{-J} + 2^{-JJ})$
$NG = t_{11}^2 w_{11} + t_{22}^2 w_{22}$
IF $NG < NG_{opt}$ and $t_{11} \geq 1$ AND $t_{22} \geq 1$

THEN

$NG_{opt} = NG$

$\boldsymbol{T} = diag(t_{11}, t_{22})$

END IF

END FOR

END FOR

END FOR

END FOR

PRINT NG_{opt}, $\boldsymbol{T}$

This algorithm will give a lower roundoff noise than *class* 1 and *class* 2 structures proposed in [40], [41]. The reason for the output roundoff noise improvement is that the values of the entries of the diagonal rescaling matrix (i.e., t_{11} and t_{22}) in the proposed technique will be lower than their corresponding values obtained by using the technique in [39]. The new *class* 1 structure has two zero multiplies and two sum-of-two power-of-two multiplies. The new *class* 2 has one zero multiply, one power-of-two multiply and two sum-of-two power-of-two multiplies.

Method 2:

The first step of this method is a modified step 1 of the technique of [40], [41] summarized in the previous section. The restriction that one or two of the design parameters be sum-of-two power-of-two values is imposed. This leads to a lower output roundoff noise than the technique of [40], [41] due to the fact that the design parameters (γ, ρ, μ) in that case can be chosen more closer to their optimal values. The last three steps in that case will remain the same as in the technique of [40], [41]. This modification can be applied to the second and the third classes. For the new *class* 2, the same values in Table III in [40] can be used after replacing every $\mathrm{npt}(x)$ by $\mathrm{nspt}(x)$ where $\mathrm{npt}(x)$ denotes the nearest power-of-two value to x and $\mathrm{nspt}(x)$ denotes the nearest sum-of-two power-of-two value to x. Also each $(x)^H$ will be interpreted as the nearest sum-of-two power-of-two value greater than or equal to $|x|$. Through this modification, the new *class* 2 has one

zero multiply, one sum-of-two power-of-two multiply and two power-of-two multiplies while the new *class* 3 has two sum-of-two power-of-two terms multiplies and two power-of-two multiplies.

In all cases (method 1 and method 2), the new structures have four nontrivial multiplies and four trivial multiplies and are guaranteed to be free of zero-input limit-cycle oscillations. A combination of the two previous modifications can be applied to *class* 2 and *class* 3 to gain more improvement in the output roundoff noise at the expense of increasing the algorithm complexity. The use of difference of two power-of-two terms or the sum (or difference) of several power-of-two terms may improve the output roundoff noise at the price of increasing the algorithm complexity.

2.5 Numerical Examples

Example 2.1:

The proposed structures will be applied to an example that was presented in [41] and the result will be compared with the result reported in [41]. The second-order section considered is

$$H(z) = \frac{10^{-3}(0.87715z + 2.40610)}{z^2 - 1.95556z + 0.96249}$$

The only suitable realization for this example is the *class* 3. It has been shown in [40] that ρ and γ should satisfy the following conditions in order to make the output structure free of overflow oscillations:

$$-0.02519 \le \rho \le 0.02607$$

$$-4.0047\times10^{-7} \le \gamma \le 3.44226\times10^{-3}$$

In order to apply the new technique (method 2), the range of sum-of-two power-of-two

values for ρ to be considered is

$$\rho = \{-\left(2^{-6}+2^{-7}\right), -\left(2^{-6}+2^{-8}\right), \ldots, -2^{-6}, \ldots, \left(2^{-6}+2^{-8}\right), \left(2^{-6}+2^{-7}\right)\}$$

The value of ρ equal to $-\left(2^{-6}+2^{-9}\right)$ or $\left(2^{-6}+2^{-9}\right)$ leads to the lowest roundoff noise gain. Using the first value, we can show by using (2.15) that γ should satisfy either of the following conditions

$$\gamma \leq -5.50521$$

$$\gamma \geq 6.3967\times10^{-4}$$

for μ to be real. The overlap of the requirements on γ is then

$$6.3967\times10^{-4} \leq \gamma \leq 3.44226\times10^{-3}$$

Within this range the choices for γ to be considered are:

$$\gamma = \{2^{-11}, \left(2^{-11}+2^{-10}\right), 2^{-9}, \left(2^{-9}+2^{-11}\right), \left(2^{-9}+2^{-10}\right)\}$$

By taking $\gamma = 2^{-9}+2^{-11}$ and by following steps 2,3 and 4 of the original technique listed in Section 2.3, we can get the following realization

$$\boldsymbol{A} = \begin{bmatrix} 0.9789573 & -0.102994 \\ 2^{-4} & 0.976864 \end{bmatrix} \quad \boldsymbol{B} = \begin{bmatrix} 2^{-2} \\ 0.118114 \end{bmatrix}$$

$$\boldsymbol{C}^T = \begin{bmatrix} -\left(2^{-4}+2^{-7}\right) \\ 2^{-3}+2^{-5} \end{bmatrix} \qquad g^2 = 0.727$$

Comparing the noise gain value of the proposed structure with that of the structure in Example 1 in [41] ($g^2 = 1.14$), the improvement in the output noise gain is about 36% at the expense of extra two additions and two shift operations (see Table 2.1).

Example 2.2:

In order to compare the proposed structure with other well known structures, an example treated by [36] will be considered. An eighth-order Chebyshev low-pass filter with a passband ripple of 0.17 dB, and passband edge at 0.02. The poles and residues of the four second-order sections for a parallel realization of this filter are given by [36]

Section 1

$$\lambda = 0.98468 + 0.12716i$$
$$\alpha = -0.00800 - 0.00149i$$

Section 2

$$\lambda = 0.97397 + 0.10651i$$
$$\alpha = 0.02375 - 0.0071i$$

Section 3

$$\lambda = 0.96727 + 0.07057i$$
$$\alpha = -0.02897 + 0.03047i$$

Section 4

$$\lambda = 0.96416 + 0.02468i$$
$$\alpha = 0.01231 - 0.04957i$$

The number of multiplications, addition and shift operations required for realizing a second-order section using different seven structures are listed in Table 2.1.

Table 2.1. Arithmetic Operations Required For Each Second-Order Structure.

Structure	Multiplication No.	Addition No.	Shift-operation No.
Scaled Direct	5	4	-
Normal [34]	9	6	-
Minimum-noise [34]	9	6	-
Barnes [36]	6	5	2
Bomar [39]	6	5	2
Bomar [41]	5	4-6	2-4
Proposed Struct.	5	6-8	4-6

By using the software tool presented in [61], the output noise gains of the seven different structures for the four sections and the total noise gains for the parallel connection

of all sections are listed in Table 2.2. From Table 2.2, it is clear that the proposed structure provides better roundoff noise gain than that of many other well known structures with the same number of nontrivial multiplies as the direct structure. The only price paid for that roundoff improvement is two extra additions and two extra shift operations over what required by the structure proposed in [41].

Table 2.2. Noise Gains for Example 2.2

Structure	Scaled Direct	Scaled Normal [34]	Minimum-noise [34]	Barnes [36]
Section 1	039.59	0.653	0.651	1.170
Section 2	072.25	0.767	0.748	3.466
Section 3	205.20	0.973	0.962	1.715
Section 4	325.70	1.071	0.749	0.992
Overall	642.74	3.464	3.110	7.343

Structure	Bomar [39]	Bomar [41]	Proposed Structure
Section 1	0.919	1.084	0.672
Section 2	1.003	1.770	0.971
Section 3	1.715	1.987	0.977
Section 4	0.992	1.269	1.050
Overall	4.629	6.110	3.670

2.5 Conclusion

New structures for IIR filters as combination of cascade or parallel of second-order sections have been proposed. They yield lower output roundoff noise than many well-known low-roundoff structures. Further, they have the same number of nontrivial multiplies as the direct structure and are guaranteed to be free of zero-input overflow oscillations.

Chapter 3

New Residue-Feedback IIR Digital Filter Realizations

3.1 Introduction

The residue feedback (RF) technique has been efficiently used to reduce the output quantization noise and/or to eliminate limit cycles of IIR digital filters in both direct forms [25-27], [43-45] and state-space forms [46], [47]. The RF technique is implemented by extracting the quantization error after product quantization and feeding the error signal back through a feedback filter. The idea of RF technique is to place zeros in the passband of the transfer function from the quantization sources to the filter output. It should be emphasized that the RF technique affects only the transfer function of the quantization error signal, while the transfer function of the filter itself remains unchanged. RF schemes can be divided into two categories according to how the coefficients of residue feedback scheme are related to the filter coefficients:

Variable feedback schemes: The IIR filter structure determines the order and coefficients of the residue feedback filter. These RF schemes have been applied to second-order direct sections [44], [49], [62], high-order direct sections [27], [63] and state-space forms [46], [64]. It has been pointed out in [27] that for most applications, it is sufficient to have the order of the residue feedback filter less than or equal to the IIR filter order. The

coefficients of the feedback scheme are more or less related to the coefficients of the denominator (or the system matrix) of the IIR filter. If the coefficients of the feedback filter are chosen to be the same as the coefficients of the denominator (or the system matrix) of IIR filter, the resulting RF structures correspond to double-precision arithmetic [65]. Alternatively, suboptimal (in terms of output roundoff noise) RF structures were suggested. In such suboptimal structures, the coefficients of the residue feedback filter assume integer values [25], [48], [66] or power-of-two values [49] or to have desirable properties such as symmetry [27]. However, all these suboptimal RF structures trade the reduction in computational complexity for increase in output roundoff noise.

Fixed feedback schemes: These RF schemes use a simple feedback filter (usually first or second-order FIR filter). The coefficients of the feedback filter are not related to the IIR filter coefficients. Fixed RF schemes have been applied to second-order direct forms [26] and state-space forms [46], [47], [51], [67]. The coefficients of the residue feedback filter are usually restricted to take values of ± 1 which results in achieving less computational complexity compared to the variable RF schemes. These fixed RF schemes are well suited to narrow-band low-pass filters (LPF) and high-pass filters (HPF) since they allow the filter designer to place zeros in the error transfer function at points $z = \pm 1$ using simple first-order feedback schemes.

By applying first-order feedback filter to Nth-order state-space IIR digital filters, Williamson in [51] has obtained optimal[2] RF structures which may provide lower output roundoff noise than that of the MRH structures in [28], [29] (which don't use any RF technique) at the price of only N extra additions (or subtractions). However, these optimal RF

2. We use the term "optimal" to refer the state-space realization that provides the minimum output roundoff under l_2-scaling constraint using a fixed feedback scheme. By using this terminology, the structures in [28], [29] may be called optimal structures with zero feedback scheme.

structures cover only the cases of LPF and HPF and the resulting RF optimal structures have full state-space matrices which require many arithmetic operations in the implementation.

In this chapter, the use of residue feedback for implementations of filters with arbitrary specifications is considered. Residue feedback of second-order were used to deal with band-pass filters (BPF) and band-stop filters (BSF) in addition to low-pass filters (LPF) and high-pass filters (HPF). The method presented in [51] has been extended to obtain optimal structures for such types of filters. A disadvantage of these optimal structures is the fact that the resulting system matrix has in general all elements not equal to zero and thus requires a large number of arithmetic operations for implementations. To overcome this drawback new suboptimal structures have been proposed which provide near-optimal output roundoff noise with a saving of at least $N(N-2)/2$ multiplies over the optimal ones. Moreover, the update state-space matrices of these suboptimal RF structures can be chosen to take the form of block-triangular which is more suitable for high-speed array-processor implementations.

The chapter is organized as follows: In Section 3.2 the synthesis problem is formulated and the necessary background material is presented. In Section 3.3 the optimal residue feedback structures for filters with various frequency specifications are presented. In Section 3.4 techniques to obtain the suboptimal structures which require less arithmetic operations for implementation are presented. In Section 3.5, the proposed RF structures and other low-noise structures are compared using extensive numerical examples. The comparison is carried out in terms of output roundoff noise, coefficient sensitivity, and computational complexity and demonstrates the usefulness of the proposed structures.

3.2 Problem Formulation

The infinite-precision state-space description of Nth-order IIR digital filter in (1.5) and (1.6) are recalled here for convenience:

$$\boldsymbol{x}(n+1) = \boldsymbol{A}\boldsymbol{x}(n) + \boldsymbol{B}u(n) \tag{3.1}$$

$$y(n) = \boldsymbol{C}\boldsymbol{x}(n) + du(n) \tag{3.2}$$

The matrices $\{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},d\}$ are of dimension $N\times N$, $N\times 1$, $1\times N$ and 1×1, respectively. FWL implementation of (3.1) and (3.2) with second-order RF can be described by the following equations

$$\tilde{\boldsymbol{x}}(n+1) = \boldsymbol{A}Q[\tilde{\boldsymbol{x}}(n)] + \boldsymbol{A}_1\boldsymbol{e}(n) + \boldsymbol{A}_2\boldsymbol{e}(n-1) + \boldsymbol{B}u(n) \tag{3.3}$$

$$\tilde{y}(n) = \boldsymbol{C}Q[\tilde{\boldsymbol{x}}(n)] + du(n) \tag{3.4}$$

where $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ are the residue feedback matrices and $\boldsymbol{e}(n)$ is the roundoff residue vector. The quantizer $Q[\ \cdot\ ]$ rounds the states $\tilde{\boldsymbol{x}}(n)$ to $\hat{\boldsymbol{x}}(n)$ of b bits after the multiplications and additions are completed. i.e., $\hat{\boldsymbol{x}}(n) = Q[\tilde{\boldsymbol{x}}(n)]$.[3] The diagram of the IIR structure with the second-order RF described by (3.3) and (3.4) is shown in Fig. 3.1. The coefficients of $\{\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, d, \boldsymbol{A}_1, \boldsymbol{A}_2\}$ are assumed to be exactly represented.

In (3.3) and (3.4), residue feedforward scheme has not been used since it has been pointed out in [27] that this residue feedforward is not effective for most cases. Fixed-point arithmetic is implemented using a two's complement representation so that

$$\hat{\boldsymbol{x}}(n) = Q[\tilde{\boldsymbol{x}}(n)] = \tilde{\boldsymbol{x}}(n) - \boldsymbol{e}(n) \tag{3.5}$$

The roundoff residue vector $\boldsymbol{e}(n)$ is modeled as a zero-mean white noise process with covariance

3. In (3.3) and (3.4), the effect of secondary residue is neglected.

$$E\{\boldsymbol{e}(n)\boldsymbol{e}^{t}(n)\} = \sigma^{2}_{r}\boldsymbol{I} \tag{3.6}$$

where $\boldsymbol{I}$ is the $N \times N$ identity matrix.

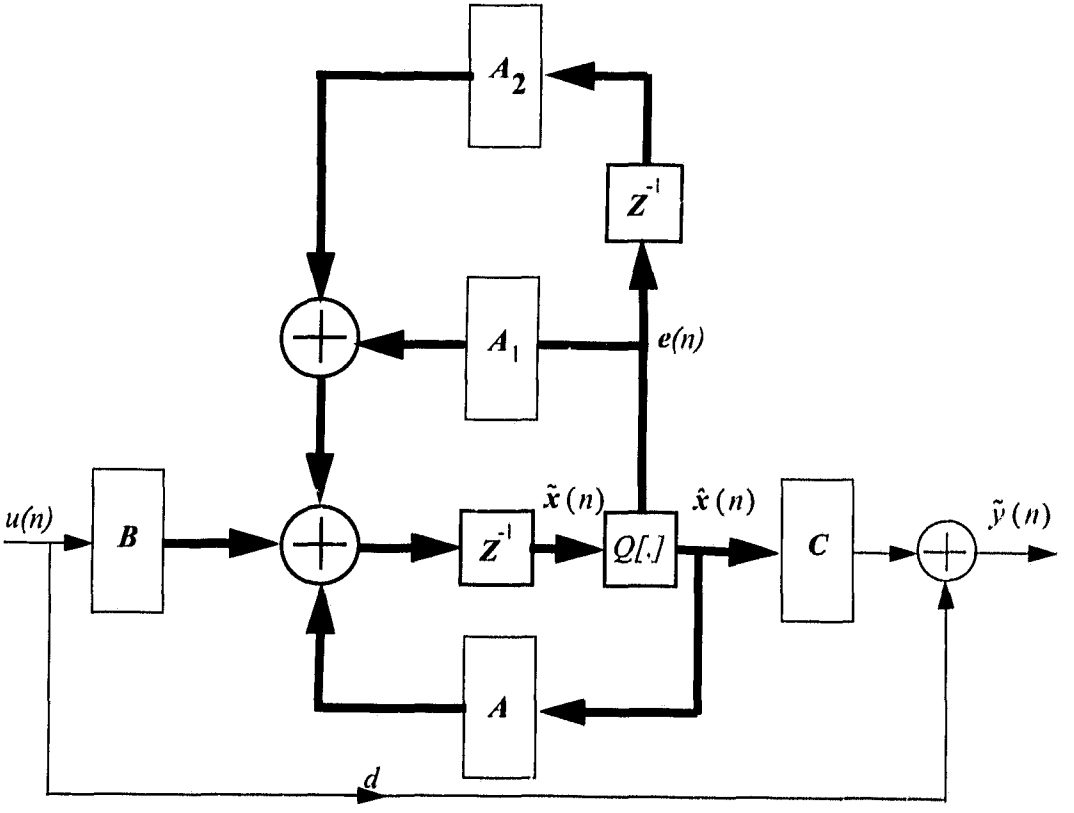


Figure 3.1: The block diagram of Nth order state-space filter with second-order RF scheme.

The samples of the residue vector $\boldsymbol{e}(n)$ are assumed to be uncorrelated with each other and uncorrelated with the input sequence $u(n)$. Substituting (3.5) in (3.3) and (3.4) leads to

$$\tilde{\boldsymbol{x}}(n+1) = \boldsymbol{A}\tilde{\boldsymbol{x}}(n) + (\boldsymbol{A}_1 - \boldsymbol{A})\boldsymbol{e}(n) + \boldsymbol{A}_2\boldsymbol{e}(n-1) + \boldsymbol{B}u(n) \tag{3.7}$$

$$\tilde{y}(n) = \boldsymbol{C}\tilde{\boldsymbol{x}}(n) - \boldsymbol{C}\boldsymbol{e}(n) + du(n) \tag{3.8}$$

From (3.7) and (3.8) the output propagate error $y_e(n)$ due to quantization errors can be

calculated as

$$y_e(n) = \boldsymbol{C}\sum_{i=0}^{\infty}\boldsymbol{A}^i(\boldsymbol{A}_1-\boldsymbol{A})\,\boldsymbol{e}(n-i-1) + \boldsymbol{C}\sum_{i=0}^{\infty}\boldsymbol{A}^i\boldsymbol{A}_2\boldsymbol{e}(n-i-2) - \boldsymbol{C}\boldsymbol{e}(n) \qquad (3.9)$$

Therefore, the corresponding output noise variance σ^2 can be obtained as

$$\sigma^2 = E\{y_e(n)y_e^t(n)\} = \sigma_r^2 g^2 \qquad (3.10)$$

g^2 can be derived as

$$g^2 = E\{y_e(n)y_e^t(n)\}/\sigma_r^2 = \text{trace}\left(\boldsymbol{S}_1\boldsymbol{W}+\boldsymbol{S}_2\boldsymbol{W}+\boldsymbol{S}_2^t\boldsymbol{W}+\boldsymbol{S}_3\boldsymbol{W}+\boldsymbol{C}^t\boldsymbol{C}\right) \qquad (3.11)$$

where

$$\boldsymbol{S}_1 = (\boldsymbol{A}_1-\boldsymbol{A})\,(\boldsymbol{A}_1-\boldsymbol{A})^t$$

$$\boldsymbol{S}_2 = \boldsymbol{A}_2(\boldsymbol{A}_1-\boldsymbol{A})^t\boldsymbol{A}^t$$

$$\boldsymbol{S}_3 = \boldsymbol{A}_2\boldsymbol{A}_2^t$$

For proof of (3.11), see Appendix A.

An l_2-scaling technique that equalizes overflow probability for all the states of FWL implementation in (3.3) and (3.4) is equivalent to imposing the constraint in (1.11) where k_{ii} is the ith diagonal element of the covariance matrix $\boldsymbol{K}$ which here is given by

$$\boldsymbol{K} = \boldsymbol{A}\boldsymbol{K}\boldsymbol{A}^t + \boldsymbol{B}\boldsymbol{B}^t + \sigma_r^2\left[(\boldsymbol{A}-\boldsymbol{A}_1)\,(\boldsymbol{A}-\boldsymbol{A}_1)^t + \boldsymbol{A}_2\boldsymbol{A}_2^t\right] \qquad (3.12)$$

For high input signal-to-quantization noise ratio (i.e., $\sigma_r^2 \ll 1$), the third term in RHS of (3.12) can be neglected and $\boldsymbol{K}$ can be calculated by (1.12).

Under any similarity transformation $\boldsymbol{T}$, any realization $\{\boldsymbol{A}_0, \boldsymbol{B}_0, \boldsymbol{C}_0, d\}$ can be

transformed to another realization $\{\boldsymbol{A}_T, \boldsymbol{B}_T, \boldsymbol{C}_T, d\}$ which performs differently under FWL effects as was shown in (1.15).

The residue feedback matrices $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ will be determined based on the type and the specifications of the IIR filter (i.e., they are independent of the state-space realization).

The problem to be considered can now be formulated as follows: starting from a realization with specified residue feedback matrices $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$, it is required to find a similarity transformation $\boldsymbol{T}$ which minimizes the output roundoff noise gain in (3.11) under the l_2-scaling constraint in (1.11). Two different groups of realizations will be considered. In the first group discussed in Section 3.3, no constraint is imposed on the system matrix and hence it will be, in general, a full matrix. In the second group discussed in Section 3.4, the system matrix is required to have special forms where some of the coefficients are zero. It will be shown that this type of constraint leads to not only computational efficient structures but also structures which are amenable to fast VLSI implementation with only a slight increase in the output roundoff noise gain.

3.3 Full State-Space Matrix Realizations

The choice of $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ depends on the filter characteristics and leads to different realizations of *H(z)*. In this section, realizations for low-pass filter (LPF), high-pass filter (HPF), band-stop filter (BSF) and band-pass filter (BPF) using residue feedback will be presented. All these realizations have in general full system matrices and can be obtained by minimizing the noise gain under l_2-scaling constraint. The cases for LPF and HPF have been discussed in [51] and are included here for completeness.

3.3.1 Optimal LPF And HPF Realizations Using RF

For the LPF, we could take $A_1 = I$ and $A_2 = 0$ (i.e., first-order RF scheme) which corresponds to placing zeros in the error transfer function at $z = 1$. For this case, (3.11) can be simplified to

$$g_0^2 = \text{trace}(P_0) \tag{3.13}$$

where P_0 is the residue matrix of any initial realization for LPF defined as

$$P_0 = (I-A_0)^t W_0 + W_0 (I-A_0) \tag{3.14}$$

Under any similarity transformation T, the residue matrix P_T will satisfy the following equation

$$P_T = T^t P_0 T \tag{3.15}$$

The noise gain g_T^2 after transformation can be obtained as

$$g_T^2 = \text{trace}\left(T^t P_0 T \right) \tag{3.16}$$

A similar scheme has been obtained for the HPF case taking $A_1 = -I$ and $A_2 = 0$ which leads to a residue matrix P_0 given by

$$P_0 = (I+A_0)^t W_0 + W_0 (I+A_0) \tag{3.17}$$

The positive definiteness of P_0 defined in (3.14) and (3.17) are shown in Appendix B, lemma 1. Both LPF and HPF cases have been solved by Williamson in [51] and RF optimal structures have been obtained where the output noise gain is a function of the eigenvalues of KP which are invariant under any similarity transformation. The square roots of these eigenvalues are called residue modes. It has been shown in [51] that those RF optimal structures may provide lower output roundoff noise compared to the MRH

structures if and only if the sum of the residue modes is lower than the sum of second-order modes [51]. An algorithm to find the similarity transformation $\boldsymbol{T}$ which leads to The RF optimal structures has been presented as theorem 5.2 in [51]. This algorithm has been applied only for the cases of LPF and HPF in [51]. In the next two subsections, the BSF and BPF cases will be considered and new RF structures will be obtained.

3.3.2 Optimal and suboptimal realizations for BSF using RF

For BSF, it is possible to place zeros in the noise spectra at $z = 1$ or at $z = -1$ or at both. This can be accomplished by either of the following schemes:

Scheme 1

Consider $\boldsymbol{A_1} = \boldsymbol{J_0} = \begin{bmatrix} \pm \boldsymbol{I}_{r \times r} & \boldsymbol{0} \\ \boldsymbol{0} & \mp \boldsymbol{I}_{(N-r) \times (N-r)} \end{bmatrix}$ and $\boldsymbol{A_2} = \boldsymbol{0}$. The output roundoff noise gain of any initial realization can be obtained as

$$g_0^2 = \text{trace}\left((\boldsymbol{I} - \boldsymbol{A_0 J_0})^t \boldsymbol{W_0} + \boldsymbol{W_0} (\boldsymbol{I} - \boldsymbol{A_0 J_0}) \right) \tag{3.18}$$

If the signs of both block-diagonal sub-matrices in $\boldsymbol{J_0}$ are the same (i.e., $\boldsymbol{J_0} = \pm \boldsymbol{I}$), the scheme will be equivalent to the schemes presented in Subsection 3.3.1 and the optimal structures can be easily obtained as discussed in Subsection 3.3.1. If the signs are not the same, one can proceed as follows: applying a similarity transformation $\boldsymbol{T}$, the noise gain g_T^2 will be

$$\begin{aligned} g_T^2 &= \text{trace}\left[\boldsymbol{T}^t \boldsymbol{W_0} \boldsymbol{T} \left(\boldsymbol{I} - \boldsymbol{T}^{-1} \boldsymbol{A_0} \boldsymbol{T} \boldsymbol{J_0} \right)^t + \left(\boldsymbol{I} - \boldsymbol{T}^{-1} \boldsymbol{A_0} \boldsymbol{T} \boldsymbol{J_0} \right) \boldsymbol{T}^t \boldsymbol{W_0} \boldsymbol{T} \right] \\ &= \text{trace}\left(\boldsymbol{T}^t \boldsymbol{W_0} \boldsymbol{T} - \boldsymbol{T}^t \boldsymbol{W_0} \boldsymbol{T} \boldsymbol{J_0} \boldsymbol{T}^t \boldsymbol{A_0}^t \boldsymbol{T}^{-t} + \boldsymbol{T}^t \boldsymbol{W_0} \boldsymbol{T} - \boldsymbol{T}^{-1} \boldsymbol{A_0} \boldsymbol{T} \boldsymbol{J_0} \boldsymbol{T}^t \boldsymbol{W_0} \boldsymbol{T} \right) \\ &= \text{trace}\left(\boldsymbol{T}^t \boldsymbol{W_0} \boldsymbol{T} - \boldsymbol{J_0} \boldsymbol{T}^t \boldsymbol{A_0}^t \boldsymbol{W_0} \boldsymbol{T} + \boldsymbol{T}^t \boldsymbol{W_0} \boldsymbol{T} - \boldsymbol{T}^t \boldsymbol{W_0} \boldsymbol{A_0} \boldsymbol{T} \boldsymbol{J_0} \right) \\ &= \text{trace}\left[\boldsymbol{T}^t \left(\boldsymbol{W_0} - \boldsymbol{T}^{-t} \boldsymbol{J_0} \boldsymbol{T}^t \boldsymbol{A_0}^t \boldsymbol{W_0} + \boldsymbol{W_0} - \boldsymbol{W_0} \boldsymbol{A_0} \boldsymbol{T} \boldsymbol{J_0} \boldsymbol{T}^{-1} \right) \boldsymbol{T} \right] \end{aligned}$$

$$= \text{trace}\left[T^t\left(W_0 - J_T^t A_0^t W_0 + W_0 - W_0 A_0 J_T\right)T\right]$$

where $J_T = TJ_0T^{-1}$.

i.e.,

$$g_T^2 = \text{trace}\left\{T^t\left[(I - A_0 J_T)^t W_0 + W_0 (I - A_0 J_T)\right]T\right\} \tag{3.19}$$

Assuming T has the following form:

$$T = \begin{bmatrix} T_{1_{r \times r}} & 0 \\ 0 & T_{2_{(N-r) \times (N-r)}} \end{bmatrix} \tag{3.20}$$

The output noise gain g_T^2 can be written as

$$g_T^2 = \text{trace}\left(T^t P_0 T\right) \tag{3.21}$$

where P_0 can be defined as

$$P_0 = (I - A_0 J_0)^t W_0 + W_0 (I - A_0 J_0) \tag{3.22}$$

P_0 defined in (3.22) is shown in Appendix B, lemma 2, to be positive definite if $\lambda_{max}\left(\frac{\tilde{A}J_1 + J_1^t \tilde{A}^t}{2}\right) < 1$. The output noise gain g_T^2 in (3.21) can be rewritten as

$$g_T^2 = \text{trace}\left(T_1^t P_{11} T_1\right) + \text{trace}\left(T_2^t P_{22} T_2\right) \tag{3.23}$$

where P_{11} is the upper left $r \times r$ submatrix of P_0 in (3.22) and P_{22} is the lower right $(N-r) \times (N-r)$ submatrix of P_0. For the particular choice of T in (3.20), the l_2-scaling condition in (1.11) can be divided into two separate constraints:

$$\left(\boldsymbol{T}_1^{-1}\boldsymbol{K}_{11}\boldsymbol{T}_1^{-t}\right)_{ii} = 1 \text{ for } 1 \le i \le r \tag{3.24}$$

$$\left(\boldsymbol{T}_2^{-1}\boldsymbol{K}_{22}\boldsymbol{T}_2^{-t}\right)_{ii} = 1 \text{ for } r < i \le N \tag{3.25}$$

where $\boldsymbol{K}_{11}$ is the upper left $r \times r$ submatrix of $\boldsymbol{K}_0$ and $\boldsymbol{K}_{22}$ is the lower right $(N-r) \times (N-r)$ submatrix of $\boldsymbol{K}_0$. From (3.23), (3.24) and (3.25), it can be seen that the synthesis problem has been divided into two separate subproblems. The first subproblem is to find $\boldsymbol{T}_1$ that minimizes the first term in the RHS of (3.23) under l_2-constraint in (3.24). The second subproblem is to find $\boldsymbol{T}_2$ that minimizes the second term in the RHS of (3.23) under l_2-constraint in (3.25). Each subproblem can easily be obtained using an algorithm similar to that reported in [51]. Therefore, $\boldsymbol{T}$ in (3.20) can be obtained and can be applied to the initial realization.

The resulting RF structure obtained by using this scheme is not optimal since the similarity transformation $\boldsymbol{T}$ is restricted to take the block-diagonal form in (3.20). Despite this restriction, a good choice of r (depending on the filter specifications) leads to suboptimal structures with low roundoff noise gain. Examples of those suboptimal structures will be presented in Section 3.5.

Scheme 2

In this scheme, we will take $\boldsymbol{A}_1 = \boldsymbol{0}$ and $\boldsymbol{A}_2 = \boldsymbol{I}$. This scheme corresponds to placing double zeros in the noise spectra at both $z = 1$ and $z = -1$ for each state. The output roundoff noise gain of any initial realization can be obtained as

$$g_0^2 = \text{trace}\,(\boldsymbol{P}_0) \tag{3.26}$$

where $\boldsymbol{P}_0$ for this scheme is given by

$$\boldsymbol{P}_0 = \left[2\boldsymbol{W}_0 - \boldsymbol{W}_0\boldsymbol{A}_0^2 - \left(\boldsymbol{A}_0^2\right)^t\boldsymbol{W}_0\right] \tag{3.27}$$

$\boldsymbol{P}_0$ defined in (3.27) is shown in Appendix B, lemma 3, to be positive definite. The noise

gain g_T^2 will satisfy (3.16) where $\boldsymbol{P}_0$ is defined in (3.27). This optimal RF structures for this scheme can be easily obtained by applying the algorithm mentioned in [51] by using the definition of $\boldsymbol{P}_0$ in (3.27). The resulting structures will be optimal since no restriction has been applied on $\boldsymbol{T}$.

It should be noted that the RF schemes proposed in Subsections 3.3.1 and 3.3.2 for LPF, HPF, and BSF require only N additions to implement the RF technique (no additional multiplications are required).

3.3.3 Optimal Realization For BPF Using RF

For the BPF, it is required to place complex zeros at $z = e^{\pm j\omega_\kappa}$, $\kappa = 1, \ldots, N$ in the passband of the noise spectra where ω_κ, $\kappa = 1, \ldots, N$ are frequencies at the passband. This can be achieved by taking $\boldsymbol{A}_1 = \boldsymbol{U} = \text{diag}\{\gamma_\kappa\}$, $\boldsymbol{A}_2 = -\boldsymbol{I}$ where $\gamma_\kappa = 2\cos\omega_\kappa$ [67]. In general γ_κ assumes real values and therefore this scheme requires N multiplications plus $2N$ additions for implementing it. If ω_κ could be chosen to be one of the frequencies $\pi/2, \pi/3$ (i.e., if $\pi/2$ or $\pi/3$ lies in the passband), the RF scheme can be implemented without any need for additional multiplication. The noise gain g_0^2 for this RF structure can be obtained as

$$g_0^2 = \text{trace}(\boldsymbol{P}_0) \tag{3.28}$$

where

$$\boldsymbol{P}_0 = \boldsymbol{W}_0 + (\boldsymbol{I}-\boldsymbol{A}_0\boldsymbol{U})^t\boldsymbol{W}_0 + \boldsymbol{W}_0\left(\boldsymbol{U}^2-\boldsymbol{A}_0\boldsymbol{U}\right) + (\boldsymbol{A}_0-\boldsymbol{U})^t\boldsymbol{A}_0^t\boldsymbol{W}_0 + \boldsymbol{W}_0\boldsymbol{A}_0(\boldsymbol{A}_0-\boldsymbol{U}) \tag{3.29}$$

γ_κ, $\kappa = 1, \ldots, N$ is determining the error transmission zero for each state [67]. One possible choice is $\gamma_\kappa = \gamma$, $\kappa = 1, \ldots, N$ which leads to $\boldsymbol{A}_1 = \boldsymbol{U} = \gamma\boldsymbol{I}$. In such case, the noise gain satisfies (3.16) where $\boldsymbol{P}_0$ form (3.29) becomes

$$P_0 = W_0 + (I-\gamma A_0)^t W_0 + W_0\left(\gamma^2 I-\gamma A_0\right) +$$

$$(A_0-\gamma I)^t A_0^t W_0 + W_0 A_0 (A_0-\gamma I) \qquad (3.30)$$

It is shown in Appendix B, lemma 4, that P_0 defined in (3.30) is positive definite if $\lambda_{min}\left((\gamma I-\tilde{A})^2 + \left((\gamma I-\tilde{A})^t \right)^2 \right) > \gamma^2 - 2$. The optimal RF structures for BPF can be obtained by applying the algorithm presented in [51] by using the expression for P_0 in (3.30).

Remark: The optimal RF structures presented in Subsections 3.3.1, 3.3.2 and 3.3.3 provide lower output roundoff noise than those of MRH if and only if the sum of the residue modes (using the corresponding definition of residue matrix) is lower than the sum of the corresponding second-order modes [51].

All RF structures presented in this section will be called full residue feedback (FRF) structures since they have, in general, full state-space matrices. A disadvantages of such structures, as well as, of the MRH structures is the high number of arithmetic operations required. In the next section, new structures will be presented which require significantly less arithmetic operations at the expense of a slight increase of the noise gain.

3.4 New Low-Complexity Suboptimal RF Structures

In this section, structures will be presented which require less arithmetic operations than the ones presented in the previous section. The approach to be used consists in finding a triangular similarity transformation which minimizes the noise gain subject to the l_2-scaling constraint. This approach was used in [68] to minimize the roundoff noise and obtain realization which has system matrix in Hessenberg form and thus requires lower number of arithmetic operations. This approach will be extended here to RF implementations.

Consider an RF realization $\{A_0, B_0, C_0, d\}$ in (3.7) and (3.8) where A_1 and A_2 are

chosen depending on the filter types (LPF, HPF, BSF, BPF) as discussed in the previous section. The residue matrix $\boldsymbol{P}_0$ is given by (3.14) for LPF, (3.17) for HPF, (3.22) for BSF scheme 1, (3.27) for BSF scheme 2, and (3.30) for BPF. The noise gain g_0^2 for all the cases can be calculated from (3.28) using the corresponding residue matrix for each filter type.

The noise gain can be minimized under the l_2-scaling constraint and the constraint that the transformation matrix is triangular using an iterative algorithm which is a modified version of the minimization algorithm of [68]. The modification counts in using the residue matrix $\boldsymbol{P}_0$ instead of the noise matrix $\boldsymbol{W}_0$. The resulting minimization algorithm is discussed in the next subsection.

3.4.1 Minimization Algorithm

The minimization algorithm can be summarized as follows:

The initial realization $\{\boldsymbol{A}_0, \boldsymbol{B}_0, \boldsymbol{C}_0, d\}$ is l_2-scaled using

$$\boldsymbol{T}(\boldsymbol{0}) = \text{diag}\{\sqrt{k_{ii}(0)}\}, \; i = 1, \ldots, N \tag{3.31}$$

where $k_{ii}(0)$ is the ith diagonal element of $\boldsymbol{K}_0$, $\boldsymbol{K}(\boldsymbol{1}) = \boldsymbol{T}^{-1}(\boldsymbol{0})\boldsymbol{K}_0\boldsymbol{T}^{-t}(\boldsymbol{0})$ and $\boldsymbol{P}(\boldsymbol{1}) = \boldsymbol{T}^{t}(\boldsymbol{0})\boldsymbol{P}_0\boldsymbol{T}(\boldsymbol{0})$.

For $l > 0$, $\boldsymbol{T}(l)$ will have the form of identity matrix except two entries only, which are denoted by $t_{ii}(l)$ and $t_{ij}(l)$, where $j > i$ for upper triangular matrices. t_{ii} will be chosen to retain l_2-scaling condition in (1.11) and t_{ij} will be chosen to reduce the noise gain in (3.16). This reduction under l_2-scaling constraint is given by

$$\Delta g^2_{\boldsymbol{T}(l)} = \frac{p_{ii}(l)}{2}\left[k_{ij}(l) - \frac{p_{ij}(l)}{p_{ii}(l)}\right]^2 \tag{3.32}$$

where p_{ij} is the ijth element of the corresponding residue matrix $\boldsymbol{P}(l)$, p_{ii} is the ith diagonal element of the corresponding residue matrix $\boldsymbol{P}(l)$ and k_{ij} is the ijth element of the matrix $\boldsymbol{K}(l)$. The indices i, j , $j > i$ which give the maximum reduction for $\Delta g^2_{T(l)}$ are obtained through a search process for all $i, j = 1, \ldots, N$ and $j > i$. The value of t_{ij} is obtained from

$$t_{ij}(l) = \frac{1}{2}\left(k_{ij}(l) - \frac{p_{ij}(l)}{p_{ii}(l)} \right) \tag{3.33}$$

The value of t_{ii} will be obtained from [68]

$$t_{ii}(l) = \sqrt{t_{ij}^2(l) - 2t_{ij}(l)\, k_{ij}(l) + 1} \tag{3.34}$$

The process is repeated until the maximum possible reduction in noise gain falls below a given threshold and convergence is achieved. The similarity transformation $\boldsymbol{T}$

$$\boldsymbol{T} = \boldsymbol{T}(0)\,\boldsymbol{T}(1)\,\boldsymbol{T}(2) \ldots\ldots\ldots \tag{3.35}$$

will be an upper triangular matrix and can be applied to the RF structure as shown in (1.15) and (3.15). The resulting structures are suboptimal since the similarity transformation $\boldsymbol{T}$ is limited to be a triangular matrix.

The above algorithm can be applied to the cases of LPF, HPF, BSF scheme 2, and BPF since their corresponding residue matrices satisfy (3.15) and the output noise gains satisfy (3.16). For BSF scheme 1 (with different signs of block-diagonal sub-matrices in $\boldsymbol{J}_0$), a slight modification of the above algorithm is needed since the output noise gain defined in (3.19) does not satisfy (3.16). One way to make the algorithm applicable to this case is to ensure the index j of each $\boldsymbol{T}(l)$ does not exceed the index r of the matrix $\boldsymbol{J}_0$. In that case, the noise gain after each individual similarity transformation will satisfy (3.16) and the definition of $\boldsymbol{P}_0$ in (3.22) can be used for the algorithm.

3.4.2 Initial Structure For RF Suboptimal Realizations

The suboptimal realizations obtained by applying the minimization algorithm may differ according to the initial structure $\{\boldsymbol{A}_0, \boldsymbol{B}_0, \boldsymbol{C}_0, d\}$. These different realizations will perform, in general, differently under finite wordlength effects. Two schemes will be presented in the following paragraphs. In scheme 1, the initial realization is the direct form (one section) while in scheme 2 the initial structure is a cascade of second-order sections each in direct form.

Scheme 1:

If the direct form realization (one section) is chosen as an initial structure, the resulting realization after applying the minimization algorithm will have a state matrix in Hessenberg form. This implies that the resulting realization has $N(N-1)/2$ coefficients equal to zero. The new realization provides excellent performances under finite wordlength effects in terms of output roundoff noise and coefficient sensitivity as will be shown in Section 3.5.

Scheme 2:

In this case, the initial realization is the cascade of first- and second-order sections each in direct form. The resulting realization has a state matrix which in the form of block-triangular which are more suitable for high-speed VLSI hardware implementations. This realization has also $N(N-2)/2$ coefficients which are equal zero and it provides excellent performance in terms of output roundoff noise and coefficient sensitivity as it will be demonstrated in Section 3.5. This initial structure can easily be obtained as shown in [16], [17]. The initial structure is neither required to be optimized in the sense of zero-pole pairing or section ordering nor to be well l_2-scaled.

Remark: It is clear that the choice of the initial structure has a significant effect on the final realization obtained from the minimization algorithm. Other variations of the initial

structure proposed in Scheme 2 is the parallel structure, or the use of any of the many low-roundoff structures for each of the individual sections [69].

All these new suboptimal RF realizations (based on the choice of the initial structure) have a reduced number of multiplication and addition operations compared to the structures presented in Section 3.3. These new realizations have near-optimal output roundoff noise which may be lower than that of many other low-noise realizations including the MRH structures as will be shown in Section 3.5. These structures (scheme 1 and scheme 2) will be called suboptimal residue feedback (SRF) structures.

3.5 Performance Analysis

In this section, the performance of the structures discussed in Sections 3.3, 3.4 is compared with that of two other low-noise realizations in terms of computational complexity, roundoff noise and coefficient sensitivity. The five low-noise realizations to be considered are:

1. MRH structure: The MRH realizations [28], [29] (without RF technique).
2. SB structure: The low-noise realizations (without RF technique) proposed by Smith and Bomar in [68] and they will be called SB structures for the sake of brevity.
3. FRF structure: The FRF realizations presented in Subsections 3.3.1, 3.3.2 and 3.3.3. The FRF structures for LPF and HPF have been proposed by Williamson in [51] and briefly discussed in Subsection 3.3.1. The FRF structures for BSF and BPF are proposed in Subsection 3.3.2 and 3.3.3.
4. SRF scheme 1 structure: The SRF scheme 1 realizations proposed in Section 3.4.

5. SRF scheme 2 structure: The SRF scheme 2 realization proposed in Section 3.4

The computational complexity in terms of number of required multiplications and additions per output sample for the above structures are summarized in Table 3.1. It can be seen from Table 3.1 that SRF scheme 1 structure has a saving of $N(N-1)/2$ multiplies compared to MRH and FRF structures and also has the same number of multiplies as SB structures. SRF scheme 2 structure has a saving of $N(N-2)/2$ multiplies compared to MRH and FRF structures. They also requires $N/2$ multiplies more than SB structures. It should also be noted that for BPF, FRF structures and SRF structures may require extra N additions and extra N multiplications over what is listed in Table 3.1 due to possible real value of γ as explained in Subsection 3.3.3.

In order to compare the coefficient sensitivities of the proposed structures with other low-noise structures, the coefficient sensitivity measure reported in [31] will be used. This measure Φ_T for any l_2-scaled state-space realization $(\boldsymbol{A}_T, \boldsymbol{B}_T, \boldsymbol{C}_T, d)$ is given by [31]

$$\Phi_T = (N+1)\,\mathrm{trace}\,(\boldsymbol{W}_T) + N \tag{3.36}$$

Three examples will be considered for the comparison of the proposed RF structures. The first example deals with the cases of LPF and HPF. The second example deals with the case of BSF. Finally, the case of BPF is investigated in the third example. For each of the three examples, the output roundoff noise gain g_T^2 and coefficient sensitivity measure Φ_T for each of the five structures will be calculated.

Table 3.1: Computational complexity of different IIR realizations (***N* is even**).

	Number of Multiplications	Number of Additions
MRH structure [28], [29]	$(N+1)^2$	$N(N+1)$
SB structures [68]	$\left(N^2+5N+2\right)/2$	$\left(N^2+3N\right)/2$
FRF structure[a] (Section 3.3)	$(N+1)^2$	$N(N+2)$
SRF scheme1[a] (Section 3.4)	$\left(N^2+5N+2\right)/2$	$\left(N^2+5N\right)/2$
SRF scheme 2[a] (Section 3.4)	$\left(N^2+6N+2\right)/2$	$\left(N^2+6N\right)/2$

a. More multiplications and additions may be required for the BPF case.

Example 3.1 (LPF and HPF study-case)

In this example, four sixth-order narrow-band IIR filters which include an elliptic LPF, an elliptic HPF, a Chebyshev LPF and a Chebyshev HPF are considered. The specifications of the four filters are listed in Table 3.2. Each filter in Table 3.2 has been realized by the five different structures in Table 3.1 and output noise gains g_T^2 and coefficient sensitivity measures Φ_T are listed in Table 3.3. The following observations can be made from Table 3.3:

- Both SRF scheme 1 and SRF scheme 2 structures (proposed in Section 3.4) provide output roundoff noise gains which are very close to that of optimal FRF structures proposed by Williamson in [51]. Therefore, both schemes provide near-optimal output roundoff noise gains with significant reduction in computational complexity (see Table 3.1).

- SRF structures provide significant reduction (more than 8 dB for Chebyshev and 11 dB for elliptic) in output noise gains compared to MRH structures with significant reduction in the computational complexity (see Table 3.1) at the price of increasing the coefficient sensitivity.
- SRF structures provide significant reduction (more than 8 dB for Chebyshev and 11 dB for elliptic) in output roundoff noise gains compared to SB structures. This improvement in output roundoff noise may compensate for the N extra additions required for scheme 1 and $N/2$ extra multiplications and $3N/2$ extra additions required for scheme 2 over that is required for SB structures.

Based on the above results, it can be concluded that the proposed SRF structures provide a good compromise in terms of output roundoff noise, coefficient sensitivity and computational complexity for narrow-band LPF and HPF.

Table 3.2: Filter Specifications for Example 3.1.

Parameters	elliptic LPF	Chebyshev LPF	elliptic HPF	Chebyshev HPF
A_p dB	1.00	0.60	0.80	0.60
A_a dB	72.89	40.00	66.50	48.55
ω_p rad/s	250.00	400.00	4,650.00	4,400.00
ω_a rad/s	400.00	700.00	4,450.00	3,800.00
ω_s rad/s	10,000.00			

A_p = maximum passband ripple, dB
A_a = minimum stopband attenuation, dB
ω_p = passband edge, rad/s
ω_a = stopband edge, rad/s
ω_s = sampling frequency, rad/s.

Table 3.3: Results of Example 3.1.

	elliptic LPF		Chebysh. LPF		elliptic HPF		Chebysh. HPF	
	g_T^2 dB	Φ_T	g_T^2 dB	Φ_T	g_T^2 dB	Φ_T	g_T^2 dB	Φ_T
MRH Struct.	1.42	15.71	1.45	15.77	1.34	15.54	1.38	15.63
SB Struct.	1.45	15.77	1.54	15.98	1.76	16.56	1.57	18.58
FRF Struct.	-13.28	19.96	-10.86	19.52	-11.61	18.86	-7.96	16.05
SRF Scheme 1	-13.19	20.04	-10.56	18.943	-11.08	19.53	-7.83	18.01
SRF Scheme 2	-12.92	25.00	-9.96	19.22	-10.71	21.47	-7.07	18.95

Example 3.2 (BSF study-case)

In this example, four eighth-order narrow-band BSF are considered. The specifications of the four filters are listed in Table 3.4. The first three filters are elliptic BSF with the same specifications except the locations of the stopband. They will be called elliptic-1, elliptic-2, and elliptic-3. Elliptic-1 has its stopband at the lower range of frequency-band, elliptic-2 has its stopband at the higher portion of the frequency-band, and elliptic-3 has its stopband at the middle portion of the frequency-band. The fourth filter is a Chebyshev BSF.

Each filter in Table 3.4 has been realized by five different structures which are MRH structure, SB structure, FRF scheme 1, FRF scheme 2, and SRF scheme 2. Here SRF scheme 1 has not been considered since it has been shown in the previous example that the both schemes of SRF perform similarly. The output noise gains g_T^2 and coefficient sensitivity measures Φ_T for the four filters realized by the five realizations are

listed in Table 3.5. Many observations can be obtained from Table 3.5 and they can be summarized below:

Table 3.4. BSF Specifications.

Parameters	elliptic-1	elliptic-2	elliptic-3	Chebyshev
A_p dB	0.50	0.50	0.50	1.00
A_a dB	50.60	50.60	50.60	40.40
ω_{p1} rad/s	750.00	3,450.00	2,450.00	3,850.00
ω_{p2} rad/s	1,650.00	4,350.00	3,350.00	4,500.00
ω_{a1} rad/s	1,050.00	3,750.00	2,750.00	4,150.00
ω_{a2} rad/s	1,350.00	4,050.00	3,050.00	4,350.00
w_s rad/s	10,000.00			

ω_{p1}, ω_{p2} = passband edges, rad/s
ω_{a1}, ω_{a2} = stopband edges, rad/s

- From Table 3.5 it can be seen that FRF scheme 1 and SRF scheme 2 are the best structures (in terms of output roundoff noise) for all the examples except for elliptic-3 case. They have been obtained after extensive search for the optimal choice of $\boldsymbol{J_0}$. These choices of $\boldsymbol{J_0}$ are included in Table 3.5. For FRF scheme 1, the choices of $\boldsymbol{J_0}$ are deliberately chosen to show the performance of this scheme with different signs in block-diagonal sub-matrices of $\boldsymbol{J_0}$. These choices of $\boldsymbol{J_0}$ are also included in Table 3.5. It is obvious that the choice of $\boldsymbol{J_0}$ depends on the specifications of the filter (specifically on the location of the stopband). From the results presented in Table 3.5 the recommendation for the choice of $\boldsymbol{J_0}$ is that: for elliptic-1 (i.e., stopband at the lower portion of the frequency band), it is recommended to place most of error transfer function zeros at

$z = 1$; for elliptic-2 it is recommended to place most of error transfer function zeros at $z = -1$. For elliptic-3, the RF technique seems not to be particularly effective.

Table 3.5: Results of Example 3.2.

	elliptic-1 BSF		elliptic-2 BSF		elliptic-3 BSF		Chebysh. BSF	
	g_T^2 dB	Φ_T	g_T^2 dB	Φ_T	g_T^2 dB	Φ_T	g_T^2 dB	Φ_T
MRH Struct	2.38	23.58	2.38	23.58	2.38	23.58	2.65	24.56
SB Struct.	3.16	26.63	3.10	26.36	3.32	27.35	3.14	26.54
FRF Scheme 1	1.09	29.28	0.548	30.26	4.15	25.39	-0.92	32.55
	$J_0 = \begin{bmatrix} I_{7\times 7} & 0 \\ 0 & -1 \end{bmatrix}$		$J_0 = -I$		$J_0 = -I$		$J_0 = \begin{bmatrix} -I_{7\times 7} & 0 \\ 0 & 1 \end{bmatrix}$	
FRF Scheme 2	4.52	25.26	4.01	26.40	7.05	23.85	2.83	28.73
SRF Scheme 2	1.02	26.31	0.55	27.26	4.45	26.06	-0.88	33.42
	$J_0 = I$		$J_0 = -I$		$J_0 = -I$		$J_0 = -I$	

- From Table 3.5, it can be observed that both FRF scheme 1 and SRF scheme 2 provide lower output roundoff noise gains (less than 3 dB) compared to that of both MRH and SB structures (except for elliptic-3 case). However, the noise gain reductions are not as significant as those achieved for LPF and HPF (see Example 3.1) which implies that the RF technique is not particularly effective for the BSF case. A similar observation has been presented in [49] and explained by the inability of the RF technique to attenuate the error spectra over the entire band (BSF has a significant response over most of the frequency band). This explanation may be confirmed by the modest behavior of the FRF scheme 2 since it provide higher output roundoff error gains than MRH and SB structures which don't use RF technique at all!. Consider also that two zeros are

inserted at the error transfer function of each state when using FRF scheme 2. This may lead to the conclusion that increasing the order of RF scheme for BSF will adversely affect the output roundoff noise.

- The results in Table 3.5 may imply the RF technique is more effective for Chebyshev BSF than for elliptic BSF with the same filter order.

Example 3.3 (BPF study-case)

In this example, three sixth-order and one eighth-order narrow-band BPF are considered. The specifications of the four filters are listed in Table 3.6. Each filter has been realized by the four realizations of Table 3.1. These are the realizations of Table 3.1 except SRF scheme 1 which leads to close performance to that of SRF scheme 2. The measures g_T^2, Φ_T and the γ values used for FRF and SRF structures are listed in Table 3.7. Many observations and conclusions can be derived from Table 3.7 summarized below:

Table 3.6: BPF specifications for Example 3.3.

Parameters	Elliptic	Elliptic	Chebyshev	Chebyshev
A_p dB	0.50	0.80	1.00	0.80
A_a dB	65.50	51.80	40.50	46.87
ω_{p1} rad/s	980.00	2,400.00	3,200.00	1,450.00
ω_{p2} rad/s	1,020.00	2,520.00	3,400.00	1,700.00
ω_{a1} rad/s	850.00	2,100.00	2,900.00	1,150.00
ω_{a2} rad/s	1,150.00	2,950.00	3,700.00	2,000.00
ω_s rad/s	10,000.00			

- Both FRF and SRF structures provide significant reduction (more than 12 dB) in output roundoff noise gains compared to both MRH and SB structures at the price of increasing the coefficient sensitivity. These improvements in roundoff noise gains prove that RF technique is very effective for narrow-band BPF cases in reducing the output roundoff noise.
- For the second and the fourth BPF examples, FRF and SRF structures provide excellent performance in terms of output roundoff noise and coefficient sensitivity with γ chosen to be 0 and 1, respectively. This means that for some cases of BPF's (depending on the location of the passband), efficient RF schemes can be implemented without the need for any extra multiplication.
- SRF structures provide close output roundoff gains to that of FRF structures which implies that SRF structures provide near-optimal output roundoff noise gain with significant reduction in computational complexity (see Table 3.1).

Table 3.7: Results of Example 3.3.

	elliptic BPF		elliptic BPF		Chebysh. BPF		Chebysh. BPF	
	g_T^2 dB	Φ_T	g_T^2 dB	Φ_T	g_T^2 dB	Φ_T	g_T^2 dB	Φ_T
MRH Struct.	1.15	15.13	1.30	15.46	1.38	15.63	1.43	15.75
SB Struct.	1.16	15.15	1.31	15.47	1.40	15.67	1.49	15.88
FRF	-22.01	38.71	-14.44	23.17	-12.74	21.76	-11.02	21.54
	γ = 1.618		γ = 0		γ = –0.9635		γ = 1	
SRF Scheme 2	-19.54	21.92	-13.67	17.31	-12.04	18.68	-10.41	17.34
	γ = 1.618		γ = 0		γ = –0.9635		γ = 1	

3.6 Conclusion

In this chapter, the problem of synthesizing finite wordlength fixed-point realizations of Nth-order IIR digital filters, which use second-order residue feedback to minimize the output roundoff noise subject to l_2-scaling, has been considered. It has been shown that by proper choice of simple residue feedback schemes, new optimal structures can be obtained for narrow band-stop and band-pass filters. These optimal structures provide lower output roundoff noise compared to the MRH structures proposed by Mullis, Roberts [28] and Hwang [29] if the sum of the residue modes is less than the sum of the second-order modes.

Further, new suboptimal residue feedback structures for narrow-band IIR filters are also proposed which provide near-optimal roundoff noise and have a saving of at least $N(N-2)/2$ multiplies over the optimal structures. Moreover, these suboptimal structures can be chosen to have block-triangular state-update matrices which are more suitable for high-speed hardware implementations. Extensive numerical comparisons between the proposed suboptimal structures and three other low-noise structures show that the proposed suboptimal structures provide excellent performance in terms of output roundoff noise and coefficient sensitivity as well as low computational complexity. It has also been found that the involved simple residue feedback schemes are very powerful and versatile methods to reduce the quantization noise for narrow-band low-pass, high-pass and band-pass filters. For narrow band-stop filters, the proposed residue feedback schemes seem to be less effective.

Chapter 4

Implementations of Residue-Feedback IIR Digital Filter Realization

4.1 Introduction

The digital filter application sets certain specifications to be met such as input sampling rate, device cost, and S/N ratio. On the other hand, the implementation technique imposes certain limitations on those specifications. Therefore, there is usually a tradeoff between the required specifications and implementation cost.

In this chapter, two implementations are presented for the SRF scheme 2 realization proposed in Chapter 3. This particular realization was chosen among the proposed realizations in Chapter 3 due to its block-triangular system matrix which leads to simple and high-performance implementations. The first implementation uses a standard off-the-shelf fixed-point digital signal processor (Motorola DSP56001). The second implementation is an application specific integrated circuit (ASIC) VLSI architecture suitable for high-speed applications.

4.2 DSP Implementation of SRF Realization

The DSP chip has a fixed architecture which is usually designed to accommodate most digital signal processing applications. Therefore, most of the digital filter realization

parameters such as states and coefficient wordlengths are fixed (as determined by the specifications of the DSP) and must be taken as they are. It is not possible to take advantage of the low roundoff noise and low coefficient sensitivity of the proposed SRF realization in reducing the states and coefficient wordlengths. The main purpose of implementing SRF realization using a DSP is to investigate its FWL performance for real-time applications. This real-time performance will be also compared with other realizations using the same DSP56001 environment.

4.2.1 Motorola DSP56001

Motorola DSP56001 is a fourth-generation signal processor [70], that is widely used and well known. The processor is capable of high processing speed due to its dual architecture and the incorporation of a parallel multiplier-accumulator. The processor has 24-bit data and coefficient wordlength which provides a 144 dB dynamic range; intermediate results can be held in 56-bit accumulators. The DSP56001 architecture has two independent expanded data memory spaces (up to 64k $\times$ 24 each), two address arithmetic units, and an arithmetic logic unit (ALU) which has two accumulators. The duality of the architecture allows a 24-bit $\times$ 24-bit multiplication, a 56-bit addition, two data moves, and two address-pointer updates in a single instruction cycle. The instruction cycle takes 100 ns (equivalent to 10M instructions per second.)

4.2.2 Implementation of the SRF Realization

The assembly code for the implementation of the SRF scheme 2 realization of LPF is listed in Appendix C. This program code can be easily modified to accommodate the HPF, BSF, BPF cases discussed in Section 3.4. Two's complement truncation is used for quantization process i.e., the lower part of the 48-bit product is simply removed and temporarily stored. By storing the lower part, it is possible to feed it back to the next iteration in order to incorporate the residue-feedback process. However, the lower part must

be provided with the correct sign bit, which requires extra instructions. Saturation arithmetic is used to deal with the overflow of the states and output. This saturation process is a feature provided by the DSP chip.

4.2.3 Cost and Performance Analysis

The traditional cost criteria of a digital filter implementation such as the number of multiplications and additions are not relevant in the DSP environment but the suitability of the realization to the DSP architecture has a more relevant impact. Therefore the cost criteria of any filter implementation using DSP should be measured in terms of:

1. Maximum sampling frequency
2. Data memory requirements

The speed (i.e., maximum input sampling rate) of a filter is inversely proportional to the length of the assembly code (or more accurately, the total number of instructions that are performed during a single sample interval), so the question of how efficiently the filter structure can be turned to assembly code is often of atmost importance. The data memory requirements are related to the number of the filter coefficients stored in memory and to the intermediate data needed at each cycle. They indicate the complexity of the filter realization and its degree of suitability to the DSP architecture.

The FWL performance criteria of any filter implementation usually include coefficient sensitivity, output roundoff noise, and tendency to self-sustaining periodic limit cycles. Coefficient sensitivity is not of prime importance in DSP56001 signal processing implementation since the available wordlength (24 bit) is usually sufficient for most practical cases. The output roundoff noise is also not of prime importance for the same reason. However, attention will be given to the output roundoff noise for two reasons:

1. The prediction of the theoretical noise model explained in Section 1.2.2 will be compared with actual measurements.

2. The noise performance measurement of the actual DSP implementation, based on a specific realization, can be also used for any other implementation (e.g., VLSI array-processor implementation). To obtain this general measurement, the noise performance will be measured normalized to the noise power of one rounding operation.

The limit cycle behavior of the filter implementation is important since it can take over the filtering operation and it is difficult to be theoretically determined.

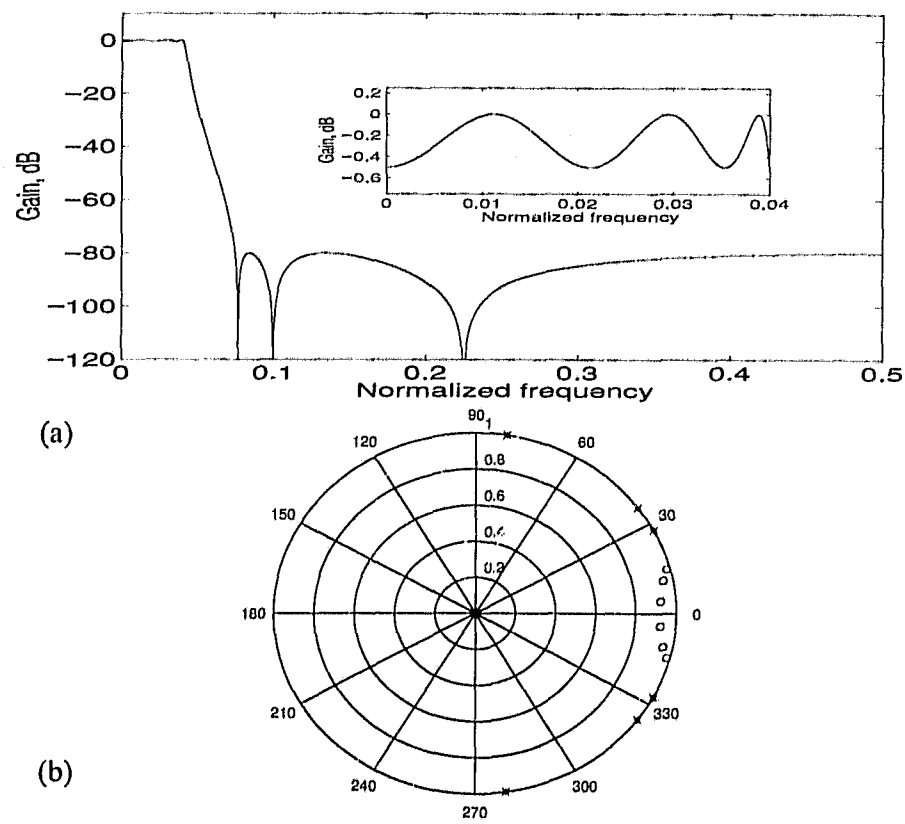


Figure 4.1: Narrow-band LPF test example filter. (a) Amplitude response. The inset shows the response in the passband. (b) zero-pole plot ('o' is a pole and '*' is a zero).

4.2.4 Test example

A specific narrow-band sixth-order elliptic LPF is chosen to be DSP implemented using three different realizations. The specifications of the filter are:

A_p = 0.5 dB
A_a = 80 dB
ω_p = 400 rad/s
ω_a = 800 rad/s
ω_s = 10,000 rad/s.

The amplitude response and the zero-pole plot of the filter are shown in Fig. 4.1. The three realizations are :

1. SRF scheme 2 realization proposed in Chapter 3.
2. MRH realization [28], [29].
3. Cascade of second-order sections in direct form II [20].

The code for the three realizations was written in assembly. All realizations are designed to satisfy the l_2-scaling constraint in (1.11). Two's complement truncation and saturation arithmetic have been used for the three realizations. A great effort was made to optimize the code as much as possible.

4.2.5 Testing

The actual transfer function with quantized coefficients was tested for the three realizations. All realizations met the filter specifications. The code length (in instruction cycles) of each structure has been calculated for the complete filter loop which includes the I/O instructions. Therefore, the real-time maximum sampling frequency can be determined based on this code length. To run the program, some extra instruction cycles are required to load the coefficients and set the pointers. The code lengths, maximum sampling rates and the data memory requirements of the three realizations for this specific test filter example are listed in Table 4.1.

The relative power spectral density (RPSD) [71] was measured. This RPSD can be calculated from

$$RPSD = S_0(w)/\sigma^2_r \tag{4.1}$$

where $S_0(w)$ is the power spectral density of the output noise due to product quantization. RPSD was experimentally determined using the following method. A digitally generated Gaussian noise has been filtered with each structure. The output sequences were subtracted from the corresponding sequences obtained by filtering the identical input signal with a filter with the same coefficients but without quantization of states and outputs (simulated on a SPARC workstation). By doing so, it is guaranteed that the resulting errors are only due to the quantization operations without the influence of input data and coefficient quantization. The resulting errors were analyzed using the Fast Fourier Transform (FFT) and the corresponding RPSDs were calculated. Some 10 spectra were averaged by altering the amplitude of the input noise which resulted in a smoother estimate. For the three realizations considered, the noise spectra have been theoretically calculated from the common noise model [17], [16], [Section 3.4] and the theoretical predictions are in good agreement with the experimental results. The analytical and experimental noise spectra of the SRF implementation are shown in Fig. 4.2. The noise powers for the three structures obtained by integrating the noise spectra are shown in Table 4.1.

Limit cycles and overflow behavior have been examined using a signal generator at the A/D converter and an oscilloscope at the output of the D/A converter. Constant-input quantization limit cycles have been searched using zero and constant inputs. The overflow behavior was checked using large-amplitude sinusoidal and square waves with possible positive and negative DC components.

4.2.6 Results

Not surprisingly, the direct form II structure provides the fastest structure in this comparison. Straightforward sum-of-product type calculations are very suitable to DSP architectures. On the other hand, output roundoff noise is quite high. In addition, DC and AC

quantization limit cycles have been observed in the analog test. Furthermore, self-sustaining overflow oscillations have also been found. These overflow oscillations occasionally (depending on the amplitude of the input signal and the DC component) take the whole dynamic range. Other scaling technique such as L_∞-scaling, section ordering choices and zero-pole pairing choices have failed to eliminate these overflow limit cycles.

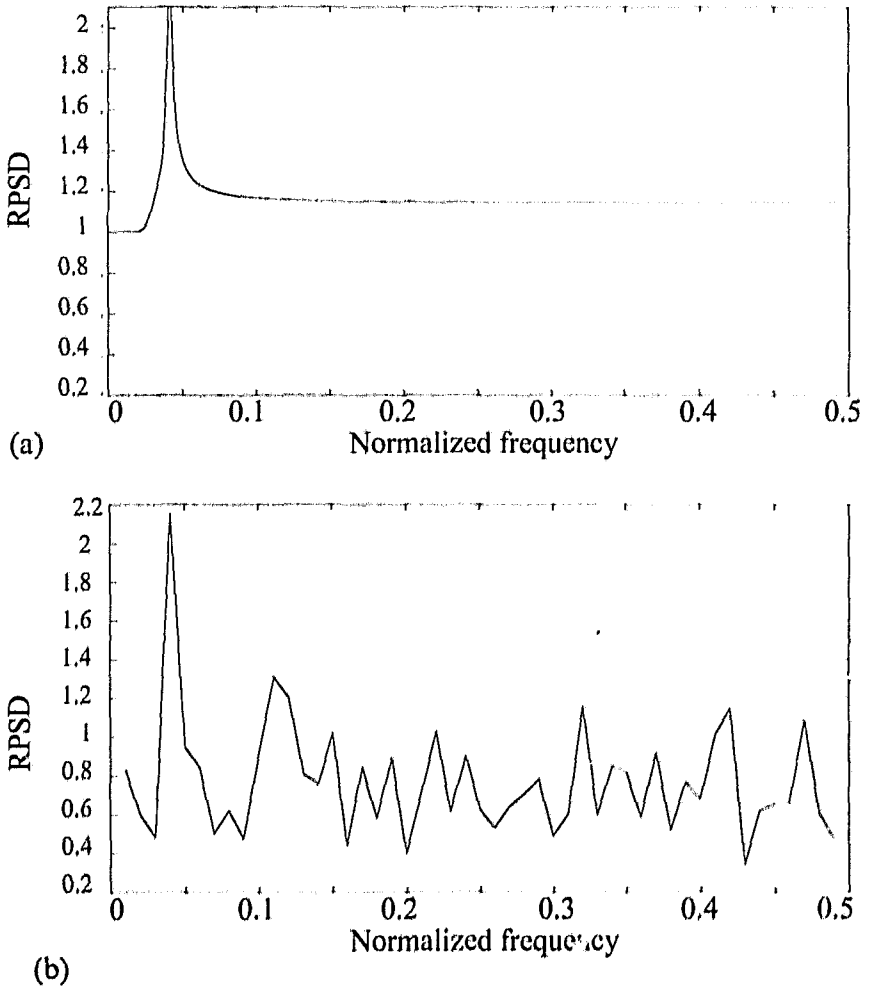


Figure 4.2: The noise spectra for the SRF implementation of the test filter. (a) analytical and (b) experimental results.

Table 4.1: Code complexity and performance of three different implementations for the sixth-order LPF test example.

	Code length (instruction cycle)	Max sampling frequency (kHz)	Data memory (states and coeffs)	Noise gain in dB from experimental data	Noise gain in dB from theory
SRF scheme 2 structure	140	70	58	0.45	0.39
MRH structure [28], [29]	120	83	61	4.10	3.70
Cascade of direct form II [20]	51	196	21	46	41

Surprisingly, the SRF scheme 2 structure requires longer code length (for this 6th-order test filter) compared to the MRH structure although it requires less arithmetic operations (see Table 3.1 in Chapter 3). This is because the SRF structure requires extra instruction cycles to extract the residues (with their proper signs) in each iteration. These extra cycles are significant for low-order filters. However, for high-order filters, these extra cycles will be less significant compared to the extra multiplication cycles required by MRH structure. For example, SRF implementation of $14th$-order filter requires 348 instruction cycles while the corresponding MRH implementation requires 440 instruction cycles.

The SRF scheme 2 structure provides the least output roundoff noise in this comparison which is consistent with the theoretical conclusion of Chapter 3. Both SRF and MRH structures have excellent limit cycle performance for this test example where neither overflow oscillations nor quantization limit cycles have been observed.

Many other narrow-band filter examples have been implemented for SRF scheme 2 structures. For all of the cases, no sign of limit cycles have been found. In conclusion, it can be said that DSP implementation of SRF structure leads to high S/N ratio, most likely freedom of limit cycles and moderate filtering speed. In the next section, all the features of the SRF scheme 2 structure discussed in Chapter 3 will be combined to obtain high-speed VLSI array-processor implementation.

4.3 An Array-Processor Implementation For SRF Realization

Compared to DSP implementations, VLSI implementations allow far more opportunities to optimize the data, states and coefficient wordlengths. VLSI implementation of narrow-band IIR digital filters based on the SRF scheme 2 structure has the following merits:

1. Short wordlengths for states and coefficients since the SRF realization has low roundoff noise and low coefficient sensitivity.
2. Simple RF scheme which generally requires only N extra additions (except for the BPF case) and can be executed in parallel with the filter state-space computations.
3. The amenability to high-speed implementation since the proposed SRF state-update (system) matrix has a block-triangular form.

In this section, a high-speed VLSI array-processor implementation is obtained for the proposed SRF structure. It has been shown in Chapter 3 that for most digital filter types, the first-order feedback scheme is sufficient. Therefore, the realization with first-order RF scheme will only be considered. The case of a second-order RF scheme can be easily obtained by a slightly modifying the resulting implementation.

The required computations for the proposed structure with first-order RF scheme

are described in (3.3) and (3.4) with $\boldsymbol{A}_2 = \boldsymbol{0}$. It has been also shown that the coefficients of the diagonal matrix $\boldsymbol{A}_1$ are ± 1. This indicates that no multiplication is involved in the implementation of the first-order RF scheme. Therefore, without loss of generality, $\boldsymbol{A}_1$ will be assumed equal to $\boldsymbol{I}$ (i.e., the case of LPF). So, the SRF realization for the LPF can be described as

$$\tilde{\boldsymbol{x}}(n+1) = \boldsymbol{A}\hat{\boldsymbol{x}}(n) + \boldsymbol{e}(n) + \boldsymbol{B}u(n) \tag{4.2}$$

$$\tilde{y}(n) = \boldsymbol{C}\hat{\boldsymbol{x}}(n) + du(n) \tag{4.3}$$

The required computations described in (4.2) and (4.3) can be divided into four subcomputations. Each subcomputation will be carried out by a different array-processor network. The first array-processor network will perform the computations $\boldsymbol{A}\hat{\boldsymbol{x}}(n) + \boldsymbol{e}(n)$ in (4.2). This network is called state-update network (SUN). The other three array-processor networks are assigned to the computations $\boldsymbol{B}u(n)$, $\boldsymbol{C}\hat{\boldsymbol{x}}(n)$ and $du(n)$, respectively. The SUN network contains the feedback operation of the IIR filter and this will determine the maximum throughput rate [72]. Specifically, the computation delays inside the feedback loop determines the maximum throughput rate.

4.3.1 State Update Network (SUN)

Figure 4.3 shows the array-processor implementation for the lower block-triangular state-update matrix of the proposed structure. The internal details of the diagonal processor elements are shown in Fig. 4.4. This SUN structure is an extension of the one presented in [73]. The design in [73] was for block-state filters, while our proposed design is for single input/output filters and the RF technique is incorporated. It can be seen from Fig. 4.4 that the computation delay inside the feedback loop is equal to the delay of one multiplication and three additions (the quantizer delay is neglected). This computational delay determines the maximum throughput rate and it is essential to reduce it. One way to

reduce the SUN computation delay is to use the accumulator-multiplier (ACM) reported in [6], [74]. The ACM processor can be used to execute one multiplication and one double-precision addition concurrently without any need for a separate adder. Since both $e(n)$ and $\hat{x}(n)$ are available at the same time, the ACM processor can be advantageously used to generate the new states. An I/O icon of the ACM is shown in Fig. 4.5(a). The diagonal processor element of the SUN using the ACM is shown in Fig. 4.5(b). It can be seen from Fig. 4.5(b) that the use of ACM reduces the computation delay inside the feedback loop to the delay of one ACM operation and two additions and eliminates the need for two adders (one adder inside each feedback loop). The delay of ACM is slightly higher than that of the corresponding array multiplier (AM) and its area is also slightly larger than that of the corresponding array-multiplier [6], [74]. The internal details of the off-diagonal processor element are shown in Fig. 4.5(c). The delay required for the computation of the off-diagonal processor element is equal to the delay of one multiplication operation and two addition operations.

4.3.2 The Pipelined Array-Processor

The complete array-processor implementation of the proposed realization is shown in Fig. 4.6(a) in which the slice pipeline technique [75] has been applied to maximize the computation rate. The internal details of the processor elements required to compute $\boldsymbol{B}u(n)$, $\boldsymbol{C}\hat{\boldsymbol{x}}(n)$ and $du(n)$ are shown in Fig. 4.6(b). From Fig. 4.5(c) and Fig. 4.6(b), it is obvious that the delays associated with the elements are not synchronized with that of SUN diagonal processor elements. However all these delays are less than the delay required for SUN diagonal processor element and therefore will not put any burden on the hardware system.

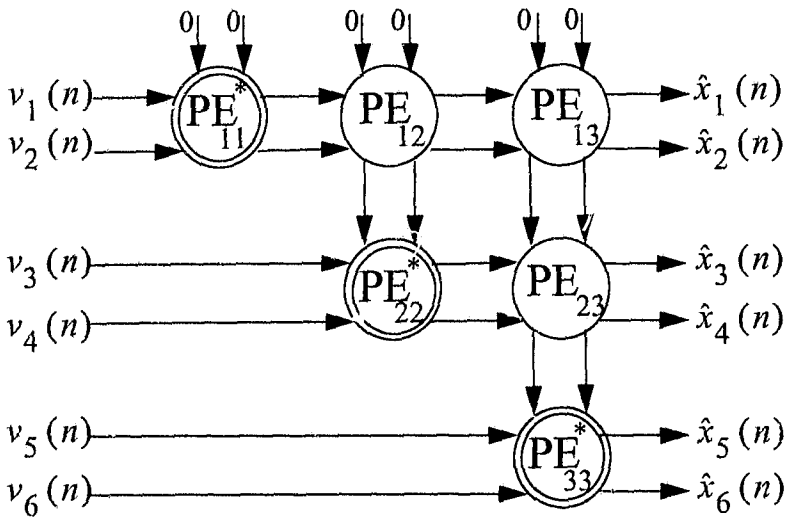


Figure 4.3: State-update network (SUN) array-processor (N=6).

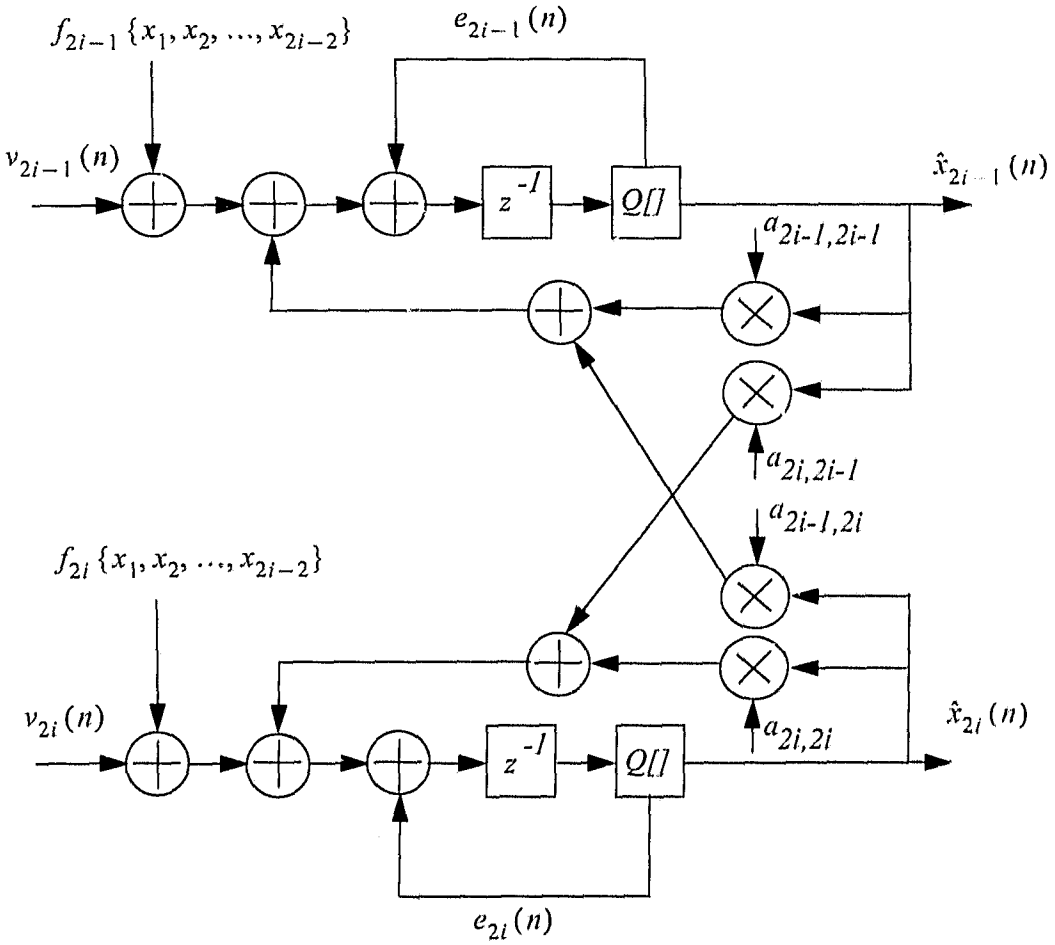
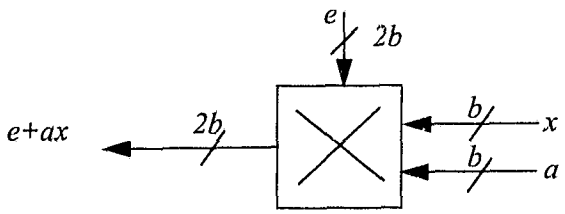
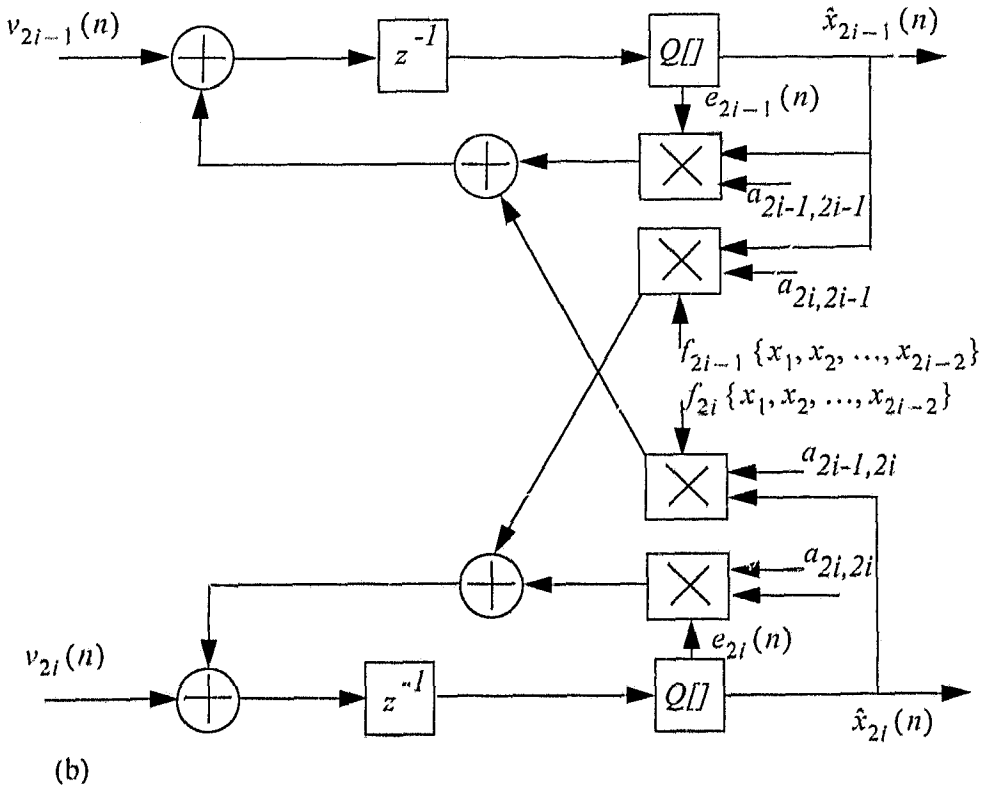


Figure 4.4: Details of the *ii*th diagonal processor element PE*.


e
2b
e+ax
2b
b
x
b
a

(a)


$v_{2i-1}(n)$
z^{-1}
$Q[]$
$\hat{x}_{2i-1}(n)$
$e_{2i-1}(n)$
$a_{2i-1,2i-1}$
$a_{2i,2i-1}$
$f_{2i-1}\{x_1, x_2, \ldots, x_{2i-2}\}$
$f_{2i}\{x_1, x_2, \ldots, x_{2i-2}\}$
$a_{2i-1,2i}$
$a_{2i,2i}$
$e_{2i}(n)$
$v_{2i}(n)$
z^{-1}
$Q[]$
$\hat{x}_{2i}(n)$

(b)

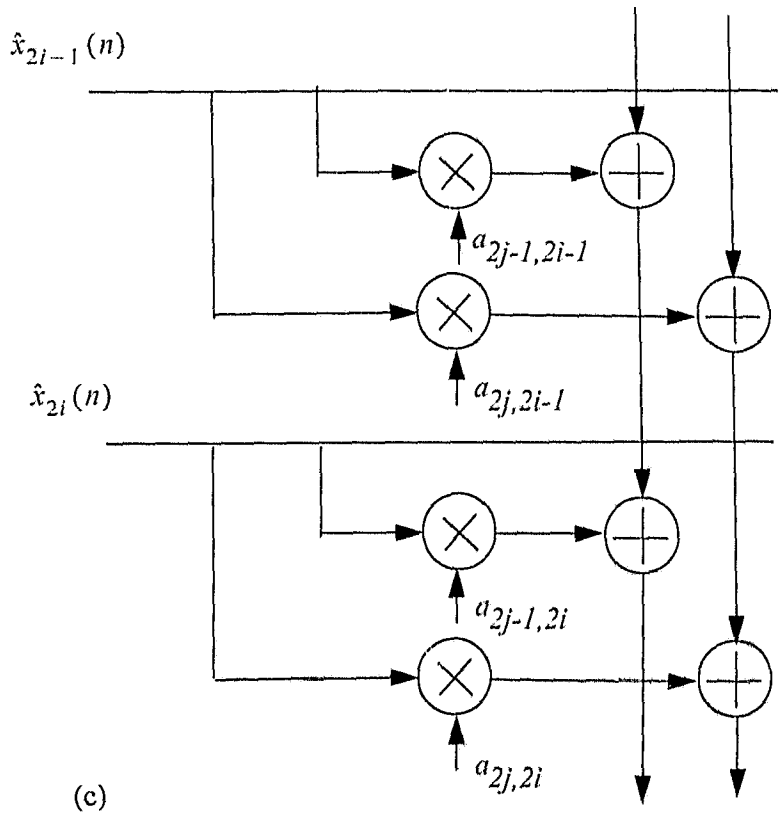


Figure 4.5: The processor elements of the SUN array processors. (a) I/O model of an ACM [6], [74]. (b) The details of the *ii*th SUN diagonal processing element using the ACM. (c) The details of the *ij*th SUN off-diagonal processing element ($j>i$).

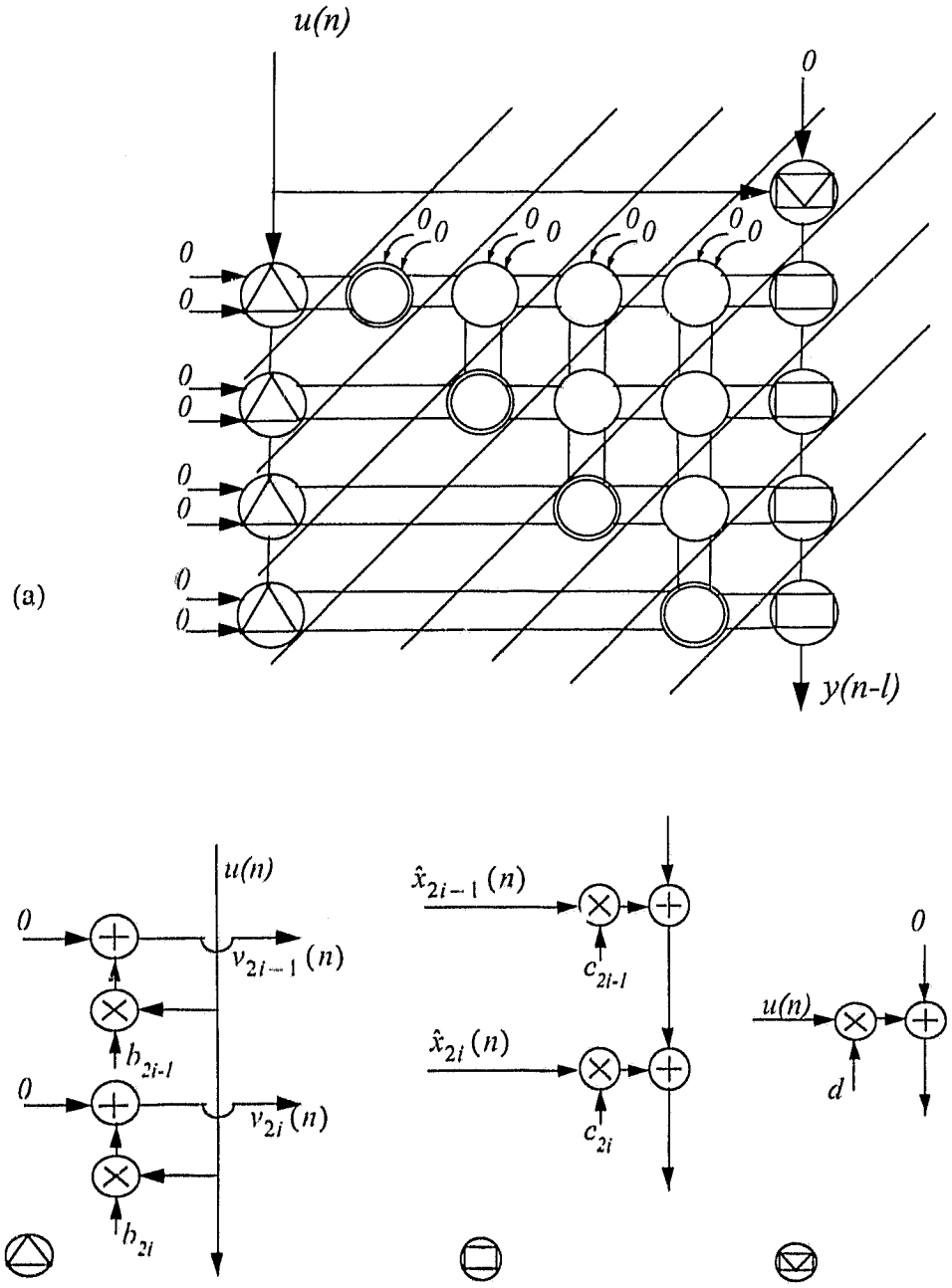


Figure 4.6: The VLSI array-processor for a 8*th*-order IIR digital filter. (a) The fully pipelined array-processor implementation of the proposed IIR filter realization. *l* is the latency. (b) The internal details and the symbols of the involved processor elements.

4.3.3 Performance Analysis

In this section, performance analysis of the proposed array-processor in terms of computation area complexity, delay, and latency is presented.

4.3.3.1 Area Complexity

It can be seen from Fig. 4.6(a) that the proposed implementation requires $N^2/2+3N+1$ multipliers ($2N$ ACMs and the other are AMs) and $(N/2)(N+6)$ double-precision adders. The area required for the buffers can not be neglected since the implementation is fully pipelined. The proposed implementation requires $(N/2)(N+1)$ double-precision buffers and $(N/2)(N+3)$ single-precision buffers. It should be noted that the sizes of the multipliers and the lengths of the adders are small since the SRF scheme 2 realization has low output roundoff noise and low coefficient sensitivity as was shown in Chapter 3. This subject will be discussed in more detail in Section 6.5.

4.3.3.2 Computational Delay

The sampling rate, T_s, of the proposed array-processor implementation for the IIR digital filter satisfies the following inequality

$$T_s \geq T_c \tag{4.4}$$

where T_c is the clock period which is used to trigger all the latches in Fig. 4.6 (a). T_c can be calculated from

$$T_c = T_{ACM} + 2T_{add} \tag{4.5}$$

where T_{ACM} is the delay required for an ACM of Fig. 4.5(a) and T_{add} is the delay required for a double-precision additions.

4.3.3.3 Latency

Latency is defined as the delay between applying the first input sample and obtaining the first output sample. From Fig. 4.6 (a), it can be seen that the latency l for the proposed array-processing implementation is given by

$$l = NT_c \tag{4.6}$$

Therefore the latency is proportional to the filter order. The latency is expected to be high as is to be expected from any fully pipelined system [75]. However, in many digital filter applications, latency is not of prime importance.

A general comparison in terms of speed and area complexity between the proposed VLSI array-processor implementation and two other VLSI implementations (one is direct implementation) is discussed in more detail in Chapter 6.

4.4 Conclusion

Two implementations of the SRF scheme 2 realization of Chapter 3 are presented. One by using a fixed-point DSP56001, while the second is an ASIC VLSI array-processor. It has been shown that for many narrow-band IIR filters, the DSP implementation based on the SRF scheme 2 realization provides high S/N ratio, freedom of limit cycles for the examples considered and moderate input-sampling frequency. The maximum input sampling frequency is 70 KHz for sixth-order filters which is more than sufficient for most audio applications.

On the other hand, the VLSI implementation of SRF scheme 2 realization provides high input sampling rate. This sampling rate is determined by the computation delay required for the accumulator-multiplier operation and two double-precision additions. This efficient VLSI implementation is obtained by taking advantage of both the parallelism inherent in the residue feedback technique and the block-triangular update state

matrix of the proposed realization. A performance comparison (in terms of hardware area and speed) which confirms the advantages of the proposed implementation over other existing implementations is discussed in Chapter 6.

Chapter 5

Design of a High-Speed Inner-Product Processor

5.1 Introduction

The inner product operation is encountered in many signal processing applications and has a direct impact on many digital filter implementations [74], [76]. In this chapter, a new scheme for two's complement fixed-point inner-product processor is presented. This novel inner-product processor will be used in the next chapter as the processing element in an efficient implementation for full-matrix state-space IIR digital filters.

5.2 Existing Inner-Product Processors

The conventional way to obtain an inner-product is to use a multiplier followed by an accumulator where the result of each multiplication is added to the previous result stored in the accumulator. In order to increase the speed of the inner-product operation, a pipelined inner-product step processor (PIPSP) has been adopted for most DSP chips [74]. The schematic diagram of a conventional PIPSP is shown in Fig. 5.1(a). The speed of the operation is limited by either the multiplier or the double-precision adder.

Further increase in speed can be achieved by eliminating the need for the separate adder of the PIPSP which also leads to a reduction in the required area [74], [77]. By

using an array-multiplier as a multiplier and adder in the same time, the multiplication and the addition operations required for each inner-product step can be executed at the same clock cycle. In this case, the modified scheme of the array-multiplier is called an accumulator-multiplier (ACM). The schematic diagram of the inner-product processor proposed in [6], [77] is shown in Fig. 5.1(b). The discussion of Fig. 5.1(c) is postponed to the next section. The ACM structure proposed in [6] is shown in Fig. 5.2 where the two's complement array-multiplier (AM) of [78] has been used. There was a minor error in the ACM and it is corrected here. In the $b \times b$ ACM, b^2 terms (binary products) are generated in parallel. The sums of the binary products propagate along the diagonals, which represent lines of equal binary weights. The carries propagate vertically downward and become inputs for the higher binary weights. For each cycle, the previously accumulated result ($V_{i-1}[0]$,..., $V_{i-1}[2b-1]$) is present at the input of the ACM at the time of multiplication. Therefore, an inner-product of length M can be executed in only M ACM cycles. The overall inner-product operation speed is governed by the ACM speed. In Fig. 5.2, it can be seen that the most significant b-bits of the $2b$-bit product are formed by the carry-propagate adder (CPA) (carry look-ahead can also be used) at the bottom of the ACM. The adder delay is a significant part of the multiplier delay. Therefore the adder delay influences the overall inner-product operation speed.

In the following section, a new scheme for two's complement fixed-point inner-product processor is presented. It will be shown that the proposed scheme provides an increase in the computation speed (approximately doubling the speed for long inner-product operation) with a slight increase in the required area.

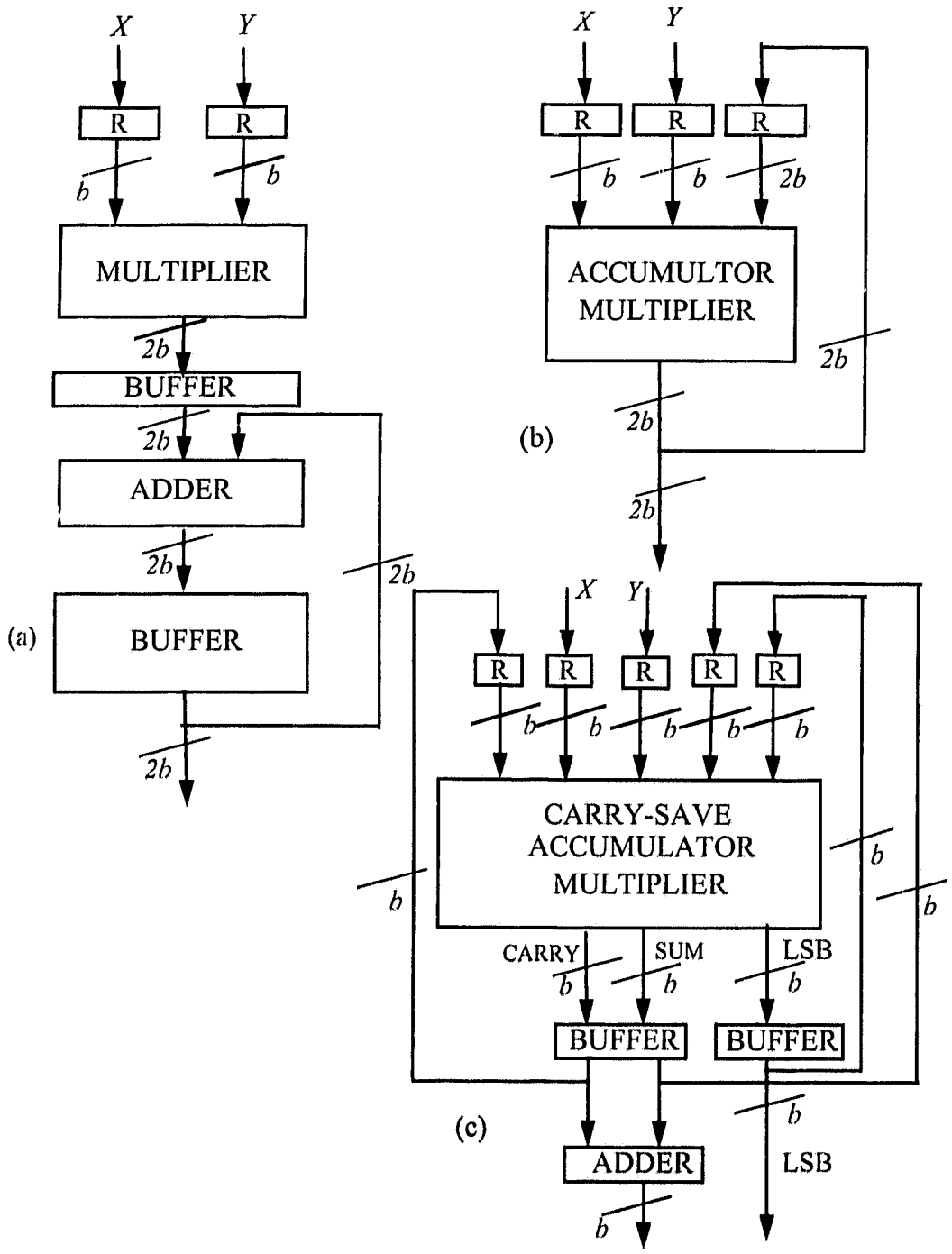


Figure 5.1: The schematic diagrams of different inner-product processors. (a) Pipelined inner-product step processor (PIPSP). (b) Accumulator-multiplier (ACM) [6], [74]. (c) Carry save inner-product processor (CSIPP).

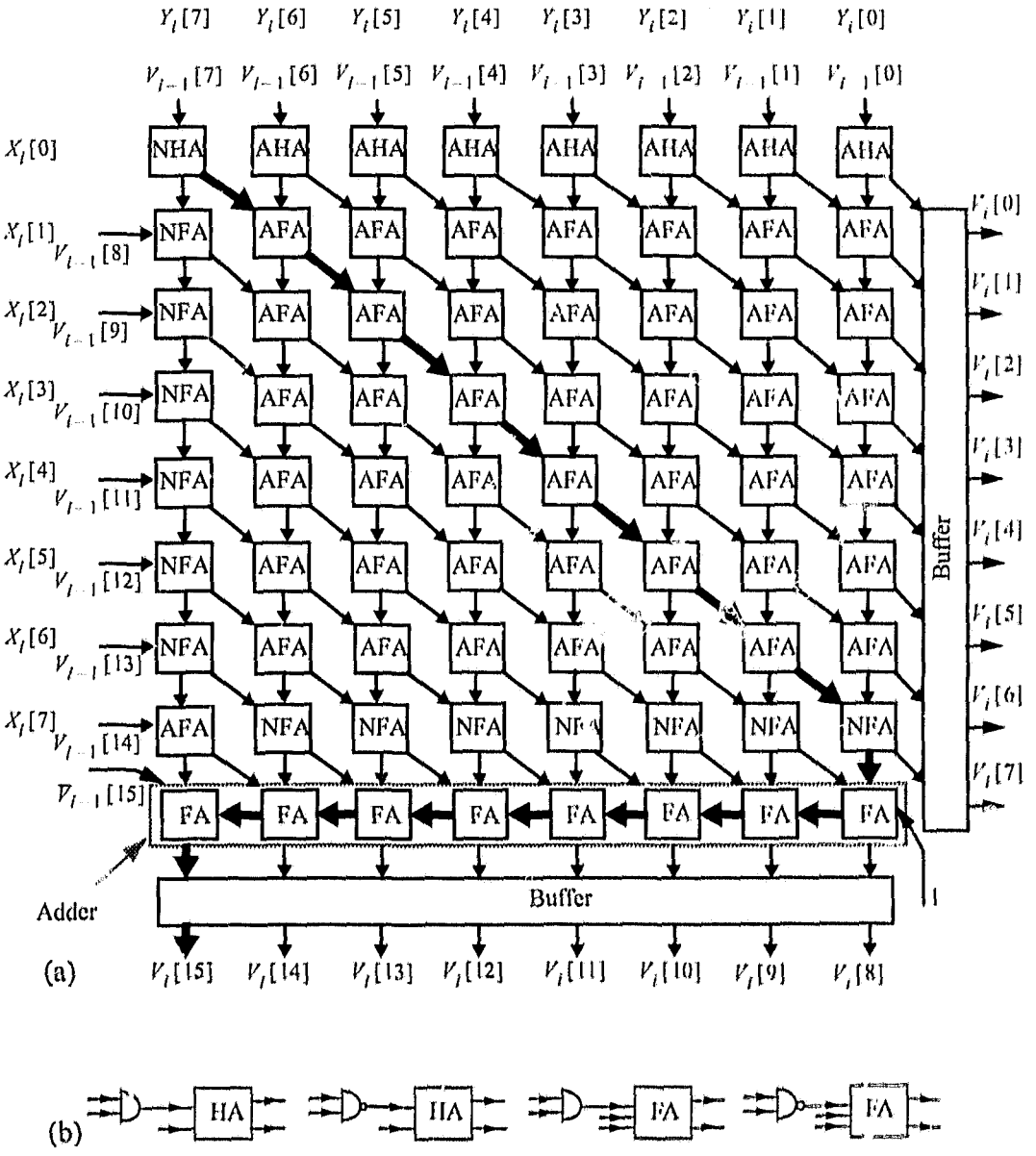


Figure 5.2: Accumulator Multiplier (ACM) [6]. (a) 8×8 ACM. (b) Details of the blocks AHA, NHA, AFA, NFA cells.

5.3 Carry-Save Inner-Product Processor

The schematic diagram of the inner-product processor proposed in this chapter is shown in Fig. 5.1(c). The details of the new inner-product processor are shown in Fig. 5.3 which is based on a modification of the ACM. The inputs for the carry-propagate adder at the bottom ($U_{i-1}[0]$,..., $U_{i-1}[b-1]$ and $O_{i-1}[0]$,..., $O_{i-1}[b-1]$) and the least significant byte (LSB) ($V_{i-1}[0]$,..., $V_{i-1}[b-1]$) of the previous output product are fedback to the corresponding binary weights. In order to achieve that feedback, one column of full adders has been added to the ACM. In this new scheme, the addition operation is performed concurrently with the multiplication operation without using the adder. This carry-propagate (or carry look-ahead) adder will be used only after all inner product steps are performed in order to get the most significant byte (MSB) of the final result ($V_M[b]$,..., $V_M[2b-1]$). To initiate the computation of the inner-product, the output registers are reset and the first multiplication is performed and the results ($V_1[0]$,..., $V_1[b-1]$,$U_1[0]$,..., $U_1[b-1]$) and ($O_1[0]$,... $O_1[b-1]$) stored in the output registers (buffers). At the next clock cycle, the bits of the new two operands are input to the multiplier along with the bits stored in the output registers. Such a process is repeated M times (M is the length of the inner-product operation). An extra cycle is needed to get the b-MSB of the final result ($V_M[b]$,..., $V_M[2b-1]$) by adding the bits $U_M[0]$,..., $U_M[b-1]$ and $O_M[0]$,..., $O_M[b-1]$ through the carry-propagate (or carry look-ahead) adder. The two operands of the new inner-product can be loaded simultaneously to the processor during this extra cycle. The scheme is still highly regular and thus very amenable for VLSI implementation. The new scheme of Fig. 5.3 is called carry-save inner-product processor (CSIPP) for obvious reasons. A circuit to detect the overflow in the final addition can be easily incorporated to the proposed scheme [6], [30].

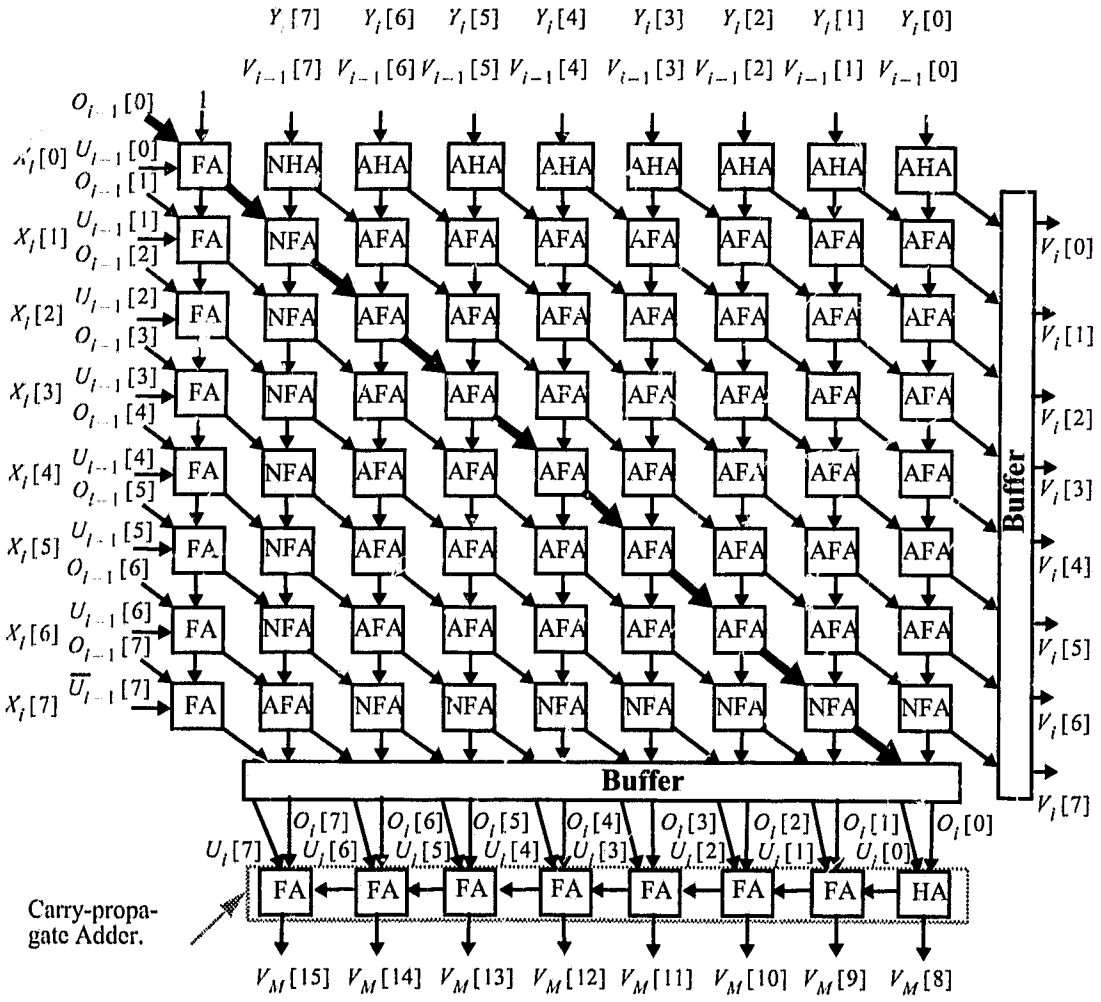


Figure 5.3: An 8×8 carry-save inner-product processor (CSIPP) [79].

5.4 Performance Analysis

In this section, a comparison of three inner product schemes is presented with respect to the following two criteria. The first criterion is the delay required for performing inner-product of length M and the second criterion is the required hardware area. The three inner-product schemes to be considered are:

1. Scheme IP-1: Conventional pipeline inner-product step processor (PIPSP).
2. Scheme IP-2: Accumulator-multiplier (ACM) [6], [74].
3. Scheme IP-3: Carry-save inner-product processor (CSIPP).

For the sake of comparison, a carry-propagate adder will be adopted for the adders in all three schemes. The array-multiplier (AM) will be adopted for the multipliers. The processor area complexity will be considered first.

5.4.1 Processor Area

Estimations for the layout areas of the three schemes in terms of the area required for gates performing basic operation such as full addition are presented. The area required for the PIPSP can be estimated as

$$A_{PIPSP} = A_{AM} + A_{Add} + 4NA_{latch} + A_{routl} \tag{5.1}$$

where A_{AM} is the area of 2's complement array multiplier, A_{Add} is the area for two's complement $2b$-bit (double-precision) addition, A_{latch} is the area for 1-bit latch and A_{routl} is the area required for routing of PIPSP scheme. The layout area for the circuits which perform 1-bit full-addition, 1-bit half-addition, 2-input And gate, 2-input Nand gate, and an inverter are denoted by A_{FA}, A_{HA}, A_A, A_{NA} and A_{inv}, respectively. Therefore, the PIPSP area A_{PIPSP} can be calculated as:

$$A_{PIPSP} = 4bA_{latch} + \left(b^2 + 1\right)A_{FA} + \left((b-1)^2 + 1\right)A_A + (b-1)A_{HA} + 2(b-1)A_{NA} + A_{inv} + A_{routl} \tag{5.2}$$

The area required for the ACM can be estimated from Fig. 5.2 as

$$A_{ACM} = 2bA_{latch} + b^2 A_{FA} + \left((b-1)^2 + 1 \right) A_A + bA_{HA} + 2(b-1)A_{NA} + A_{inv} + A_{rout2} \quad (5.3)$$

where A_{rout2} is the area required for the routing of ACM.

The area required for the carry-save inner-product processor (CSIPP) in Fig. 5.3 can be estimated as

$$A_{CSIPP} = 3bA_{latch} + \left(b^2 + b - 1 \right) A_{FA} + \left((b-1)^2 + 1 \right) A_A + (b+1) A_{HA} + (2b-2) A_{NA} + A_{inv} + A_{rout3} \quad (5.4)$$

where A_{rout3} is the area required for the routing of CSIPP.

5.4.2 Processor Delay

The time required for the pipelined M-step conventional PIPSP can be estimated as [75]

$$T_{PIPSP} = (M+1) \left[T_{latch} + \max (T_{AM}, T_{add}) \right] \quad (5.5)$$

where

T_{AM} Critical (longest) path delay of two's complement parallel array-multiplier.

T_{latch} Set-up and propagation delay time of 1-bit latch.

The delay T_{AM} of $b \times b$ two's complement AM based on the algorithm reported in [78] can be estimated as

$$T_{AM} = T_A + T_{HA} + (2b-3) T_{FA} + T_{inv} \quad (5.6)$$

where T_A, T_{HA}, T_{FA} and T_{inv} are the delay-times for standard 2-input AND gate, 1-bit half adder, 1-bit full adder and 1-bit Inverter, respectively. The delay of double-precision two's complement carry-propagate adder T_{add} is given by

$$T_{add} = 2bT_{FA} \quad (5.7)$$

From (5.6) and (5.7), T_{add} is slightly higher than T_{AM}. Therefore the time required for the conventional PIPSP can be estimated as

$$T_{PIPSP} = (M+1)\,T_{latch} + 2b\,(M+1)\,T_{FA} \tag{5.8}$$

The time required for ACM in Fig. 5.2 to perform an inner-product of length M is equal to M times the time required for one multiply-add step. The time required for one step is determined by critical path indicated by the thick arrow line in Fig. 5.2. Therefore, the time required for the ACM to perform an inner-product of length M can be estimated as:

$$\begin{aligned} T_{ACM} &= M[T_{latch} + T_{NA} + T_{HA} + (b-1)\,T_{FA} + bT_{FA} + T_{inv}] \\ &= MT_{latch} + MT_{NA} + MT_{HA} + 2M\left(b-\frac{1}{2}\right)T_{FA} + MT_{inv} \end{aligned} \tag{5.9}$$

The time required for the suggested scheme (CSIPP) in Fig. 5.3 is equal to *(M+1)* times the time required for one step. The time required for one step is determined by the critical path indicated by the thick arrow line in Fig. 5.3. Note that this longest computation path does not include the CPA. Therefore, the time required for the suggested scheme (CSIPP) in Fig. 5.3 can be estimated as

$$T_{CSIPP} = (M+1)\,[T_{latch} + bT_{FA}] \tag{5.10}$$

It can be seen that all the above terms used in the calculation of the computation time and the area are dependent on several factors such as the particular type of technology implementation and the layout style. In order to compare the area and delay performances of the three implementations, some normalization is required. All areas are normalized relative to an inverter area. Similarity, all delays are normalized relative to an inverter delay. The results are summarized in Table 5.1 based on the assumptions reported in [80], [81]:

1. $T_{NA} \approx T_{inv}$, $A_{NA} \approx 2A_{inv}$
2. $T_{A} \approx 2T_{inv}$, $A_{A} \approx 3A_{inv}$,
3. $T_{HA} \approx 5T_{inv}$, $A_{HA} \approx 10A_{inv}$
4. $T_{FA} \approx 6T_{inv}$, $A_{FA} \approx 15A_{inv}$
5. $T_{latch} \approx 4T_{inv}$, $A_{latch} \approx 12A_{inv}$

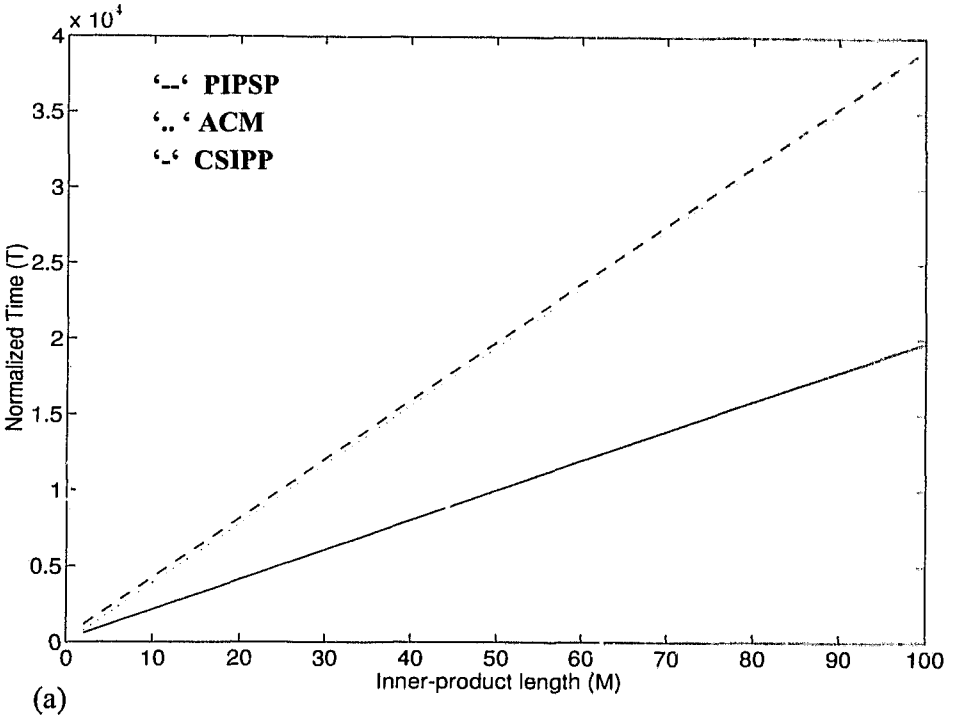
The area required for routing is also dependent on the technology used. However, because the communication in the CSIPP is relatively local compared to that of PIPSP, A_{rout3} is expected to be lower than A_{rout1}. In order to simplify the comparison, these routing areas have been neglected for the present analysis.

From Table 5.1 and Fig. 5.4, it is evident that reduction in the computation time ranging from 20 to 50 percent can be obtained by using the proposed carry-save inner-product processor with a slight increase in the required area. Therefore, the proposed scheme provides better results in terms of $area \times time$ (AT) and $area \times time^2$ (AT^2) measurements compared to the two other schemes. The AT performance comparison is shown in Fig. 5.4(c).

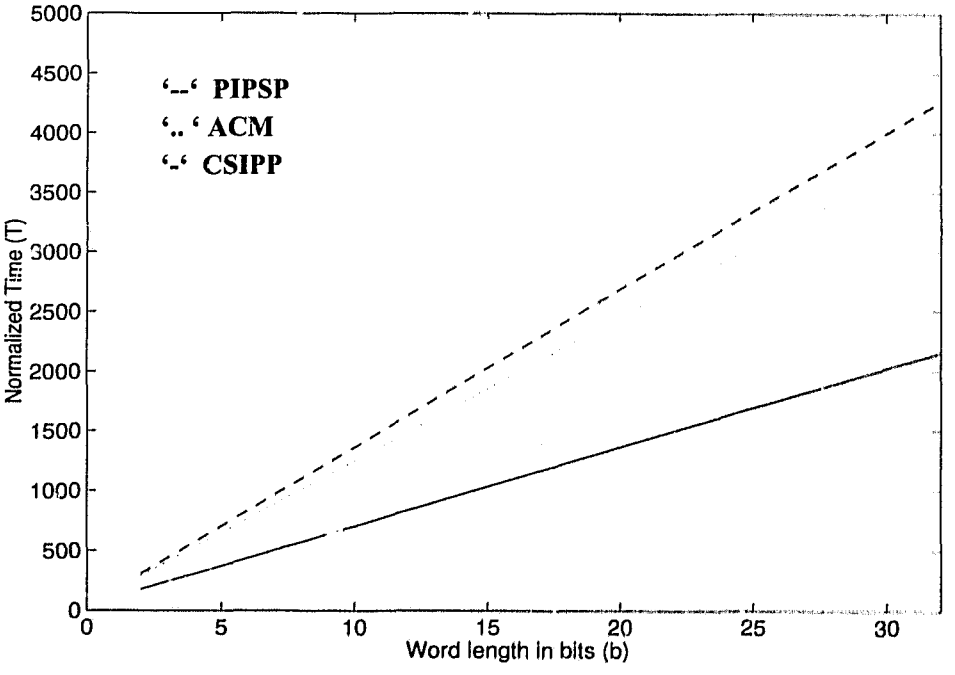
The operation of CSIPP has bee verified by simulation using Verilog and it has been fabricated using the technology of 1.2 μ CMOS [82].

Table 5.1: Area and computational delay for different inner-product schemes.

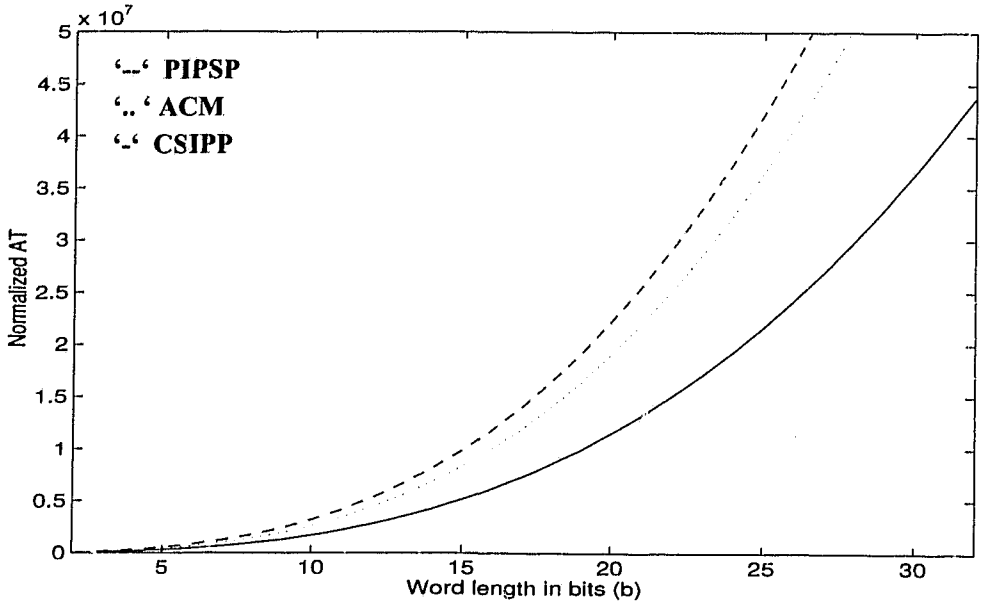
Inner-product Processor	Normalized area	Normalized delay
PIPSP	$18b^2 + 52b + 8$	$12Mb + 4M + 12b + 4$
ACM	$18b^2 + 27b + 3$	$12Mb + 5M$
CSIPP	$18b^2 + 59b - 2$	$6Mb + 4M + 6b + 4$


x 10^4
'--' PIPSP
'..' ACM
'-' CSIPP
Normalized Time (T)
Inner-product length (M)
0
0.5
1
1.5
2
2.5
3
3.5
4
0
10
20
30
40
50
60
70
80
90
100

(a)


'--' PIPSP
'..' ACM
'-' CSIPP
Normalized Time (T)
Word length in bits (b)
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
0
5
10
15
20
25
30

(b)



(c)

Figure 5.4: Delay and AT performances for the three schemes: PIPSP ACM and CSIPP. (a) Normalized computation time performance where b=32. (b) Normalized computation time performance where M=10. (c) Normalized AT performance where M=10.

5.5 Conclusion

In this chapter, a new scheme for area-time efficient two's complement fixed-point inner-product processor is presented. This scheme provides better performance compared to existing inner-product schemes in terms of computation speed, $area \times time$, and $area \times time^2$ performances. The proposed inner-product processor has a high computational speed which is double (for long inner-product operation) the speed of the conventional pipelined inner-product. The new processor is well suited to any DSP application requiring computation of inner-products and is exceptionally well suited to VLSI implementation.

Chapter 6

Implementation of Full-Matrix State-Space IIR Digital Filter

6.1 Introduction

Any realization for IIR digital filters requires feedback loops to realize the recursion. These loops in the case of conventional binary arithmetic imposes an upper bound on the system's throughput rate. This is referred to as the bottleneck barrier [72] in the literature. In the case of full-matrix state-space IIR digital filter, the maximum throughput rate is limited by the delay required for one multiplication and one N-operand addition. This maximum throughput rate can be achieved if a multiplier is assigned to each state-space matrix coefficient. This assumption has been adopted by the majority of the researchers and leads to expensive hardware implementations especially when considering implementing a high-order filter as one section with full state matrices. As an example, a 16-bit full-matrix state-space filter of order 10 approximately requires 1M transistors. Such a chip would be of comparable complexity to some current 32-bit microprocessors [2]. This high complexity is the reason behind the preference to implement IIR digital filters using direct structures in spite of their bad performance due to the finite wordlength effects [5], [6]. One way to obtain a feasible VLSI implementation for one-section state-space filter is by reducing the computational complexity through zeroing some of the state-space matrices coefficients using suitable transformations. This solution has been

adopted in Chapter 3 and it leads to the high-speed VLSI array-processor implementation proposed in Chapter 4. Another way to make the VLSI implementation feasible is to map the large number of computations onto a few number of processor elements and thus reducing the hardware complexity.

In this chapter, a new systolic array implementation for a one-section Nth-order filter with full state space matrices is proposed. The number of the processor elements is proportional to the filter order N. This makes it feasible to implement the full-matrix state-space structure of IIR digital filters using VLSI. The mapping of the state-space filter structure onto a systolic array is done in such a way as to make the computations for each state locally executed. The novel CSIPP presented in Chapter 5 can then be used for the local computations and thus enhance the computation speed. A performance comparison between the proposed implementation and two other implementations (including the one proposed in Chapter 4) is also presented.

6.2 The Proposed Systolic Architecture

The behavior of a general Nth-order one-section state-space digital filter can be described by equations in (1.5) and (1.6) where the matrices $\{\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}\}$ are full.

The method for developing a systolic design involves two steps [52]. In the first step, a Dependance Graph (DG) is obtained to indicate the dependencies between different variables and computations. A Signal Flow Graph (SFG) is developed in the second step by selecting an appropriate scheduling vector and a projection vector on the index set. The computations required for implementing the Nth-order state-space digital filter are described by (1.5) and (1.6). It can be seen from (1.5) and (1.6) that the required computations are iterative. Each iteration requires a new input and the updated states. It is possible to study the present iterative algorithm using the concepts presented in [55]

where a Reduced Dependance Graph (RDG) may be obtained to provide an abstraction for the involved computations. The RDG for the required multiplication and addition steps is shown in Fig. 6.1. The input sample $u(n)$ and the states $x_1(n)$, $x_2(n)$,....., $x_N(n)$ are propagated along the j-axis of the RDG. The states $\boldsymbol{x}(n)$ and the input sample $u(n)$ are weighted by different matrix elements and added (accumulated) to get output sample $y(n)$ and the new states $\boldsymbol{x}(n+1)$.

The concepts of schedule and projection vectors explained in [52] will be used to obtain a valid schedule and processor configurations. One suitable and obvious choice for the scheduling vector is $\boldsymbol{s}^t$=[1,1]. Such a schedule gives rise to equi-temporal hyperplanes. All computations along the equi-temporal hyperplanes (i-j=constant) are performed at the same clock cycle. However, they are performed at different locations by selecting an appropriate projection vector which determines the locations of the computations. The projection vector is selected to lie along the i-axis, i.e., $\boldsymbol{p}^t$=[1,0]. That choice of both the scheduling and projection vectors produces a systolic structure in which all the required double-precision operations are locally performed inside each processor element (using inner-product processors). The resulting SFG is shown in Fig. 6.2(a). Processor elements PE_1:PE_N are assigned to calculate the states x_1:x_N, respectively. Processor element PE_0 is assigned to calculate the output. That particular choice of the projection vector assigns all the state-space feedback connections and the input to a single processor element PE_0. The feedback line carries the N states to PE_0 at different instants of time.

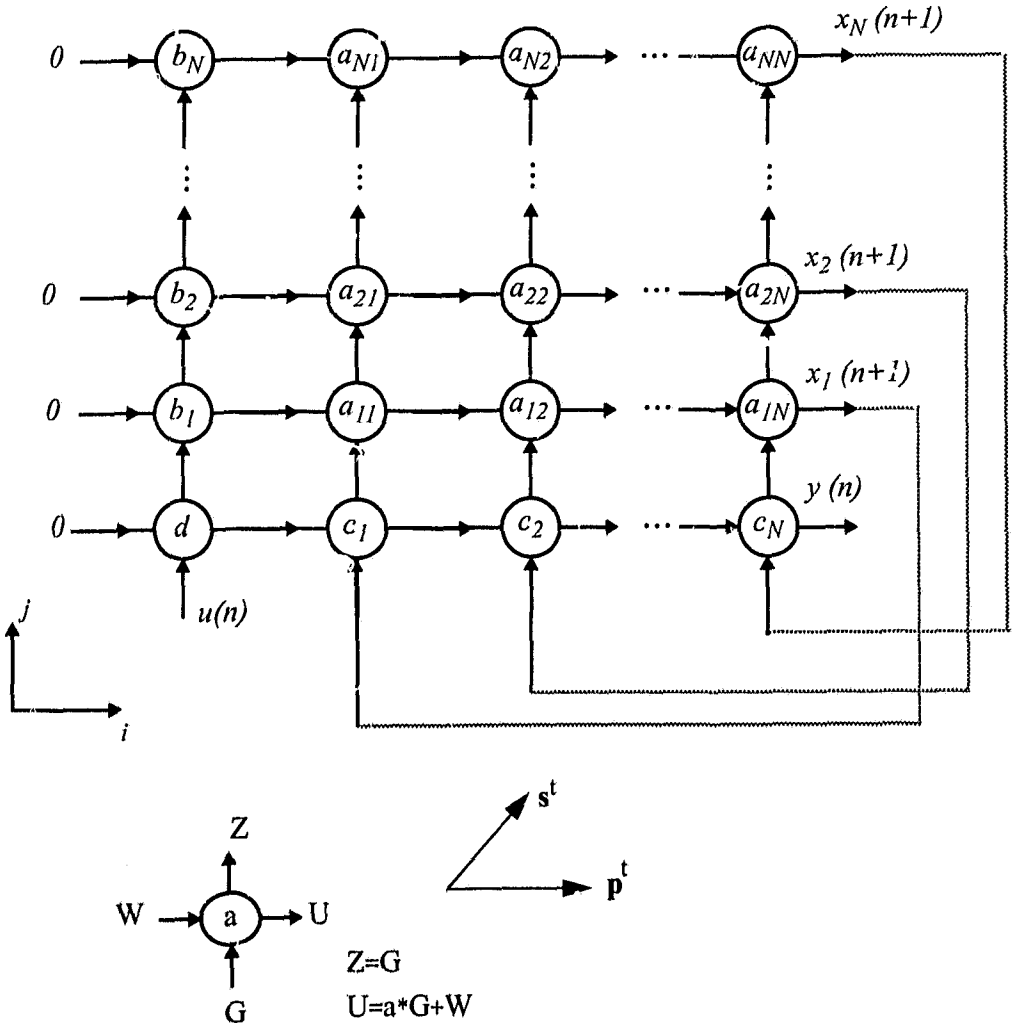
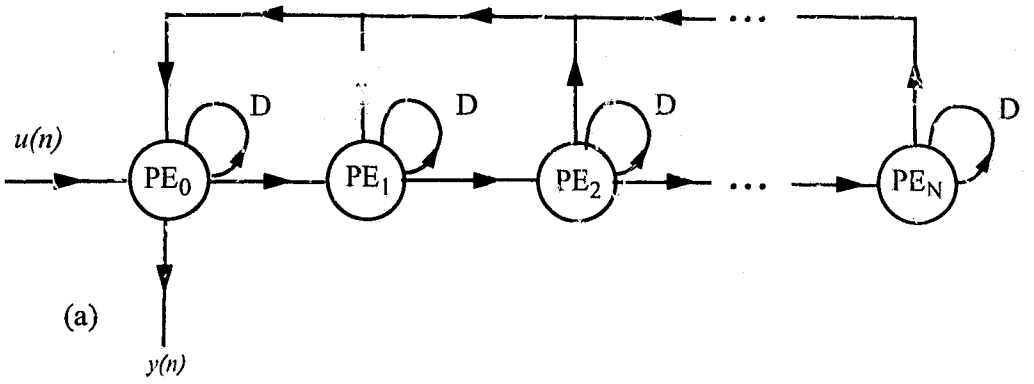


Figure 6.1: Reduced dependence graph of Nth-order state-space digital filter.



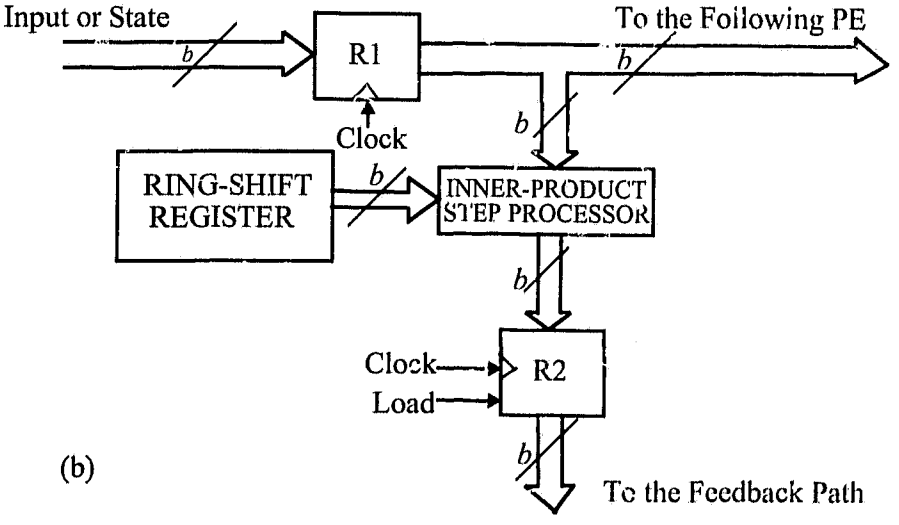


Figure 6.2: The systolic implementation of the Nth-order state-space IIR digital filter. (a) Signal flow graph of Nth-order state-space digital filter. (b) Internal details of the PE.

6.2.1 Internal Organization of The Processor Elements

Each PE in Fig. 6.2(a) basically performs an inner product operation. The internal structure of a PE is shown in Fig. 6.2(b). Each PE contains two single-precision registers (of length b bits where b is assumed to be the input and coefficient wordlengths), one two's complement inner product step processor (IPSP), a ring-shift register of size $(N+1) \times b$. The ring-shift register is $(N+1)$ words long and stores a row of the system matrix A. Register R1 stores either a new state sample or the input sample. Register R2 stores the output of the inner-product processor and delivers the result to the feedback path. Each processor is responsible for calculating either the present output or one component of the next state vector using double-precision accuracy. The proposed systolic implementation has the following features:

- All operations are locally performed inside each processor unit using double-precision arithmetic.
- All data communication between processors are single-precision which results in decreasing the communication overhead.
- The main operation inside each processor element is inner-product operation which can be executed by using the novel high-speed inner-product processors presented in Chapter 5.
- Rounding operations are performed after the double-precision inner-product operations which results in high S/N ration without increasing computation delay, layout area or communication overhead.

Any available inner-product processor can be used inside each PE to perform the required inner-product operations. It can be seen from Fig. 6.2(b) that the inner-product processors are the main elements of the proposed implementation. Therefore, the area and

the computation delay of the involved inner-product processor will determine the area and the computation delay of the whole implementation in Fig. 6.2(a). Therefore, the incorporation of the proposed CSIPP of Chapter 5 will approximately double the speed of filter without a significant increase in area.

6.2.2 Data-flow and Timing

In order to maintain the synchronization of the input data and the updated states, CSIPP requires one extra cycle in order to produce the MSB of the final output. This extra cycle can be easily accommodated in the implementation of Fig. 6.2(a) by augmenting each ring shift-register in Fig. 6.2(b) by one zero-coefficient. In order to show the timing operations of the systolic architecture in Fig. 6.2, the input $u(n)$ is assumed to be accepted by PE_0 at instants $n=(N+2)m$ for $m=0,1,....$ The output $y(n)$ is produced at $n=(N+2)(m+1)$ for $m=0,1,....$ The kth state $\{x_k(0), x_k(1),...\}$ *(i.e., the* k*th component of the state vector* $\mathbf{x}$*)* is computed at instants $n=(N+2)(m+1)+k$, for $m=0,1,...,$ and $k=1,...,N$. These states are fedback to the structure. Register R1 of each PE is loaded at every clock cycle. Register R2 delivers the result (the output of the inner-product processor) to the feedback path in synchronism with a load strobe at every $(N+2)$ clock cycles. This arrangement makes the kth state to be only delivered to the feed-back path at the appropriate clock cycle. The distribution and the flow of the coefficients in the ring shift-register of each PE are demonstrated in Table 6.1.

6.3 Performance Analysis

In this section, a complexity analysis is presented for the layout area and the computational delay of the proposed systolic architecture in Fig. 6.2

Table 6.1: Timing flow of coefficients, $N=4$.

Time slot	PE_0	PE_1	PE_2	PE_3	PE_4
0	d	-	-	-	-
1	c_1	b_1	-	-	-
2	c_2	a_{11}	b_2	-	-
3	c_3	a_{12}	a_{21}	b_3	-
4	c_4	a_{13}	a_{22}	a_{31}	b_4
5	0	a_{14}	a_{23}	a_{32}	a_{41}
6	d	0	a_{24}	a_{33}	a_{42}
7	c_1	b_1	0	a_{34}	a_{43}
8	c_2	a_{11}	b_2	0	a_{44}
9	c_3	a_{12}	a_{21}	b_3	0
10	c_4	a_{13}	a_{22}	a_{31}	b_4
11	0	a_{14}	a_{23}	a_{32}	a_{41}
12	d	0	a_{24}	a_{33}	a_{42}
13	c_1	b_1	0	a_{34}	a_{43}
14	c_2	a_{11}	b_2	0	a_{44}
-	-	-	-	-	-

6.3.1 Area Complexity

The area complexity of the proposed implementation is mainly due to the area of the CSIPP's and the associated storage elements. Therefore, the area required for the proposed implementation A_{FSYST} can be expressed as

$$A_{FSYST} = (N+1)A_{CSIPP} + 2(N+1)A_R + (N+1)(N+2)A_R \qquad (6.1)$$

where A_R is the area required for a single-precision register. By substituting (5.4) in (6.1), A_{FSYST} can be estimated as

$$A_{FSYST} = (N+1)\left[\left(b^2-2b+2\right)A_A + 2(b-1)A_{NA} + (b+1)A_{HA}\right] + (N+1)\left[\left(b^2+b-1\right)A_{FA} + (N+7)bA_{latch}\right] \quad (6.2)$$

The routing area for the systolic structure is negligible since most of the interconnections are local.

6.3.2 Computational Delay

The delay required for each output sample for the proposed architecture in Fig. 6.2 can be expressed as

$$T_{FSYST} = (N+2)T_{CSIPP} \quad (6.3)$$

The time required for the computation of one CSIPP operation is calculated in (5.10) (the inner product length M in (5.10) is equal to $N+1$). Substituting (5.10) in (6.3) leads to

$$T_{FSYST} = (N+2)(bT_{FA} + T_{latch}) \quad (6.4)$$

6.3.3 Latency

The latency l_{FSYST} of the proposed implementation is given by

$$l_{FSYST} = T_{FSYST} = (N+2)T_{CSIPP} \quad (6.5)$$

6.4 Comparison with Other State-Space Architectures

In this section, a comparison of the proposed architecture with other state-space architectures for IIR digital filters is presented. The first architecture is the parallel nonsystolic architecture shown in Fig. 6.3. The second architecture is the VLSI architecture based on SRF scheme 2 realization presented in Section 4.3. This second implementation is called SRF systolic implementation.

Parallel architecture:

The parallel architecture of Fig. 6.3 requires $(N+1)^2$ array-multipliers, $(N+1)$ double-precision multi-operand adders, N single-precision registers for storing the states and $(N+1)^2$ single-precision registers for storing the digital filter coefficients. The multi-operand addition is implemented using υ 3-input-operand Carry-Save-Adder (CSA) units arranged in κ levels and one Carry-Propagation Adder (CPA) unit [80]. So, the area required by the parallel architecture can be estimated as

$$A_{PARAL} = (N+1)^2 A_{AM} + (N+1) A_{mult-oper} + NA_R + (N+1)^2 A_R \tag{6.6}$$

$A_{mult-oper}$ is the area required for one multi-operand addition. The area required for the parallel architecture can be expressed as

$$A_{PARAL} = (N+1)^2 \left[\left(b^2-2b+2\right)A_A + 2(b-1)A_{NA} + A_{inv}\right] + (N+1)^2 \left[(b-1)A_{HA} + \left(b^2-2b+1\right)A_{FA}\right] + (N+1)(\upsilon+1)bA_{FA} + \left[\upsilon(N+1) + N + (N+1)^2\right]bA_{latch} \tag{6.7}$$

The routing area required for the parallel implementation is significant since the communications between blocks are not local. However, to simplify the comparison this routing area will not be taken into consideration.

The computation delay required for the parallel structure will be estimated noticing that this non-systolic parallel state-space architecture has the same bottle-neck delay bound as all full state-space architectures when assigning a multiplier for each coefficient of the state-space matrices. The longest computation path consists of one array-multiplier and one multi-operand addition. The CSA does not require carry propagation across the wordlength of an operand [80]. Therefore, the computation delay for each output sample

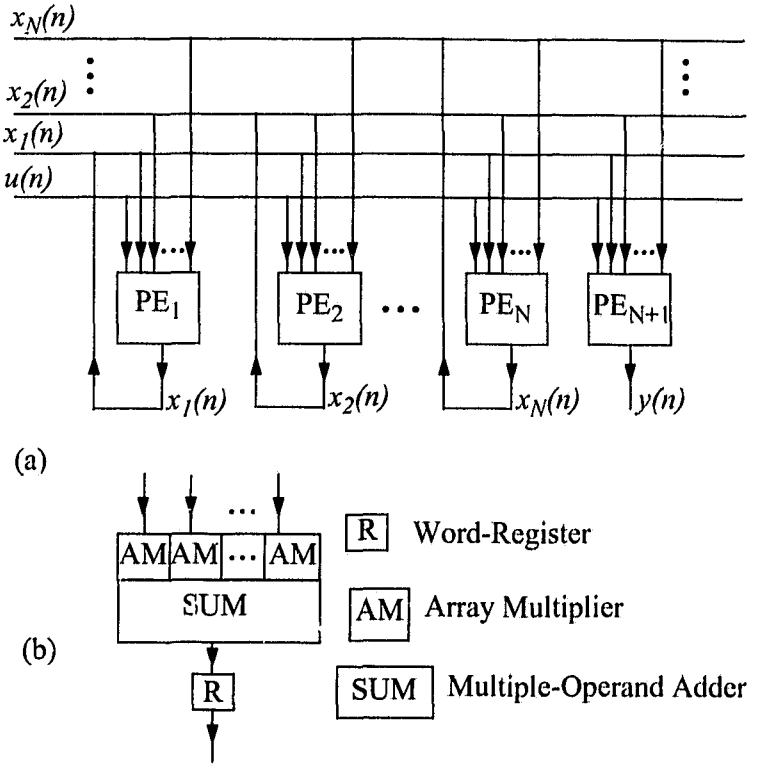


Figure 6.3: The parallel nonsystolic implementation. (a) Parallel architecture of Nth-order state-space digital filter. (b) Details of a processor elements (PE) involved.

for the parallel architecture is given by

$$T_{PARAL} = T_{mult} + T_{mult-oper}$$
$$= T_A + T_{HA} + T_{inv} + (4b + \kappa - 3)\,T_{FA} + \kappa T_{latch} \quad (6.8)$$

SRF systolic implementation proposed in Chapter 4

The area complexity of the SRF systolic implementation presented in Section 4.3 is discussed in Subsection 4.3.3.1. This area A_{RFSYST} can be expressed as

$$A_{RFSYST} = \left(b^2 - 2b + 2\right)\left(\frac{N^2}{2} + 3N + 1\right)A_A + (2b-2)\left(\frac{N^2}{2} + 3N + 1\right)A_{NA} +$$
$$\left(\frac{N^2}{2} + 3N + 1\right)A_{inv} + \left(2Nb + (b-1)\left(\frac{N^2}{2} + N + 1\right)\right)A_{HA} +$$
$$\left(2Nb^2 + (b-1)^2\left(\frac{N^2}{2} + N + 1\right) + 2Nb\,(N+6)\right)A_{FA} +$$
$$\frac{1}{4}\left(11N^2 b + 59Nb + 4b\right)A_{latch} \quad (6.9)$$

The routing area for the SRF systolic implementation can be neglected since the implementation is fully pipelined. The computational delay of the VLSI implementation in Section 4.3 is calculated in (4.5). This computational delay can be expressed as

$$T_{RFSYST} = T_{ACM} + T_{3-oper}$$
$$= T_{latch} + T_{NA} + T_{HA} + T_{inv} + (6b-1)\,T_{FA} \quad (6.10)$$

Discussion

It can be seen that all the above terms used in the estimation of areas and delays are dependent on several factors such as the particular type of technology implementation and the layout style. In order to compare the performance of the implementations, a normalization process similar to that in Section 5.4 is carried out; i.e., all areas are normalized relative to area of an inverter A_{inv}. Similarity, all delays are normalized relative to

delay of an inverter T_{inv}. The results are summarized in Table 6.2 based on the assumptions discussed in details in Section 5.4

Table 6.2: Normalized area and computational delay of different state-space implementations.

	$A = Area/A_{inv}$	$T = Delay/T_{inv}$
Parallel implem.	$(N+1)^2[18b^2-22b+8]+$ $27(N+1)vb+\left(12N^2+51N+27\right)b$	$24b+10\kappa-10$
Full-matrix Systolic implem.	$(N+1)\left[18b^2+107b+8bN-3\right]$	$6Nb+4N+12b+8$
SRF systolic implem.	$(N+1)^2\left(9b^2+11b+4\right)+$ $b^2(21N+9)+Nb(667-N/4)/4-$ $32b-6N+4$	$24b+15$

Figure 6.4 show the normalized computational delay, normalized area and normalized AT performances of the three implementations for $b = 16$. Figure 6.5 show the normalized computational delay, normalized area, and normalized AT performances of the three implementations for $N = 12$. It can be seen that the non-systolic parallel implementation provides neither the lowest delay nor the lowest area. From Fig. 6.4(a) and Fig. 6.5(a), it can be seen that the SRF systolic implementation provides the lowest computational delay which is independent of the filter order N. Although the non-systolic implementation provides a delay which is close to that of SRF systolic implementation, this delay increases with the filter order and the implementation requires much larger area. From Fig. 6.4(b) and Fig. 6.5(b), it is apparent that the full-matrix systolic implementation requires much less area compared to the other implementations which indicates that it is more feasible for hardware implementation. From Fig. 6.4(c) and Fig. 6.5(c), it can be also seen that the full-matrix systolic implementation provides the best $area \times time$

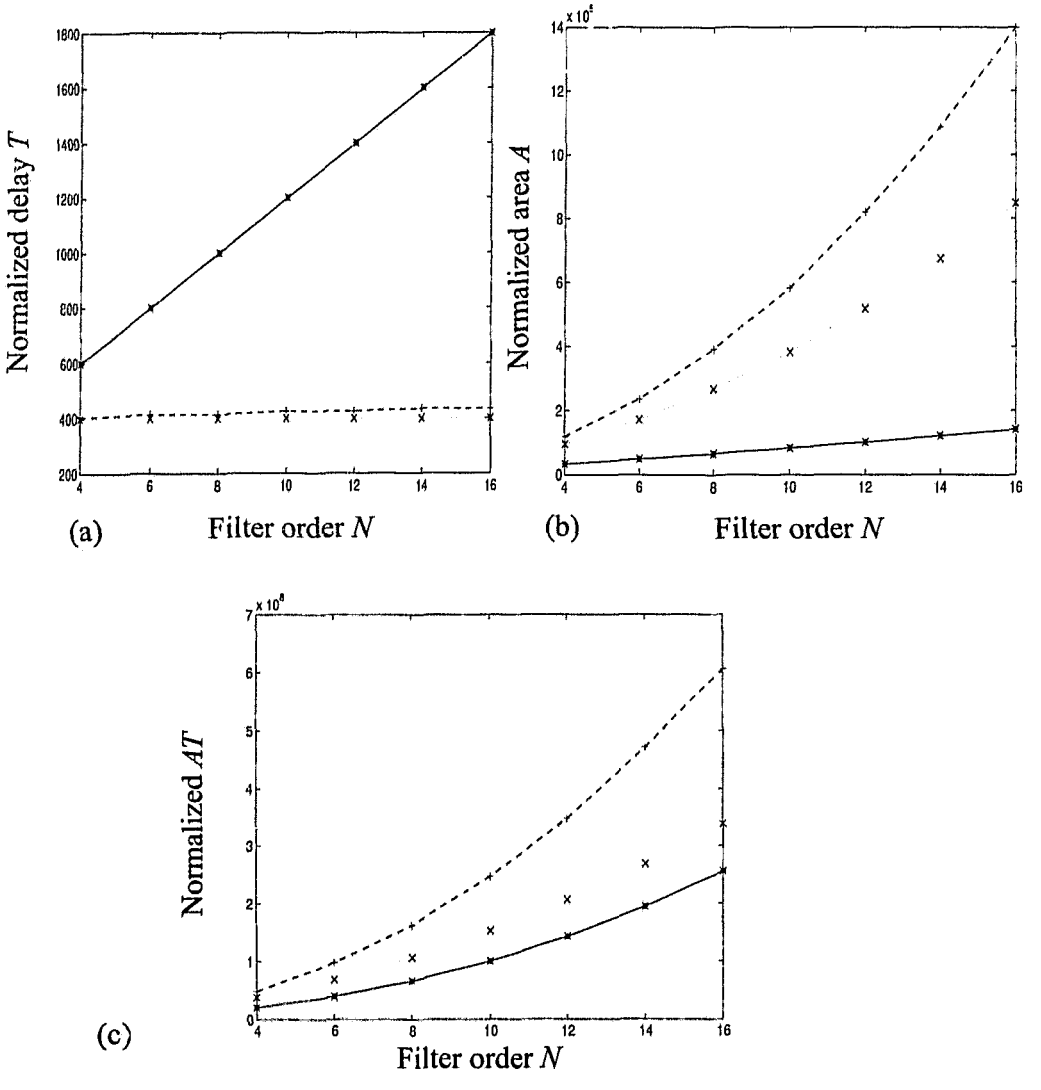


Figure 6.4: Performance of three different implementations for state-space IIR digital filter (b=16). (a) Normalized computational delay T. (b) Normalized area complexity A. (c) Normalized AT performance. '-' full-matrix systolic implementation, '..' SRF systolic implementation, and '--' non-systolic parallel implementation

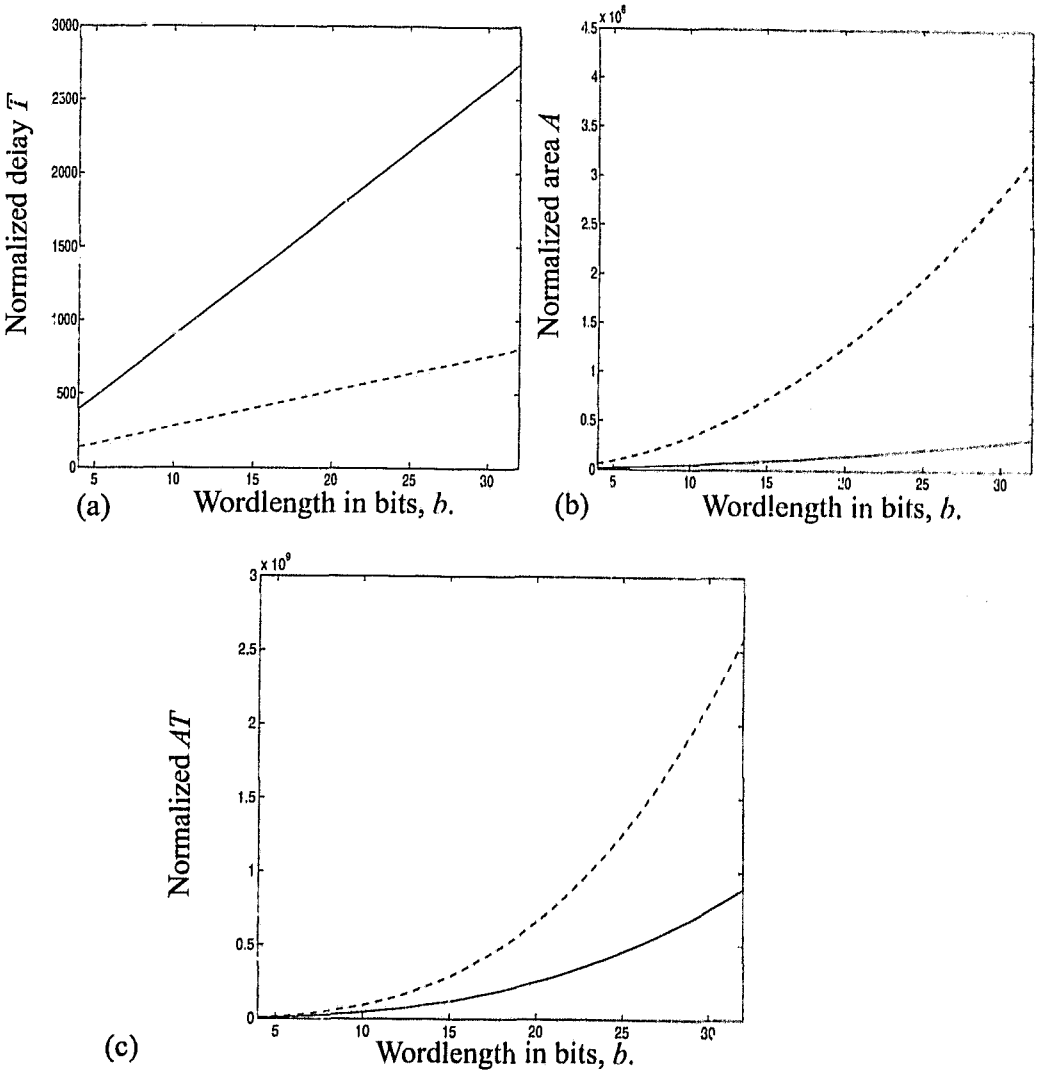


Figure 6.5: Performance of three different implementations for state-space IIR digital filter ($N=12$). (a) Normalized computational delay T. (b) Normalized area complexity A. (c) Normalized AT performance. '-' full-matrix systolic implementation, '..' SRF systolic implementation, and '--' non-systolic parallel implementation

performance compared with the other two implementations.

6.5 Comparison with Direct Implementation

There are many implementations of IIR digital filters which are based on the standard direct form. Examples of these implementations can be found in [83-85]. One systolic implementation which provides excellent performance in terms of computation delay and area is reported in [83]. In this section, the hardware performances of the two implementations proposed in Section 4.3 and in Section 6.2 (which are based on the state-space realization) are compared with the implementation in [83] (which is based on direct realization). The performance analysis in terms of computation delay, latency, area and the involved operations is summarized in Table 6.3. In order to compare the hardware performance of the three implementations, the performance of the corresponding realizations in terms of frequency response and output roundoff noise should be similar [86]. This requires using different wordlengths for each implementation (i.e., different values of b). The determination of b for each realization depends on the filter specifications. Therefore, a general comparison of the three implementations can not be carried out. This comparison can be only done for specific filter examples.

In this section, the sixth-order elliptic narrow-band LPF in Table 3.2 will be taken as a typical example for the sake of comparison. The l_2-scaled direct realization of the LPF elliptic filter requires 25-bit coefficient wordlength to approximately fit the required frequency response while the SRF scheme 2 realizations proposed in Chapter 3 (on which the SRF systolic implementation in Section 4.3 is based on) and MRH realization (on which the proposed implementation in this chapter is based on) require only 12-bit coefficient wordlength [69].

Table 6.3: Performance analysis of three different implementations.

	The full-matrix systolic Implementation	The RF systolic implementation (proposed in Chapter 4)	The direct implementation in [83]
multipliers	$N+1$	$\frac{N^2}{2}+3N+1$	$2N$
double-precision adders	-	$\frac{N^2}{2}+3N$	$2N$
single-precision latches	$(N+1)\ (N+3)$	$\frac{3}{4}N^2+15\frac{N}{4}+1$	$3N$
double-precision latches	-	$\frac{N}{2}(N+1)$	N
Computational Delay	$(N+2)\,T_{CSIPP}$	$T_{mult}+T_{3-oper}$	$T_{mult}+T_{3-oper}$
Latency	$(N+2)\,T_{CSIPP}$	$N(T_{mult}+T_{3-oper})$	$\frac{N}{2}(T_{mult}+T_{3-oper})$

The l_2-scaled direct realization provides noise gain of approximately 5×10^{11} while SRF scheme 2 realization for the same filter provides noise gain of only 1.054 (the roundoff noise generated from the output node has been taken into consideration) and the MRH realization provides noise gain of 2.387 (the roundoff noise generated from the output node has been taken into consideration.) In order to equate the output roundoff noise of the three realizations (i.e., the same signal to noise ratio,) the direct structure requires 13 bits for the internal states wordlengths more than what is required for the other two realizations.[4] By choosing the wordlength for the filter states in the state-space realiza-

4. If the wordlength of the output samples can be taken different than the wordlength of the internal states, SRF scheme 2 (and consequently the SRF systolic implementation) can use shorter wordlengths for the internal states compared to the full systolic implementation. This will further improve the hardware performance of the SRF systolic implementation.

tions to be 12 bits implies that the wordlength for the states of the scaled direct realization should be equal 25 bits. Therefore, the wordlength for both the coefficients and the states in the proposed implementations (SRF systolic and Full-matrix systolic implementations) will be taken 12 bits (i.e., $b = 12$) while the wordlengths for both the coefficients and the states of the direct implementation in [83] will be taken as 25 bits (i.e., $b = 25$).

By following the normalization process discussed in Section 5.4, the performance analysis of the three implementations in terms of area A, computational delay T, $area \times time$ (AT), $area \times time^2$ (AT^2) and latency are summarized in Table 6.4. It can be seen from Table 6.4 that the SRF systolic implementation proposed in Section 4.3 provides better delay, area, and AT performance compared to the direct implementation in [83] for this specific example. As the computational delay of the SRF systolic implementation is dependent only on the wordlength b, it is guaranteed that this implementation provides better computational delay compared to the direct implementation in [83] for any narrow-band filter. However, it should be noted that increasing the filter order will quadratically increase the area required for the SRF implementation and may degrade its area and the AT performance.

The full-matrix systolic implementation proposed in Section 6.2 provides a computational delay close to that of the direct implementation with much smaller area for this specific LPF. The full-matrix implementation provides the best AT performance which is much lower than that of the direct implementation. Therefore, the full-matrix implementation presented in this chapter provides the best compromise between the required area and the computational delay, for this specific narrow-band LPF example.

From Table 6.4, it is also obvious that both proposed state-space implementations provide better performance in the AT^2 sense compared to the direct implementation in

[83], for this specific LPF example.

Table 6.4: Performance analysis of three different implementation for the sixth-order elliptic LPF filter.

	The full-matrix systolic implementation	The SRF systolic implementation (proposed in Chapter 4)	The implementation of [83]
Wordlength in bits, b	12	12	25
Normalized area, A	31, 143	101, 171	128, 676
Normalized delay, T	608	303	600
Normalized $AT \times 10^{6}$	18.93	30.65	77.21
Normalized $AT^{2} \times 10^{10}$	1.15	0.92	4.63
Normalized latency	608	1818	1800

It should be noted that the computational delays of the architecture presented in this chapter and Chapter 4 are limited by the computational delays inside the feedback paths. So, the increase in the computational delay of any of these architectures is dependant on the performance of the elementary operators such as the adder and multiplier; i.e., the computational delay is dependent on the advance in technology which seems to be relatively limited. In the next chapter, new implementations will be presented which can provide lower computational delays (i.e., higher input frequency) even using slow processing elements

6.6 Conclusion

An efficient systolic implementation for Nth-order IIR digital filter with full state-space matrices has been proposed. The new architecture is amenable to VLSI implementation because the number of the required processor elements linearly increases with the filter order. The performance of the proposed implementation has been enhanced by basing the PE design on the CSIPP proposed in Chapter 5. MRH state-space structure of Nth-order IIR filters (with its desirable features) can now be implemented by the proposed architecture and can provide better performance in the $area \times time$ sense compared to other existing state-space implementations. It has also been shown that the proposed implementation may provide better performance in terms of $area \times time$ and $area \times time^2$ compared to existing direct implementation for narrow-band filters. Therefore, the proposed implementation may provide the best compromise between the required complexity area and the computational delay for narrow-band IIR digital filters.

Chapter 7

VLSI Array Processor Implementation of Block-State IIR Digital Filters

7.1 Introduction

The maximum computation speeds which can be achieved by the implementations of IIR digital filters proposed in Chapter 6 and in Chapter 4 are estimated in (6.4) and in (6.10), respectively. The only possibility to improve the computational speed of such implementation is to improve the computational speed of the involved processing elements which is VLSI technology dependent. In order to have implementations with speeds not limited by the processor unit speed, various approaches have been proposed in the literature. Some of these speed-up techniques are pipelining techniques [87], [88], block techniques [58], [59], [89], multi-path techniques [90], or by a combination of two of these techniques [91]. By using any of these speed-up techniques, the input sampling rates can be much higher than that of a single "arithmetic unit" or "processing element". The lower iteration period bound $\underline{T}$, which is a low limit for the input sampling period, is given by [72]

$$\underline{T} = \frac{T_L}{L\Omega} \tag{7.1}$$

where T_L represents the computation delay of each loop, Ω represents the number of

latches inside each loop and L represents the input block size length. In (7.1) equal latency and equal number of latches inside each loop has been assumed. The case of the state-space filter where $L = 1$ and $\Omega = 1$ is called conventional state-space filter without pipelining (e.g. the implementations proposed in Chapter 4 and Chapter 6). Pipeline techniques ($\Omega > 1$) have been successfully applied to direct-form structures, as in [5], but the utility of these techniques is doubtful for the state-space structures because of the complexity of the implementations. Moreover, the maximum speed-up factor that can be achieved by using pipeline techniques is limited by the largest number of stages that a multiplier can be broken into. On the other hand, the only limitation on the speed-up factor that can be achieved by using block techniques ($L > 1$) is technology available [58], [59], [89]. Theoretically, any high input sampling rate can be achieved by increasing the input block size used at the expense of hardware complexity. The reason for the increased complexity is that in order to obtain the maximum throughput rate, a multiplier is typically assigned to each state-space matrix entry [73], [92], [93]. Table 7.1 shows estimations of the hardware complexity in terms of the number of multipliers required for the conventional direct form structure, conventional state-space structure [28] and three state-space realizations using three different block techniques [58], [91]. All the state-space structures in Table 7.1 are assumed to have full state-space matrices. It can be seen from Table 7.1 that the number of multipliers required for all the state-space structures using block techniques are considerably larger than those required for both the direct and conventional state-space structures (under the assumption of assigning a multiplier for each state-space matrix entry). Given the limitations of the present VLSI technology, hardware implementations of state-space filters based on block description would require a massive amount of VLSI chips. Therefore, any hardware implementation for block realization should strive to reduce the number of involved multipliers.

In this chapter, two new array processor implementations will be proposed for IIR

digital filters with high input sampling rates which are not limited by the speed of the involved processing element. The first implementation is based on block-state description [58], [59] in which the state-update matrix is full corresponding to implementing the corresponding conventional state-space filter as one section. The other implementation is also based on the block-state description [58], [59] in which the state-update matrix is block-diagonal corresponding to the case of parallel combination of second-order sections.

Table 7.1: Number of multiplies required for direct structure and different state-space realizations.

Implementation	Number of multipliers [a]
Direct	$2N+1$
SISO state-space [28]	$(N+1)^2$
Block-State [58], [59]	$\left[N^2+2LN+L\frac{(L+1)}{2}\right]$
Parallel Block-State PBS [89]	$L\left(N^2+LN+N+\frac{(L+1)}{2}\right)$
Incremental Block State [b] IBS [91]	$L\left\{3N+\frac{\left(2N^2+[[\sqrt{2}N]]^2\right)}{2[[\sqrt{2}N]]}+\frac{1}{2}\right\}$

a. For state-space structures, a multiplier has been assumed to be assigned for each matrix entry.
b. $[[x]]$ denoted the nearest integer to x and L is assumed to be high (i.e., asymptotic complexity)

7.2 Block-State IIR Filter Description

In this section, the description of the block-state space IIR filter algorithm and a general framework for obtaining the required array processor implementation are presented.

7.2.1 Block-state IIR Digital Filter Algorithm

In the block-state description of the IIR digital filter, block of output samples $y(kLT_s)$,

$y((kL+1)T_s),\ldots, y((kL+L-1)T_s)$, is computed based on the single state vector $\boldsymbol{x}(kLT_s)$ and the corresponding L inputs, where T_s is the input sampling time. The block-state update equation and the block-output equation can be described as [59]

$$\boldsymbol{x}((k+1)LT_s) = \boldsymbol{A}^{(L)}\boldsymbol{x}(kLT_s) + \boldsymbol{B}^{(L)}\boldsymbol{u}(kLT_s) \tag{7.2}$$

$$\boldsymbol{y}(kLT_s) = \boldsymbol{C}^{(L)}\boldsymbol{x}(kLT_s) + \boldsymbol{D}^{(L)}\boldsymbol{u}(kLT_s) \tag{7.3}$$

where

$$\boldsymbol{A}^{(L)} = \boldsymbol{A}^{L}$$

$$\boldsymbol{B}^{(L)} = \left[\boldsymbol{A}^{L-1}\boldsymbol{B}\ \boldsymbol{A}^{L-2}\boldsymbol{B}\ \boldsymbol{A}^{L-3}\boldsymbol{B}\ \ldots\ \boldsymbol{B}\right]$$

$$\boldsymbol{C}^{(L)} = \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{CA} \\ \boldsymbol{CA}^{2} \\ \ldots \\ \boldsymbol{CA}^{L-1} \end{bmatrix}$$

$$\boldsymbol{D}^{(L)} = \begin{bmatrix} d & 0 & 0 & \ldots & 0 \\ \boldsymbol{CB} & d & 0 & \ldots & 0 \\ \boldsymbol{CAB} & \boldsymbol{CB} & d & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \boldsymbol{CA}^{L-2}\boldsymbol{B} & \boldsymbol{CA}^{L-3}\boldsymbol{B} & \boldsymbol{CA}^{L-4}\boldsymbol{B} & \ldots & d \end{bmatrix}$$

$$\boldsymbol{u}(kLT_s) = \left[u(kLT_s)\ u((kL+1)T_s)\ u((kL+2)T_s)\ \ldots\ u((kL+L-1)T_s)\right]^{t}$$

$$\boldsymbol{y}(kLT_s) = \left[y(kLT_s)\ y((kL+1)T_s)\ y((kL+2)T_s)\ \ldots\ y((kL+L-1)T_s)\right]^{t}$$

$\boldsymbol{A}^{(L)}$ is $N\times N$, $\boldsymbol{B}^{(L)}$ is $N\times L$, $\boldsymbol{C}^{(L)}$ is $L\times N$, $\boldsymbol{D}^{(L)}$ is $L\times L$, $\boldsymbol{u}(kLT_s)$ is $L\times 1$ and $\boldsymbol{y}(kLT_s)$ is $L\times 1$. $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ and d are the corresponding conventional state space matrices

for the IIR filter.

7.2.2 Dependence Graph of the Algorithm

The computations required for implementing the block-state IIR filter are described by (7.2) and (7.3). These computations are graphically depicted in Fig. 7.1. It can be seen from Fig. 7.1 that the required computations are iterative. Each iteration requires a new input block and the updated states. It is possible to study the present iterative algorithm using the RDG concepts presented in [55] to provide an abstraction for the involved computations. Furthermore, Fig. 7.1 illustrates that the computations required for updating the states are independent of that required to get the output block even though they share the same inputs. Therefore, two separate RDG's can be obtained for the block-state IIR filter as shown in Fig. 7.2. The RDG in Fig. 7.2(a) depicts the computations required for (7.2) and the other RDG in Fig. 7.2(b) depicts the computations required for (7.3). For both RDG's, the input sample block $\boldsymbol{u}(kLT_s)$ and the states $x_1(kLT_s)$, $x_2(kLT_s)$,....., $x_N(kLT_s)$ are propagated along the j-axis. The states $\boldsymbol{x}(kLT_s)$ and the input samples $\boldsymbol{u}(kLT_s)$ are weighted by different matrices elements and are added (accumulated) to get output samples $\boldsymbol{y}(kLT_s)$ and the new states $\boldsymbol{x}((k+1)LT_s)$.

7.2.3 Processor Assignment and Data Scheduling

The method for developing an array processor design involves two steps. In the first step, an RDG will be obtained. An array processor implementation is then developed in the second step by selecting an appropriate schedule vector and a projection vector on the index set [52]. The concepts of schedule and projection vectors explained in [52] are used to obtain a valid schedule and processor configurations. Each RDG of Fig. 7.2 is divided into two different subregions as shown by the vertical dotted line. Every subregion is handled separately to get the complete array processor architecture. The block diagram of the whole implementation is shown in Fig. 7.3.

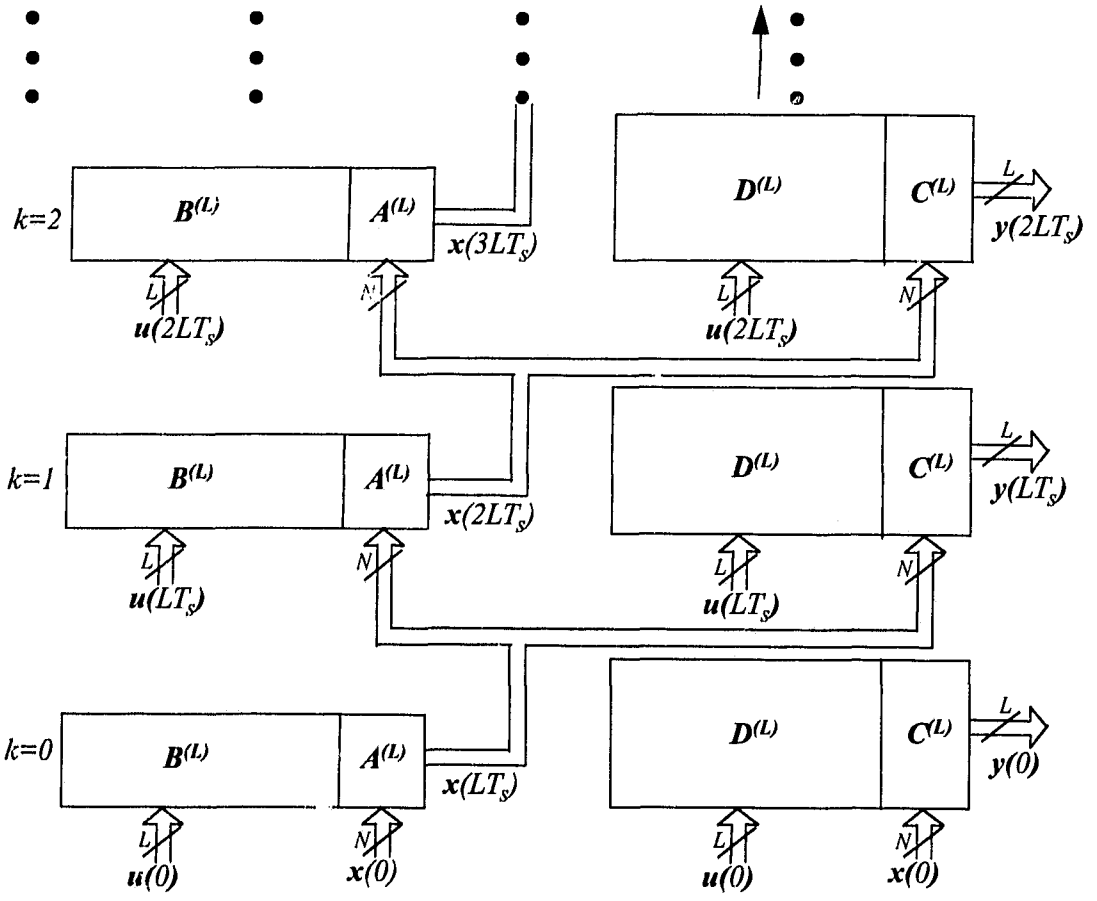


Figure 7.1: Block graph of the block-state IIR filter for different iterations.

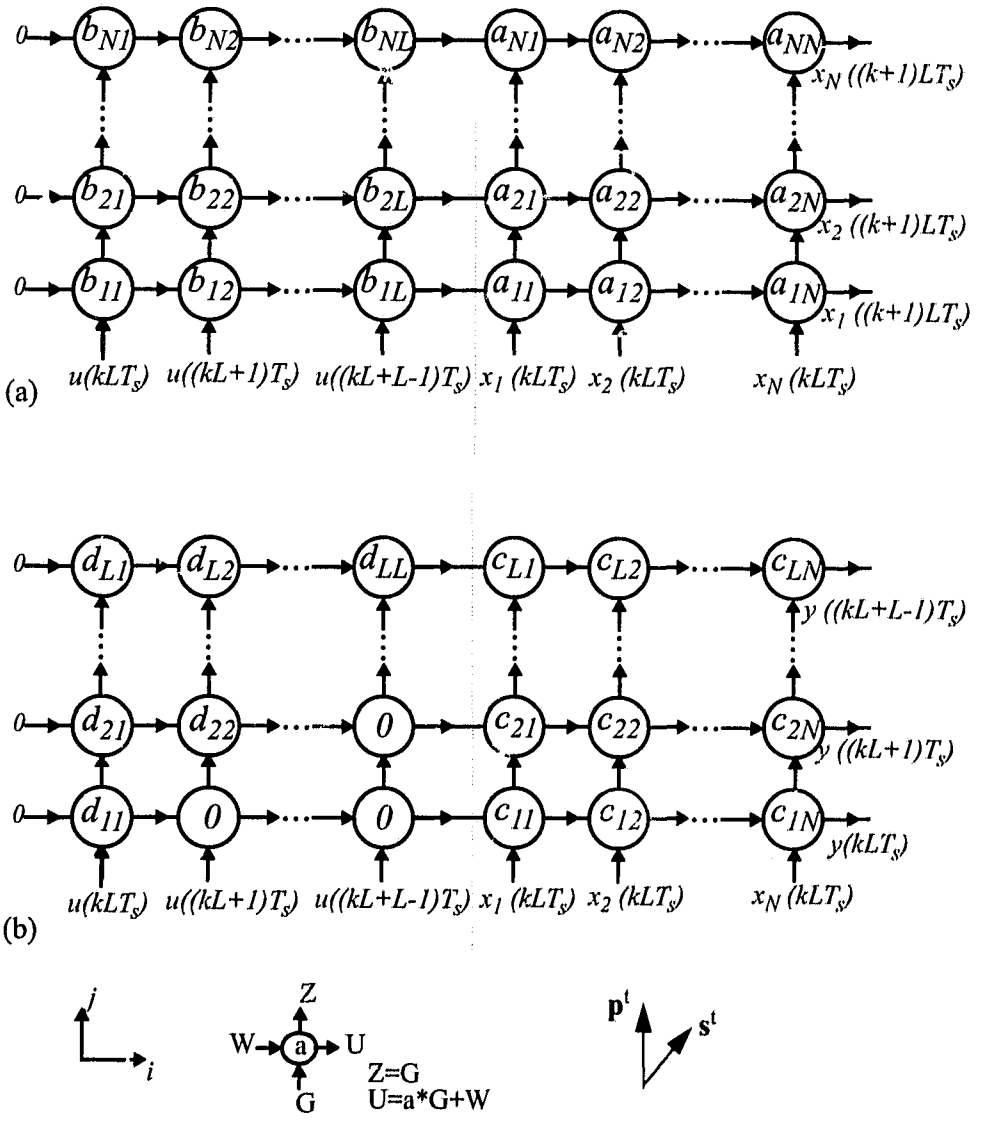


Figure 7.2: Reduced dependent graphs for block-state IIR digital filter. N is the filter order and L is the block size. The feedback path is not shown. (a) RDG for $\boldsymbol{A}^{(L)}$ and $\boldsymbol{B}^{(L)}$. (b) RDG for $\boldsymbol{C}^{(L)}$ and $\boldsymbol{D}^{(L)}$.

The subnetworks which implement the operations $\boldsymbol{A}^{(L)}\boldsymbol{x}(kLT_s)$, $\boldsymbol{B}^{(L)}\boldsymbol{u}(kLT_s)$, $\boldsymbol{C}^{(L)}\boldsymbol{x}(kLT_s)$ and $\boldsymbol{D}^{(L)}\boldsymbol{u}(kLT_s)$ are implemented by different special purpose array processor networks. Our concern now is to implement each matrix vector multiplication (MVM) block and the state update network (SUN) block such that the resulting array processor implementation provides good compromise between hardware complexity and system throughput rate.

7.3 Implementation of The Full State-update Matrix IIR Digital Filter

In this section, the case of a block-state IIR digital filter, in which the corresponding conventional state update matrix $\boldsymbol{A}$ is full, is considered. From (7.2), it is obvious that, in the general case, $\boldsymbol{A}^{(L)}$ will also be full. The SUN block will be first handled because this block determines the system throughput rate [72].

7.3.1 State Update Network (SUN)

By using the scheduling and projection vectors shown in Fig. 7.2 on the block $\boldsymbol{A}^{(L)}\boldsymbol{x}(kLT_s)$, the array processor implementation shown in Fig. 7.4(a) can be obtained [94]. The number of processing elements is equal to the filter order N. Figure 7.4(b) shows the details of the ith processing element (PE^*_i). Each PE is essentially a multiply-add processor. The ring-shift register contains the ith column of $\boldsymbol{A}^{(L)}$. All delays elements and ring-shift registers are clocked at the rate of $1/T_c$ where T_c is defined as

$$T_c = T_{mult} + T_{add} \tag{7.4}$$

where T_{mult} is the delay required for a multiplication. State $x_i(kLT_s)$ is loaded through the feedback path in synchronism with a load strobe every N clock cycles. This arrangement loads the ith processor with the ith state component of the state vector at the

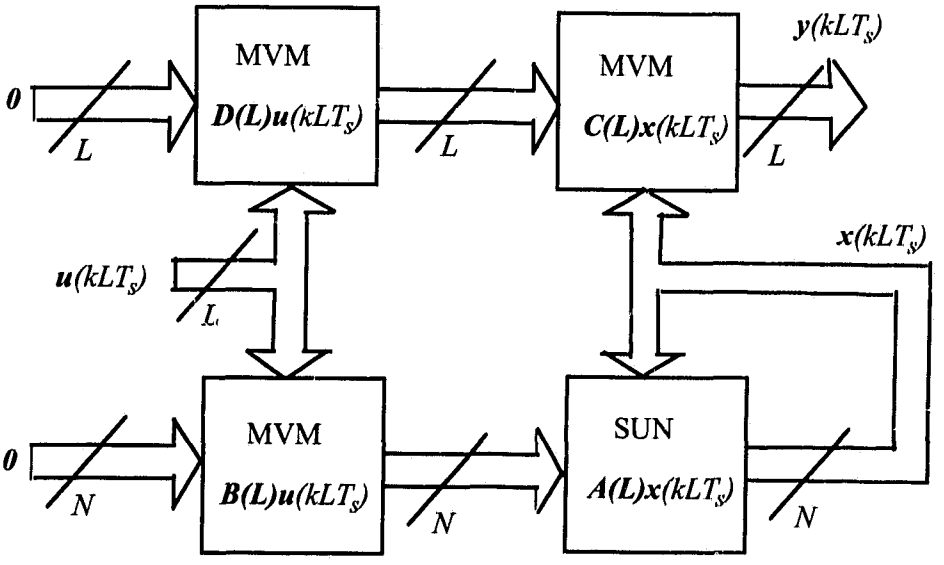


Figure 7.3: Block diagram of the block-state IIR digital filter implementation.
MVM: Matrix-Vector Multiplication.
SUN: State Update Network.

appropriate clock cycle. The register R will save the state for the next N cycles. Table 7.2 shows the coefficients available in different PE*s at various time instants. The a's in Table 7.2 are the coefficients of $\boldsymbol{A}^{(L)}$.

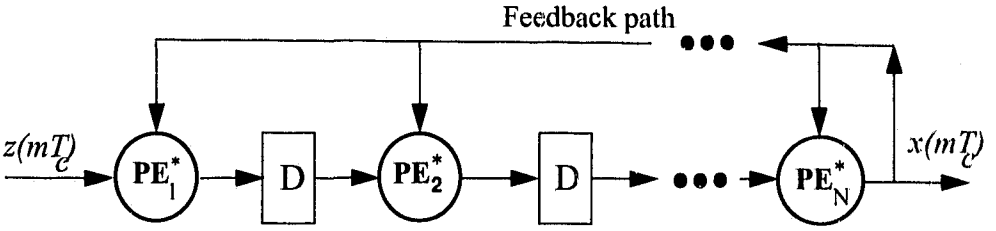
The delay required for updating each state (T_{SUN}) can be defined as

$$T_{SUN} = NT_c \tag{7.5}$$

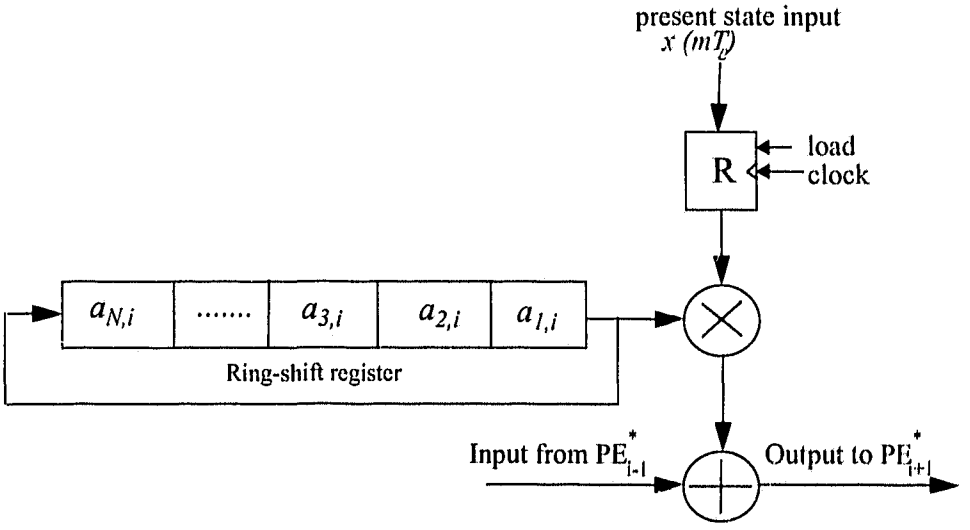
The SUN array processor implementation of Fig. 7.4 has latency l_{SUN} which is equal to the T_{SUN}. The input $z(mT_c)$ to PE^*_1 in Fig. 7.4(a) and in Table 7.2 is the output of the array processor for the block $\boldsymbol{B}^{(L)}\boldsymbol{u}(kLT_s)$. This array processor architecture will be presented in the next subsection.

Table 7.2: Timing for data flow and coefficients for the SUN array processor.

Time in T_c	Input to PE_1	PE_1	PE_2	-	-	PE_N	output
1	$z(T_c)$	a_{11}	-	-	-	-	-
2	$z(2T_c)$	a_{21}	a_{12}	-	-	-	-
3	$z(3T_c)$	a_{31}	a_{22}	-	-	-	-
4	$z(4T_c)$	a_{41}	a_{32}	-	-	-	-
5	$z(5T_c)$	a_{51}	a_{42}	-	-	-	-
-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-
N	$z(NT_c)$	a_{N1}	-	-	-	a_{1N}	$x_1(T_c)$
N+1	$z((N+1)T_c)$	a_{11}	a_{N2}	-	-	a_{2N}	$x_2(T_c)$
N+2	$z((N+2)T_c)$	a_{21}	a_{12}	-		a_{3N}	$x_3(T_c)$
N+3	$z((N+3)T_c)$	a_{31}	a_{22}	-	-	a_{4N}	$x_4(T_c)$
-	-	-	a_{32}	-	-	-	-
-	-	-	-	-	-	-	-



(a)



(b)

Figure 7.4: State-update network (SUN). (a) Array processor architecture for the SUN network $z(mT_c)$ is the output of the $\boldsymbol{B}^{(L)}$ array-processor). (b) The details of PE_i^*.

7.3.2 The MVM Networks

The same schedule and projection vectors are applied to the block $\boldsymbol{B}^{(L)}\boldsymbol{u}(kLT_s)$. The resulting pipelined array processor implementation is shown in Fig. 7.5. The PE structure is the same as that of Fig. 7.4(b) without the need for the register R and each ring shift-register contains the column coefficients of the matrix $\boldsymbol{B}^{(L)}$. Equal delay for all data samples $\boldsymbol{u}(kLT_s)$ has been taken into consideration in order to make the delay from any input to the output a constant. The delay for this subblock (T_B) can be calculated as:

$$T_{\boldsymbol{B}^{(L)}} = LT_c \tag{7.6}$$

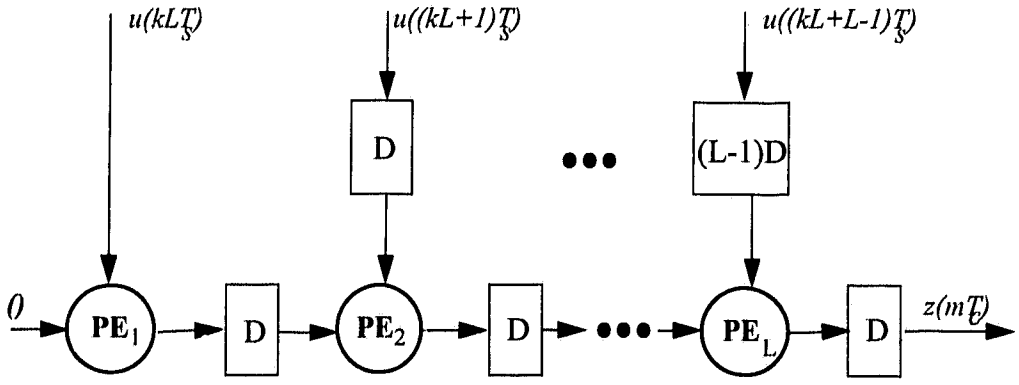


Figure 7.5: Array processor implementation for $\boldsymbol{B}^{(L)}$ block.

Fig. 7.6 shows the computation timing sequence for two consecutive input blocks when the array processors of $\boldsymbol{A}^{(L)}\boldsymbol{x}(kLT_s)$ and $\boldsymbol{B}^{(L)}\boldsymbol{u}(kLT_s)$ are connected. For the sake of demonstration, we take $N = 2$ and $L = 4$. The new update state component $x_1(8T_s)$ for the first block is obtained at $t = 6$ and then will be used as an input for $\boldsymbol{A}^{(L)}\boldsymbol{x}(kLT_s)$ of the second block at $t = 7$. Figure 7.6 also shows that the time inter-

val between two consecutive input blocks is equal to LT_s (i.e., the feeding rate of the input blocks is equal to F_s/L where $F_s = 1/T_s$).

Now, the blocks $\boldsymbol{D}^{(L)}\boldsymbol{u}(kLT_s)$ and $\boldsymbol{C}^{(L)}\boldsymbol{x}(kLT_s)$ are considered. As shown in Fig. 7.3, $\boldsymbol{D}^{(L)}\boldsymbol{u}(kLT_s)$ and $\boldsymbol{B}^{(L)}\boldsymbol{u}(kLT_s)$ are using the same input block and $\boldsymbol{A}^{(L)}\boldsymbol{x}(kLT_s)$ and $\boldsymbol{C}^{(L)}\boldsymbol{x}(kLT_s)$ are using the same states. Therefore, the computations of the blocks $\boldsymbol{D}^{(L)}\boldsymbol{u}(kLT_s)$ and $\boldsymbol{C}^{(L)}\boldsymbol{x}(kLT_s)$ should be executed at the same time required for the computations of the blocks $\boldsymbol{B}^{(L)}\boldsymbol{u}(kLT_s)$ and $\boldsymbol{A}^{(L)}\boldsymbol{x}(kLT_s)$, respectively. This can be achieved under the restriction that the block size L is an integer multiple of N, *i.e.*, $L = rN$ and $r \geq 1$. This restriction is not critical because L is a designer speed-up choice. Based on this restriction, the RDG in Fig. 7.2(b) can be horizontally divide into r subblocks. Each subblock is similar to the RDG in Fig. 7.2(a). By applying the same schedule and the projection vectors to each subblock, r array processors for the block $\boldsymbol{D}^{(L)}\boldsymbol{u}(kLT_s)$ shown in Fig. 7.7(a) can be obtained.

In order to demonstrate the distribution of the $\boldsymbol{D}^{(L)}$ coefficients among the processors, a diagram of the detail of one processor element of Fig. 7.7(a) is shown in Fig. 7.7(b). The computational delay for $\boldsymbol{D}^{(L)}\boldsymbol{u}(klT_s)$ block can be calculated as

$$T_{\boldsymbol{D}^{(L)}} = LT_c \tag{7.7}$$

The r array processors for the block $\boldsymbol{C}^{(L)}\boldsymbol{x}(kLT_s)$ is shown in Fig. 7.7(c). Each array processor in Fig. 7.7(c) delivers N output samples in a serial manner. As the number of these arrays is equal to r, the number of output samples delivered by them will be the L output samples. The computational delay for the $\boldsymbol{C}^{(L)}\boldsymbol{x}(kLT_s)$ block can be calculated

as

$$T_{C^{(L)}} = NT_c \tag{7.8}$$

The whole array processor implementation of the block-state digital filter where N=2 and L=4 is shown in Fig. 7.8.

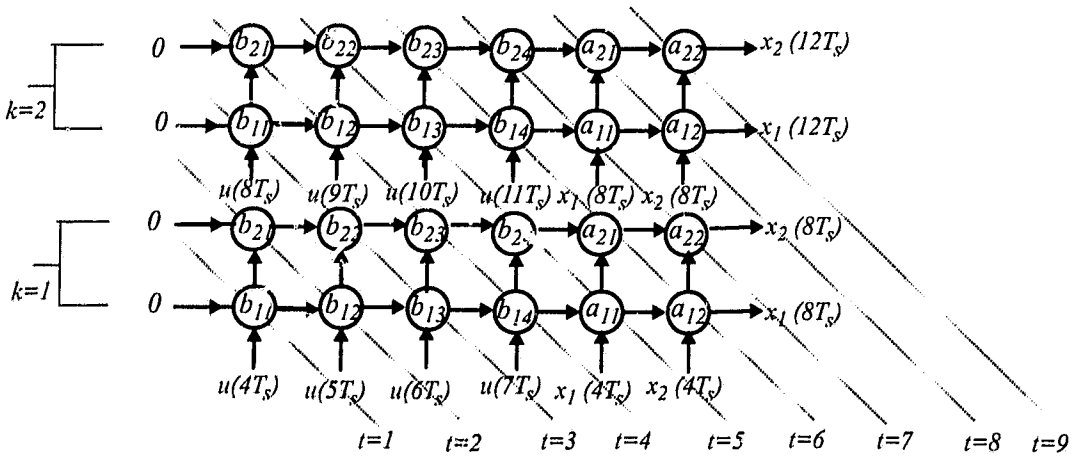
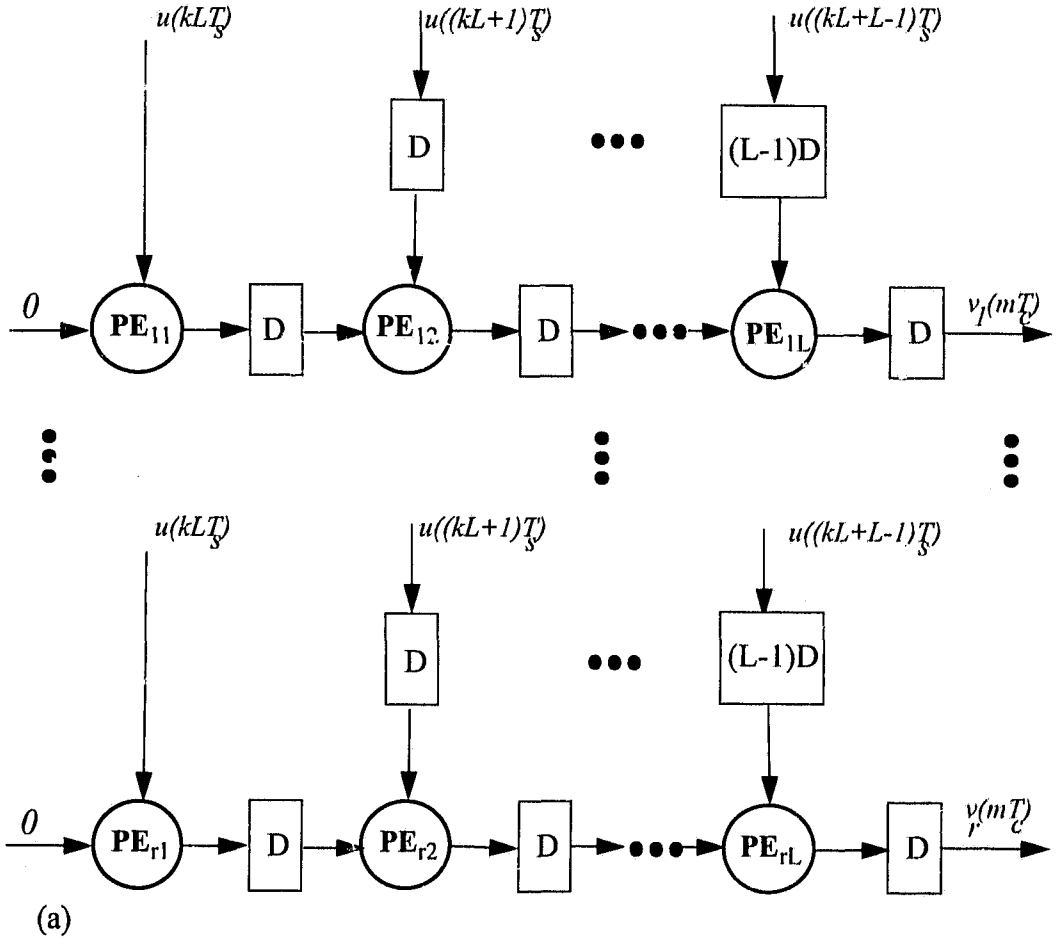
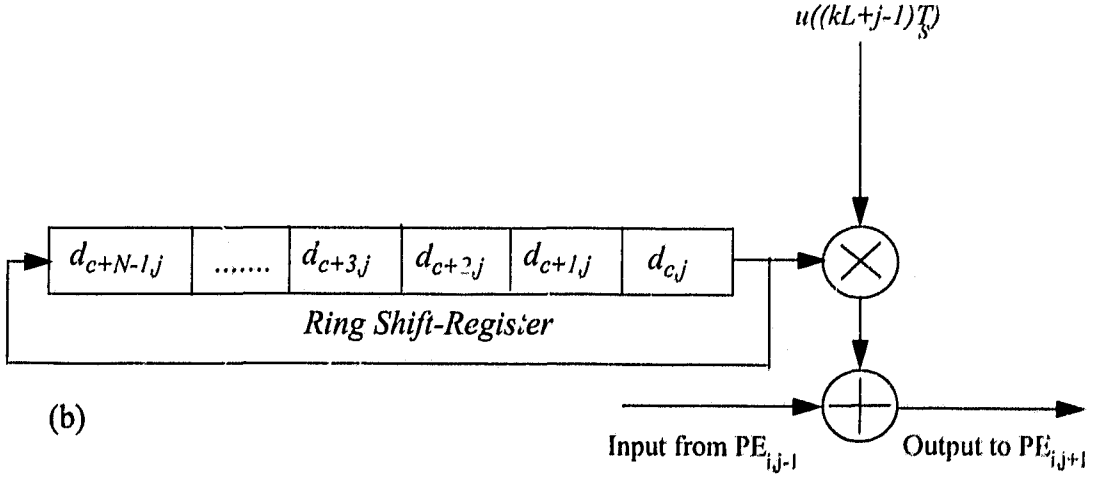


Figure 7.6: Timing diagram showing the timing sequence for two consecutive input blocks when connecting the array processors for both $\boldsymbol{A}^{(L)}$ and $\boldsymbol{B}^{(L)}$ (N=2 and L=4).


u(kLT_S)
u((kL+1)T_S)
u((kL+L-1)T_S)
D
(L-1)D
0
PE11
PE12
PE1L
v1(mTc)
PEr1
PEr2
PErL
vr(mTc)

(a)


u((kL+j-1)T_S)
dc+N-1,j
.....
dc+3,j
dc+2,j
dc+1,j
dc,j
Ring Shift-Register
Input from PEi,j-1
Output to PEi,j+1

(b)

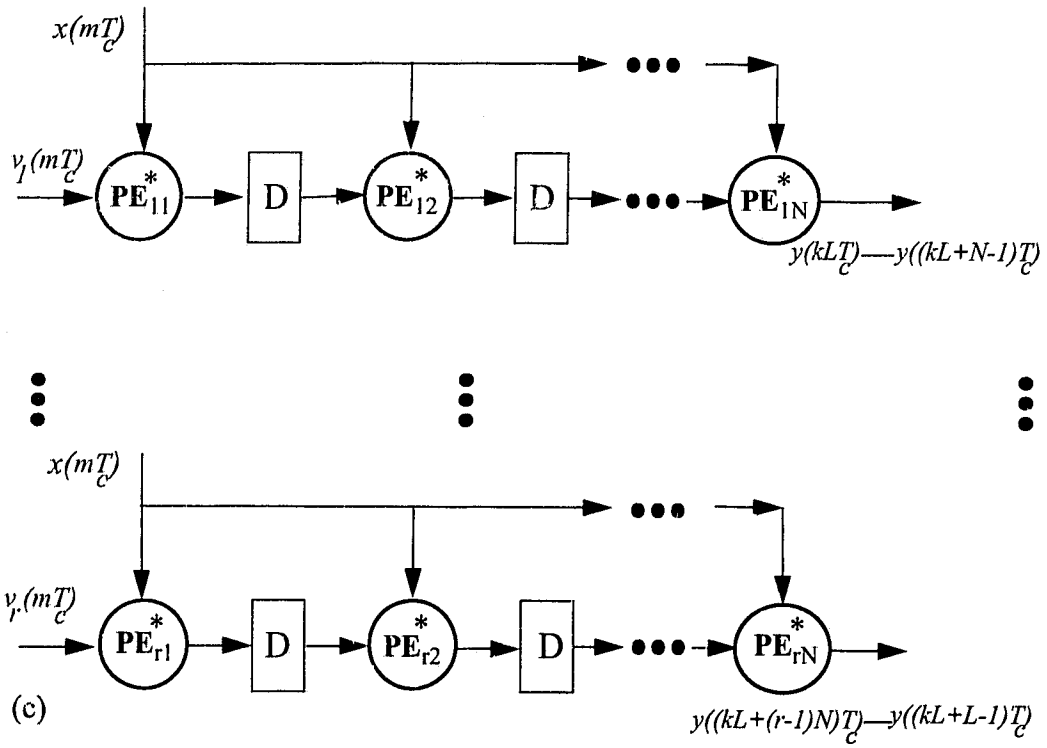


Figure 7.7: Array-processor networks for $\boldsymbol{D}^{(L)}$ and $\boldsymbol{C}^{(L)}$. (a) Array processor implementation for $\boldsymbol{D}^{(L)}$ block where $r=L/N$. (b) The internal structure of the ijth processor element involved in (a) ($1 \le j \le L$, $1 \le i \le r$, $c = (i-1)N+1$, and d's are the coefficients of $\boldsymbol{D}^{(L)}$). (c) Array processor implementation for $\boldsymbol{C}^{(L)}$ block. The PE* involved is similar to that in Fig. 7.4(b).

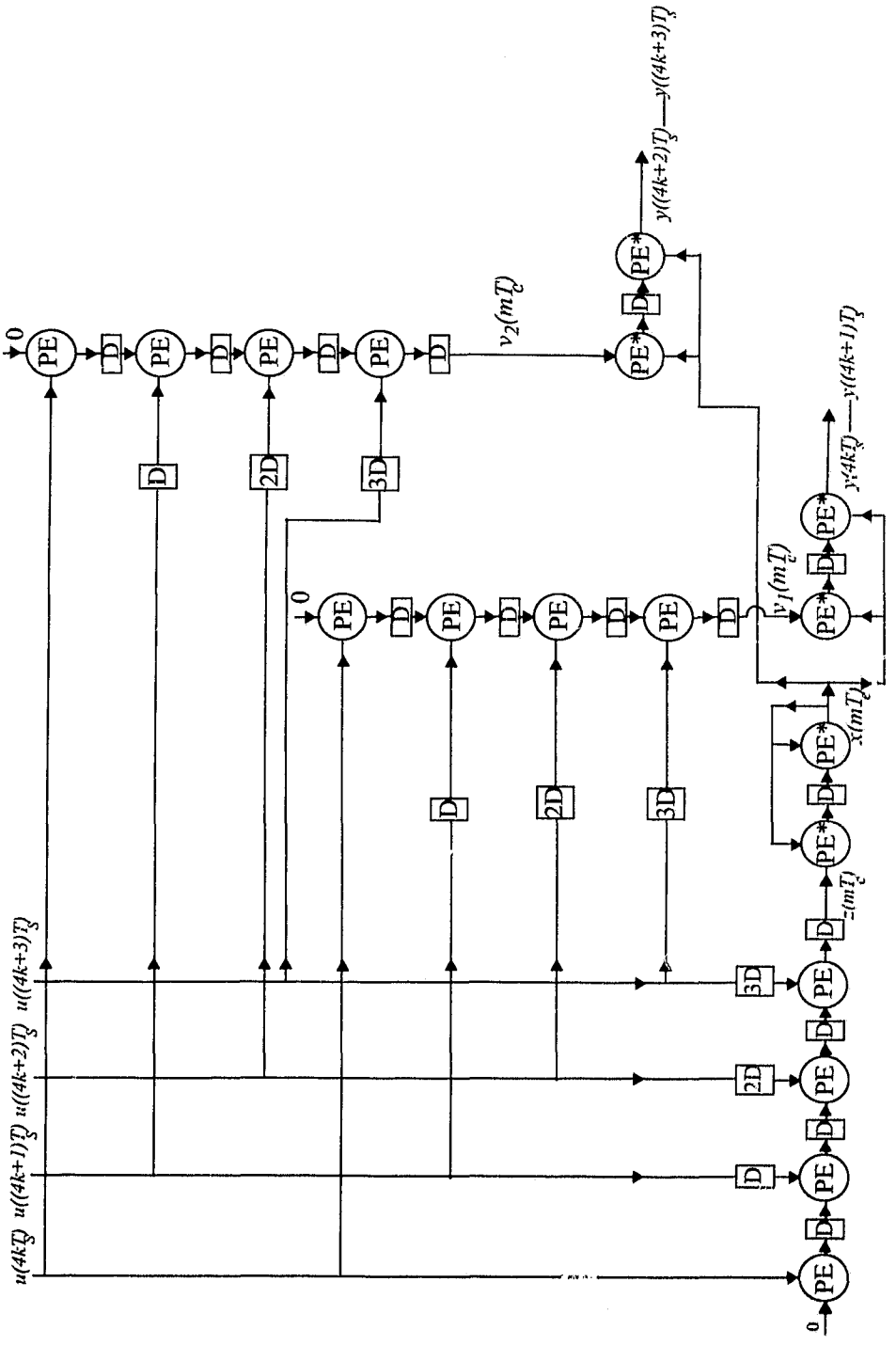


Figure 7.8: The complete implementation of the block-state IIR digital filter (N=2, L=4 i.e., T_c=2T_s).

7.3.3 Performance Analysis

In this subsection, the performance of the proposed architecture is discussed.

Area Complexity

The area complexity of the proposed architecture will be measured by the number of required processing elements PEs. Each PE is mainly composed of one multiplier and one adder. It can be seen from Figs. 7.4, 7.5 and 7.7 that the number of PEs ($Comp_1$) is given by

$$Comp_1 = N + L + rL + rN = N + 2L + rL \tag{7.9}$$

Throughput

The maximum throughput of the recursive implementation is determined by the delay of the feedback loop. In the present case the throughput is set by the delay of the SUN; i.e., the delay of the recursive subnetwork determines the upper bound on the throughput [72]

$$\Im \le \frac{L}{T_{SUN}} \tag{7.10}$$

where $\Im$ denotes the throughput. The upper bound on $\Im$ which appears in the RHS of (7.10) is also called the "iteration bound". Therefore the throughput $\Im$ of the whole implementation is given by

$$\Im \le \frac{L}{NT_c} \tag{7.11}$$

The proposed digital filter implementation can be used to filter input data at a sampling rate up to $\Im$ samples/sec. The input sampling period T_s can be calculated as

$$T_s = \frac{1}{\Im} \ge \frac{NT_c}{L} = \hat{T}_s \tag{7.12}$$

where $\hat{T}_s$ is the lower bound on the input sampling period. As $L \ge N$, the upper bound of $\hat{T}_s$ is given by

$$\hat{T}_s \leq T_c = T_{mult} + T_{add} \tag{7.13}$$

So, the input sampling frequency exceeds the limitation imposed by the processor element speed for $L > N$. From (7.12), it is evident that high input sampling frequencies can be achieved by increasing L. However, the input sampling frequency is limited by the available hardware for the input serial/parallel and the output parallel/serial converters. This limitation will be discussed in Section 7.5.

Latency

Using Figs. 7.4, 7.5 and 7.7, the latency of the proposed filter architecture with full state update matrix can be computed. The delays of the serial-to-parallel and parallel-to-serial converters are neglected in the present discussion. The total latency (l) (expressed in units of T_s) can be calculated as:

$$\begin{aligned} l &= \frac{L}{N}\left(l_{C^{(L)}} + l_{D^{(L)}} \right) = \frac{L}{N}(N+L) \\ &= L(1+r) \end{aligned} \tag{7.14}$$

Finite-Word length Effects

It has been shown [97] and experimentally verified [95] that block-state realizations have less output roundoff noise than that of the corresponding SISO filter, assuming that roundoff operation for the states is done after additions rather than after the multiplications in (7.2). This assumption can be easily met in the proposed architecture by executing all the additions inside all PEs in double-precision and rounding the output of block $\boldsymbol{A}^{(\boldsymbol{L})}\boldsymbol{x}(kLT_s)$. It has been also proved that the block realization has no stability problem if the corresponding conventional filter is originally stable [96]. However, it has been proved that the block realization for IIR filters may exhibit time-varying response under finite-word effects due to coefficient sensitivity [59],[97], [98].

7.3.4 Comparison with Existing Architecture

In this subsection, the performance of the proposed architecture is compared to another architecture presented in [92] for implementation of block-state IIR filters. It is assumed that L is a multiple of N. This assumption is needed for the proposed architecture but not for the architecture in [92]. The comparison is done with respect to the number of processor elements (PEs) required, the $area \times time$ (AT) estimation and the $area \times time^2$ (AT^2) estimation. The corresponding values as a function of N and L are presented in Table 7.3. The AT estimation is obtained from the area multiplied by the minimum input sampling period $\hat{T}_s$, while the AT^2 estimation is obtained by multiplying the area by the square of the minimum input sampling period. The area complexity has been defined in terms of the area required for all processor elements normalized by the area of one processor element area. The area required for latches has been neglected. The minimum input sampling period has been normalized by the delay required for one addition assuming that $T_{mult} \approx T_{add}$ i.e., we assume the use of array multipliers for multiplications and carry-propagate adders for additions. The improvements in performance of the proposed implementation relative to the one proposed in [92] can be quantified by the following three parameters

$$G_1 = \frac{Comp_2 - Comp_1}{Comp_2} \times 100$$

$$G_2 = \frac{AT_2 - AT_1}{AT_2} \times 100$$

$$G_3 = \frac{AT_2^2 - AT_1^2}{AT_2^2} \times 100$$

where $Comp_2$ is the number of PEs required for the implementations in [92], $Comp_1$ is calculated in (7.9), AT_2 and AT_2^2 are the normalized AT and AT^2 of the implementa-

tion proposed in [92], respectively. The AT_1 and AT_1^2 are the normalized AT and AT^2 of the proposed implementation, respectively. G_1, G_2 and G_3 as function of the filter order N are plotted in Fig. 7.9(a), (b) and (c). Simple calculations show that G_1, G_2 and G_3 are insensitive to the variation of L (or r).

Table 7.3: Performance properties for two block-state IIR digital filter implementations of full state-update matrix.

	The Proposed Arch.	Arch. of [92]
Min. input Sampling period ($\hat{T}_s$).	$N\frac{(T_{mult}+T_{add})}{L}$	$\frac{(T_{mult}+NT_{add})}{L}$
Number of PEs (*Comp*)	$N+2L+\frac{L^2}{N}$	$N^2+2LN+L^2$
Normalized AT	$2\frac{N^2}{L}+4N+2L$	$\left(\frac{N^2}{L}+2N+L\right)(N+1)$
Normalized AT^2	$4\frac{N^3}{L^2}+8\frac{N^2}{L}+4N$	$\left(\frac{N^2}{L^2}+2\frac{N}{L}+1\right)(N+1)^2$

It can be seen from Fig. 7.9(a) that the reduction in the processor elements for the proposed architecture reaches at least 50% compared to the architecture in [92] for all values of N and L. Further, from Fig. 7.9(b), it can be seen that the proposed architecture provides better perfor mance (at least 33%) in AT sense for all values N and L. Fig. 7.9 (c) shows the proposed architecture provides better performance (at least 11%) in AT^2 sense for all values N and L. The performance improvements G_1, G_2 and G_3 gained by using the proposed architecture increase as N increases.

It should be noticed that the above comparison is based on the assumption of using the same values of N and L for the proposed architecture and the architecture proposed in

[92]. This implies the use of a different input sampling rate for each architecture. Another comparison can be made under the assumption that both architectures for the same filter order N are used to filter the same input data rate. This implies the use of different values for block size (L) for each architecture. The relation between the both block sizes L_1 and L_2 for the proposed architecture and the one proposed in [92], respectively, can be easily obtained from the following relation

$$L_2 = \left\lceil \frac{(N+1)}{2N} L_1 \right\rceil \tag{7.15}$$

where $\lceil\ \rceil$ is the ceiling function. Evaluation of the parameters G_1, G_2 and G_3 based on the modified assumption shows that the proposed implementation provides better performance for all values of L and N.

7.4 Implementation of Block-Diagonal IIR Digital Filter

An important IIR filter realization is the realization of the filter as a parallel combination of second-order sections. In this case, a higher input sampling rate can be achieved because of the reduced number of the computations required for updating the state matrix and the possibility of decoupling these computations. The state update matrix $\boldsymbol{A}$ of the conventional filter for the parallel realization takes the block-diagonal form. i.e.,

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A_1} & \mathbf{0} & \cdot & \cdot & \mathbf{0} \\ \mathbf{0} & \boldsymbol{A_2} & \cdot & \cdot & \mathbf{0} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \mathbf{0} & \mathbf{0} & \cdot & \cdot & \boldsymbol{A_{N/2}} \end{bmatrix} \tag{7.16}$$

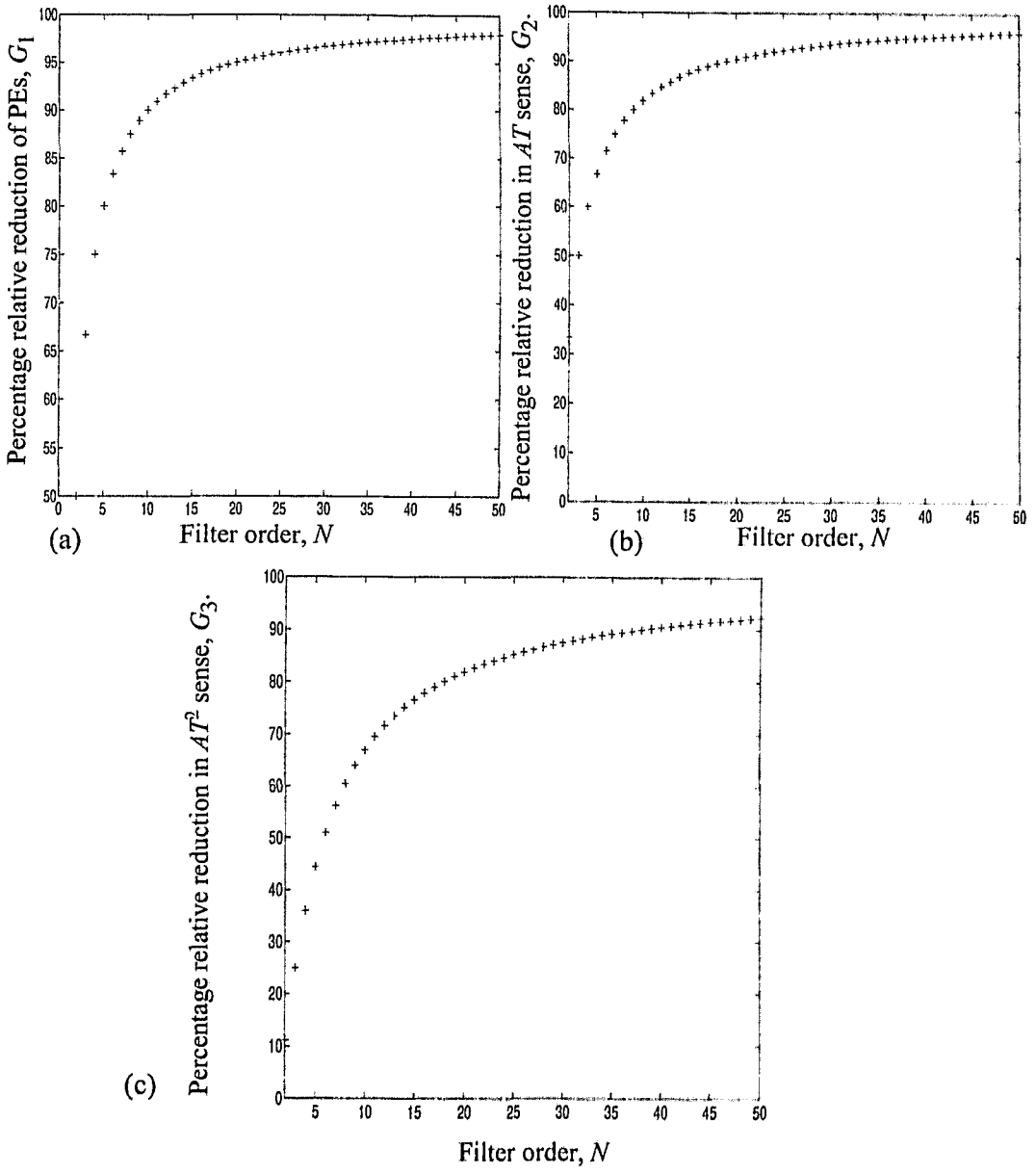


Figure 7.9: Performance gains in percentage obtained by using the new proposed architecture. (a) The percentage relative reduction of the number of the processor elements, G_1. (b) The percentage relative reduction in AT sense, G_2. (c) The percentage relative reduction in AT^2 sense, G_3.

where $\mathbf{0}$'s are 2×2 matrices filled with zeros and $A_1, A_2, \ldots, A_{N/2}$ describe the update state matrices for the second-order sections. If N is odd, one of the submatrices A_i will represent a first-order section. For the sake of brevity, the case of N even will only be considered and the case of N odd can be obtained following a similar procedure.

Similar to the procedure of Section 7.3, the array processors for the state-update network (SUN) $A^{(L)}$ of (7.2) when the conventional update state matrix takes the form in (7.16) will be obtained. Then, the matrix vector multiplications (MVM) array processors for $B^{(L)}$, $C^{(L)}$ and $D^{(L)}$ networks will be presented. The array processor implementation for the case of a filter with even-order block-diagonal update state matrix will be obtained under the restriction that the block size is also even.

7.4.1 State Update Network (SUN)

It is obvious that if matrix A takes the block-diagonal form as in (7.16), then the update state matrix of the block description $A^{(L)}$ in (7.2) will also take the form of block-diagonal matrix where each 2×2 block diagonal matrix is $A_i^{(L)}$ and $1 \le i \le N/2$. Therefore, the state update matrix can be written as $N/2$ decoupled equations. These decoupled equations can be implemented using $N/2$ decoupled systems as shown in Fig. 7.10(a) [73]. Each decoupled system can be implemented by a separate array processor. This array processor has a similar structure of Fig. 7.4(a) but contains only 2 processor elements. Each register R inside the processor element holds the corresponding state for two clock cycles. The array processor is responsible of updating two states and therefore $N/2$ array processors are required to update the whole N states. The detailed structure for SUN is shown in Fig. 7.10(b) where each ring shift-register contains the two-column entries of the corresponding $A_i^{(L)}$.

The delay required for updating each state in this case T_{SUN} is

$$T_{SUN} = 2\,(T_{mult} + T_{add}) = 2T_c \tag{7.17}$$

7.4.2 MVM Networks

The networks $\boldsymbol{B}^{(L)}$, $\boldsymbol{D}^{(L)}$ and $\boldsymbol{C}^{(L)}$ are going to be mapped onto array processors in a way similar to that presented in Section 7.3.2. The difference is that the mapping should produce array processors which are compatible with the decoupled array processors in Fig. 7.10(a).

To compute the block $\boldsymbol{B}^{(L)}\boldsymbol{u}\,(kLT_s)$, the matrix $\boldsymbol{B}^{(L)}$ is divided into $N/2$ submatrices each of dimension $2 \times L$. Each submatrix contains two rows of the $\boldsymbol{B}^{(L)}$ matrix. Each submatrix will be mapped separately by the same scheduling and projection vectors as in Fig. 7.2 onto array processor. The resulting $N/2$ array processors are shown in Fig. 7.11(a). The structure of each processor element is identical to that in Fig. 7.5 with the exception that the ring shift-register inside each processor element contains only two coefficients which are the column entries of the corresponding submatrix.

Similarly, the matrix $\boldsymbol{D}^{(L)}$ is also divided into $L/2$ submatrices. Each submatrix contains two rows of the $\boldsymbol{D}^{(L)}$. The $L/2$ array processors resulting from separately mapping the submatrices are shown in Fig. 7.11(b). The internal structure of the involved processor element is identical to that in Fig. 7.7(b). Each ring shift-register in Fig. 7.11(b) carries two coefficients of $\boldsymbol{D}^{(L)}$ which are the column entries of the corresponding submatrix.

The $L/2$ array processors for the block $\boldsymbol{C}^{(L)}$ are shown in Fig. 7.11(c) where the involved processor elements are similar to those in Fig. 7.10(b). Each shift-register in Fig. 7.11(c) carries two coefficients of $\boldsymbol{C}^{(L)}$ which are the column entries of the correspond-

ing submatrix. Each array processor produces two samples of the output in serial manner.

7.4.3 Performance Analysis and Comparison

The performance properties of the proposed array processor architecture for the block-diagonal update-state-matrix digital filter are summarized in the first column of Table 7.4 The second column of Table 7.4 contains the performance properties of the architecture presented in [73]. The calculations of both the input sampling rate and the required area are based on the same assumptions and parameters proposed in Subsection 7.3.4. It can be seen from Tables 7.3 and Table 7.4 that the minimum input sampling rate achieved by the architecture presented in this section is higher than that of the architecture presented in Section 7.3. This is expected because of the decoupling property of the block-diagonal update state matrix of Fig. 7.10(a). From Table 7.4, it can also be seen that the proposed array processor structure for the block-diagonal update-state matrix case has 50% reduction of the involved processor elements compared to that of the corresponding structure proposed in [73] for all values of N and L. The proposed architecture also provides 33% reduction in AT performance over the structure presented in [73] for all values of N and L. The proposed architecture also provides 11% reduction in AT^2 performance over the structure presented in [73] for all values of N and L.

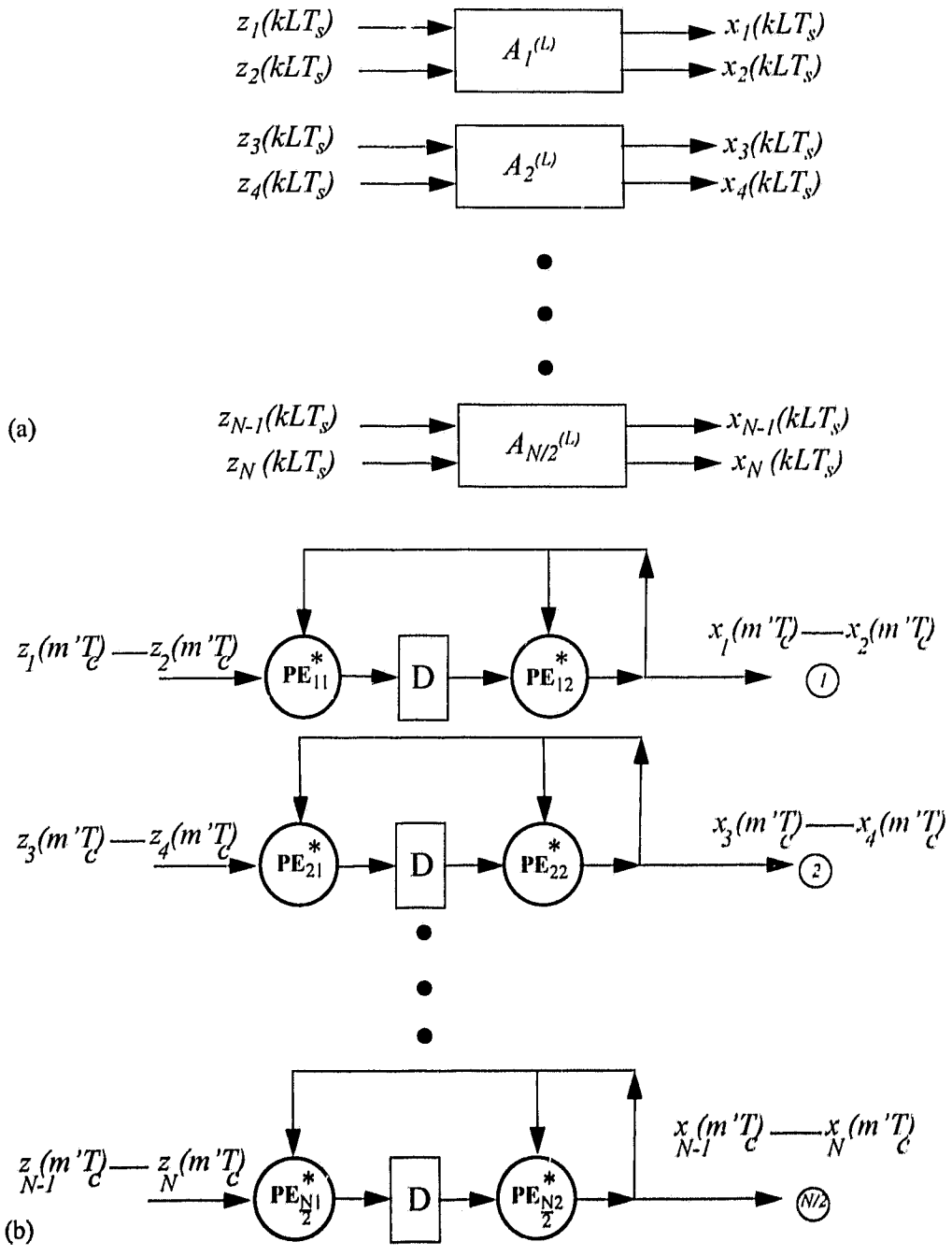
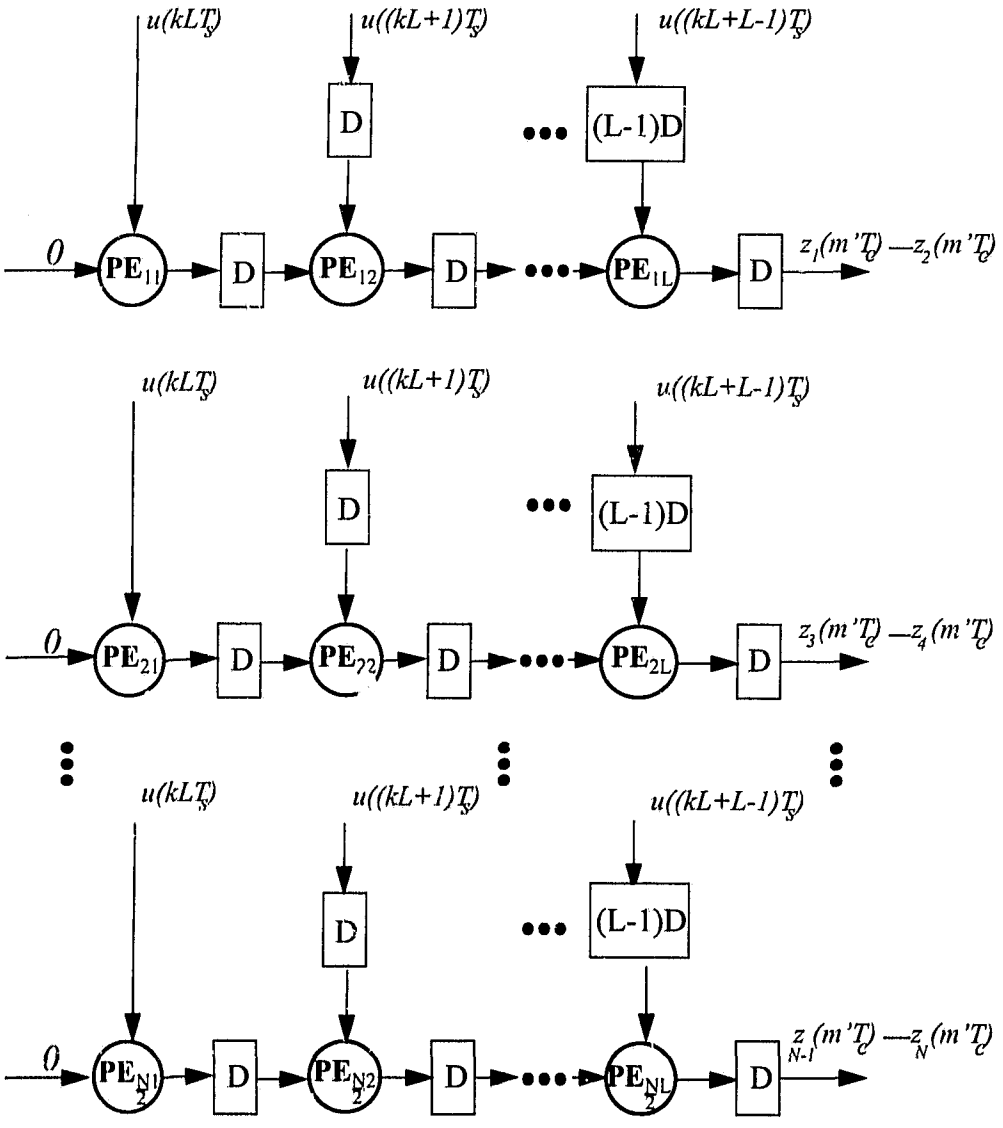
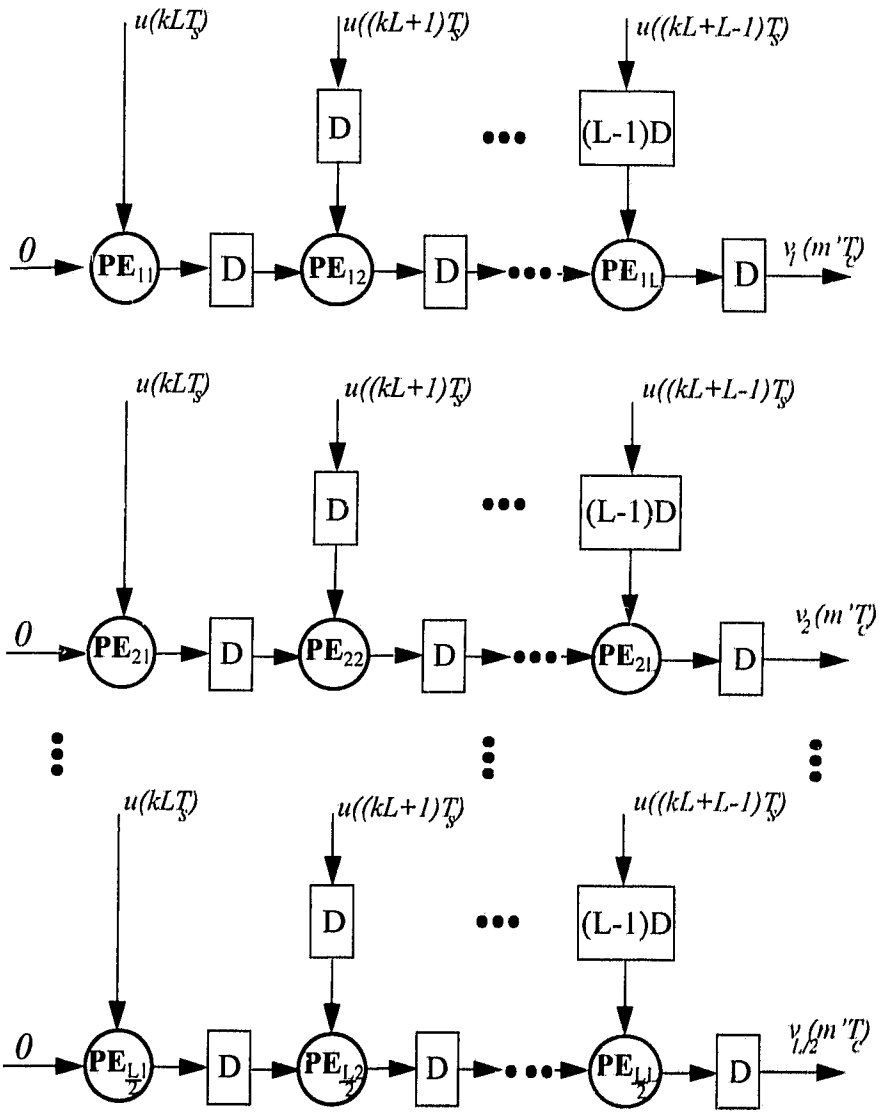


Figure 7.10: The state update network (SUN) for the block-diagonal case. (a) Block diagram of the decoupled systems. (b) The SUN array processor.

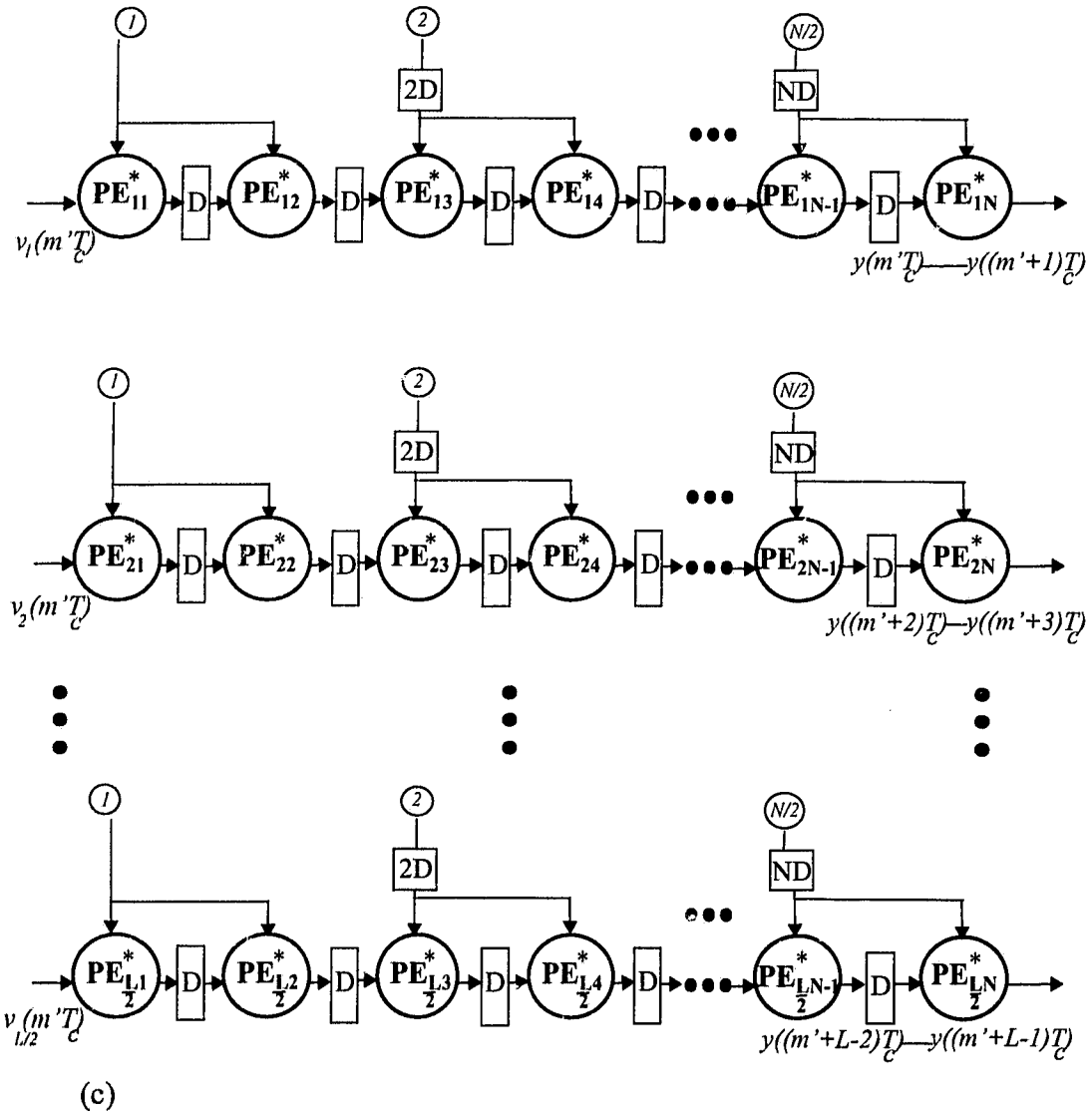

u(kLT_s)
u((kL+1)T_s)
u((kL+L-1)T_s)
D
(L-1)D
0
PE_11
PE_12
PE_1L
z_1(m'T_c) —z_2(m'T_c)
PE_21
PE_22
PE_2L
z_3(m'T_c) —z_4(m'T_c)
PE_N1/2
PE_N2/2
PE_NL/2
z_N-1(m'T_c) —z_N(m'T_c)

(a)

*


u(kLT_s)
u((kL+1)T_s)
u((kL+L-1)T_s)
D
(L-1)D
0
PE_11
PE_12
PE_1L
v_1(m'T_c)
PE_21
PE_22
PE_2L
v_2(m'T_c)
PE_L1/2
PE_L2/2
PE_LL/2
v_L/2(m'T_c)

(b)



(c)

Figure 7.11: The array processor implementations for the MVM networks for the block-diagonal case. (a) Array processor implementation for $B^{(L)}$ block. (b) Array processor implementation for $D^{(L)}$ block. (c) Array processor implementation for $C^{(L)}$ block.

Table 7.4: Performance properties for different block-state implementations of block-diagonal state-update matrix.

Parameter	Proposed Arch.	Arch. of [73]
Min. input Sampling period ($\hat{T}_s$).	$2\frac{(T_{mult}+T_{add})}{L}$	$\frac{(T_{mult}+2T_{add})}{L}$
Number of PEs (*Comp*)	$N+NL+\frac{L^2}{2}$	$2N+2LN+L^2$
Normalized AT	$2\left(2N+2\frac{N}{L}+L\right)$	$3\left(2N+2\frac{N}{L}+L\right)$
Normalized AT^2	$8\left(1+2\frac{N}{L}+2\frac{N}{L^2}\right)$	$9\left(1+2\frac{N}{L}+2\frac{N}{L^2}\right)$

7.5 Input/Output Constraints

From Section 7.3 and Section 7.4, it can be seen that any high input sampling rate can be achieved for Nth-order filter by increasing the block size L at the expense of increasing the hardware complexity. The serial/parallel input converter and the parallel/serial output converter required for the block structure may impose further constraints on the input sampling rate. These converters are clocked at the high-sampling rate F_s. Therefore, if the available technology provides converters of maximum converter rate of F'', a limitation on the block size L for Nth-order full update-state array-processor architecture proposed in Section 7.3 is given by

$$L \leq NF''/F_c \tag{7.18}$$

where $F_c = 1/T_c$. For the case of the array-processor architecture presented in Section 7.4, the constraint on the block size is given by

$$L \leq 2F''/F_c \tag{7.19}$$

The outputs of the proposed array-processor architectures are obtained in a form of blocks of serial data. The size of this block is N for Fig. 7.7(c) and 2 for Fig. 7.11(c). Therefore, r serial/parallel converters for Fig. 7.7(c) and $L/2$ serial/parallel converters for Fig. 7.11(c) are needed. These additional converters are required to be operated at a rate equal to F_c which is equal to or lower than the input sampling rate F_s. Therefore, these additional converters will not put any additional constraint on the input block size.

7.6 Conclusion

Two new array processor implementations have been proposed for IIR digital filters with high input sampling rates which are not limited by the speed of the involved processing element. The first implementation is based on block-state description in which the state-update matrix is full corresponding to implementing the corresponding conventional state-space filter as one section. The other implementation is also based on the block-state description in which the state-update matrix is block-diagonal corresponding to the case of parallel combination of second-order sections. Both implementations have been obtained by mapping the block description of the IIR filter onto an array processor, so that to reduce the number of the PEs involved and in the same time to make these PEs operate at a clock rate which is higher than that of the input sampling rate divided by the block size. The proposed array processor architectures are amenable to VLSI implementation because they require a significantly reduced number of processor elements. The proposed implementations have been compared with other implementations in the $area \times time$ and $area \times time^2$ sense and also with respect to the number of PEs required. It has been shown that the proposed architectures give better performance over existing ones.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

The objective of this dissertation has been to obtain realizations of IIR digital filters which perform well under finite wordlength constraints and can be efficiently implemented using DSP or in a VLSI environment.

In Chapter 2, new structures for IIR filters as combination of cascade or parallel of second-order sections have been proposed. They yield lower output roundoff noise than many well-known low-roundoff structures. Further, they have the same number of non-trivial multiplies as the direct structure and are guaranteed to be free of zero-input overflow oscillations.

In Chapter 3, the problem of synthesizing finite wordlength fixed-point realizations of Nth-order IIR digital filters using second-order residue feedback to minimize the output roundoff noise subject to l_2-scaling, has been considered. It has been shown that by proper choice of simple residue feedback schemes, new optimal structures can be obtained for narrow band-stop and band-pass filters. These optimal structures provide lower output roundoff noise compared to the MRH structures proposed by Mullis, Roberts [28] and Hwang [29] if the sum of the residue modes is less than the sum of the second-order modes.

Further, new suboptimal residue feedback structures for narrow-band IIR filters are

also proposed which provide near-optimal roundoff noise and have a saving of at least $N(N-2)/2$ multiplies over the optimal structures. Moreover, these suboptimal structures can be chosen to have block-triangular state-update matrices which are more suitable for high-speed hardware implementations. Extensive numerical comparisons between the proposed suboptimal structures and three other low-noise structures show that the proposed suboptimal structures provide excellent performance in terms of output roundoff noise and coefficient sensitivity as well as low computational complexity. It has also been found that the involved simple residue feedback schemes are very powerful and versatile methods to reduce the quantization noise for narrow-band low-pass, high-pass and band-pass filters. For narrow band-stop filters, the proposed residue feedback schemes seem to be less effective.

In chapter 4, two implementations of the SRF scheme 2 realization of Chapter 3 are presented. One by using a fixed-point DSP56001, while the second is an ASIC VLSI array-processor. It has been experimentally shown that for many narrow-band IIR filters, the DSP implementation based on the SRF scheme 2 realization provides high S/N ratio, freedom of limit cycles for the examples considered and moderate input-sampling frequency. A VLSI implementation of the SRF scheme 2 realization which provides high input sampling rate has been presented. This sampling rate is determined by the computation delay required for the accumulator-multiplier operation and two double-precision addition. This efficient VLSI implementation is obtained by taking advantage of both the parallelism inherent in the residue feedback technique and the block-triangular update state matrix of the proposed realization. It has been also shown that this implementation is guaranteed to provide higher speed than that of corresponding direct implementation for narrow-band IIR filters.

In Chapter 5, a new scheme for an area-time efficient two's complement fixed-point inner-product processor is presented. This scheme provides better performance

compared to existing inner-product schemes in terms of computation speed, $area \times time$, and $area \times time^2$ performances. The proposed inner-product processor has a high computational speed which is double (for long inner-product operation) the speed of the conventional pipelined inner-product processor. The new processor is well suited to any DSP application requiring computation of inner-product and is amenable for VLSI implementation.

In Chapter 6, an efficient systolic implementation for Nth-order full-matrix state-space digital filter has been proposed. The new architecture is amenable to VLSI implementation because the number of the involved processor elements is linear w.r.t the filter order. The performance of the proposed implementation has been enhanced by using the CSIPP structure proposed in Chapter 5 as the inner-product processor inside each processing element. All the double-precision operations are locally performed which results in decreasing the communication overhead. The MRH Nth-order state-space structure for IIR filters (with its desirable features) can now be implemented by the proposed architecture and can provide better performance in terms of $area \times time$ compared to other existing state-space implementations. It has also been shown that the proposed implementation provides in many cases better performance in terms of $area \times time$ and $area \times time^2$ compared to existing direct implementation for narrow-band filters. Therefore, the proposed implementation provides a good compromise between the required complexity area and the computational delay for narrow-band IIR digital filters.

In Chapter 7, two efficient array processor implementations are proposed for IIR digital filters with high input sampling rate which are not limited by the speed of the processing element involved. The first implementation is based on block-state description in which state-update matrix is full corresponding implementing the corresponding filter as one section. The other implementation is also based on the block-state description in which the state-update matrix is block-diagonal corresponding to the case of parallel

combination of second-order sections. Both have been obtained by mapping the block description of the IIR filter onto an array processor, so that to reduce the number of the PEs involved and in the same time to make them operate at a clock rate which is higher than that of the input sampling rate divided by the block size. The proposed array processor architectures are amenable to VLSI because they require significantly reduced number of processor elements. The proposed implementations have been compared with other implementations in the $area \times time$ and $area \times time^2$ and also with respect to the number of PEs required. It has been shown that the proposed architectures give better performance over existing techniques.

8.2 Recommendations for Future Research

The area of IIR digital filter realization and implementation is a very active due to the several possible applications of digital filters. There are several possible extensions and developments of the work presented here.

The implementation in Chapter 6 is proposed for the IIR digital filter structures with full system matrix without residue-feedback technique. However, it has been shown in Chapter 3 that the full residue feedback realizations proposed in Section 3.3 can provide lower output roundoff noise compared to the structures without residue feedback. Therefore, it would be interesting to investigate the possibility of extending the implementation in Chapter 6 to accommodate residue feedback schemes. This may require the development of new inner-product processor (extension to CSIPP proposed in Chapter 5) where the inner-product processor step can be executed in parallelism with external addition operation.

The proposed implementation in Chapter 7 is based on the block-state description of IIR digital filters. The computational complexity of this description is proportional to the square of the input block size. Another description which is called incremental block-

state has been proposed in [91] where its computational complexity is linear with respect to the input block size. However, the required computations are not suitable for direct mapping onto array processors. It would be interesting to investigate the mapping of incremental block-state onto array processor using a similar approach presented in Chapter 7.

Several VLSI implementations for IIR digital filters have been proposed in Chapters 4, 6 and 7. It would be natural extension to the work done here to fabricate these implementation on Silicon using available CAD tool and therefore they can be experimentally tested. The proposed implementations can also be tested using an available Field Programmable Gate Array (FPGA).

It has been shown in [99], [100] that the computational bottleneck barrier of IIR digital filter is due to the nature of the conventional binary arithmetic and it can be overcome by switching to existing nonconventional arithmetic such as residue arithmetic technique. Many bit-level implementations have been proposed based on these nonconventional arithmetic. However, most of the proposed implementations have been proposed for direct filter section which suffer from bad FWL effects [99], [100]. Moreover, the hardware complexity is significant since each multiplication is eventually mapped to a corresponding multiplier. It would be interesting to investigate the possibility of transforming the word-level systolic implementation of the state-space filter proposed in Chapter 6 to a bit-level implementation based on residue arithmetic. It is expected that the resulting implementation would achieve high computational speed with reasonable hardware area since the number of multipliers is linear w.r.t the filter order.

As was shown in Chapter 1, the realization of IIR digital filters using fixed-point arithmetic has a limited dynamic range which may cause overflow. This problem can be alleviated by switching to floating-point arithmetic. However, the bits assigned to the exponent in the floating-point format are bits of precision given up by the mantissa.

Block-floating-point format is an attractive compromise between the fixed-point format and the floating-point format when wordlength is limited. The roundoff error analysis of signal processing systems utilizing block-floating-point representation has been based on simulation results. It may be possible to analytically study roundoff error and data quantization using block-floating-point format. A study of the block-floating-point use in IIR digital filters could be considered. In addition, the incorporation of the residue-feedback technique to the block-floating-point system can be also investigated.

Bibliography

[1] M. Johnsdon, "*Superscalar Microprocessor Design*", Prentice Hall, Englewood cliffs, CA, 1991.

[2] J. Schutz, " A CMOS 100MHz microprocessor", *IEEE International Solid-State Circuits Conference ISSCC*, pp. 90-91, 1991.

[3] Texas Instruments, "*TMS320C3x User's Guide, Digital Signal Products*", Texas Instruments Inc., Huston, TX, 1990.

[4] Motorola Inc., "*DSP96002 IEEE floating-point dual-port processor User's Manual* ", Motorola Inc., Phoenix, AZ, 1989.

[5] K. K. Parhi and M. Hatamian, " A high sample rate recursive digital filter chip", *Proc. IEEE VLSI Signal Processing Workshop*, Monterey, CA, pp. 3-14, 1988.

[6] S. Sunder, F. El-Guibaly, and A. Antoniou, " VLSI implementation of a second-order digital filter ", *Canadian Journal of Elec. & Comp. Engineering*, vol. 19, pp. 143-147, 1994.

[7] A. Antoniou, "*Digital filters : Analysis, Design, and Applications*", McGraw-Hill, NewYork, 1993.

[8] T. Thong and Y. C. Jenq, " Ch. 11" *Handbook for digital signal processing* , S. K. Mitra and J. F. Kaiser (eds.), Wiley, New York, 1993.

[9] S. Magur, S. Shen, G. Luikuo, M. Fleming, and P. Aguilar, " An application specific DSP chip set for 100 MHz data rate" *Proc. IEEE Int. Conf. Acoustic, speech and Signal Processing,* New York, pp. 1989-1992, 1988.

[10] J. F. Kaiser, " Digital Filters ", *Systems by Digital Computers*, F. F. Kuo and J. F. Kaiser (eds.), Wiley, NY, 1966.

[11] J. F. Kaiser, " Some practical consideration in the realization of linear digital filter ", *Proc. 3rd allerton Conf. Circuit and systems Theory*, pp. 621-633, 1965.

[12] B. Widrow, " A study of rough amplitude quantization by means of Nequist sampling theory ", *IRE Trans. Circuits Theory*, vol. CT-3, pp. 266-276, Dec. 1956.

[13] S. Y. Hwang, " Roundoff noise in state-space digital filtering: A general analysis" , *IEEE Trans. Acoustics, Speech, Signal Processing*, vol. pp. 256-262, June 1976.

[14] P. W. Wong, " Quantization noise, fixed-point multiplicative roundoff noise, and dithering ", *IEEE Trans. Acoustic, Speech, and Signal Processing*, vol. 38, pp. 286-300, Feb. 1990.

[15] L. B. Jackson, " Limit cycles in state-space structures for digital filters ", *IEEE Trans. Circuits Syst.*, vol. CAS-27, pp. 67-68, Jan. 1979.

[16] R. A. Roberts and C. T. Mullis, " *Digital Signal Processing* ", Addison Wesely, Reading, MA, 1987.

[17] L. B. Jackson, " *Digital filters and signal Processing* ", Kluwer Academic, Boston, 1989.

[18] P. M. Ebert, J. E. Mazo, and M. G. Taylor, " Overflow oscillations in digital filters " , *Bell Syst. Tech. J.*, vol. 48, pp. 2999-3020, Nov. 1969.

[19] S. R. Parker and S. F. Hess, " Limit cycle oscillations in digital filters ", *IEEE Trans. Circuit Theory*, vol. CT-18, pp. 687-696, Nov. 1971.

[20] L. B. Jackson, " Roundoff-noise analysis for fixed-point digital filters realized in cascade or parallel form ", *IEEE Bell Telephone Labs,* vol. 18, pp. 107-122, June 1970.

[21] K. Meekotter, " Realization of limit cycles-free second-order digital filter ", *Proc. IEEE Int. Symp. Circuits and Syst.*, pp. 295-298, 1976.

[22] T. Claassen, W. Mecklenbrauker, and J. Peek, " Second-order digital filter with only one-magnitude truncation quantizer and having practically no limit cycles ", *Electronic Lett.*, vol. 9, No. 2, pp. 531-532, Nov. 1973.

[23] D. C. Munson and B. Liu, " Low-noise realizations for narrow-band recursive digital filters ", *IEEE Trans. Acoustics Speech, Signal Processing,* vol. ASSP-28, pp. 41-54, Feb. 1980.

[24] R. Agarwal and S. Burrus, " New recursive digital filter structures having very low sensitivity and roundoff noise ", *IEEE Trans. Circuits Syst.*, vol. CAS-22, pp. 921-927, Dec. 1975.

[25] A. I. Abu-El-Haija and A. M. Peterson, " An approach to eliminate roundoff errors in digital filters ", *IEEE Trans. Acoustics, Speech, Signal Processing,* vol. ASSP-27, pp. 195-198, Apr.1979.

[26] T. L. Chang, " Suppression of limit cycles in digital filters designed with one magnitude truncation quantizer ", *IEEE Trans. Circuits Syst.*, vol. CAS-28, pp. 107-111, Feb. 1981.

[27] T. I. Laakso and I. O. Hartimo, " Noise reduction in recursive digital filters using high-order error feedback ", *IEEE Trans. Signal Processing*, vol. 40, pp. 1096-1107, May 1992.

[28] C.T. Mullis and R. A. Roberts, " Synthesis of minimum roundoff noise fixed-point digital filters " *IEEE Trans. Circuit Syst.*, vol. CAS-23, pp. 551-561, Sept. 1976.

[29] S. Y. Hwang, " Minimum uncorrelated unit noise in state space digital filtering" , *IEEE Trans. Acoustics, Speech, Signal Processing*, vol. ASSP-25, pp. 273-281, Aug. 1977.

[30] C.W. Barnes and A.T. Fam, " Minimum norm recursive digital filter that are free of overflow limit cycles ", *IEEE Trans. Circuits syst.*, vol. Cas-24, pp. 569-574, Oct. 1977.

[31] V. Tavsanoglu and L. Thiele, " Optimal design of state-space digital filters by simultaneous minimization of sensitivity and roundoff noise ", *IEEE Trans. Circuits Syst.*, vol. CAS-31, pp. 884-888, Oct. 1984.

[32] W. L. Mills, C. T. Mullis and R. A. Roberts, " Digital filter realizations without overflow oscillations ", *IEEE Trans. Acoustics, Speech, Signal Processing*, vol. ASSP-26, pp. 334-338, Aug. 1978.

[33] M. Kawamata and T. Higuchi, " On the absence of limit cycles in a class of state-space digital filters which contains minimum noise realizations ", *IEEE Trans. Acoust. Speech, and Signal Processing*, vol. 32, pp. 928-930, Aug. 1984. .

[34] W. L. Mills, C. T. Mullis and R. A. Roberts, " Low roundoff noise and normal realizations of fixed point IIR digital filters ", *IEEE Trans. Acoustics, Speech, Signal Processing*, vol. ASSP-29, pp. 893-903, Aug. 1981.

[35] L. B. Jackson A. G. Lindgren and Y. Kim, " Optimal synthesis of second-order state-space structures for digital filters ", *IEEE Trans. Circuits Syst.* vol. CAS 26, pp. 149-153, Mar. 1979.

[36] C. W. Barnes, " Computationally efficient second-order digital filter sections with low roundoff noise gain ", *IEEE Trans. Circuit Syst.*, vol. CAS-31, pp. 841-847, Oct. 1984

[37] B. W. Bomar and J. C. Hung, " Minimum roundoff noise digital filters with some power-of-two coefficients ", *IEEE Trans. Circuits Syst.*, vol. CAS-31, pp. 833-840, Oct. 1984

[38] B. W. Bomar, " New second-order state-space structures for realizing low roundoff noise digital filters ", *IEEE Trans. Acoustics, Speech, Signal Processing,* vol. ASSP-33, pp. 106-110, Feb. 1985.

[39] B. W. Bomar, " Computationally efficient low roundoff noise second-order state-space structures ", *IEEE Trans. Circuits Syst.*, vol. CAS-33, pp.35-41, Jan. 1986.

[40] B. W. Bomar, " Computationally efficient low roundoff noise second-order digital filter sections with no overflow oscillations ", *Conf. Proceedings IEEE southeaston*, Knoxvill, TN, USA, pp. 606-613, Apr. 1989.

[41] B. W. Bomar, " On the design of second-order state-space digital filter sections " , *IEEE Trans. Circuits Syst.*, vol. CAS-36, pp. 542-552, Apr. 1989.

[42] T. Thong and B. Liu, " Error spectrum shaping in narrow-band recursive filters ", *IEEE Trans. Acoustics, Speech, Signal Processing,* vol. ASSP-25, pp. 200-203, Apr. 1977.

[43] D. Williamson, P.G. McCrea and S. Sridharan, " Residue feedback in digital filters using fractional coefficients and block floating point arithmetic", *Proc. IEEE Symp. Circuit and Systems,* CA, pp. 831-834, May 1983

[44] W. E. Higgins, and D. Munson, " Noise reduction strategies for digital filters: error spectrum shaping versus the uptimal linear state-spce formulation ", *IEEE Trans. Acoustics, Speech, Signal Processing,* vol. 30, pp. 963-973, Dec. 1983.

[45] V. Singh, " Formulation of a criterion for the absence of limit cycles in digital filters designed with one quantizer " , *IEEE Trans. Circuits Syst.,* vol. CAS-32, pp. 1062-1064, Oct. 198

[46] D. Williamson and S. Sridharan, " Residue feedback in digital filters using fractional feedbackcoefficients" , *IEEE Trans. Acoustics Speech, Signal Processing,* vol. ASSP-33, pp. 477-483, Apr. 1985.

[47] C. W. Barnes, " Error feedback in normal realization of recursive digital filters " , *IEEE Trans. Circuits Syst.,* vol. CAS-28, pp. 72-75, Jan. 1981.

[48] A.I. Abu-El Haija, " Determining coefficients of error-feedback digital filters to obtain minimum roundoff errors with minimum complexity" , *Proc. IEEE Symp. Circuit and System, CA.*, pp. 819-822, May 1983

[49] W. E. Higgins, and D. Munson, " Optimal and suboptimal error-spectrum shaping for cascade-form digital filters" , *IEEE Trans. Circuits Syst.,* vol. CAS-31, pp. 429-437, May. 1984

[50] T. I. Laakso, P. S. R. Diniz, I. Hartimo, and T. C. Macedo, " Elimination of zero-input and constant-input limit cycles in single-quantizer recursive filter structures ", *IEEE Trans. Circuits Syst*-II: *Analog and Digital, Signal Processing.*, vol. 39, pp. 638-646, Sept. 1992.

[51] D. Williamson, " Roundoff noise minimization and pole-zero sensitivity in fixed-point digital filters using residue feedback " , *IEEE Trans. Acoustics, Speech, Signal Processing,* vol. ASSP-34, pp. 1210-1220, Oct. 1986.

[52] S. Y. Kung, " *VLSI Array Processors* ", Prentice Hall, Englewood Cliffs, NJ, 1988.

[53] H. T. Kung, " Why systolic architectures?" , *IEEE Computer*, vol. 15, pp. 37-46, Jan. 1982.

[54] F. El-Guibaly, S. Sunder, and A. Antoniou, " Systolic implementation of FIR filters ", *Signal Processing V: Theories and Applications*, L. Torres, E. Masgrau, and M. A. Lagunas (eds.), Elsevier, NY, pp. 1423-1426, 1990.

[55] S. K. Rao and T. Kailath, " Regular iterative algorithms and their implementation on processor arrays", *Proc. IEEE*, vol. 76, pp. 259-269, Mar. 1988.

[56] F. El-Guibaly and A. Tawfik, " Mapping 3-D IIR digital filter onto systolic array", to be appear in *Multidimensional Systems and Signal Processing.*

[57] P. Quinton, " The systematic design of systolic arrays ", *IRISA Reseaech Report,* No. 193, Apr. 1983.

[58] C. S. Burrus, " Block implementation of digital filters", *IEEE Trans. Circuit Theory*, vol. CT-18, pp. 697-701, Nov. 1971.

[59] C. W. Barnes and S. Shinneka, " Block shift invariance and block implementation of discrete-time filter", *IEEE Trans. Circuits Syst.*, vol. CAS-27, pp. 667-672, Aug. 1980.

[60] P. P. Vaidyanathan and V. Liu, " An improved sufficient condition for absence of limit cycles in digital filters", *IEEE Trans. Circuits Syst.*, vol. CAS-34, pp. 319-322, March 1987.

[61] A. Tawfik, P. Agathoklis and F. El-Guibaly, " A software tool for analyzing finite wordlength effects in fixed-point implementation", *IEEE Pacific Rim*

Conferenceon Comm., Comp., and Signal Processing, Victoria, B.C., pp. 116-119, May 1993.

[62] D. Williamson, S. Sridharan and P.G. McCrea, A new approach for block floating-point arithmetic in recursive filters, *IEEE Trans. Circuits Syst.,* vol. CAS-32, pp. 719-722, July 1985.

[63] D. Williamson and S. Sridharan " Implementation of high-order direct-form digital filter structures", *IEEE Trans. Circuit Syst.* vol. CAS-33, pp. 818-822, Aug. 1986.

[64] M. Renfors, " Roundoff noise in error-feedback state-space filters", *Proc. Int. Conf. Acoust. Speech, Signal Processing (ICASSP'83)*, pp. 619-622, Apr. 1983.

[65] C. T. Mullis and R. A. Roberts, " An interpretation of error spectrum shaping in digital filters" , *IEEE Trans. Acoust. Speech, and signal Processing,* vol. 30, pp. 1013-1015, Dec. 1982.

[66] T. L. Chang, " A low roundoff noise digital filter structure", *in Proc. IEEE Int. Symp. Circuit Syst.*, NY, pp. 1004-1008, May 1978.

[67] P.P. Vaidyanathan, " On error-spectrum shaping in state-space digital filters ", *IEEE Trans. Circuits Syst.,* vol. 32, pp. 88-92, Jan. 1985.

[68] L. M. Smith and B. W. Bomar, " An algorithm for constraint roundoff noise minimization in digital filters with application to two-dimensional filters ", *IEEE Trans. Circuits Syst.,* vol. CAS-35, pp. 1359-1368, Nov. 1988.

[69] A. Tawfik, P. Agathoklis, and F. El-Guibaly, " New realization and Implementation of IIR digital filters using residue feedback ", *Asilomar Conf. signals, Syst., and Comp.*, Pacific Grove, CA, pp. 167-171, Oct. 1994.

[70] Motorola, "DSP56000 User Guide", 1986.

[71] A. Antoniou and M. Rezek, " A comparison of cascade and wave fixed-point digital-filter structures", *IEEE Trans. Circuits Syst.*, vol. CAS-27, pp. 1184-1193, Dec. 1980.

[72] M. Renfors and Y. Neuvo, " The maximum sampling rate of digital filters under hardware speed constraints", *IEEE Trans. Circuits Syst.*, vol. CAS-28, pp. 196-202, Mar. 1981.

[73] H. H. Lu, E. A. Lee, and D. G. Messerschmitt, " Fast recursive filtering with multiple slow processing elements", *IEEE Trans. Circuits Syst.*, vol. CAS-32, pp. 1119-1129, Nov. 1985.

[74] M. O. Ahmad and D. V. Poornalah "Design of an efficient VLSI inner-product processor for real-time DSP applications", *IEEE Trans. Circuits Syst.*, vol. CAS-36, pp. 324-329, Feb. 1989.

[75] J. R. Jump and S. R. Ahuja, "Effective pipelining of digital systems", *IEEE Trans. Computers*, vol. C-27, pp. 855-865, Sept. 1978.

[76] H. T. Kung, "Systolic algorithms for the CMU WARP processor", *in Proc. IEEE Int. Conf. on Pattern Recognition*, pp. 570-577, 1984.

[77] S. Sunder, "*VLSI Implementation of Digital Filters*", Ph.D. Dissertation, University of Victoria, 1992.

[78] M. Hatamian and G. L. Cash, "A 70-MHZ 8×8 parallel pipelined multiplier in 2.5 μm CMOS," *IEEE J. Solid-State Circuits*, Vol. Sc-21, pp. 505-513, Aug. 1986.

[79] A. Tawfik, F. El-Guibaly, M. Fahmi, E. Abdel-Raheem, and P. Agathoklis, " High-speed area-efficient inner-product processor", *Canadian Journal of Electrical Engineering*, pp. 187-191, Oct. 1994.

[80] K. Hwang, *"Computer Arithmetic: Principles architecture, and Design"*, John Wiely, NewYork, 1979.

[81] N. E. Weste and K. Eshraghian, *"Principles of CMOS VLSI Design: A system Perspectives"*, Addison-Wesley, 1993.

[82] M. Fahmi, *" Inner product processor design for DSP"*, M.Sc. Thesis, University of Victoria, Nov. 1994.

[83] W. Luk and G. Jones, " Systolic recursive filters", *IEEE Trans. Circuits Syst.*, vol. CAS-35, pp. 1067-1068, . 1988.

[84] Jayadeva, "A new systolic design for digital IIR filters", *IEEE Trans. Circuits Syst.*, vol. CAS-37, pp. 653-654, 1990.

[85] Shaw-Min Lei and Kung Yao, " Efficient systolic array implementations of IIR digital filtering", *IEEE Trans. Circuits Syst.*, vol. CAS-39, pp. 581-584, Aug. 1992

[86] Arjmand and R. A. Roberts, "On comparing hardware implementations of fixed point digital filters", *IEEE Magazine*, vol. 3, pp. 2-8, June 1981.

[87] H. H. Loomis and B. Sinha, " High speed recursive digital filter realization", *Circuits, Syst., Signal Processing*, vol. 3, pp. 267-294, 1984.

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

[89] C. L. Nikias, " Fast block data processing via a new IIR digital filter structure", *IEEE Trans.* Acoustics, Speech, Signal Processing, vol. ASSP-32, pp. 770-779 Aug. 1984.

[90] K. Hatashi, K. Dhar, K. Sugahara and K. Hirano,''Design of high-speed digital filters suitable for multi-DSP implementation", *IEEE Trans. Circuits Syst.*, vol. CAS-33, pp. 202-217, Feb. 1986

[91] K. K. Parhi and D. G. Messerschmitt, '' Pipeline interleaving and Parallelism in recursive digital filters-Part II: Pipelined incremental block filtering", *IEEE Trans.* Acoustics, Speech, Signal Processing, vol. 37, pp. 1118-1134, July 1989.

[92] B. G. Mertzios and V. L. Syrmos, '' Implementation of digital filters via VLSI array processor", *IEE Proceedings*, vol. 135, Part G, pp. 78-82, Apr. 1988.

[93] K. K. Parhi and D. G. Messerschmitt, '' A bit-parallel bit level recursive filter architecture", *in Proc. IEEE Int. Conf. Comput. Design*, New York, pp. 284-289, 1986.

[94] S. Ghanekar, *Realization and Architectures for Digital Filters Using Periodically Time-Varying Systems,* Ph.D Dissertation, University of Massachusettes, 1993.

[95] J. Zeman and A. G. Lindgren,'' Fast digital filters with low round-off noise", *IEEE Trans. Circuits Syst.*, vol. CAS-28, pp. 716-723, July 1981.

[96] S. Mitra and R. Ganasekaran,''Block implementation of recursive digital filters-new structures and properties", *IEEE Trans. Circuits Syst.*, vol. CAS-25, pp. 200-207, Apr. 1978.

[97] C. W. Barnes and S. Shinnaka, ''Finite word effects in block-state realizations of fixed point digital filters", *IEEE Trans. Circuits Syst.*, vol. CAS-27, pp. 200-207, May 1980.

[98] K.S. Arun and D.R. Wagner, "High-speed digital filtering: structures and finite wordlength effects", *Journal of VLSI Signal Processing*, pp. 355-370, Apr. 1992.

[99] M. Lapointer, P. Fortier and H. T. Huynh, "A new faster and simpler systolic structure for IIR filters", *Proc. IEEE Int. Symposium on Circuits and Syst. (ISCAS)*, New Orleans, pp. 1227-1230, 1990.

[100] O.C. McNally, J.V. McCanny and R.F. Woods, " A 40 Megasamples IIR Filter Chip", *Application Specific Array Processor*, 1991.

Appendix A

We will show how the noise gain in (3.11) has been obtained. The noise gain g^2 can be expressed as

$$g^2 = E\{y_e(n)y_e^t(n)\}/\sigma_r^2 = E\{\boldsymbol{Q}_1 + \boldsymbol{Q}_2 + \boldsymbol{Q}_3 + \boldsymbol{Q}_4\}/\sigma_r^2 + \boldsymbol{C}\boldsymbol{C}^t \qquad \text{(A.1)}$$

where E is the statistical average and

$$\boldsymbol{Q}_1 = \left[\boldsymbol{C}\sum_{l=0}^{\infty}\boldsymbol{A}^l(\boldsymbol{A}_1-\boldsymbol{A})\,\boldsymbol{e}(n-l-1)\right]\left[\sum_{l=0}^{\infty}\boldsymbol{e}^t(n-i-1)\,(\boldsymbol{A}_1-\boldsymbol{A})^t\left(\boldsymbol{A}^l\right)^t\boldsymbol{C}^t\right] \qquad \text{(A.2)}$$

$$\boldsymbol{Q}_2 = \left[\boldsymbol{C}\sum_{l=0}^{\infty}\boldsymbol{A}^l(\boldsymbol{A}_1-\boldsymbol{A})\,\boldsymbol{e}(n-l-1)\right]\left[\sum_{l=0}^{\infty}\boldsymbol{e}^t(n-i-2)\,(\boldsymbol{A}_2)^t\left(\boldsymbol{A}^l\right)^t\boldsymbol{C}^t\right] \qquad \text{(A.3)}$$

$$\boldsymbol{Q}_3 = \left[\boldsymbol{C}\sum_{l=0}^{\infty}\boldsymbol{A}^l\boldsymbol{A}_2\boldsymbol{e}(n-l-2)\right]\left[\sum_{l=0}^{\infty}\boldsymbol{e}^t(n-i-1)\,(\boldsymbol{A}_1-\boldsymbol{A})^t\left(\boldsymbol{A}^l\right)^t\boldsymbol{C}^t\right] \qquad \text{(A.4)}$$

$$\boldsymbol{Q}_4 = \left[\boldsymbol{C}\sum_{l=0}^{\infty}\boldsymbol{A}^l\boldsymbol{A}_2\boldsymbol{e}(n-l-2)\right]\left[\sum_{l=0}^{\infty}\boldsymbol{e}^t(n-i-2)\boldsymbol{A}_2^t\left(\boldsymbol{A}^l\right)^t\boldsymbol{C}^t\right] \qquad \text{(A.5)}$$

Consider the statistical average of (A.2)

$$E\{\boldsymbol{Q}_1\} = \boldsymbol{C}\sum_{l=0}^{\infty}\sum_{l=0}^{\infty}\boldsymbol{A}^l(\boldsymbol{A}_1-\boldsymbol{A})\,E\{\boldsymbol{e}(n-l-1)\boldsymbol{e}^t(n-i-1)\}\,(\boldsymbol{A}_1-\boldsymbol{A})^t\left(\boldsymbol{A}^l\right)^t\boldsymbol{C}^t$$

The noise error samples have been assumed to be uncorrelated with each other. Using (3.6)

$$E\{\boldsymbol{Q}_1\} = \sigma_r^2 \boldsymbol{C} \sum_{i=0}^{\infty} \boldsymbol{A}^i (\boldsymbol{A}_1 - \boldsymbol{A}) (\boldsymbol{A}_1 - \boldsymbol{A})^t \left(\boldsymbol{A}^i\right)^t \boldsymbol{C}^t \tag{A.6}$$

(A.6) can be simplified and expressed as

$$E\{\boldsymbol{Q}_1\} = \sigma_r^2 \text{trace}(\boldsymbol{S}_1 \boldsymbol{W}) \tag{A.7}$$

where $\boldsymbol{S}_1$ is defined in (3.11) and $\boldsymbol{W}$ is defined as

$$\boldsymbol{W} = \sum_{i=0}^{\infty} \left(\boldsymbol{C}\boldsymbol{A}^i\right)^t \boldsymbol{C}\boldsymbol{A}^i \tag{A.8}$$

The noise matrix $\boldsymbol{W}$ can also be calculated from (1.14). By following the same procedure, all the terms in (A.1) can be derived and (3.11) can be obtained.

Appendix B

In this appendix, the positive definiteness (P.D.) of the residue matrices P defined in (3.14), (3.17), (3.22), (3.27), and (3.30) are discussed.

Lemma 1

Let $\{A,B,C,d\}$ be a minimal realization of a stable $H(z)$, then the residue matrices P defined in (3.14) and (3.17) are positive definite.

Proof: For any minimal realization $\{A,B,C,d\}$ of $H(z)$, the noise matrix W in (1.14) is P.D. and can be expressed as

$$W = V^t V \tag{B.1}$$

where $det(V) \neq 0$. Therefore (1.14) can be expressed as

$$V^t V = A^t V^t V A + C^t C$$

or

$$I = \left(V^{-t} A^t V^t\right)\left(V A V^{-1}\right) + \left(V^{-t} C^t\right)\left(C V^{-1}\right)$$

$$I = \tilde{A}^t \tilde{A} + \tilde{C}^t \tilde{C} \tag{B.2}$$

where $\tilde{A} = VAV^{-1}$ and $\tilde{C} = CV^{-1}$. The residue matrix P defined in (3.14) can be expressed as

$$P = W - A^t W + W - WA$$

$$= V^t V - A^t V^t V + V^t V - V^t V A$$

Therefore,

$$V^{-t} P V^{-1} = I - \tilde{A}^t + I - \tilde{A}$$

$$= (I - \tilde{A})^t (I - \tilde{A}) + \left(I - \tilde{A}^t \tilde{A} \right) \qquad \text{(B.3)}$$

By using (B.2), (B.3) can be written as

$$V^{-t} P V^{-1} = (I - \tilde{A})^t (I - \tilde{A}) + \tilde{C}^t \tilde{C} \qquad \text{(B.4)}$$

The RHS of (B.4) is guaranteed to be P.D. and therefore P defined in (3.14) is P.D. The case of P defined in (3.17) can be proved to be a P.D. in a similar way.

Lemma 2

Given any minimal realization $\{A,B,C,d\}$ of a stable Nth-order $H(z)$ and $J_0 = \begin{bmatrix} \pm I_{r \times r} & 0 \\ 0 & \mp I_{(N-r) \times (N-r)} \end{bmatrix}$ where $0 < r < N$, the residue matrix P defined in (3.22) is guaranteed to be positive definite provided that $\lambda_{max}\left(\frac{\tilde{A} J_1 + J_1^t \tilde{A}^t}{2} \right) < 1$ where $J_1 = V J_0 V^{-1}$, $\tilde{A} = V A V^{-1}$ and V id defined in (B.1).

Proof: Since $\{A,B,C,d\}$ is a minimal realization of a stable $H(z)$, then W is P.D. and can be expressed as in (B.1).The residue matrix P defined in (3.18) can be written as

$$P = W - J_0^t A^t W + W - W A J_0$$

$$= V^t V - J_0^t A^t V^t V + V^t V - V^t V A J_0$$

Therefore,

$$V^t P V^{-1} = I - \left(V^{-t} J_0^t V^t \right) \left(V^{-t} A^t V^t \right) + I - \left(V A V^{-1} \right) \left(V J_0 V^{-1} \right)$$

$$= 2I - J_1^t \tilde{A}^t - \tilde{A} J_1$$

where $J_1 = VJ_0V^{-1}$ and $\tilde{A} = VAV^{-1}$.

Thus, P defined in (3.22) is guaranteed to be P.D. if $\lambda_{max}\left(\frac{\tilde{A}J_1 + J_1^t\tilde{A}^t}{2}\right) < 1$.

Lemma 3

Let $\{A,B,C,d\}$ be a minimal realization of a stable $H(z)$, then the residue matrix P defined in (3.27) is positive definite.

Proof: Since $\{A,B,C,d\}$ is a minimal realization of stable $H(z)$, then W is P.D. and can be expressed as in (B.1). P defined in (3.27) can be written as

$$P = 2V^tV - V^tVA^2 - \left(A^2\right)^tV^tV$$

Therefore,

$$\begin{aligned} V^{-t}PV^{-1} &= 2I - \tilde{A}^2 - \left(\tilde{A}^2\right)^t \\ &= \left(I-\tilde{A}^2\right)^t\left(I-\tilde{A}^2\right) - \left(\tilde{A}^2\right)^t\tilde{A}^2 + I \end{aligned} \qquad \text{(B.5)}$$

By using (B.2), (B.5) can be written as

$$V^{-t}PV^{-1} = \left(I-\tilde{A}^2\right)^t\left(I-\tilde{A}^2\right) + \tilde{C}^t\tilde{C} + \tilde{A}^t\tilde{C}^t\tilde{C}\tilde{A} \qquad \text{(B.6)}$$

where $J_1 = VJ_0V^{-1}$, $\tilde{A} = VAV^{-1}$ and V id defined in (B.1). The RHS of (B.6) is guaranteed to be P.D. and therefore P defined in (3.27) is P.D.

Lemma 4

Given any minimal realization $\{A,B,C,d\}$ of a stable $H(z)$, then P defined in (3.30) is guaranteed to be P.D. provided that $\lambda_{min}\left(R + R^t\right) > \gamma^2$ 2 where $R = (\gamma I - \tilde{A})^2$, $\tilde{A} = VAV^{-1}$ and V is defined in (B.1).

Proof: Since $\{A,B,C,d\}$ is a minimal realization of stable $H(z)$, then W is P.D. and can be expressed as in (B.1). P defined in (3.27) can be written as

$$\boldsymbol{P} = \left(2+\gamma^2\right)\boldsymbol{W} - 2\gamma\boldsymbol{A}^t\boldsymbol{W} - 2\gamma\boldsymbol{W}\boldsymbol{A} + \left(\boldsymbol{A}^t\right)^2\boldsymbol{W} + \boldsymbol{W}\boldsymbol{A}^2$$

$$= \left(2+\gamma^2\right)\boldsymbol{V}^t\boldsymbol{V} - 2\gamma\boldsymbol{A}^t\boldsymbol{V}^t\boldsymbol{V} - 2\gamma\boldsymbol{V}^t\boldsymbol{V}\boldsymbol{A} + \left(\boldsymbol{A}^t\right)^2\boldsymbol{V}^t\boldsymbol{V} + \boldsymbol{V}^t\boldsymbol{V}\boldsymbol{A}^2$$

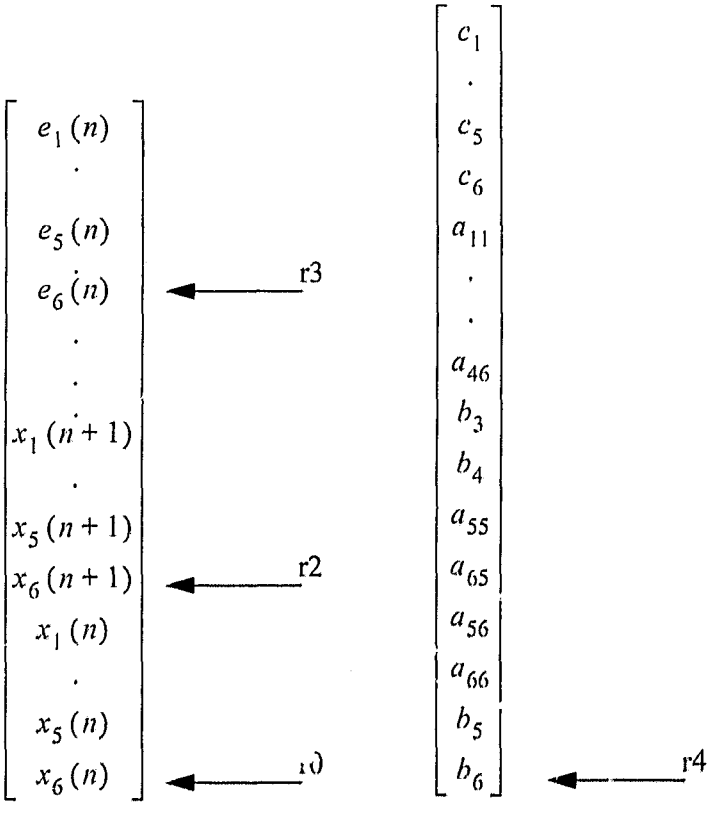
Therefore

$$\boldsymbol{V}^{-t}\boldsymbol{P}\boldsymbol{V}^{-1} = (\gamma\boldsymbol{I} - \tilde{\boldsymbol{A}})^2 + \left((\gamma\boldsymbol{I} - \tilde{\boldsymbol{A}})^t\right)^2 - \gamma^2\boldsymbol{I} + 2\boldsymbol{I}$$

Thus, $\boldsymbol{P}$ defined in (3.30) is guaranteed to be P.D. provided that
$\lambda_{min}\left((\gamma\boldsymbol{I} - \tilde{\boldsymbol{A}})^2 + \left((\gamma\boldsymbol{I} - \tilde{\boldsymbol{A}})^t\right)^2\right) > \gamma^2 - 2$.

Appendix C

In this appendix, the list of DSP56001 Assembly code to implement a 6th-order LPF based on the SRF scheme 2 realization is listed. The SPECTRUM board based on DSP56001 has been used.



```
;       sixth-order one-section LPF with 70k sampling rate
;       The program is based on SRF scheme 2 realization discussed in Chapter 3
;       Links LK1a, LK2a AND LK9b must be present.

output          EQU     $FFC1
state_adr       EQU     $0000
coef_adr        EQU     $0000
new_state       EQU     $0006
resi_adr        EQU     $0040
temp_adr        EQU     $0049
Channel0        EQU     $FFC0
timer           EQU     $FFC3
speed           EQU     $FF72
filt_ord        EQU     6
half_ord        EQU     3
no_coef         EQU     36
dcgain          EQU     $
        org y:coef_adr
    dc
    ..
    dc
        org x:state_adr
    dc
    ..
    dc
        org x:new_state
    dc
    ..
    dc
        org x:resi_adr
    dc
    ..
    dc
```

```
        org y:temp_adr
    dc      2
    dc      4
    dc      6
;   Set INERUPT VECTOR FOR IRBQ
        org p:$A
        jsr  i_s_r
;   Main Program
        org p:$60
;   Set IO WAIT STATES =2
        movep #$2,x:$FFFE
;   Initialize pointers
        move  #state_adr,r0
        move  #state_adr,r6
        move  #resi_adr,r3
        move  #resi_adr,r1
        move  #new_state,r2
        move  #coef_adr,r4
        move  #temp_adr,r5
;   Set linear modulo address
        move  #2*filt_ord-1,m0
        move  #2*filt_ord-1,m2
        move  #filt_ord-1,m3
        move  #filt_ord-1,m1
        move  #no_coef-1,,m2
        move  #half_ord-1,m4
        move  #2*filt_ord-1,m6
        move  #filt_ord,n6
;   Unmask edge triggered interrupt
;   and wait for interrupt to occur
        movep #$FC3C, x:$FFFF
```

```
        ;       Start timer running at 70kHz

                movep  #speed,y:timer
main            jmp main

i_s_r           movep #$20,y:channel0
                movep  y:output,y1
                move  y:(r4)+,y0
                move  #$0,x1
                do #half_ord,_firsloop
                clr a
                clr b
        ;       Calculate B*u term

                mac y1,y0,a     y:(r4)+,y0
                mac y1,y0,b     y:(r4)+,y0      x:(r0)+,x0
        ;       Calculate A*x  term in blocks of two states

                do y:(r5),_secloop
                mac x0,y0,a     y:(r4)+,y0
                mac x0,y0,b     y:(r4)+,y0      x:(r0)+,x0
_secloop
        ;       The incorporation of the residue part

                move x:(r1)+,x0
                add x,a
                move x:(r1)+,x0
                add x,b
                move b,L:$1234
                clr b
        ;       The calculation of the residue part

                do #2,_quant
                move x:(r1)+,x0
                add a,b
                move #$0,a0
                sub a,b      a,x:(r2)+
                move b0,x:(r3)+
```

```
                clr b
                move L:$1234,a
_quant
                move (r5)+
                move (r6),r0
_firsloop
        ;       The calculation of C*x term
                move x:(r0)+,x0
                do #filt_ord,_outp
                mac x0,y0,b     y:(r4)+,y0      x:(r0)+,x0
_outp
        ;       The calculation of d*u
                move #dcgain,x1
                mac x1,y1,a
        ;       To output the data
                movep #$20,y:channel0
                movep a,y:output
        ;       To adjust the pointers
                move (r6)+n6
                move (r4)-
                move (r0)-
                rti
        ;       The end of INERRUPT service routines
```

VITA

Surname: Tawfik Given Names: Ayman
Place of Birth: Damiatte, Egypt Date of Birth: Nov. 21, 1960

Educational Institutions Attended:

University of Victoria, Victoria, Canada	1991-1995
Ain Shams University, Cairo, Egypt	1978-1989

Degrees Awarded:

M. Sc.	1989	Ain-Shams University
B. Sc.	1983	Ain-Shams University

Honours and Awards:

Distinction with Honor degree, (1983) B.Sc. in electrical engineering.
Egyptian government excellence scholarship (1979-1983).

Publications:

Journal Publications

[1] A. Tawfik, P. Agathoklis, and F. El-Guibaly, " New low roundoff noise realization of second-order IIR digital filter", *IEEE Trans. Signal Processing*, pp 1255-1258, May 1995.

[2] A. Tawfik, F. El-Guibaly and P. Agathoklis, " Systolic implementation of fixed-point state-space digital filter ", accepted in *IEE Pt. G*, April, 1995.

VITA. Continued

Publications, Continued:

[3] A. Tawfik, F. El-Guibaly, M. Fahmi, E. Abdel-Raheem, and P. Agathoklis, " High-speed area-efficient inner-product processor", *Canadian Journal of Electrical Engineering*, pp. 187-191, Oct. 1994.

[4] F. El-Guibaly and A. Tawfik, " Mapping 3-D IIR digital filter onto systolic array", accepted in *Multidimensional systems and signal processing*, Sept. 1994.

Conference Publications

[1] A. Tawfik, P. Agathoklis, and F. El-Guibaly, " New residue-feedback state-space IIR digital filter realizations, *IEEE Int. Symp. Circuits, Sys.*, Seattle, WA, pp. 61-64, April 1995.

[2] E. Abdel-Raheem, a. Tawfik, F. El-Guibaly, and A. Antoniou, " A new inner-product processor for FIR filter implementation", *IEEE Pac. Rim Conf. Comm., Comp., Signal Processing*, pp. 395-389, Victoria, B.C., May 1995.

[3] A. Tawfik, P. Agathoklis, and F. El-Guibaly, " New realization and implementation of IIR digital filters using residue feedback technique", *Asilomar Conf. Signals, sys., Co. ..*, Pacific Grove, CA, USA, pp. 167-171, Oct. 1994.

[4] A. Tawfik, F. El-Guibaly and P. Agathoklis, " VLSI array processors implementation of block-state digital filters", *IEEE Int. Symp. circuits, Sys.*, London, UK, vol. 4, pp. 267-270, May 1994.

[5] F. El-Guibaly, A. Tawfik, E. Abdel-Raheem, and M. Fahmi, " systolic arrays for 2-D filters using a computational geometric approach", *Int. Conf. Microelectronics*, Dhahran, Saudia Arabia, pp. 185-188 , Dec. 1993.

[6] E. Abdel-Raheem, F. El-Guibaly, A. Tawfik, and A. Antoniou, " Novel systolic implementation of multirate polyphase filters", *Int. Conf. Microelectronics*, Dharhan, Saudia Arabia, pp. 193-196, Dec. 1993.

VITA, Continued

Publications, Continued:

[7] F. El-Guibaly, A. Tawfik, E. Abdel-Raheem, M. Fahmi, and W. -S. Lu, " 2-D IIR filters architecture using computational geometry concepts", *Proc. CCVLSI,* pp. 7-22, 7-27, Banff, Alta, Nov. 1993.

[8] A. Tawfik, F. El-Guibaly, and P. Agathoklis, " Efficient systolic implementation of fixed-point state-space digital filter", *Proc. Canadian conf. Elec. Comp. Eng.,* Vancouver, B.C., pp. 39-42, Sept. 1993.

[9] A. Tawfik, P. Agathoklis and F. El-Guibaly, " New low roundoff noise realization of second-order digital filter sections with coefficients which are sum of power-of-two", *Proc. 36th Midwest Symp. Circuits, Sys.* Detriot, Michigan, pp. 272-275, Aug. 1993.

[10] A. Tawfik, P. Agathoklis and F. El-Guibaly, " A tool for analyzing finite wordlength effects in fixed-point digital filter implementation", *Proc. IEEE Pac. Rim. Conf. Comm. Comp. signal Processing,* Victoria, B.C., pp. 116-119, May 1993.

Other Publications

[1] A. Tawfik, P. Agathoklis and F. El-Guibaly, " Finite word length effects in fixed-point IIR digital filters", *Technical Report VMC-93-1,* ECE Dept., University of Victoria, B.C., 1993.

[2] F. El-Guibaly, A. Tawfik, W.-S. Lu, and E. Abdel-Raheem, " Synthesising systolic arrays: basic tools", *Technical Report VMC-92-2,* ECE Dept., University of Victoria, B.C., 1993.

[3] A. Tawfik, " Digital spectrum analyzer using the polyphase technique", *Master Thesis,* Electronics and Computer Engineerings Dept., Ain-shams University, Cairo, Sep. 1989.

Partial Copyright License

I hereby grant the right to lend my thesis (or dissertation) to users of the University of Victoria Library, and to make single copies only for such users or in response to a request from the Library of any other university, or similar institution, on its behalf or for one of its users. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by me or a member of the University designated by me. It is understood that copying or publication of this dissertation for financial gain shall not be allowed without my written permission.

Title of Thesis/Dissertation:

REALIZATION AND IMPLEMENTATION OF STATE-SPACE IIR DIGITAL FILTERS

Author

AYMAN EL-SAYED TAWFIK

June 27, 1995