# A Stabilized Parallel Algorithm for Direct–Form Recursive Filters

Richard P. Brent and Zhou Bing Bing†

Computer Sciences Laboratory
Australian National University
Canberra, Australia

**Abstract**

A stabilized parallel algorithm for direct-form recursive filters is obtained using a new method of derivation in the $Z$ domain. The algorithm is regular and modular, so very efficient VLSI architectures can be constructed to implement it. The degree of parallelism in these implementations can be chosen freely, and is not restricted to be a power of two.

## 1. Introduction

Recursive filtering is one of the most important techniques in digital signal processing. Because recursion is involved, high sampling-rate computation is not straightforward. The look-ahead computation concept may be applied to the implementation of recursive filters to achieve parallel computation. Although this conventional method has been used successfully in the parallel realization of recursive filters in state-variable form [1,4,6,7,10], it may cause numerical instability when applied to the direct-form implementation of recursive filters because of the effect of finite wordlength (see the detailed discussion in [2,3,8,9,11]). This paper introduces a new method to obtain a stabilized parallel algorithm for computing direct-form recursive filters. The structure of the algorithm is regular and modular. Thus it is suitable for VLSI implementation.

Our algorithm is described in Section 2. Section 3 is concerned with the degree of parallelism, stability and complexity of the algorithm. The degree of parallelism is considered in Section 3.1. In Section 3.2 we show how to reduce the number of multiplications compared to the number required in a naive implementation. Numerical stability is considered in Section 3.3. Finally, some conclusions and comments on related work by Moyer [5] and by Parhi and Messerschmitt [8,9] are given in Section 4.

---

† Current address: Department of Radio Engineering, Southeast University, Nanjing, Jiangsu 210018, Peoples Republic of China.

Appeared in *IEEE Transactions on Computers* 40 (1991), 333–336.

 rpb105 typeset using TEX

## 2. The Algorithm

In this section, we derive a new parallel algorithm. This algorithm is cost-effective in VLSI implementation, and is guaranteed to be stable if the original (serial) algorithm is stable.

The impulse response of an $N^{th}$ order recursive filter can be expressed as

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{j=0}^{N} w_j z^{-j}}{1 - \sum_{j=1}^{N} r_j z^{-j}} \tag{1}$$

where $X(z)$ and $Y(z)$ represent the $Z$ transforms of input and output, respectively. Conversely, a rational function of the form (1) implies a recursive filter. Our approach is to transform (1) into a different form which is mathematically equivalent (since it represents the same rational function) but has a different denominator (see (10) below). The denominator is chosen so that the coefficients of $z^{-1}, z^{-2}, \cdots z^{1-M}$ vanish, which makes an implementation with $M$-fold parallelism possible. Here $M$ is a parameter which may be chosen freely, although at a certain cost in computational complexity (discussed in Section 3.2).

We introduce a well known $N \times N$ matrix $\mathbf{B}$, called a "companion matrix":

$$\mathbf{B} = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ -b_0 & -b_1 & -b_2 & \dots & -b_{N-1} \end{pmatrix} \tag{2}$$

In this matrix, the elements on the first superdiagonal are unity and the $j^{th}$ element of the last row is $-b_{j-1}$; all other elements are zero. It is known that

$$\begin{aligned} \det(z\mathbf{I} - \mathbf{B}) &= z^N + b_{N-1} z^{N-1} + \cdots + b_1 z + b_0 \\ &= z^N + \sum_{j=1}^{N} b_{N-j} z^{N-j} \end{aligned} \tag{3}$$

where $\det(\mathbf{X})$ denotes the determinant of the matrix $\mathbf{X}$ and $\mathbf{I}$ is an $N \times N$ identity matrix. Let $-b_j = r_{N-j}$. Then

$$\mathbf{B} = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ r_N & r_{N-1} & r_{N-2} & \dots & r_1 \end{pmatrix} \tag{4}$$

and

$$\det(z\mathbf{I} - \mathbf{B}) = z^N - \sum_{j=1}^{N} r_j z^{N-j} \tag{5}$$

Multiplying both sides of (5) by $z^{-N}$, we obtain

$$\det(\mathbf{I} - \mathbf{B}z^{-1}) = 1 - \sum_{j=1}^{N} r_j z^{-j} \tag{6}$$

From (6) we see that, by using the companion matrix $\mathbf{B}$, the denominator of the impulse response function in (1) can be expressed in matrix form. We can rewrite $H(z)$ as

$$H(z) = \frac{\sum_{j=0}^{N} w_j z^{-j}}{\det(\mathbf{I} - \mathbf{B}z^{-1})} \tag{7}$$

We multiply the numerator and denominator of (7) by a common factor

$$\det(\sum_{j=0}^{M-1} \mathbf{B}^j z^{-j}),$$

where $\mathbf{B}^0 = \mathbf{I}$ and $\mathbf{B}^j$ is a product of $j$ matrices $\mathbf{B}$. The impulse response function then becomes

$$\begin{aligned} H(z) &= \frac{(\sum_{j=0}^{N} w_j z^{-j})\det(\sum_{j=0}^{M-1} \mathbf{B}^j z^{-j})}{\det(\mathbf{I} - \mathbf{B}z^{-1})\det(\sum_{j=0}^{M-1} \mathbf{B}^j z^{-j})} \\ &= \frac{(\sum_{j=0}^{N} w_j z^{-j})\det(\sum_{j=0}^{M-1} \mathbf{B}^j z^{-j})}{\det((\mathbf{I} - \mathbf{B}z^{-1})(\sum_{j=0}^{M-1} \mathbf{B}^j z^{-j}))} \end{aligned} \tag{8}$$

Now

$$(\mathbf{I} - \mathbf{B}z^{-1})(\sum_{j=0}^{M-1} \mathbf{B}^j z^{-j}) = \mathbf{I} - \mathbf{B}^M z^{-M} \tag{9}$$

Substituting (9) into (8), we obtain

$$H(z) = \frac{(\sum_{j=0}^{N} w_j z^{-j})\det(\sum_{j=0}^{M-1} \mathbf{B}^j z^{-j})}{\det(\mathbf{I} - \mathbf{B}^M z^{-M})} \tag{10}$$

which implies an algorithm in the usual way. We emphasize that, when regarded as a rational function, (10) is equivalent to (1). However, the algorithm derived from (10) is different from the algorithm derived from (1). From Lemma 1 below, the denominator in (10) is a polynomial in $z^{-M}$, so it is easy to implement the algorithm derived from (10) with $M$-fold parallelism. For details of the parallel implementation, see [3,11].

## 3. Analysis

To analyse the algorithm, it is useful to consider(10) as a product:

$$H(z) = H_1(z)H_2(z) \tag{11}$$

where

$$H_1(z) = \frac{1}{\det(\mathbf{I} - \mathbf{B}^M z^{-M})} \tag{12}$$

and

$$H_2(z) = (\sum_{j=0}^{N} w_j z^{-j})\det(\sum_{j=0}^{M-1} \mathbf{B}^j z^{-j}) \tag{13}$$

are the $Z$ transformations of the impulse response of recursive convolution and linear convolution, respectively.

### 3.1. Recursive convolution

We first analyze the recursive convolution part $H_1(z)$. We show that the denominator of $H_1(z)$ is a polynomial of degree $MN$ in $z^{-1}$ with only $N+1$ nonzero terms, and that $N$ multiplications are required for implementing the filter corresponding to $H_1(z)$.

**Lemma 1**. If $\mathbf{B}$ is an $N \times N$ matrix, then $\det(\mathbf{I}-\mathbf{B}^M z^{-M})$ is a polynomial of degree $MN$ in $z^{-1}$ with only $N+1$ nonzero terms, in fact

$$\det(\mathbf{I} - \mathbf{B}^M z^{-M}) = 1 - \sum_{j=1}^{N} b_j z^{-jM} \tag{14}$$

where $b_j$ is a combination of some elements in $\mathbf{B}^M$.

**Proof :** The result follows from the fact that

$$z^{NM}\det(\mathbf{I} - \mathbf{B}^M z^{-M}) = \det(z^M \mathbf{I} - \mathbf{B}^M)$$

is a polynomial of degree $N$ in $z^M$.

From (12) and (14), we have

$$H_1(z) = \frac{1}{1 - \sum_{j=1}^{N} b_j z^{-jM}} \tag{15}$$

Converting (15) into the time domain, we obtain

$$y_n = \sum_{j=1}^{N} b_j y_{n-jM} + u_n \tag{16}$$

It is easy to see from (16) that $N$ multiplications are required for computing the recursive convolution by the modified algorithm. Because $y_n$ in (16) depends only on $y_{n-jM}$ for $j = 1$ to $N$, up to $M$ outputs can be computed simultaneously.

*Example*

We give an example with $N = 2$.

Suppose that $\mathbf{B}^M = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$. Then

$$\mathbf{I} - \mathbf{B}^M z^{-M} = \begin{pmatrix} 1 - b_{11}z^{-M} & -b_{12}z^{-M} \\ -b_{21}z^{-M} & 1 - b_{22}z^{-M} \end{pmatrix} \tag{17}$$

We have

$$\begin{aligned} \det(\mathbf{I} - \mathbf{B}^M z^{-M}) &= (1 - b_{11}z^{-M})(1 - b_{22}z^{-M}) - b_{12}b_{21}z^{-2M} \\ &= 1 - (b_{11} + b_{22})z^{-M} - (b_{12}b_{21} - b_{11}b_{22})z^{-2M} \\ &= 1 - \mathrm{tr}(\mathbf{B}^M)z^{-M} + \det(\mathbf{B}^M)z^{-2M} \\ &= 1 - \sum_{j=1}^{2} b_j z^{-jM} \end{aligned} \tag{18}$$

where $\mathrm{tr}(\mathbf{B}^M)$ denotes the trace of $\mathbf{B}^M$, $b_1 = \mathrm{tr}(\mathbf{B}^M)$ and $b_2 = -\det(\mathbf{B}^M)$.

Since $H_1(z) = 1/(1 - \sum_{j=1}^{2} b_j z^{-jM})$, we have

$$y_n = b_1 y_{n-M} + b_2 y_{n-2M} + u_n \tag{19}$$

### 3.2. Linear convolution

We now analyse the linear convolution part $H_2(z)$.

**Lemma 2**. If $\mathbf{B}$ is an $N \times N$ matrix, then $\det(\sum_{j=0}^{M-1} \mathbf{B}^j z^{-j})$ is a polynomial of degree $N(M-1)$ in $z^{-1}$.

**Proof** : From Lemma 1, $\det(\mathbf{I} - \mathbf{B}^M z^{-M})$ is a polynomial of degree $NM$ in $z^{-1}$. We also know that $\det(\mathbf{I} - \mathbf{B}z^{-1})$ is a polynomial of degree $N$ in $z^{-1}$. However, we have $\det(\mathbf{I} - \mathbf{B}^M z^{-M}) = \det(\mathbf{I} - \mathbf{B}z^{-1})\det(\sum_{j=0}^{M-1} \mathbf{B}^j z^{-j})$. Thus, the degree of $\det(\sum_{j=0}^{M-1} \mathbf{B}^j z^{-j})$ must be $NM - N = N(M-1)$.

Since $\det(\sum_{j=0}^{M-1} \mathbf{B}^j z^{-j})$ is a polynomial of degree $N(M-1)$ in $z^{-1}$, with all its coefficients nonzero (in general), $N(M-1)$ multiplications are required to implement the linear filter associated with it. We see that an additional $N(M-1)$ multiplications have been introduced in the modified algorithm. This is impractical if either $N$ or $M$ is large. However, by using the decomposition technique of Parhi and Messerschmitt [7], the number of multiplications can be reduced. In [7], the decomposition technique was used for state-variable form recursive filters with $M$ restricted to be a power of two. The following lemmas extend the use of this technique to direct-form recursive filters without restriction on $M$.

**Lemma 3**. If $M = m_1 m_2$, where $m_1$ and $m_2$ are integers, then $\sum_{j=0}^{M-1} \mathbf{B}^j z^{-j}$ can be expressed as

$$\sum_{j=0}^{M-1} \mathbf{B}^j z^{-j} = (\sum_{j=0}^{m_2-1} (\mathbf{B}z^{-1})^{jm_1})(\sum_{j=0}^{m_1-1} (\mathbf{B}z^{-1})^j) \tag{20}$$

**Proof :** The result follows by straightforward algebra.

In the following, the integers $m_k$ are not necessarily distinct.

**Lemma 4.** If $M = \prod_{k=1}^{K} m_k$, where $m_k$ is an integer, then

$$\sum_{j=0}^{M-1} \mathbf{B}^j z^{-j} = \prod_{k=1}^{K} \Big( \sum_{j=0}^{m_k-1} (\mathbf{B} z^{-1})^{j \prod_{i=1}^{k-1} m_i} \Big) \tag{21}$$

**Proof :** By induction on $K$ from Lemma 3.

*Example*

We give an example with $M = 12$. Since $12 = 3 \cdot 2 \cdot 2$, we have

$$\sum_{j=0}^{11} \mathbf{B}^j z^{-j} = (\mathbf{I} + \mathbf{B} z^{-1} + \mathbf{B}^2 z^{-2})(\mathbf{I} + \mathbf{B}^3 z^{-3})(\mathbf{I} + \mathbf{B}^6 z^{-6}) \tag{22}$$

In the example, we see that a "large" polynomial has been decomposed into a product of three "small" polynomials. Thus, the filter associated with $\sum_{j=0}^{11} \mathbf{B}^j z^{-j}$ can be implemented on a three-stage cascaded structure with only $4N$ multiplications, instead of $11N$ multiplications for the naive implementation. This reduction in the number of multiplications can be expressed formally by the following two lemmas.

**Lemma 5.** Suppose that $\mathbf{B}$ is an $N \times N$ matrix and $q$ and $p$ are constants. Then

$$\det\Big(\sum_{j=0}^{q-1} (\mathbf{B} z^{-1})^{jp}\Big) = 1 + \sum_{j=1}^{(q-1)N} d_j z^{-jp}, \tag{23}$$

where $d_j$ is a combination of some elements in $\mathbf{B}^p$.

**Proof :** Let $z^p = \lambda$. Then

$$\det\Big(\sum_{j=0}^{q-1} (\mathbf{B} z^{-1})^{jp}\Big) = \det\Big(\sum_{j=0}^{q-1} \mathbf{B}^{jp} \lambda^{-j}\Big) \tag{24}$$

From Lemma 2, we know that $\det(\sum_{j=0}^{q-1} \mathbf{B}^{jp} \lambda^{-j})$ is a polynomial of degree $N(q-1)$ in $\lambda^{-1}$, and it can then be expressed as

$$\det\Big(\sum_{j=0}^{q-1} \mathbf{B}^{jp} \lambda^{-j}\Big) = \sum_{j=0}^{(q-1)N} d_j \lambda^{-j}, \tag{25}$$

where $d_j$ is a combination of some elements in $\mathbf{B}^p$.

Since the coefficient of $\lambda^0$ is unity both in $\det(\mathbf{I} - \mathbf{B}^M \lambda^{-M})$ and in $\det(\mathbf{I} - \mathbf{B}\lambda^{-1})$, the coefficient of $\lambda^0$ in (25) must also be unity. Thus

$$\det\Big(\sum_{j=0}^{q-1} \mathbf{B}^{jp} \lambda^{-j}\Big) = 1 + \sum_{j=1}^{(q-1)N} d_j \lambda^{-j} \tag{26}$$

Replacing $\lambda$ by $z^p$ in (26), we obtain (23).

We see from Lemma 5 that only $N(q-1)$ multiplications are required to implement the filter associated with $\det(\sum_{j=0}^{q-1}(\mathbf{B}z^{-1})^{jp})$, although the polynomial has degree $Np(q-1)$. Extending this result, we have Lemma 6.

**Lemma 6**. If $\mathbf{B}$ is an $N \times N$ matrix and $M = \prod_{k=1}^{K} m_k$, then the linear filter associated with $\det(\sum_{j=0}^{M-1} \mathbf{B}^j z^{-j})$ can be implemented with $N(\sum_{k=1}^{K} m_k - K)$ multiplications by using the decomposition technique.

**Proof :** From Lemma 4, we have

$$\begin{aligned} \det(\sum_{j=0}^{M-1} \mathbf{B}^j z^{-j}) &= \det(\prod_{k=1}^{K}(\sum_{j=0}^{m_k-1}(\mathbf{B}z^{-1})^{j\prod_{i=1}^{k-1} m_i})) \\ &= \prod_{k=1}^{K} \det(\sum_{j=0}^{m_k-1}(\mathbf{B}z^{-1})^{j\prod_{i=1}^{k-1} m_i}) \end{aligned} \tag{27}$$

From (27), $\det(\sum_{j=0}^{M-1} \mathbf{B}^j z^{-j})$ can be expressed as a product of $K$ small polynomials. The filter associated with it can be implemented on a $K$-stage cascaded structure. From Lemma 5, $N(m_k - 1)$ multiplications are required in the $k^{th}$ stage. The total number of multiplications is $\sum_{k=1}^{K} N(m_k - 1) = N(\sum_{k=1}^{K} m_k - K)$.

When $m_k$ is not a prime number, the $k^{th}$ polynomial on the right-hand side of (27) can be decomposed further. In the case $M = \prod_{k=1}^{K} p_k$ where $p_k$ is prime, the total number of multiplications is reduced to $N(\sum_{k=1}^{K} p_k - K)$. If $M$ is a power of two, then this expression simplifies to $N\log_2 M$.

### 3.3. Stability

We can rewrite (10) as

$$H(z) = \frac{(\sum_{j=0}^{N} w_j z^{N-j})\det(\sum_{j=0}^{M-1} \mathbf{B}^j z^{M-1-j})}{\det(Iz^M - \mathbf{B}^M)} \tag{28}$$

Suppose that the original algorithm before the modification is stable. Then the roots of $\det(\mathbf{I}z - \mathbf{B})$ are all in the unit circle. This means that the eigenvalues $z_i$ of $\mathbf{B}$ are all in the unit circle. It is clear that the eigenvalues $z_i^M$ of $\mathbf{B}^M$ are also in the unit circle and closer to the origin than the corresponding $z_i$. Thus, stability of the original algorithm implies stability of our modified algorithm.

## 4. Conclusions

In this paper, we have introduced a new method of $Z$ domain derivation for obtaining parallel algorithms for direct-form recursive filters. Using this method, parallel algorithms with guaranteed stability be derived. Also, the additional complexity required for this purpose can be reduced through a decomposition technique which was originally introduced by Parhi and Messerschmitt [7] and extended to more general cases for direct-form recursive filters in this paper. Because of the regularity and modularity of the derived algorithm, very efficient pipelined and/or parallel VLSI architectures can also be constructed [2,3,8,9].

We have recently learned that Parhi and Messerschmitt [8,9] have obtained a similar result using a different approach. The disadvantage of their method is that the decomposition technique can be applied only when $M$ is a power of two. Our method has no such limitation.

We thank an anonymous referee for pointing out Moyer's work which is concerned with a parallel algorithm for first-order recursive filters [5]. It can be seen that Moyer's work is a special case of our result.

## 5. References

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

[2] Brent, R. P. and Zhou, B. B., A two-level pipelined implementation of direct-form recursive filters, Report TR-CS-88-06, Computer Sciences Laboratory, Australian National University, April 1988.

[3] Brent, R. P. and Zhou, B. B., A stabilized parallel implementation of direct-form recursive filters, Report TR-CS-88-07, Computer Sciences Laboratory, Australian National University, May 1988.

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

[5] Moyer, A. L., An efficient parallel algorithm for digital IIR filters, *in Proc. IEEE Int. Conf. Acoustics, Speech, Signal Process.*, Apr. 1976, pp. 525-528.

[6] Nikias, C. L., Fast block data processing via new IIR digital filter structure, *IEEE Trans. Acoustics, Speech, Signal Process.*, vol. 32, No. 4, Aug. 1984.

[7] Parhi, K. K. and Messerschmitt, D. G., Concurrent cellular VLSI adaptive filter architectures, *IEEE Trans. Circuits Syst.*, vol. 10, Oct. 1987, pp. 1141-1151.

[8] Parhi, K. K. and Messerschmitt, D. G., Pipelined VLSI recursive filter architectures using scattered look-ahead and decomposition, *in Proc. IEEE Int. Conf. Acoustics, Speech, Signal Process.*, Apr. 1988, pp. 2120-2123.

[9] Parhi, K. K. and Messerschmitt, D. G., Pipeline interleaving and parallelism in recursive digital filters, Part I: Pipelining using scattered look-ahead and decomposition, submitted to *IEEE Trans. Acoustics, Speech, and Signal Processing*, Nov. 1987.

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

[11] Zhou, B. B., *Systolic Architectures for Parallel Implementation of Digital Filters*, Ph. D. thesis, Australian National University, September 1988.