Journal of VLSI Signal Processing, 4, 355–370 (1992)
© 1992 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

High-Speed Digital Filtering: Structures and Finite Wordlength Effects

K.S. ARUN* AND D.R. WAGNER

University of Illinois at Urbana-Champaign, Coordinated Science Laboratory, 1101 W. Springfield Avenue, Urbana, IL 61801

Received December 15, 1989; Revised January 30, 1992.

Abstract. This paper is a study of high-throughput filter structures such as block structures and their behavior in finite precision environments. Block structures achieve high throughput rates by using a large number of processors working in parallel. It has been believed that block structures which are relatively robust to round-off noise must also be robust to coefficient quantization errors. However, our research has shown that block structures, in fact, have high coefficient sensitivity. A potential problem that arises as a result of coefficient quantization is a periodically time-varying behavior exhibited by the realized filter. We will demonstrate how finite wordlength errors can change a nominally time-invariant filter into a time-varying system. We will identify the block structures that have low coefficient sensitivity, and develop high-speed structures that are immune to the time-varying problems caused by coefficient quantization.

1. Introduction

Block realizations of digital filters generate a block of outputs at a time, were developed as early as 1970 by Voelcker and Hartquist [1] and Burrus [2], and were studied by Mitra and Gnanashekharan [3]. Because of VLSI technology and the needs of modern signal and image processing, there has been a recent resurgence of interest in high-speed filter strucures [4], [5], [6]. Block realizations of digital filters were first proposed as low-noise filter structures. They require more hardware than conventional structures, but also have the ability to handle higher data rates. Given sufficient hardware, by utilizing multiple processors working in parallel, block realizations can concurrently process a block of data in each processor-arithmetic cycle, and thus handle higher throughput rates than conventional structures. It is known that corresponding to every sequential implementation of digital filters, there exists a block implementation that uses a far larger number of processors [3], and processes a block of data in every processor cycle. As a result of the big strides made in the last decade in integrated circuit technology, and the accompanying reduction in cost and physical size of hardware, block structures have become technologically feasible. At the same time, modern signal and image processing applications have been making ever-increasing demands on throughput rates. Thus these block filter structures that were proposed 15–20 years earlier have become more relevant now. Apart from block structures, which are parallel architectures, pipelined structures employing pipelined multiplier units have also been proposed for high-speed VLSI digital filtering.

A detailed round-off noise analysis of block structures was undertaken by Barnes and Shinnaka in 1980 [7], who demonstrated their low round-off noise properties. The analysis by Barnes and Shinnaka [3] indicates that in general, block implementations of recursive or IIR filters are more robust than their sequential counterparts. Heuristically, this may be explained by their observation that the internal modes of the block implementation are much closer to the origin than the internal modes of the sequential implementation. Lost in the bargain is processor utilization. In all block implementations of IIR digital filters, processor utilization is fairly low, the biggest culprit being the block version of the general, noncanonical state-space realization where the state feedback matrix is full. In comparison, the block implementation of direct-form filters (see figure 3) is fairly efficient and uses a minimal number of processing elements. The trade-off however, is in finite precision behavior. For conventional structures, it is known that structures with low round-off noise also have low sensitivity to coefficient quantization errors. Therefore, it has been believed that block structures, which are relatively robust to round-off noise, must also be robust to coefficient quantization errors. However,

*Currently at the University of Michigan, EECS Department, Ann Arbor, MI 48103.

block structures implicitly depend on pole-zero cancellations, and the cancellations may not occur in the presence of coefficient (quantization) errors. These spurious pole-zero cancellations in the block implementation of the direct-form filter are fertile sources of finite precision errors. Coefficient quantization errors in block structures can also change a nominally time-invariant filter to a time-varying one. These effects will be pointed out and robust block structures that do not suffer from this problem will be presented.

This paper is a study of various high-speed filter structures and their finite-precision behavior. We will call a filter structure a high-speed structure if its throughput rate is not bounded by its processors' multiplication cycle time. These structures must be able to process multiple samples in the time required for one multiplication. In this paper, we investigate the finite-precision behavior of such high-speed filter structures. In the next section, we will review the more well-known block structures. While these structures were originally derived using a state-space approach, we will adopt an equivalent but perhaps simpler, difference-equation approach. The block structure for nonrecursive (or FIR) filters is more easily derived by this approach. Section 3 presents the finite-precision analysis for these structures. Coefficient quantization effects on filter stability and the coefficient sensitivity of the transfer function are also studied here. The time-varying behavior of block structures is demonstrated in this section. In Section 4, robust structures are proposed that do not exhibit this time-varying anomaly caused by finite word-length effects.

2. Block Structures

Block filtering is a method of speeding up data throughput in a digital filter by using a large number of processors operating concurrently. Systolic and wavefront implementations of digital filters, both FIR and IIR, achieve a maximum throughput rate of one output per multiply-add (MAD) cycle [8], [9]. They use at most, twice as many processors as the filter order. For filters of low order, they are not able to exploit parallelism to achieve high throughput rates. By using a large number of processors, many times larger than the filter order, block filters produce several outputs per MAD cycle. All block filter structures have a serial-to-parallel converter at the input end, that takes a sequentially arriving input and presents it to the array as a block input, and a parallel-to-serial converter at the output

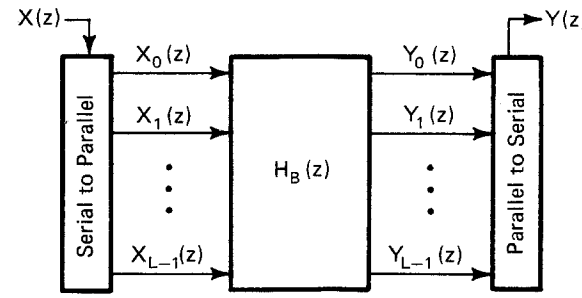

Fig. 1. The generic block realization of a digital filter.

end, that reconverts the block output to a sequential output for the outside world (see figure 1). Many of the structures presented here or similar ones have been proposed earlier [1]–[7].

2.1. Block FIR Structure

The block structure for convolution or nonrecursive (FIR) filtering is easily derived by noting the parallelism inherent in convolution. Different samples of the output sequence from a convolution can be produced independent of each other, owing to a lack of recursive dependence in the output of a convolution. Given a block of input data and a sufficient number of processors, multiple outputs can be produced in the same MAD cycle since past outputs are not part of the convolution sum.

Assume the filter being implemented is of order q, and our objective is to compute L outputs in one MAD cycle. The required architecture can be determined by examining the finite convolution sum:

$$y(n) = \sum_{m=0}^{q} b_m u(n - m)$$

and noting the similarity of this equation to the operation of multiplying two numbers in binary representation. Multiplication is essentially a convolution of two sequences of bits, and FIR filtering is a convolution of two sequences of numbers. Array multipliers have been used for fast, highly parallel multiplication in one clock cycle [10]. Borrowing the idea and applying it to FIR filtering, leads to the following *array convolver* of figure 2—a Block FIR filter structure that generates a block of outputs per MAD cycle [11]. In figure 2, the filter coefficients are $b_0, b_1, \ldots, b_q$; the input and output sequences are x and y respectively, and they may be infinitely long. The input sequence is shifted into the shift register, and at the start of a MAD cycle, the contents of the input shift register are loaded into the

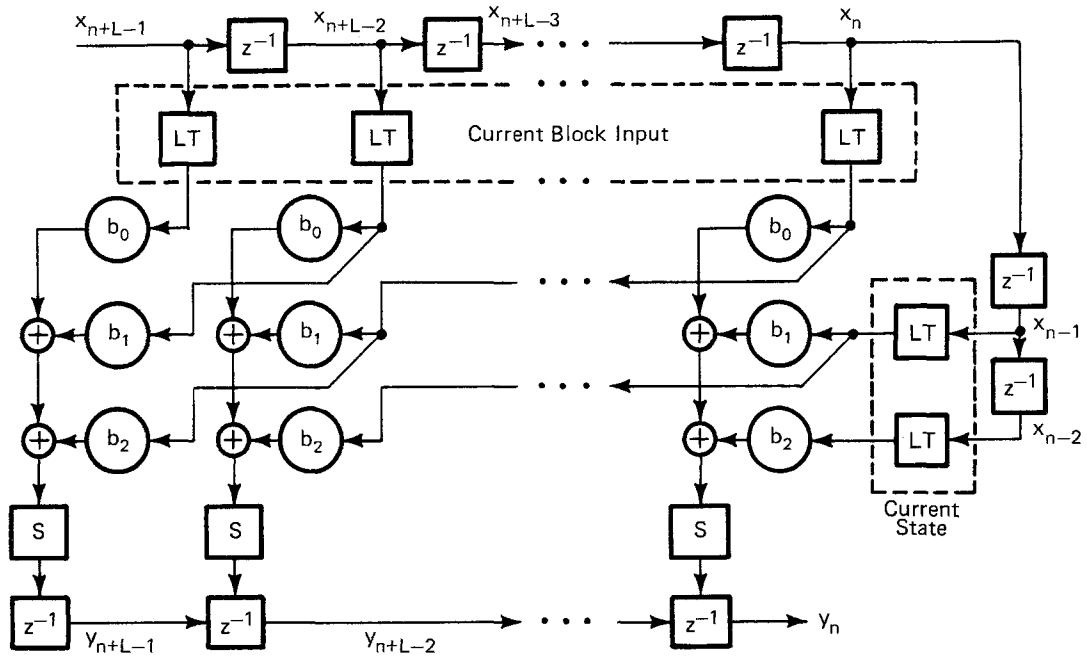

Fig. 2. A block FIR structure of order-2.

latches labelled *LT*. Together, the shift register and the latches act as a serial-to-parallel converter. The data in the latches are presented to the array of multipliers, through which it traverses diagonally. At the end of the MAD cycle, a block of outputs is ready, switches *S* open and load the output shift register in parallel. The array of switches and the output shift register constitute the parallel-to-serial converter in the generic diagram of figure 1.

Note that the structure essentially has L copies of the conventional direct form FIR filter structure. Each copy corresponds to one column in the array, and the throughput rate is L times the rate of the conventional structure. The input and output shift registers are clocked in unison at the throughput rate, L times per MAD cycle. It is instructive to note that while L outputs are shifted out sequentially for each MAD cycle, L inputs are being shifted in. Hence, during a new MAD cycle, the q rightmost inputs to the array are the q most recent data from the previous MAD cycle. They constitute the *current state* of the block filter.

2.2. *Block Direct Form for IIR Filters*

Block implementation of a recursive filter is not as straightforward. Each output in a recursive or IIR filter is generated recursively using past outputs as well as past and present inputs. To avoid interprocessor waiting and allow the computation of multiple outputs in one MAD cycle, the dependence on immediate past outputs has somehow to be eliminated in the implementation. All block structures must implicitly achieve this independence. The block direct form was derived by explicitly eliminating the afore-mentioned dependence. For simplicity of derivation, let us restrict our attention to all-pole filters; i.e., filter with transfer functions

$$H(z) = \frac{1}{A(z)}$$

$$= \frac{1}{1 - a_1 z^{-1} - a_2 z^{-2} - \ldots - a_p z^{-p}}.$$

There is no generality lost in doing so, because any IIR filter $B(z)/A(z)$ may be realized as a cascade of an FIR filter with polynomial transfer function $B(z)$ and an all-pole filter with transfer function $1/A(z)$, and the previous section has shown us how $B(z)$ may be realized in Block form.

The key to a block realization of IIR filters is a modification of the recursion

$$y(n) = \sum_{m=1}^{p} a_m y(n - m) + x(n)$$

to make $y(n)$ independent of other outputs in the same block, so that the block of outputs can be generated independent of each other concurrently. The trick is

to use *look-ahead*, just like carry look-ahead is used in fast parallel binary adders to eliminate interprocessor waiting. Using the above recursion for $y(n)$, with n replaced by $n - 1$, back in the above equation, we get

$$y(n) = (a_1^2 + a_2)y(n - 2) + (a_1a_2 + a_3)y(n - 3) + \ldots + (a_1a_{p-1} + a_p)y(n - p) + a_1a_py(n - p - 1) + x(n) + a_1x(n - 1)$$

which is more compactly written as

$$y(n) = a_1^{(1)}y(n - 2) + a_2^{(1)}y(n - 3) + \ldots + a_p^{(1)}y(n - p - 1) + x(n) + c_1^{(1)}x(n - 1).$$

The superscript (1) indicates that the output dependency has been pushed back by 1. This process has to be repeated $L - 1$ times to make the recursion for $y(n)$ independent of other outputs in the current output block: $y(n)$, $y(n - 1)$, ..., $y(n - L + 1)$. Using the following definition:

$$A^{(0)}(z) = A(z)$$

$$C^{(0)}(z) = 1,$$

the recursive process of eliminating immediate past dependencies can be expressed compactly as

$$A^{(k+1)}(z) = A^{(k)}(z) + a_1^{(k)}z^{-k-1}A(z)$$

$$C^{(k+1)}(z) = C^{(k)}(z) + a_1^{(k)}z^{-k-1}$$

In the above $(k + 1)$th step, the $(k + 1)$th power of z^{-1} is eliminated from the denominator, and for each k, we have

$$A^{(k)}(z) = 1 - a_1^{(k)}z^{-k-1} - a_2^{(k)}z^{-k-2} - \ldots - a_p^{(k)}z^{-k-p}$$

The new system still has the same transfer function, and it can be shown that

$$C^{(L-1)}(z) = 1 + a_1z^{-1} + a_1^{(1)}z^{-2} + \ldots + a_1^{(L-2)}z^{-L+1}$$

$$A^{(L-1)}(z) = A(z)\ C^{(L-1)}(z).$$

Recall that conventional direct forms can be realized as a cascade of all-pole and all-zero sections. In the so-called direct form I, the all-zero section precedes the all-pole section, while in direct form II, the order is reversed [12]. The block direct forms are obtained by realizing augmented polynomials $A^{(k)}(z)$ to generate the $(k + 1)$th output of a block, for each $k = 0, 1, \ldots, L - 1$. In particular, the block version of the conventional direct form I structure is obtained by realizing $B^{(k)}(z) = C^{(k)}(z)B(z)$ and $A^{(k)}(z)$ for $k = 0, 1, 2, \ldots, L - 1$, as shown in figure 3. In the block direct form I realization, each column corresponds to one value of k, and produces one output from the block of length L. In the block direct form II structure, the block FIR section for $B(z)$ is cascaded *after* the block all-pole section that realizes $C^{(k)}(z)/A^{(k)}(z)$ in the kth column, as shown in figure 4. Notice in both structures, the presence of the serial-to-parallel and parallel-to-serial converters. The p outputs of the all-pole section that are fed back as inputs constitute the state of the block structure. In figure 4, latches at the left side contain the state variables. In the block FIR structure of figure 2, the state variables are the q most recent inputs from the previous block. The block direct form II structure is shown in a condensed schematic in figure 5, which is drawn assuming $q \leq p \leq L$. Of course, similar structures can be envisioned for $q > p$ or p, $q > L$ cases as well.

2.3. Block Parallel and Cascade Structures

Conventional direct form realizations are known to have poor finite-precision behavior, and it led to the development of the more robust parallel and cascade structures [12]. In the block direct form realizations, the polynomial lengths become large and this can increase the deletirious effects of finite precision. In this section, we develop block cascade and parallel structures, which are obtained by combining 1st and 2nd order block direct form sections. Each section is a block direct form whose transfer function is of the type

$$\frac{b_0 + b_1z^{-1} + b_2z^{-2}}{1 - a_1z^{-1} - a_2z^{-2}}$$

if it is a 2nd-order section. Figure 6 shows a block parallel implementation (one section can be of first order, if needed), and a block cascade realization is depicted in figure 7. These structures were first studied by Zeman and Lindgren in 1981 using a state-space derivation [5]. They called them state-decimation filters. In the next subsection, we will present the state-space representation of the block structures, and show that

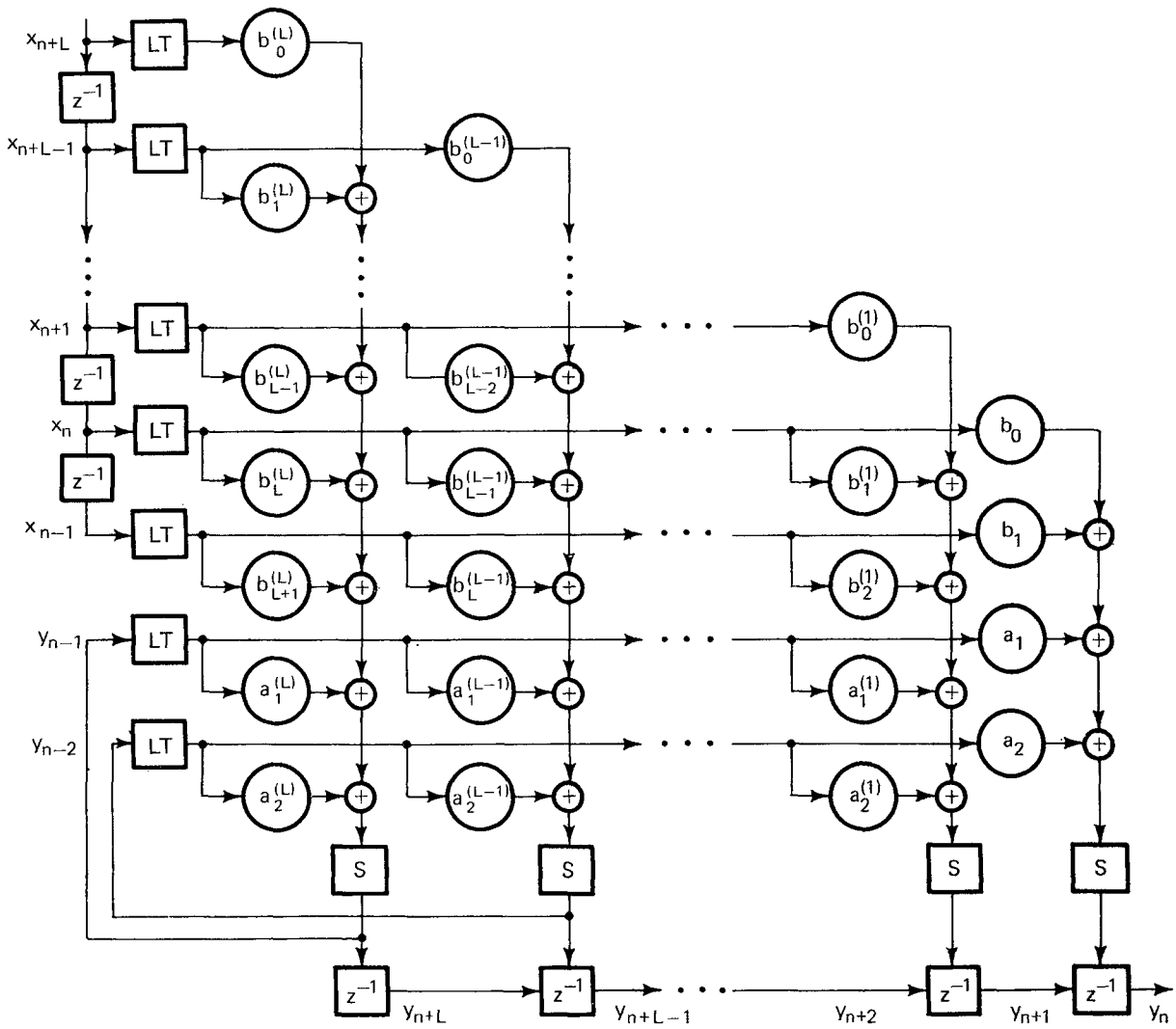

Fig. 3. The block direct form I.

they can also be obtained by a transformation (or state decimation) of the state-space equations for the corresponding conventional structures.

2.4. *State Space Representation for Block Structures*

Let us first write down the state-space equations for the block FIR structure of figure 2. For the conventional, direct form realization of an FIR filter, the state vector composed of the contents of the delay elements is

$$\underline{w}(n) = (x(n-1)\ x(n-2)\ \ldots\ x(n-q))^t$$

where the superscript t denotes matrix transposition. The state-space equations are

$$\underline{w}(n+1) = \begin{pmatrix} 0 & 0 & 0 & .. & 0 & 0 & 0 \\ 1 & 0 & 0 & .. & 0 & 0 & 0 \\ 0 & 1 & 0 & .. & 0 & 0 & 0 \\ . & . & . & .. & . & . & . \\ 0 & 0 & 0 & .. & 0 & 1 & 0 \end{pmatrix} \underline{w}(n) + \begin{pmatrix} 1 \\ 0 \\ 0 \\ . \\ 0 \end{pmatrix} x(n)$$

$$y(n) = (b_1\ b_2\ b_3\ \ldots\ b_q)\ \underline{w}(n) + b_0 x(n).$$

In the block FIR structure, the nth block input and block output may be identified from figure 2 as

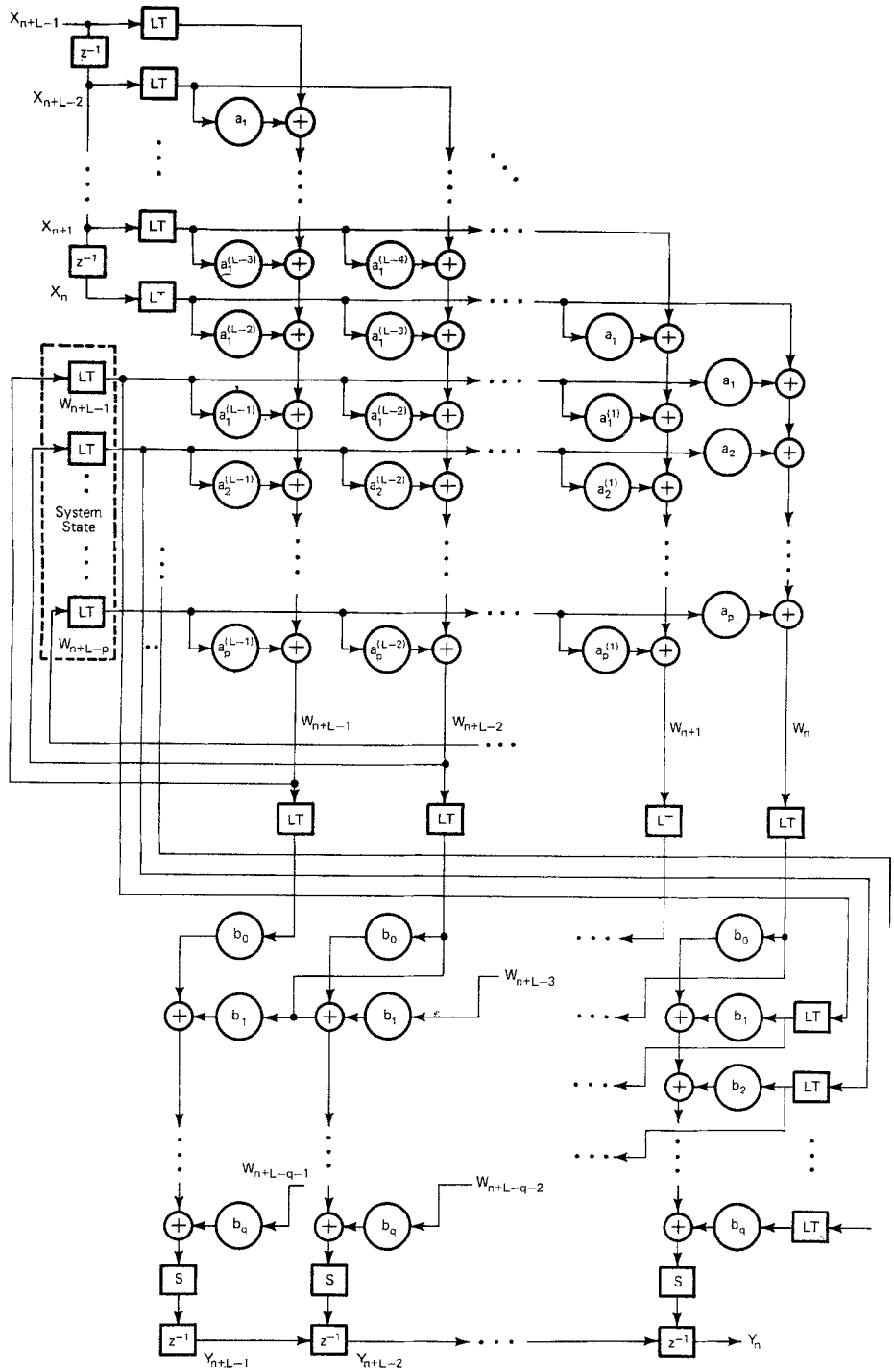

Fig. 4. The block direct form II.

$$\underline{x}_b(n) = (x(nL)\ x(nL+1)\ \ldots\ x(nL+L-1))^t$$

$$\underline{y}^b(n) = (y(nL)\ y(nL+1)\ \ldots\ y(nL+L-1))^t.$$

The nth state of the block filter is

$$\underline{W}^b(n) = (x(nL-1)\ x(nL-2)\ \ldots\ x(nL-q))^t.$$

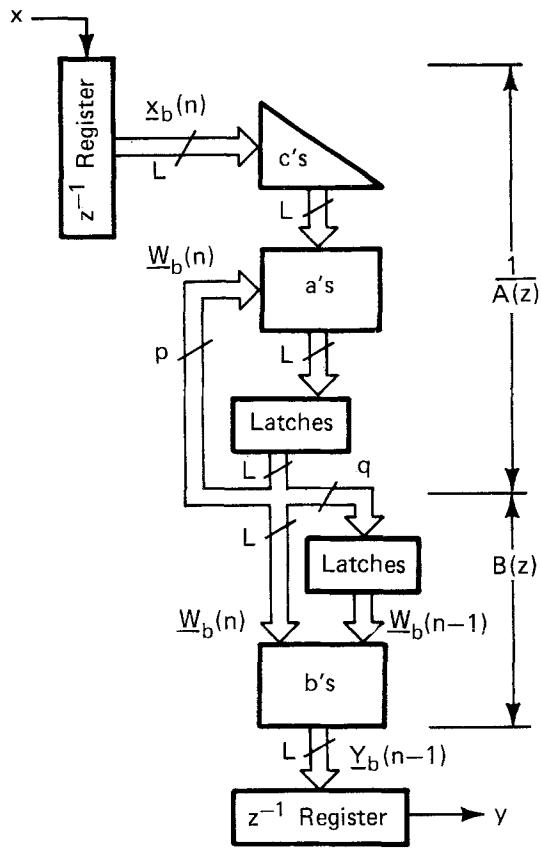

Fig. 5. Short-hand schematic for block direct form II.

The state-space equations for the block FIR structure are

$$\underline{W}_b(n+1) = F_b\underline{W}_b(n) + G_b\underline{x}_b(n)$$

$$\underline{y}_b(n) = H_b\underline{W}_b(n) + D_b\underline{x}_b(n)$$

where (for $L > q$),

$$D_b = \begin{bmatrix} b_0 & 0 & 0 & 0 & 0 \\ b_1 & b_0 & 0 & 0 & 0 \\ b_2 & b_1 & b_0 & 0 & 0 \\ . & . & . & . & . \\ . & . & . & . & . \end{bmatrix}$$

$$H_b = \begin{bmatrix} b_1 & b_2 & b_3 & \dots & b_{q-1} & b_q \\ b_2 & b_3 & b_4 & \dots & b_q & 0 \\ b_3 & b_4 & b_5 & \dots & 0 & 0 \\ . & . & . & \dots & . & . \\ . & . & . & \dots & . & . \end{bmatrix}$$

$$G_b = \left[\begin{array}{c|cccccc} 0 & 0 & 0 & .. & 0 & 0 & 1 \\ & 0 & 0 & .. & 0 & 1 & 0 \\ & 0 & 0 & .. & 1 & 0 & 0 \\ & . & . & .. & . & . & \\ & 1 & 0 & .. & 0 & 0 & 0 \end{array}\right]$$

and

$$F_b = 0.$$

Note that the sequence of states $\underline{W}_b(b)$ of the block structure is a decimated version of the state sequence $\underline{w}(n)$ of the conventional structure:

$$\underline{W}_b(n) = \underline{w}(nL).$$

This is true in general, for any pair of corresponding conventional and block realizations. There also exist direct relations between the parameter matrices of the two realizations.

The first step in developing state space equations for the block direct form structure for IIR filters is identifying the p state variables. Referring to figure 4, the current block output is clearly a function of the current block input and the contents of the p latches that store the intermediate variables w from the previous block cycle. Thus, these p variables that will be denoted collectively as $\underline{W}_b(n)$, summarize relevant information in previous input blocks that is needed for the generation of current and future output blocks. Recognizing the vector $\underline{W}_b(n)$ as the current state of the realization, we can write down the following state space equations for the block direct form II realization,

$$\underline{W}_b(n+1) = F_b\underline{W}_b(n) + G_b\underline{x}_b(n)$$

$$\underline{y}_b(n) = H_b\underline{W}_b(n) + D_b\underline{x}_b(n),$$

where the state feedback matrix is related to the coefficients as

$$F_b = \begin{bmatrix} a_1^{(L-1)} & a_2^{(L-1)} & \dots & a_p^{(L-1)} \\ a_1^{(L-2)} & a_2^{(L-2)} & \dots & a_p^{(L-2)} \\ . & . & \dots & . \\ . & . & \dots & . \\ . & . & \dots & . \\ a_1^{(L-p)} & a_2^{(L-p)} & \dots & a_p^{(L-p)} \end{bmatrix}$$

assuming as before that block length L is at least as large as filter order p. Similar equations can be obtained

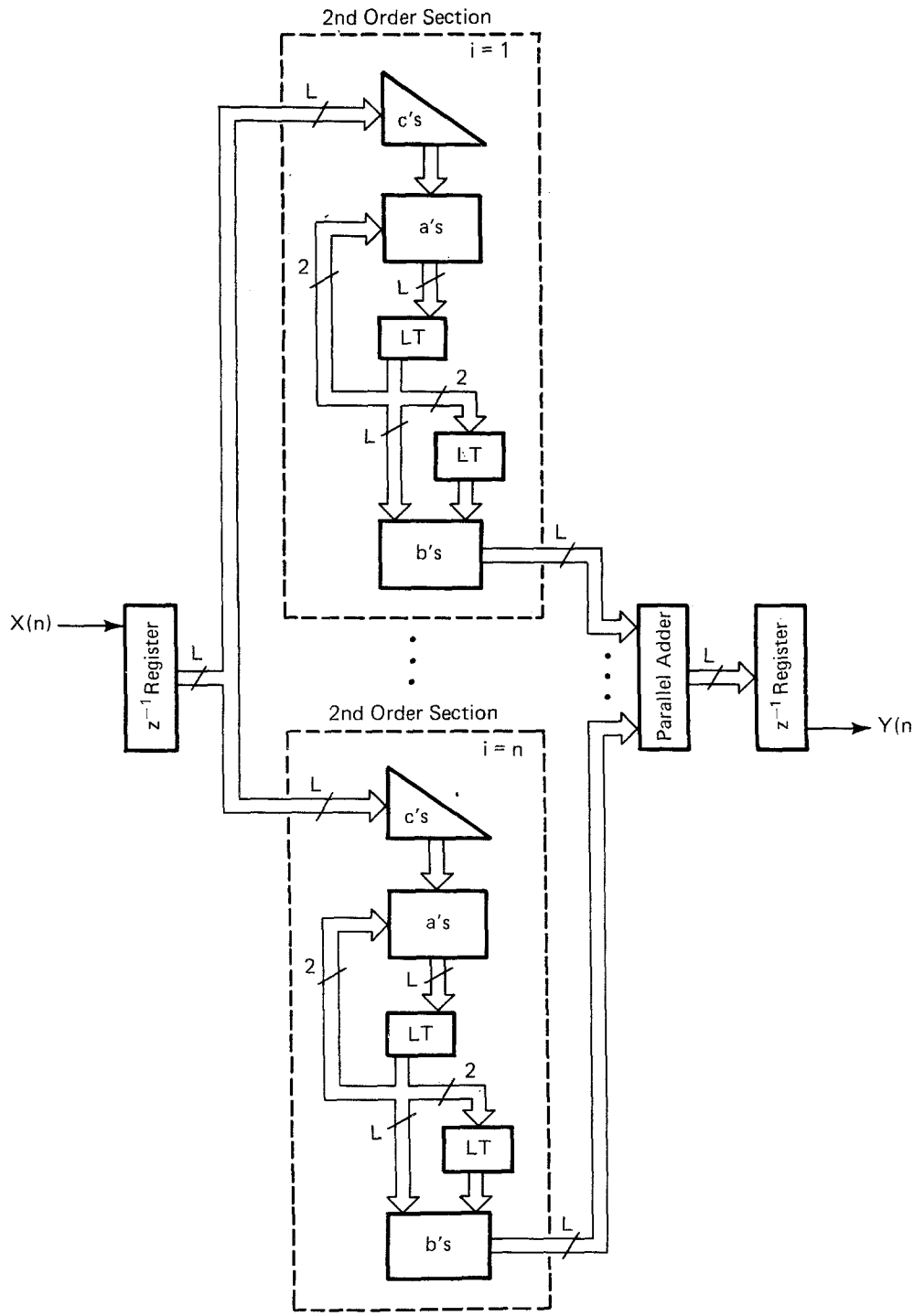

Fig. 6. Block parallel form with 2nd-order sections.

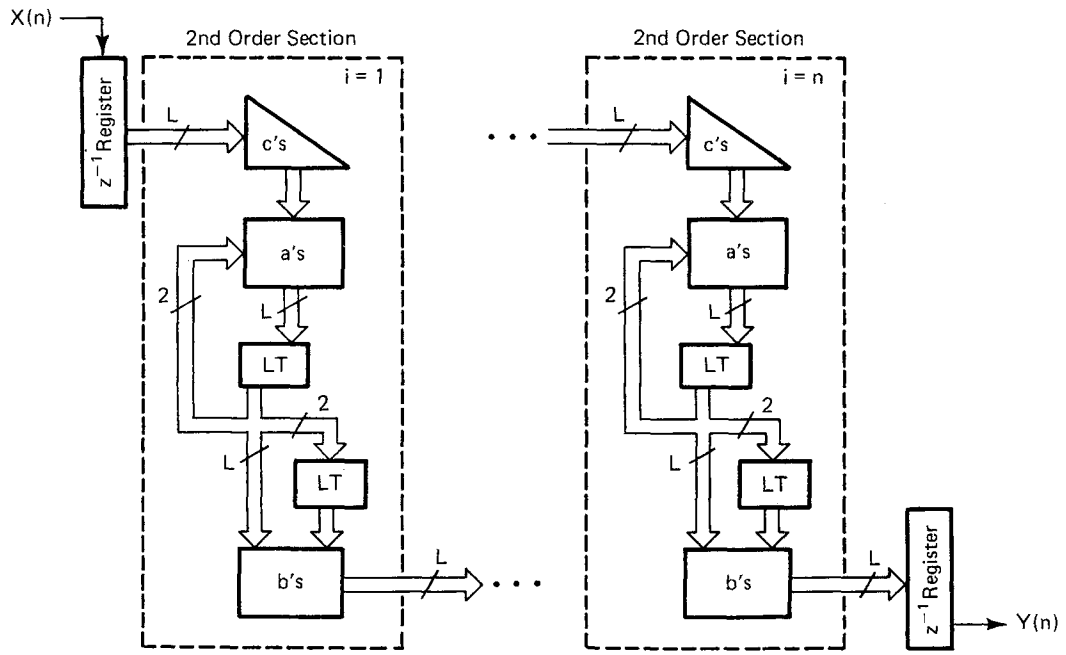

Fig. 7. Block cascade form with 2nd-order sections.

for the block parallel and cascade forms. If first order sections are used instead of second-order in the block parallel form, then the state-feedback matrix will be

$$F_b = \text{diag}\{a_1^L, a_2^L, \ldots, a_p^L\}$$

where a_k are the poles of the transfer function (assumed simple).

These state-space equations and the equations for all block forms can be obtained from the state-space equations for the conventional structures by a transformation that we will refer to as state decimation [2], [4]. Let the state-space representation for a conventional realization (with sequential processing) be

$$\underline{w}(n+1) = F\underline{w}(n) + \underline{g}x(n)$$
$$y(n) = \underline{h}\,\underline{w}(n) + dx(n)$$

where x, y, and d are scalar, $\underline{g}$ is a column vector ($p \times 1$), and $\underline{h}$ is a row vector ($1 \times p$). Let the state sequence decimated by a factor of L be $\underline{W}_b(n) = \underline{w}(nL)$. Then the state propagation equation in decimated form becomes

$$\underline{W}_b(n+1) = F^L\underline{w}(n) + [F^{L-1}\underline{g} \mid F^{L-2}\underline{g} \mid \ldots \mid F\underline{g} \mid \underline{g}]\underline{x}_b(n),$$

and in terms of the decimated state sequence, the outputs become

$$y(nL+k) = \underline{h}F^k\underline{W}_b(n) + \underline{h}F^{k-1}\underline{g}x(nL) + \ldots + \underline{h}x(nL+k-1) + dx(nL+k)$$

for $k = 0, 1, 2, \ldots, (L-1)$. Thus, we get the block structure

$$\underline{W}_b(n+1) = F_b\underline{W}_b(n) + G_b\underline{x}_b(n)$$
$$\underline{y}_b(n) = H_b\underline{W}_b(n) + D_b\underline{x}_b(n)$$

where $F_b = F^L$, G_b is the extended controllability matrix

$$[F^{L-1}\underline{g} \mid F^{L-2}\underline{g} \mid \ldots \mid F_{\underline{g}} \mid \underline{g}],$$

H_b is the extended observability matrix

$$\begin{bmatrix} \underline{h} \\ \underline{h}F \\ \vdots \\ \underline{h}F^{L-2} \\ \underline{h}F^{L-1} \end{bmatrix},$$

and D_b is the lower triangular Toeplitz matrix whose (i, j)th entry is $\underline{h}F^{i-j-1}\underline{g}$ for $i > j$ and d for $i = j$.

Simple calculations show that if we start with the state-space equations for the conventional direct form II (which will be in observer canonical form [13]) and make the above state-decimation transformation, the

resulting equations will be exactly the state-space equations for the block direct form II [14]. Similarly, if we start with the equations for the conventional parallel form using first-order parallel sections (which will be in diagonal canonical form [13]),

$$F = \text{diag}\{a_1, a_2, \ldots, a_p\},$$

the result of the above transformation will be the equations for the block parallel form with first-order sections,

$$F_b = \text{diag}\{a_1^L, a_2^L, \ldots, a_p^L\}.$$

It is also easily seen that a state-decimation transformation of the conventional direct form structure for FIR filters leads to the block FIR structure of figure 2. The state-decimation transformation can be used to generate block structures from any conventional (sequential) structure. For good finite-precision behavior, one could transform the *optimal* principal-axis realizations [15], [16] to block-form. However, that may not lead to significant reduction in round-off noise compared to other block realizations, because as was demonstrated by Barnes and Shinnaka [4], all block structures have low round-off noise for large block lengths.

2.5. *Other High-Speed Filter Structures*

An interesting alternative to block-filtering was proposed by Loomis and Sinha [17]. The Loomis and Sinha architecture is not based on parallel processing, and instead utilizes pipelining to achieve increased throughput rate. Instead of using a number of arithmetic processors working in parallel, Loomis and Sinha studied the option of pipelining the basic multiply-and-add unit. If this unit is realized as a cascade of L stages, each of which takes approximately the same amount of time for its processing, then the throughput rate through the pipelined multiplier will be approximately L per MAD cycle. The speed-up is obtained, because each stage in the pipeline takes less time than the complete arithmetic unit. Ideally, each stage should be L times faster than the complete unit, and hence the speed-up. The use of pipelined multipliers in conventional FIR structures is easily accomplished, and it will speed up the throughput rate by a factor of L. This speed-up is achieved with a small increase in hardware (the control circuitry needed for communication and synchronization between stages), unlike in block strucures where the hardware needed (for FIR filters) is also L times larger. However, the pipelined implementation is limited in speed-up improvements by the number of equally complex stages that the arithmetic unit can be broken into. Block structures, on the other hand, are only limited by the number of processors that can be devoted to the filter realization.

The use of pipelined multiply-add units to improve throughput is not as straightforward for IIR filters as it is for FIR filters. Just like with block structures, the recursive dependence on past outputs in IIR filters comes in the way. In an IIR filter, the past p outputs need to be fed back to be used in computing the present ouptut. In a pipeline, intermediate results are distributed throughout pipeline stages. Partial results for the past $L - 1$ outputs are still in the pipeline, when the processing for the present output begins in the first stage, and these outputs will not be available for feedback. The solution is to get rid of immediate past output dependency just as in the development of the block direct forms in Section 2.2. Loomis and Sinha suggest that since $A^{(L-1)}(z)$ is free of dependency on the past $(L - 1)$ samples, $B^{(L-1)}(z)/A^{(L-1)}(z)$ be realized in an L-stage pipelined implementation [17]. This novel structure can provide increased speed at very little cost in increased hardware. However, the speed-up is limited by the largest number of approximately equally complex stages that a multiplier can be broken into. More importantly, as Loomis and Sinha themselves point out, this structure has a serious stability problem when L is small, caused by finite precision effects. The next section studies the effect of finite precision on the high-speed structures surveyed here.

3. Finite Precision Effects

Let us first examine the destabilization caused by finite wordlength errors in the Loomis and Sinha pipeline structure. It was pointed out by Loomis and Sinha that finite wordlength effects can make the pipeline structure unstable, when the number of pipeline stages is small. Examining the block structures, it is seen that the columns of the block direct form II filter also realize the same augmented polynomials realized by the Loomis and Sinha implementation. Does that mean that the block direct form structure (and possibly all block structures) have stability problems for small values of L? In the next subsection, we will see that the answer is no.

3.1. *Internal Stability*

The Loomis and Sinha L-stage, pipelined implementation realizes the augmented transfer function $B^{(L-1)}(z)/A^{(L-1)}(z)$, which is ideally the same as the original

transfer function because of $(L - p)$ pole-zero cancellations. It is possible that some of the $(L - p)$ additional poles introduced by the augmentation are located outside the unit circle in the complex plane. When that happens, the Loomis and Sinha realization becomes internally unstable. Then, though $B^{(L-1)}(z)/A^{(L-1)}(z)$ and $B(z)/A(z)$ are theoretically equal, in the finite wordlength environment of the real world, they behave differently. As an illustration, consider a 2-stage pipelined implementation of the stable, 2nd-order transfer function

$$\frac{1}{1 - \frac{5}{4}z^{-1} + \frac{3}{8}z^{-2}}.$$

Here,

$$A^{(1)}(z) = \left(1 + \frac{5}{4}z^{-1}\right) A(z) = 1 - \frac{19}{16}z^{-2} + \frac{15}{32}z^{-3}$$

and the augmented 3rd-order difference equation

$$y(n) = \frac{19}{16}y(n - 2) - \frac{15}{32}y(n - 3) + x(n) + \frac{5}{4}x(n - 1)$$

is internally unstable, because of the new pole at -1.25. Internal instability manifests itself in many ways. For one, the zero-input response of the system to almost any nonzero initial conditions, blows up geometrically and quickly exceeds dynamic range limitations. Consider the zero-input response to initial conditions $y(0) = 1$, $y(-1) = 0$. Some of the output samples of the 3rd-order realization are

$$y(20) = -31,\ y(40) = -2686,$$
$$y(60) = -2 \times 10^5,\ y(80) = -2 \times 10^7,$$
$$y(100) = -1.76 \times 10^9.$$

Secondly, even if initial conditions are forced to be zero to avoid such problems, internal variables may still blow up for almost any arbitrary input. Consider the zero-state response of the direct-form II realization [12] of the augmented 3rd-order system

$$w_1(n) = \frac{19}{16}w_1(n - 2) - \frac{15}{32}w_1(n - 3) + x(n)$$
$$w_2(n) = w_1(n - 1)$$
$$y(n) = w_1(n) + \frac{5}{4}w_1(n - 1)$$

to the unit pulse input

$$x(n) = \begin{cases} 1 & n = 0 \\ 0 & \text{else.} \end{cases}$$

The output (theoretically) behaves correctly, but both state variables blow up. This would not be a problem if it were not for the dynamic range limitations of a practical system, owing to finite wordlengths. A third, and important concern regarding internally unstable realizations of externally (or BIBO) stable transfer functions in finite wordlength environments is that coefficient quantization can make the realization unstable externally as well.

In the Loomis and Sinha pipelined, realization, quantization of the coefficients of the augmented polynomials will cause the poles and zeros to be perturbed independently (and possibly differently), so that after coefficient quantization, the poles and zeros introduced by the augmentation may not cancel each other. If some of the poles introduced by augmentation are outside the unit circle, the filter will become externally unstable. In their paper, Loomis and Sinha demonstrate this problem with convincing examples, and argue that for large values of L, the $(L - 1)$ poles introduced by the augmentation do not fall outside the stability region, so that even in the presence of quantization errors, when poles and zeros do not cancel, there is no stability problem.

We now examine the stability of the block direct form structures. At first sight, it might appear that the block direct form structures face the same stability problem as does the Loomis and Sinha structure (even worse: the first several columns of the block direct form structure use low levels of augmentation). Thus, the block direct form structures also rely on pole-zero cancellations in each column of the realization. The second column from the right of the two block direct form structures realize either $C^{(1)}(z)/A^{(1)}(z)$ or $B^{(1)}(z)/A^{(1)}(z)$, both relying on one pole-zero cancellation. The $(k + 1)$th column from the right implements either $C^{(k)}(z)/A^{(k)}(z)$ or $B^{(k)}(z)/A^{(k)}(z)$, relying on k pole-zero cancellations. Just as in the Loomis and Sinha structure, these cancellations may not occur in the presence of coefficient quantization errors. However, even if the roots of $A^{(k)}(z)$ lie outside the unit circle (for any k between 1 and $L - 1$) they do *not* cause stability problems in the block direct forms. The key difference is that in the block realization, the $1/A^{(k)}(z)$ column produces only L-decimated outputs, while in the pipelined realization, $1/A^{(L)}(z)$ produces each and every output. As a result, most of the variables that are fed back to a column in

the block structure come from other columns, and this helps to prevent error accumulation and buildup.

To demonstrate that the block direct form is stable even if there are no pole-zero cancellations (as long as all roots of the nominal $A(z)$ are inside the unit circle), we will resort to the state space representation developed in the last section. Recall that the state-feedback matrix of a block structure is $F_b = F^L$. It has eigenvalues inside the unit circle, whenever the transfer function is externally stable, since eigenvalues of F^L are Lth powers of the poles of the original transfer function $B(z)/A(z)$. Thus, every block realization obtained by state decimation of a conventional, minimum-order realization is internally stable, even when the roots of the augmented polynomial are outside the unit circle. Also, when coefficient quantization prevents pole-zero cancellations in the augmented polynomial, there is still no instability introduced. Similar conclusions can be drawn for all block structures.

3.2. Round-Off Noise

Barnes and Shinnaka found that in fixed-point implementations, block structures in general, have lower round-off noise than the corresponding conventional structures. Ideally, internal variables in the block structure reproduce the state variables of the corresponding conventional structure, and hence input scaling considerations (to avoid dynamic range overflow in internal variables) are the same for corresponding conventional and block structures. For the analysis of output round-off noise, Barnes and Shinnaka used Hwang's basic model that roundoff noise is generated at the output of summing nodes, is independent from one summing node to the next, and that it is zero-mean, white and has the same variance (σ^2) at all summing nodes. With this model, there is one error source at each inner product computation node. Thus for direct form FIR structures, the output round-off noise variance is simply σ^2, whatever the model order, as long as the $(q + 1)$ terms in the inner product

$$b_0u(n) + b_1u(n - 1) + \ldots + b_qu(n - q)$$

are accumulated and summed together at one node, and that there is only one quantizer, located at this node. Under these assumptions, the block FIR structure also has the very same output round-off noise variance.

Using the same model, the direct form II structure with the all-pole section preceding the all-zero section has two error sources, one at the summing node for the a_k inner product, and one at the b_k inner product. The errors from the first source get fed back and have a cumulative effect on the output with magnitude dependent on the actual filter coefficients. Using the state-space notation introduced earlier, it can be shown that for every conventional IIR structure, the output round-off noise variance is

$$\left\{1 + \sum_{n=0}^{\infty} \underline{h}\, F^n (F^n)^t \underline{h}^t\right\}\sigma^2.$$

Similar analysis for block IIR structures, based on the same assumptions, establishes that the round-off noise variance in the $y(nL + k)$ output in the block structure is

$$\sigma_k^2 = \left\{1 + \sum_{n=0}^{\infty} \underline{h}\, F^{nL+k-1} (F^{nL+k-1})^t \underline{h}^t\right\}\sigma^2.$$

for $k = 0, 1, 2, \ldots, L - 1$. Thus the average round-off noise over one block

$$\sigma_b^2 = \frac{1}{L} \sum_{k=0}^{L-1} \sigma_k^2$$

is exactly $1/L$ times the noise variance in the output of the conventional structure. The block structure has on the average, L times lower round-off noise variance than the corresponding conventional structure.

3.3. Coefficient Sensitivity

For conventional structures, it was long believed that structures with low round-off noise also have low sensitivity to coefficient quantization errors [18], [19]. More recently, strong connections were established between coefficient sensitivity and round-off noise levels in conventional structures [16].

Since the coefficients of the block FIR structure are the same as the coefficients of the conventional direct form FIR realization, the coefficient sensitivity properties are the same for both structures, and the block realization is neither better nor worse. To study the effect of coefficient quantization on a block filter, let us examine the first partial derivatives of the poles of the transfer function with respect to the coefficients in the block realization. It is well known that the roots of a polynomial are very sensitive to polynomial coefficients, when the roots are closely spaced. This is made obvious by an examination of the partial derivative of a polynomial's roots to its coefficients. Kaiser showed

that this sensitivity measure is high for polynomials with a large number of closely spaced roots [20], and that this leads to large perturbations in the behavior of a direct-form filter with parameter quantization. The Loomis and Sinha pipelined structure realizes a high-order augmented polynomial, where this effect is further exacerbated.

A block structure also suffers from similar problems. System parameters in a block structure are the entries in the matrix $F_b = F^L$, and the poles of the transfer function are the eigenvalues of F:

$$\text{Pole} = \{\text{eigenvalue of } F_b\}^{1/L}$$

Thus, the pole's sensitivity is

$$\frac{1}{L}\{\text{eigenvalue of } F_b\}$$

$$\frac{1}{L} - 1 \{\text{eigenvalue sensitivity of } F_b\}$$

$$= \frac{1}{L}\frac{\text{pole}}{\text{corresponding eigenvalue of } F_b}$$

$$\{\text{eigenvalue sensitivity of } F_b\}.$$

Even if F_b is well-conditioned for the eigenvalue problem, and has low eigenvalue sensitivity, the sensitivity of the poles to perturbations in the entries of F_b can be high if the system pole to the corresponding eigenvalue of F_b is large. Since nominally the poles have magnitude smaller than one, this ratio is always larger than one. If L is large and the nominal poles are close to the unit circle, $1/L$ will dominate over this ratio and imply a small sensitivity. However, for poles closer to the origin, the ratio may dominate over $1/L$ and lead to large sensitivity. This suggests that block filters suffer from potential coefficient sensitivity problems. This also suggests that F must be well-conditioned for the eigenvalue problem. If F is well-conditioned, so is $F_b = F^L$.

3.4. *Periodically Time-Varying Behavior*

One factor not taken into account thus far is that quantization errors in block structures can cause the overall system response to become slightly time-varying. It is well known that multi-input, multi-output, linear, time-invariant (LTI) filters can be used in conjunction with serial-to-parallel and parallel-to-serial converters to realize periodically time-varying, single-input, single-output (SISO), linear filters [21], [22]. The structure of figure 1, where $H_B(z)$ is the matrix transfer function of the multi-input, multi-output LTI system, for instance, realizes a periodically time-varying SISO system with period L. In fact, the class of block realizations of SISO LTI filters is a subset of the general class of block realizations of periodically time-varying linear, SISO systems. Without any restriction on the block transfer function $H_B(z)$, the structure of figure 1 realizes a periodically time-varying SISO system. Barnes and Shinnaka [23], and more recently Vaidyanathan and Mitra [24], have given necessary and sufficient conditions that must be satisfied by the impulse response matrix $h_B(n)$ and the matrix transfer function $H_B(z)$ respectively, in order to make the SISO system time-invariant. A multi-input, multi-output LTI system satisfying these conditions is called *block-shift-invariant*.

To be block-shift-invariant, it was shown in [24] that the matrix transfer function must have the following Toeplitz and pseudo-circulant structure:

$$H_B(z) = \begin{bmatrix} H_1(z) & z^{-1}H_L(z) & z^{-1}H_{L-1}(z) & \cdots & z^{-1}H_2(z) \\ H_2(z) & H_1(z) & z^{-1}H_L(z) & \cdots & z^{-1}H_3(z) \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ H_L(z) & H_{L-1}(z) & H_{L-2}(z) & \cdots & H_1(z) \end{bmatrix}.$$

Under these conditions, the block structure realizes the SISO function

$$H(z) = \sum_{k=0}^{L-1} z^{-k} H_{k+1}(z).$$

Coefficient quantization in a block implementation of a nominally time-invariant SISO transfer function can cause $H_B(z)$ to be perturbed from this special structure, and cause the realized SISO system to become periodically time-varying [25].

Consider coefficient perturbations in the block FIR structure of figure 2. If all coefficients are perturbed independently, the various columns will have slightly different coefficients and will realize somewhat different FIR filters. The overall SISO system will then be periodically time-varying. However, the nominal coefficients in each column are identical, and if the same wordlength is used throughout the structure, they will be quantized identically as well. Thus, even after coefficient quantization, the columns of the block FIR structure will have identical coefficients, and the overall system will remain time-invariant. The block FIR structure

therefore, does not exhibit periodically time-varying behavior as a result of parameter quantization. The same cannot be said of block IIR structures; observe that the coefficients differ from one column to the next in figures 3 and 4. Thus, they will be perturbed independently by coefficient quantization, causing $H_B(z)$ to lose its Toeplitz and pseudo-circulant structure, and making the overall SISO system periodically time-varying. To avoid such problems, in the next section, we present block IIR filter structures that retain time-invariance even in the presence of coefficient quantization.

4. Robust Block Realization for IIR Filters

The block FIR structure remains time-invariant in the presence of coefficient quantization, because of the symmetry in its coefficients. This fact and the block shift invariance condition on the matrix transfer function $H_B(z)$ suggests a different block structure for IIR filters that is guaranteed to be time-invariant even when coefficients are quantized. This structure shown in figure 8, uses L^2 concurrent SISO filters, one for each entry in the matrix transfer function, and is based on the multi-path structure proposed by Hayashi et al. [26]. Each component SISO subsystem has order equal to the overall system order p, and produces one output per MAD cycle. Since they are all functioning concurrently, L outputs are produced every MAD cycle, the same throughput as with block structures. Referring to

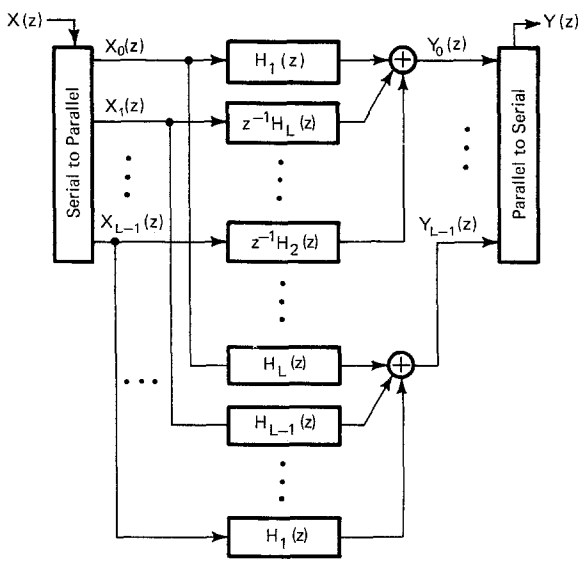

Fig. 8. The generic robust block structure—guaranteed block shift invariance.

the implementation of figure 8, observe that many of the component SISO subsystems are replicas of other component subsystems, except for an additional delay element. This symmetry is responsible for the guaranteed block shift invariance in the presence of coefficient quantization. Identical components will have identical coefficients which will be perturbed identically by coefficient quantization. Thus, even after coefficient quantization, the structure will retain its symmetry which is responsible for the block shift invariance.

For block structures, on the other hand, which do not have this symmetry in the implementation, coefficient quantization will cause the matrix transfer function to deviate in an unpredictable way from the Toeplitz and pseudo-circulant structure required for block shift invariance, causing the realized overall system to be periodically time-varying. In the suggested structure, each component SISO subsystem will also suffer from coefficient quantization, altering their transfer functions slightly, but the overall SISO system will continue to be time-invariant. This time-invariance is gained at the expense of hardware.

Two such structures with the required symmetry in their coefficients are obtained by a simple modification of the block direct form structures of figures 3 and 4. If every column in the block direct form(s) is made as big as the largest column, and realizes $A^{(L-1)}(z)$, then all columns will have identical coefficients. The underlying idea is to use the largest order augmentation in every column, and use the same recursion to generate every output in the block. Consider block direct form II and modify its all-pole half so that each column realizes

$$\begin{aligned} w(nL + k) = {} & a_1^{(L-1)} w(nL + k - L) \\ & + a_2^{(L-1)} w(nL + k - L - 1) + \ldots \\ & + a_p^{(L-1)} w(nL + k - L - p + 1) \\ & + x(nL + k) + a_1 x(nL + k - 1) \\ & + a_1^{(1)} x(nL + k - 2) + \ldots \\ & + a_1^{(L-2)} x(nL + k - L + 1) \end{aligned}$$

instead of using a different level of augmentation in each column. Such a modification will mean that instead of only p variables

$$w(nL - 1)\ w(nL - 2)\ \ldots\ w(nL - p)$$

being stored and fed back as state variables for the next block, $(L + p)$ state variables are needed:

$$w(nL - 1)\ w(nL - 2)\ \ldots\ w(nL - L - p).$$

It will also mean that the number of multipliers in the structure increases

$$\text{from } \left(\frac{1}{2}L^2 + 2pL\right) \text{ to } (L^2 + 2pL)$$

—a significant rise in hardware cost, and a corresponding drop in processor utilization, But, it provides a symmetry in coefficients that makes the structure immune to the periodic time-varying behavior seen in earlier block structures after coefficient quantization. In the new structures just described, that we will call *Robust block direct forms*, coefficient quantization will cause all columns to be perturbed identically, thus retaining time-invariance.

In terms of the block transfer function $H_B(z)$, it is easily seen that each of the component SISO transfer function $H_k(z)$ is either of order one or of order zero (provided that $L \geq p \geq q$), and that even after coefficient quantization, $H_B(z)$ retains the block shift-invariance property. Other robust block structures can be obtained from the robust block direct forms by combining first and/or second order sections in a cascade or parallel connection as in figures 6 and 7. The robust block direct and cascade forms are similar to the multipath structure proposed by Hayashi et al. [26].

All of these robust block structures are unlike any of the other known block structures discussed in Sections 2 and 3, in that they have a lot of redundancy in the form of extra hardware and extra memory elements. In fact, they each have $L + p$ state variables that are stored in memory. This redundancy in the number of state variables, indicates that unlike block structures, the robust block structures cannot be obtained by a state-decimation transformation of conventional structures.

5. Conclusions

In conclusion, this paper studied the behavior of block structures for high-throughput realization of FIR and IIR digital filters in finite wordlength environments. It has been known for some time that block IIR structures have lower round-off noise properties than conventional structures. In this paper, we have demonstrated that in spite of having low round-off noise, block IIR structures can have exceedingly high coefficient sensitivity. We have also seen that block FIR structures have excellent finite precision properties, their coefficient sensitivity is no higher than that of the corresponding conventional structure, and they do not exhibit periodically time-varying behavior. For IIR filters, new robust block structures were proposed that remain time-invariant even after the coefficients are quantized.

Acknowledgments

This work was supported partially by the SDIO/IST office under contract DAAL 03-86-K0111 administered by the US Army Research Office, and partially by the Joint Services Electronics Project under grant N00014-84-C-0149.

References

1. H.B. Voelcker and E.E. Hartquist, "Digital filtering via block recursion," *IEEE Transactions on Audio ElectroAcoustics*, vol. AU-18, 1970, pp. 169–176.
2. C.S. Burns, "Block implementation of digital filters," *IEEE Transactions on Circuit Theory*, vol. CT-18, 1971, pp. 697–701.
3. S.K. Mitra and R. Gnanashekharan, "Block implementation of recursive digital filters," *IEEE Transactions on Circuits and Systems*, vol. CAS-25, 1978, pp. 200–207, (correction on p. 890).
4. J. Zeman and A.G. Lindgren, "Fast digital filters with low round-off noise," *IEEE Transactions on Circuits and Systems*, vol. CAS-28, 1981, pp. 716–723.
5. C.L. Nikias, "Fast block data processing via a new IIR digital filter structure," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-32, 1984.
6. H.-H. Lu, E.A. Lee, and D.G. Messerschmitt, "Fast recursive filtering with multiple slow processing elements," *IEEE Transactions on Circuits and Systems*, vol. CAS-32, 1985, pp. 1119–1129.
7. C.W. Barnes and S. Shinnaka, "Finite word effects in block-state realizations of fixed-point digital filters," *IEEE Transactions on Circuits and Systems*, vol. CAS-27, 1980, pp. 345–349.
8. H.T. Kung, "Why systolic architectures," *IEEE Computer Magazine*, vol. C-15, 1982, pp. 37–46.
9. S.Y. Kung, "VLSI signal processing: From transversal filtering to concurrent array processing," pp. 127–152 in *VLSI and Modern Signal Processing*, (S.Y. Kung, H.J. Whitehouse, and T. Kailath, eds.), Englewood Cliffs, NJ: Prentice Hall, 1985.
10. N. Weste and K. Estraghian, *Principles of CMOS VLSI Design—A Systems Perspective*, Reading, MA: Addison Wesley, 1985.
11. K.S. Arun, "Ultra-high-speed parallel implementation of low-order digital filters," *Proceedings of the IEEE International Symposium on Circuits and Systems 1986*, San Jose, CA, 1986, pp. 944–946.
12. A.V. Oppenheim and R.W. Schafer, *Digital Signal Processing*, Englewood Cliffs, NJ: Prentice Hall, 1975.
13. T. Kailath, *Linear Systems*, Englewood Cliffs, NJ: Prentice Hall, 1980.
14. D.R. Wagner, *A Survey of High-Speed Digital Filtering Structures and their Finite Precision Behavior*, M.S. Thesis, Dept. of Electrical and Computer Engineering, University of Illinois, May 1988.
15. C.T. Mullis and R.A. Roberts, "Synthesis of minimum round-off noise finite precision digital filters," *IEEE Transactions on Circuits and Systems*, vol. CAS-23, 1976, pp. 551–562.

16. D.V. BhaskarRao, "Analysis of coefficient quantization errors in state-space digital filters," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-34, 1986, pp. 131–139.
17. H.H. Loomis and B. Sinha, "High-speed recursive digital filter realization," *Circuits, Systems, and Signal Processing*, vol. 3, 1984, pp. 267–294.
18. L.B. Jackson, "Round-off noise bounds derived from coefficient sensitivities for digital filters," *IEEE Transactions on Circuits and Systems*, vol. CAS-23, 1976, pp. 481–485.
19. L.B. Jackson, A.G. Lindgren, and Y. Kim, "Optimal synthesis of second-order state-space structures for digital filters," *IEEE Transactions on Circuits and Systems*, vol. CAS-26, 1979, pp. 149–153.
20. J.F. Kaiser, "Some practical considerations in the realization of linear digital filters," *Proceedings of the 3rd Allerton Conference on Circuit and System Theory*, 1965, pp. 621–633.
21. R.A. Meyer and C.S. Burrus, "A unified analysis of multirate and periodically time-varying digital filters," *IEEE Transactions on Circuits and Systems*, vol. CAS-22, 1975, pp. 162–168.
22. L.E. Crochiere and L.R. Rabiner, *Multi-Rate Digital Signal Processing*, Englewood Cliffs, NJ: Prentice Hall, 1983.
23. C.W. Barnes and S. Shinnaka, "Block-shift invariance and block implementation of discrete time filters," *IEEE Transactions on Circuits and Systems*, vol. CAS-27, 1980, pp. 667–672.
24. P.P. Vaidyanathan and S.K. Mitra, "Polyphase structures, QMF banks, and block digital filters: A unified framework," *Proc. 21st Annual Asilomar Conference on Signals, Systems, and Computers*, 1987, pp. 900–904.
25. K. Takahashi, Y. Tsunekawa, K. Seki, and J. Sehida, "Time-variant effect of coefficient quantization in block state realization of digital filters," *IEICE Technical Report on CAS*, vol. CAS88-41, 1988 (in Japanese).
26. K. Hayashi, K.K. Dhar, K. Sugahara, and K. Hirano, "Design of high-speed digital filters suitable for multi-DSP implementation," *IEEE Transactions on Circuits and Systems*, vol. CAS-33, 1986, pp. 202–207.

K.S. Arun received his B.Tech degree in electronics and electrical communication engineering from Indian Institute of Technology Kharagpur in 1980. He received the MSEE degree in computer engineering in 1982, and the Ph.D. degree in 1984 from the University of Southern California, Los Angeles. Between 1984 and 1992 he was on the faculty of electrical and computer engineering at the University of Illinois at Urbana-Champaign. Currently, he is an adjunct member of the faculty of electrical engineering and computer science at the University of Michigan at Ann Arbor, and is working toward his M.D. degree at the University of Michigan.