Efficient Bit-Level Systolic Array Implementation of FIR and IIR Digital Filters

CHIN-LIANG WANG, MEMBER, IEEE, CHE-HO WEI, SENIOR MEMBER, IEEE, AND SIN-HORNG CHEN, MEMBER, IEEE

Abstract—**High-speed digital filters are important for real-time digital signal processing applications. In this paper, we present some new bit-level systolic architectures based on a new inner product computation scheme for finite impulse response (FIR) and infinite impulse response (IIR) digital filterings. The FIR filter structure is optimized in the sense that, for a given clock rate, both the utilization efficiency and average throughput are maximized. The IIR filter structure has approximately the same utilization efficiency and throughput rate as previous related techniques for processing a single data stream (channel), but it allows two data streams to be processed concurrently to double the performance. This feature makes the new IIR system attractive for use in applications where multiple filtering and particularly bandpass analysis are required.**

I. Introduction

The application of digital filtering has expanded in and is useful in many important areas such as speech signal processing, digital communications, digital image processing, radar and sonar systems, etc., [1]. To perform digital filtering operations in real time, high-speed computing hardware is often necessary. While systems organized with bit-parallel data have been used extensively in computer hardware, the bit-serial approach to digital filter realization is efficient in digital signal processing applications where data bits are available in serial form, such as the processing of telephone voice and data signals [2]. The circuits described here are specifically intended for such applications.

A systolic architecture [3] is a network or array of interconnected cells, each capable of performing some simple operation. There is considerable interest in the use of systolic architectures to design VLSI systems because such architectures have a number of important advantages, namely nearest neighbor interconnections, regularity, and high throughput. The concept of systolic arrays has been applied at both the word and bit levels to perform convolution, FIR and IIR filtering operations (see, for example, [3]–[14]). The arrays at the bit level have lower design complexity and higher bit rates (in any given technology) than those at the word level. In this paper, we restrict our attention to the bit-level systolic architectures that can be used to implement digital filters when bit-serial fixed-point arithmetic and full precision results are desired.

Manuscript received March 15, 1987; revised October 16, 1987.

C.-L. Wang was with the Institute of Electronics, National Chiao Tung University, Hsinchu, Taiwan, Republic of China. He is now with the Department of Electrical Engineering, National Tsing Hua University, Hsinchu, Taiwan, Republic of China.

C.-H. Wei is with the Institute of Electronics, National Chiao Tung University, Hsinchu, Taiwan, Republic of China.

S.-H. Chen is with the Department of Communication Engineering, National Chiao Tung University, Hsinchu, Taiwan, Republic of China.

IEEE Log Number 8718931.

Bit-level systolic arrays with bit-serial arithmetic for FIR filtering (based on the convolution sum) and IIR filtering have been proposed earlier in [7]–[14]. The FIR structures in [7]–[10] and the IIR structures in [9]–[11] have (utilization) efficiencies less than or equal to 50 percent, while the FIR structures in [11]–[13] have an efficiency of 100 percent. For a given clock rate, neither of them achieves the maximum possible throughput, i.e., one output word per B cycles for the input data with word length B. Although the FIR filtering array in [14] has 100 percent efficiency and provides throughput in the rate of one word per B cycles, it needs a more complicated type of basic cell and at least three accumulators working alternatively. Therefore, the system has a slower clock rate and a less regular architecture. In this paper, we present some new bit-level bit-serial systolic architectures, each using a single accumulator, for FIR and IIR filterings. The FIR filtering arrays are constructed based on the convolution sum (i.e., direct-form realization), and require simpler basic cells than the array in [14]. For a given clock rate, these FIR filtering arrays are optimized in the sense that both the utilization efficiency and average throughput are maximized. The IIR filtering array has approximately the same efficiency and throughput rate as the previous related techniques [9]–[11] for processing a single data stream (channel), but it allows two data streams to be processed concurrently to double the performance. This feature makes the new IIR system attractive for use in applications where multiple filtering and particularly bandpass analysis are required.

The function of an FIR filter with N coefficients (a_i, $i = 0, 1, 2, \cdots, N-1$) is to compute the output sequence

$$y_n = \sum_{i=0}^{N-1} a_i \cdot x_{n-i}, \qquad \text{for } n = 0, 1, 2, \cdots \tag{1}$$

where $x_i (i = 0, 1, 2, \cdots)$ represents a sequence of input data words. Extending (1) to IIR filtering by adding M coefficients ($b_j, j = 1, 2, \cdots, M$) in the feedback path, the relationship between the input and output is then given by

$$y_n = \sum_{i=0}^{N-1} a_i \cdot x_{n-i} + \sum_{j=1}^{M} b_j \cdot y_{n-j}, \quad \text{for } n = 0, 1, 2, \cdots \tag{2}$$

where $y_n = 0$ for $n < 0$. For a given n, (1) and (2) are equivalent to an N-dimensional and an $(N + M)$-dimensional inner product computation, respectively. That is, for FIR filtering,

$$\begin{aligned} y_n &= A \cdot X_n^T \\ A &= [a_0 a_1 \cdots a_{N-1}] \\ X_n &= [x_n x_{n-1} \cdots x_{n-N+1}] \qquad n = 0, 1, 2, \cdots \end{aligned} \tag{3}$$

and, for IIR filtering,

$$\begin{aligned} y_n &= F \cdot D_n^T \\ F &= [a_{N-1} a_{N-2} \cdots a_0 b_M b_{M-1} \cdots b_1] \\ D_n &= [x_{n-N+1} x_{n-N+2} \cdots x_n y_{n-M} y_{n-M+1} \cdots y_{n-1}] \\ & \qquad n = 0, 1, 2, \cdots . \end{aligned} \tag{4}$$

We can see from (3) and (4) that both FIR and IIR filtering operations involve a great number of inner product computations. In the following, we first describe a bit-level systolic architecture for inner product computation, and then modify it for FIR and IIR filterings. For ease of presentation, it is assumed that all the coefficient words and input data words take the form of B-bit positive binary numbers. Thus, each inner product word of N-dimensional inner product computations has $2B + L$ bits, where L is the word length growth due to accumulation. Depending on B and N, L is equal to the number obtained by rounding $\log_2 N$ up or down.

II. A Bit-Level Systolic Architecture for Inner Product Computation

Fig. 1(a) shows the bit-level systolic array proposed by Wang *et al.* [15] for computing the inner product y of a coefficient vector $A = [a_0 a_1 \cdots a_{N-1}]$ and a data vector $X = [x_0 x_1 \cdots x_{N-1}]$, where

$$\begin{aligned} a_i &= (a_i^{(B-1)} a_i^{(B-2)} \cdots a_i^{(1)} a_i^{(0)}) \\ x_i &= (x_i^{(B-1)} x_i^{(B-2)} \cdots x_i^{(1)} x_i^{(0)}) \\ y &= (y^{(2B+L-1)} y^{(2B+L-2)} \cdots y^{(1)} y^{(0)}). \end{aligned} \tag{5}$$

This array consists of a subarray of $N \times (B + L)$ main array cells and a linear chain of $B + L + 1$ accumulator cells. The logic functions of these basic cells are indicated in Fig. 1(b) and (c), where the main array cell has an internal register for storing one coefficient bit. Throughout this paper, we assume that each basic cell (including those to be presented) contains input latches (not shown in the figures) for pipelining operations. The coefficient word a_i is stored in the $(i + 1)$th row of the array. The data word x_i enters the $(i + 1)$th row in a bit-serial format with its least significant bit first and with one zero interspersed between adjacent bits. The input sequence for the x_i is staggered by one cycle relative to that for the x_{i-1}.

As the data bits of X move left through the array, each main array cell produces a partial product $(a_i^{(k)} \cdot x_i^{(m)})$. This partial product is added to the accumulating partial product (y^{km}) from the cell above to generate the new accumulating partial product, which is then passed to the cell below. This means that the parallelogram Y [shown at the top of Fig. 1(a)] of accumulating partial products, which is initialized by zeros, will move down through the array to accumulate contributions from each term in the inner product. Since carries generated in the process are clocked to the left towards columns of higher significance, the resulting Y (appearing at the bottom of the two-dimensional subarray) will have a width of $B + L$ bits rather than only B bits. The leftmost $N \times L$ main array cells storing "0" bits are used to accommodate the growth.

Since accumulating partial products of equal significance appear at a slant in Y, the inner product y can be obtained by accumulating the resulting Y along a linear chain of $B + L + 1$ accumulator cells. Each accumulator cell propagates the accumulated result to the right if the associated PTRL bit is 1, and propagates the carry to the left in any case. As Y passes through the linear chain, the least significant B bits of y will come out serially with interspersed zeros from the rightmost accumulator cell, and the remaining $B + L$ bits will come out from the other accumulator cells in sequence, as shown at the bottom of Fig. 1(a). To ensure that the accumulated results of the present inner product computation do not affect those of the succeeding one, a PTRL pattern (a sequence of $2B - 2$ 1's followed by two 0's) recirculating continuously is used to control the propagation of the accumulated results.

It is apparent that the inner product array described above has an efficiency of only 50 percent and can yield an output result every $2B$ cycles. However, if two independent inner product computations are interleaved in the same array, the performance is doubled. The proposed inner product array can be easily modified for two's complement arithmetic by using the methods devised previously [12], [16]. This issue is briefly described in the Appendix. Comparison of this array and other related arrays are given in [15]. To update the coefficients stored in the array, the fast loading scheme described by Urquhart and Wood [11] can be used.

III. Bit-Level Systolic Arrays for FIR Filtering

As described in (3), the output y_n of an FIR filter is given as the inner product of a coefficient vector A and a data vector X_n consisting of delayed input data words. To realize such an FIR system, the bit-level systolic array shown in Fig. 2 can be used, where $B = N = 4$ and $L = 2$. This array is obtained easily from Fig. 1(a) by slightly

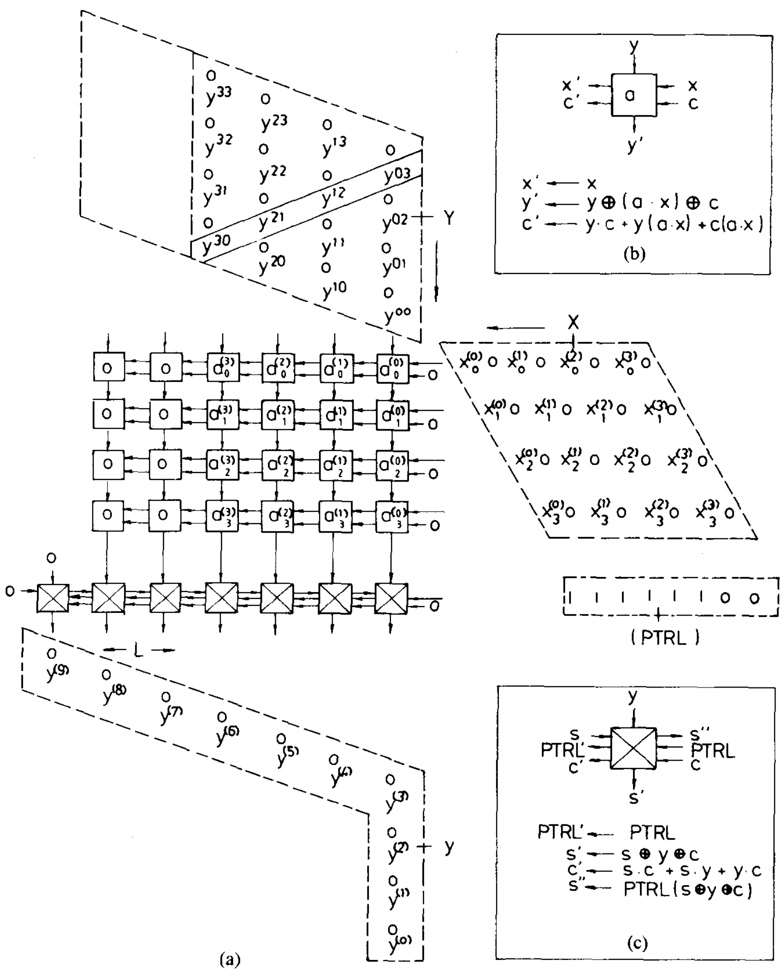


Fig. 1. (a) Inner product array for positive arithmetic. $B = N = 4$ and $L = 2$. The output word y will appear after the parallelogram Y passes through the array. (b) Main array cell. (c) Accumulator cell.

modifying the input method. The coefficient word a_i is stored in the $(N - i)$th row of the array. Following McWhirter *et al.* [7], data words $(x_i, i = 0, 1, 2, \cdots)$ with interspersed zeros are fed in serially only at the bottom row of the two-dimensional subarray (i.e., the Nth row of the array). Once a data bit has passed through a given row, it is delayed by $B - L - 1$ cycle periods $(K\tau)$ and then fed to the beginning of the row above. This arrangement is equivalent to feeding the data vector X_n to the array in a format that, when the data word x_{n-i} enters the $(N - i)$th row, it is staggered by one cycle relative to the data word x_{n-i-1} entering the $(N - i - 1)$th row, where $i = 0, 1, 2, \cdots, N - 2$. The case for $n = 3$ is shown at the right-hand side of Fig. 2. As the data bits of X_3 move left through the array, the parallelogram Y_3 of accumulating partial products moves down through the array to generate the inner product word y_3. The procedure for computing the inner product is the same as that described in the previous section.

Following McCanny *et al.* [13], data words can also be fed in serially at the top row of the array. This approach is shown in Fig. 3, where the coefficient word a_i is stored in the $(i + 1)$th row. Once a data bit has passed through a given row, it is delayed by $B - L + 1$ cycle periods and then fed to the beginning of the row below. This arrangement is equivalent to feeding the data vector X_n to the array in a format that, when the data word x_{n-i} enters the $(i + 1)$th row, it is staggered by one cycle relative to the data word x_{n-i+1} entering the ith row, where $i = 1, 2, \cdots, N - 1$. The case for $n = 3$ is shown at the right-hand side of Fig. 3. Like the array in Fig. 2, as the data bits of X_3 move left through the array, the output word y_3 is computed.

One can see that the FIR filtering arrays described so far have the same utilization efficiency and throughput performance as the inner product array described in Section II. The efficiency of these arrays can be raised to 100 percent by interleaving two independent filtering opera-

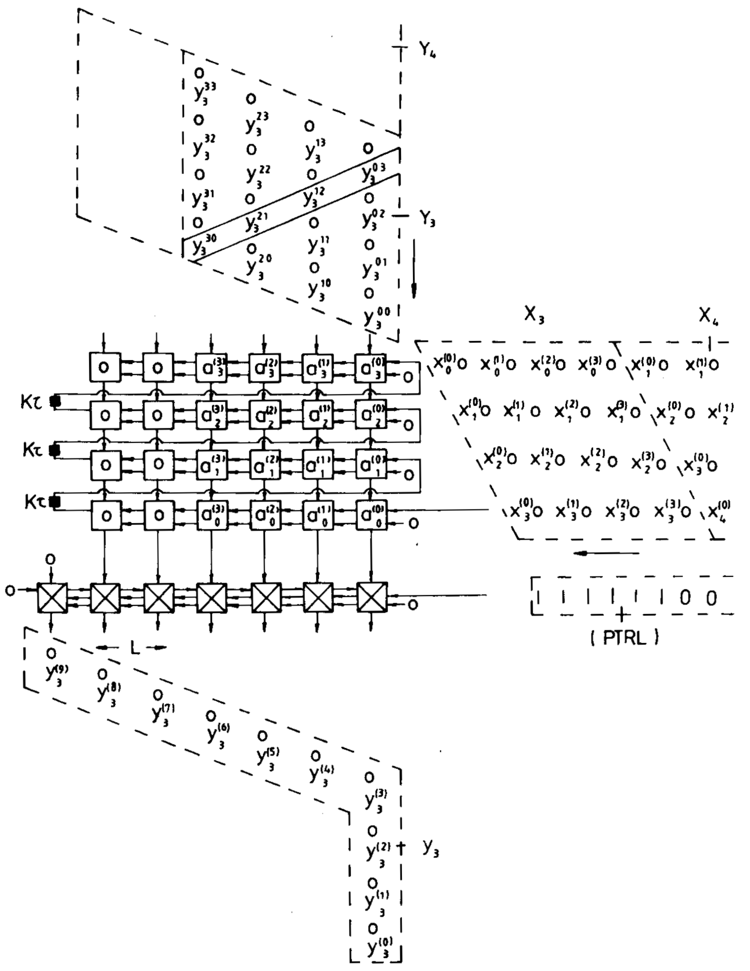


Fig. 2. FIR filtering array with input at the Nth row and with 50 percent efficiency for processing a single data stream. $B = N = 4$, $L = 2$, and $K = B - L - 1$.

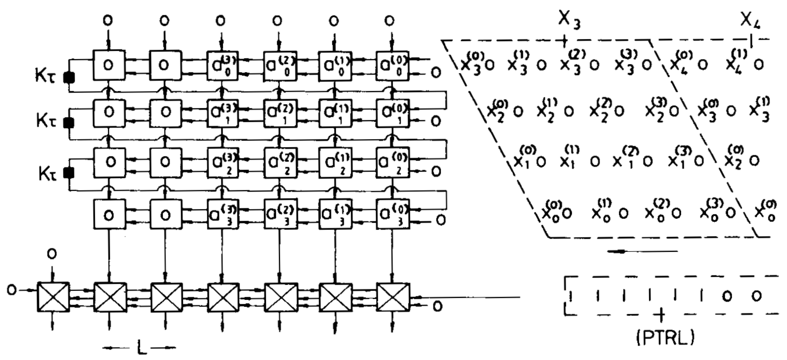
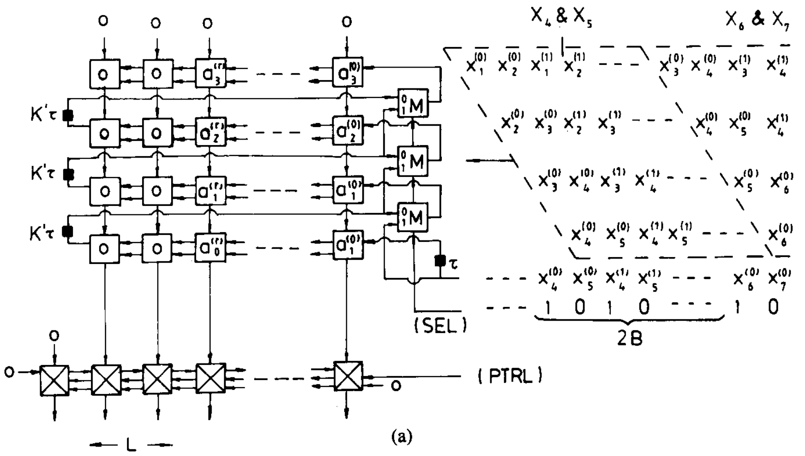


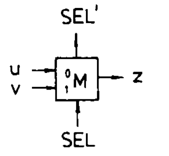
Fig. 3. FIR filtering array with input at the top row and with 50 percent efficiency for processing a single data stream. $B = N = 4$, $L = 2$, and $K = B - L + 1$.

tions (with the same coefficients) in each array. Thus, they are particularly suitable for processing two data channels whose data bits are interleaved.

To fully utilize the above arrays for a single data channel rather than two data channels, they can be modified as shown in Figs. 4 and 5. The modified arrays are very similar to the original ones. The major difference is that, for the modified arrays, several multiplexer cells (M's)



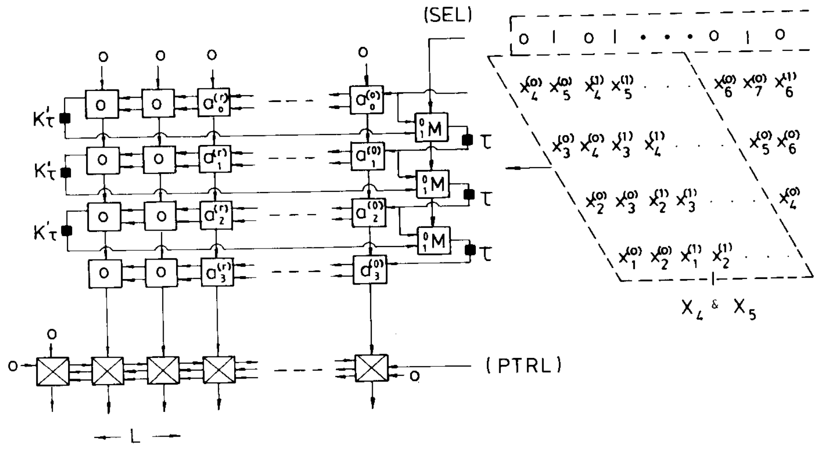
(a)



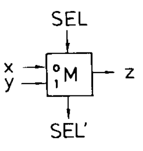
SEL' ← SEL
SEL = 0 ⇒ z ← u
SEL = 1 ⇒ z ← v

(b)

Fig. 4. (a) FIR filtering array with input at the Nth row and with 100 percent efficiency for processing a single data stream. $N = 4$, $L = 2$, $r = B - 1$, and $K' = B - L - 3$. (b) Type-I multiplexer cell.



(a)



SEL' ← SEL
if SEL = 1 then
z ← y
else
z ← x

(b)

Fig. 5.(a) FIR filtering array with input at the top row and with 100 percent efficiency for processing a single data stream. $N = 4$, $L = 2$, $r = B - 1$, and $K' = B - L - 2$. (b) Type-II multiplexer cell.

are appended and each delay between adjacent rows is rearranged properly. For the array shown in Fig. 4, the delay between adjacent rows is $B - L - 3$ instead of $B - L - 1$ cycle periods (see Fig. 2). Each type-I multiplexer cell is controlled by an SEL stream 1 0 1 0 $\cdots$ to select the input data bits for each row. The SEL stream is synchronized with the input data stream, in which the bits from two data words are interleaved. Under the above arrangement, equivalent input data vectors will be formed in a manner that X_n and X_{n+1} enter the array interleavedly for $n = 0, 2, 4, \cdots$. The case for $n = 4$ is shown at the right-hand side of Fig. 4. When the parallelogram of data vectors (X_4 and X_5) moves left through the array, the corresponding output results (y_4 and y_5) are computed interleavedly. Since pairs of equivalent input data vectors come in continuously, the modified array of Fig. 4 has 100 percent efficiency and generates two consecutive output words every $2B$ cycles.

For the modified array shown in Fig. 5, the delay between adjacent rows is partitioned into two parts: one being $B - L - 2$ units and the other being one unit. The input data stream enters the array from the top row with the same format as that of Fig. 4. Each type-II multiplexer cell is controlled by another SEL stream 0 1 0 1 $\cdots$, and works similarly to the type-I multiplexer cell. All the other issues of this array are the same as those of Fig. 4 except that, in the vertical direction, the data bits now move downwards rather than upwards through the array. The array of Fig. 5 also has an efficiency of 100 percent and yields two consecutive output words every $2B$ cycles, i.e., the average throughput is one word per B cycles.

For a given clock rate and B-bit input data words, the throughput rate of one word per B cycles is the maximum possible one that can be obtained in the bit-level systolic FIR filtering arrays with bit-serial arithmetic. Thus, the new arrays of Figs. 4 and 5 for FIR filtering are optimized in the sense that, for a given clock rate, both the utilization efficiency and average throughput are maximized. When compared to another FIR filtering array [14] which has the same feature, these arrays have simpler basic cells and use a single accumulator rather than multiple accumulators, where the multiple accumulator scheme requires a complicated switch arrangement to control the accumulators to work alternatively. We can say that the proposed arrays have a higher system clock rate and a more regular architecture than the array in [14]; however, they require an input data stream with interleaving nature. It should be noted that, in Figs. 2–5, the start positions of the feedback paths can be located inside the arrays properly to avoid the values of K and K' becoming negative.

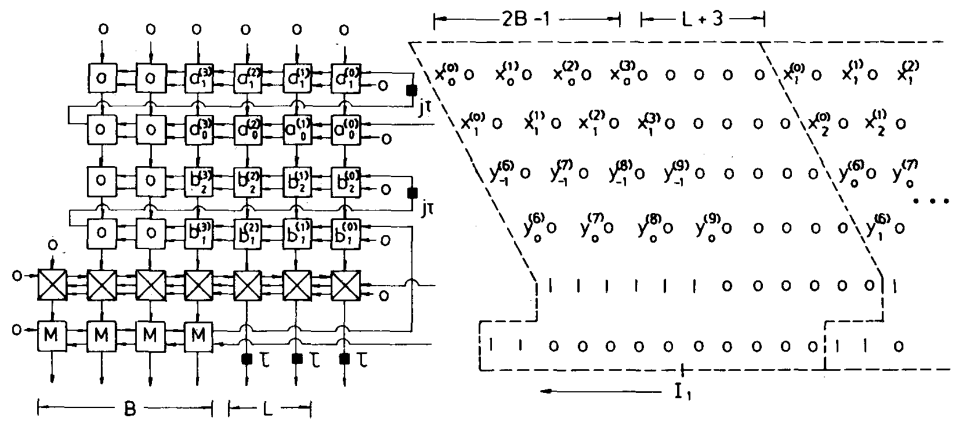
IV. A Bit-Level Systolic Array for IIR Filtering

As described in (4), the output y_n of an IIR filter is given as the inner product of a coefficient vector and a data vector consisting of delayed input and output samples. Now, if we treat input data as fractions (having no integer part) and truncate each output word to have the same word length as each input data word, the bit-level inner product array shown in Fig. 1(a) can be modified properly for IIR filtering.

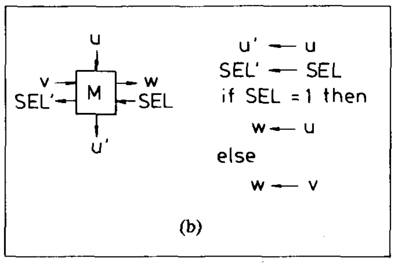
Fig. 6(a) shows a bit-level systolic array for IIR filtering with $B = 4$ and $N = M = 2$. In this array, each input data word is expressed as $x_i = (\Delta x_i^{(3)} x_i^{(2)} x_i^{(1)} x_i^{(0)})$, where Δ denotes the location of the binary point. Since the filter is assumed to have unity gain, each output word is still a fraction, i.e., $y_i = (\Delta y_i^{(9)} y_i^{(8)} \cdots y_i^{(0)})$. Thus, each desired truncated word for the feedback path computations is given as $\tilde{y}_i = (\Delta y_i^{(9)} y_i^{(8)} y_i^{(7)} y_i^{(6)})$. The first two rows of the array are used for the forward path computations, and the third and fourth rows are used for the feedback path computations. The operations of this array are very similar to those of Fig. 1(a). The main difference is in the input method. Each input data word enters the array only from the second row (in general the Nth row) in a format as shown. Once a data bit has passed through a given row, it is fed to the beginning of the row above. Since the ith truncated output word ($\tilde{y}_i$) is required for computing the $(i + 1)$th output word (y_{i+1}), it should be fed back to the input of the array properly. In Fig. 6(a), the bottom row of four type-III multiplexer cells make each truncated output word enter the fourth row (in general the $(N + M)$th row) in the same manner as the input data words enter the second row. In fact, the linear chain of type-III multiplexer cells is a special bit-parallel/bit-serial shift register. The function of the type-III multiplexer cell is shown in Fig. 6(b). Once a truncated output bit has passed through a given row (excluding the $(N + 1)$th row), it is fed to the beginning of the row above. For demonstration purpose, the equivalent input timing diagram for computing the output word y_1, which will emerge in the same manner as y in Fig. 1(a), is shown by the broken lines in Fig. 6(a). It should be noted that the actual inputs are only the Nth row and the two bottom rows.

We can see from Fig. 1(a) that $y^{(6)}$ will emerge $L + 3$ ($L = 2$ here) cycles after $x_3^{(3)}$ enters the fourth row. This means that in Fig. 6(a) $y_1^{(6)}$ will emerge $L + 4$ cycles, where one cycle is used for passing through the rightmost multiplexer cell, after $y_0^{(9)}$ enters the fourth row. Thus, a guard band of $L + 3$ 0's is required at the end of each input data word. The control pattern for the accumulator chain now becomes a sequence of $2B - 2$ 1's followed by $L + 4$ 0's, instead of a sequence of $2B - 2$ 1's followed by two 0's [see the PTRL in Fig. 1(a)]. The control pattern for the multiplexer cells is a sequence of two 1's followed by $2B + L$ 0's. The former is staggered by two cycles relative to the latter so that each desired truncated output word can be fed back to the beginning of the $(N + M)$th row properly. From the equivalent input timing diagram, it is easy to see that each delay between adjacent rows is $B + 1$ cycle periods ($j\tau$). The three one-cycle delays (τ's) are used to make the array have the same output format as Fig. 1(a).

From Fig. 6(a), one can see that the new IIR filtering array has an efficiency of $B/(2B + L + 2)$ in cell util-



(a)



(b)

Fig. 6. (a) IIR filtering array. $B = 4$, $N = M = L = 2$, and $j = B + 1$. I_l is the equivalent input timing diagram for computing the output word y_1. (b) Type-III multiplexer cell.

ization and yields outputs at a rate of one word per $2B + L + 2$ cycles. The performance is comparable to those of the related arrays presented previously [9]–[11]. (In these previous designs, the efficiency is either $B/(2B + L)$ or $B/(2B + L + 1)$, and the throughput rate is either one word per $2B + L$ cycles or one word per $2B + L + 1$ cycles.) Like the inner product array described in Section II, two independent IIR filtering operations (with the same coefficients) can be interleaved in the new array. In this case, the performance is doubled.

V. Conclusions and Discussion

In this paper, we have presented five bit-level systolic arrays, four for FIR filtering and one for IIR filtering, based on a new inner product computation method. These new arrays are highly regular, nearest neighbor connected, and thus well suited to VLSI implementation. Due to the use of the bit-serial approach, they are especially suitable for digital signal processing applications where data bits are available in serial form, such as the processing of telephone voice and data signals. As the FIR filtering arrays are based on the convolution sum, they are in fact convolution arrays. For a given clock rate, the FIR filtering arrays of Figs. 4 and 5 have the best possible utilization efficiency and average throughput that is not achieved in the previous related designs of [7]–[13]. They also gain an improvement in both processing speed and regularity of architecture over the array described in [14] with a slight increase in the complexity of the input scheme. Those of Figs. 2 and 3 can also be fully utilized (i.e., with 100 percent efficiency) for concurrent filtering of two data streams. The new IIR filtering array has utilization efficiency and throughput rate comparable to those of the previous related techniques [9]–[11] for filtering of a single data stream, and about twice as those for concurrent filtering of two data streams. This feature makes it attractive for use in applications where multiple filtering and particularly bandpass analysis are required [1]. In multiple filtering, one specific filter operation is required to be performed on many signal channels presented as input data, for example, in multiplex systems. While in bandpass analysis, many bandpass filtering operations, which in general can be performed using a single low-pass digital filter and a frequency shift operation, are performed on a single signal channel presented as input data.

It is interesting to note that the basic cells required for the new FIR filtering arrays of Figs. 3 and 5 are identical to those used in the arrays of Figs. 2 and 4. Also, all the data bits, the accumulating partial product bits, and the result bits propagate in the same direction (downwards) in the former, but not in the latter. As described by McCanny *et al.* [13], in a system with such a feature, latches can be used as a means of buffering to prevent off-chip or chip-to-chip delays from degrading system performance. This feature also makes the fault-tolerant scheme described by Kung and Lam [17], which let the bypass propagation required for the routing of information around faulty cells occupy one entire clock cycle, applicable to the two-dimensional subarray (major part of the array) without reducing the clock rate. We can say that the new FIR filtering arrays of Figs. 3 and 5 gain advantages over those of Figs. 2 and 4 in terms of chip cascadability, fault tolerance, and possible wafer-scale integration. However, the latter has smaller system latency, where the latency now is defined to be the time from when we input x_n to the time we receive y_n.

Since all the proposed arrays have a particular output format, additional circuitry may be required to handle the interface problem. To make the array chip have fewer I/O pins, the bit-parallel output part (i.e., the most significant $B + L$ bits) can be converted into two bit-serial parts. This can be implemented as shown in Fig. 7, where two linear chains of B and L type-III multiplexer cells are cascaded together. Each linear chain is controlled by a sequence of two 1's followed by $2B - 2$ 0's, and used to convert bit-parallel data (time skewed) into a bit-serial data stream with interleaving nature as shown. The $B\tau$ delay is used to make the $y^{(B)}$ and $y^{(2B)}$ emerge simultaneously. It should be noted that if the linear chain of B type-III multiplexer cells, instead of L, is placed at the left-hand side of Fig. 7, the most significant B bits of the results will be grouped into a single stream. If only the most significant $B + L + 1$ bits are desired, a parallel output form can be obtained directly from the bottom of each proposed array without any additional circuitry. However, in this case, the utilization efficiency of the output bus will be lower.

As well as for the FIR (or finite length convolution) and IIR filter problems, the proposed inner product computation scheme may also be used for other applications involving matrix and vector multiplication, such as the linear discriminant function classifier [15].

Appendix
A Bit-Level Systolic Inner Product Array For Two's Complement Arithmetic

This Appendix describes how two's complement arithmetic can be performed in the proposed systolic inner product array. A two's complement number $x = (x^{(B-1)}x^{(B-2)} \cdots x^{(0)})$ can be written as

$$x = -x^{(B-1)} \cdot 2^{B-1} + \sum_{i=0}^{B-2} x^{(i)} \cdot 2^i. \quad \text{(A-1)}$$

Hence, both positive and negative partial products are generated in the multiplication of two two's complement numbers. As described by Wood *et al.* [16], the multiplication of two two's complement numbers can be written in a form involving only positive partial products provided that all partial products which involve a sign bit and a nonsign bit are complemented. The answer is then obtained by adding an arithmetic adjustment to the final re-

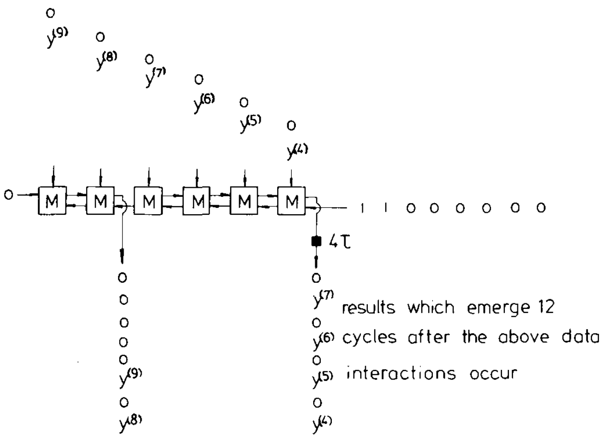

0
$y^{(9)}$
0
$y^{(8)}$
0
$y^{(7)}$
0
$y^{(6)}$
0
$y^{(5)}$
0
$y^{(4)}$
M
1 1 0 0 0 0 0 0
4τ
$y^{(7)}$ results which emerge 12
$y^{(6)}$ cycles after the above data
$y^{(5)}$ interactions occur
$y^{(4)}$

Fig. 7. Bit-parallel to bit-serial conversion scheme. $B = N = 4$ and $L = 2$.

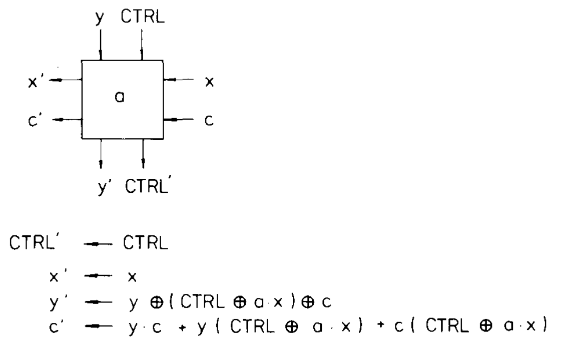

y CTRL
x'
c'
a
x
c
y' CTRL'
CTRL' ← CTRL
x' ← x
y' ← y ⊕ (CTRL ⊕ a·x) ⊕ c
c' ← y·c + y (CTRL ⊕ a·x) + c (CTRL ⊕ a·x)

Fig. 8. Modified main array cell for two's complement arithmetic.

sult. The arithmetic adjustment is fixed for given word lengths. For the multiplication of two B-bit words, the correction term is $2^B - 2^{2B-1}$ [12], i.e., a $2B$-bit number with 1's only at the most significant bit and the $(B + 1)$th least significant bit. For N-dimensional inner product computations, the required correction term becomes $N \times (2^B - 2^{2B-1})$. In the case where N is a power of 2, this becomes $2^{B+L} - 2^{2B+L-1}$, where $L = \log_2 N$.

To perform two's complement arithmetic in the proposed inner product array according to the above method, the main array cell should be modified as shown in Fig. 8, where the CTRL bit is used for controlling the inversion of partial products. The modified inner product array for two's complement arithmetic is shown in Fig. 9, where the major data flow is the same as that of Fig. 1(a) except that the parallelogram of accumulating partial products is augmented by one column. The inversion of partial products can be controlled by moving down a parallelogram of CTRL bits through the array so that CTRL bits are 1's at positions where partial products need to be complemented. The parallelogram of CTRL bits not only has the same shape (excluding the leftmost column) as that of accumulating partial products, but also synchronizes with it. In Fig. 9, the corresponding CTRL bits are 1's for the positions with parentheses, and are 0's for the others. In

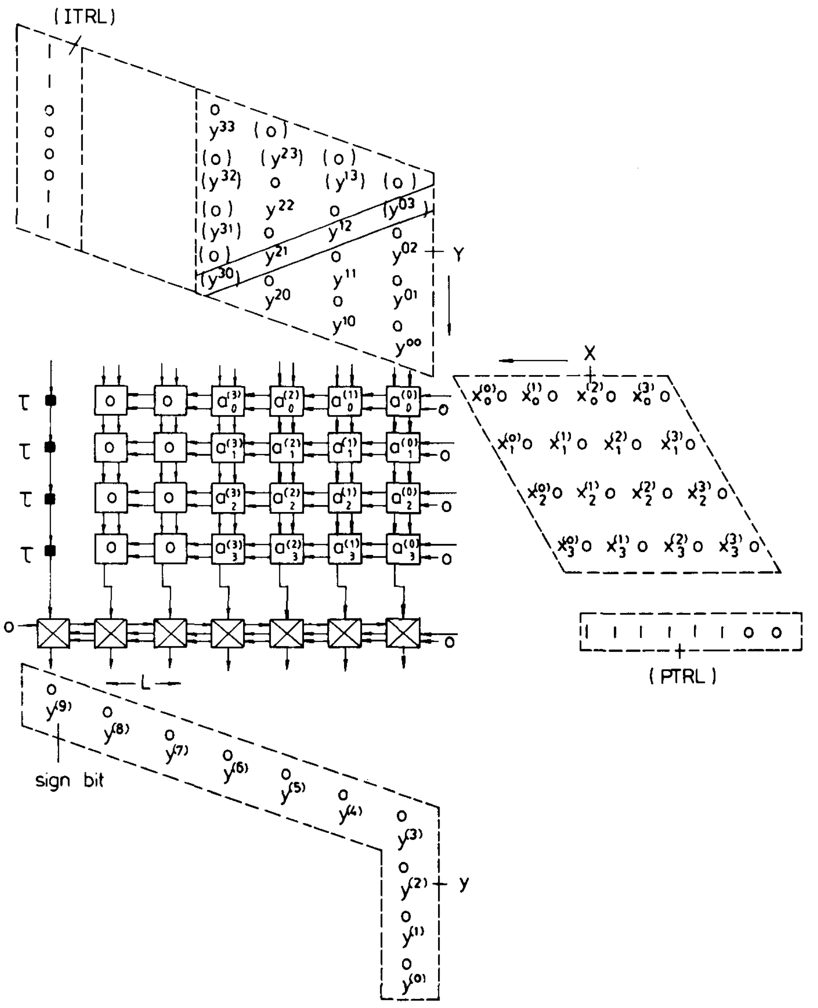


Fig. 9. Two's complement inner product array showing arithmetic adjustment. $B = N = 4$ and $L = 2$.

the case of $B = N = 4$, the required arithmetic adjustment is $2^6 - 2^9$, i.e., the correction term is the 10-bit word (1 0 0 1 0 · · · 0). The addition of the correction term to complete the final result can be handled quite simply by initializing the parallelogram of accumulating partial products properly. In Fig. 9, the leftmost column which contains terms with significance corresponding to 2^6, 2^7, 2^8, and 2^9 is added for this purpose. For two interleaved computations, this column should be initialized by an ITRL sequence of 1 1 0 0 0 0 1 1. It should be noted that the ITRL sequence is dependent on N and B. Since the input data flow has not been changed, the corresponding FIR and IIR filtering arrays for two's complement arithmetic can be easily derived by reference to Figs. 2-6.

References

[1] V. Cappellini, A. G. Constantinides, and P. Emiliani, *Digital Filters and Their Applications*. New York: Academic, 1978.

[2] L. B. Jackson, J. F. Kaiser, and H. S. McDonald, "An approach to the implementation of digital filters," *IEEE Trans. Audio Electroacoust.*, vol. AU-16, pp. 413-421, Sept. 1968.

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

[4] H. T. Kung, "Let's design algorithms for VLSI systems," in *Proc. CALTEC Conf. VLSI*, Jan. 1979, pp. 65-90.

[5] K. H. Cheng and S. Sahni, "VLSI architectures for the finite impulse response filter," *IEEE J. Select. Areas Commun.*, vol. SAC-4, pp. 92-99, Jan. 1986.

[6] P. R. Cappello and K. Steiglitz, "A note on 'free accumulation' in VLSI filter architectures," *IEEE Trans. Circuits Syst.*, vol. CAS-32, pp. 291-296, Mar. 1985.

[7] J. G. McWhirter, J. V. McCanny, and K. W. Wood, "Novel multibit convolver/correlator chip based on systolic array principles," *Proc. SPIE*, vol. 341, *Real Time Signal Processing V*, 1982, pp. 66-73.

[8] J. G. McWhirter, D. Wood, K. Wood, R. A. Evans, J. V. McCanny, and A. P. H. McCabe, "Multibit convolution using a bit level systolic array," *IEEE Trans. Circuits Syst.*, vol. CAS-32, pp. 95-99, Jan. 1985.

[9] C. Caraiscos and B. Liu, "Bit serial VLSI implementations of FIR and IIR digital filters," in *Proc. IEEE Int. Symp. Circuits Syst.*, Newport Beach, CA, May 1983, pp. 717-721.

[10] P. E. Danielsson, "Serial/parallel convolver," *IEEE Trans. Comput.*, vol. C-33, pp. 652-667, July 1984.

[11] R. B. Urquhart and D. Wood, "Systolic matrix and vector multiplication methods for signal processing," *IEE Proc.*, vol. 131, pt. F, pp. 623-631, Oct. 1984.

[12] J. V. McCanny, J. G. McWhirter, and K. Wood, "Optimized bit

level systolic array for convolution," *IEE Proc.*, vol. 131, pt. F, pp. 632-637, Oct. 1984.

[13] J. V. McCanny, R. A. Evans, and J. G. McWhirter, "Use of unidirectional data flow in bit-level systolic array chips," *Electron. Lett.*, vol. 22, pp. 540-541, May 1986.

[14] R. A. Evans and R. Eames, "Modified bit-level systolic inner product/convolver architecture with increased throughput," *Electron. Lett.*, vol. 23, pp. 460-461, Apr. 1987.

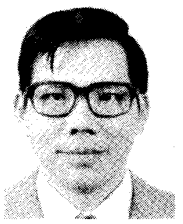
[15] C.-L. Wang, C.-H. Wei, and S.-H. Chen, "Efficient bit-level systolic array for the linear discriminant function classifier," *IEE Proc.*, vol. 134, pt. G, pp. 216-224, Oct. 1987.

[16] D. Wood, R. A. Evans, and K. W. Wood, "An 8 bit serial convolver chip based on a bit level systolic array," in *Proc. Custom Integrated Circuits Conf.*, Rochester, NY, May 1983, pp. 256-261.

[17] H. T. Kung and M. Lam, "Fault tolerance and two-level pipelining in VLSI systolic arrays," in *Proc. MIT Conf. Advanced Res. VLSI*, Cambridge, MA, Jan. 1984, pp. 74-83.

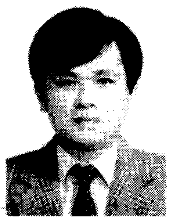
Chin-Liang Wang (S'85-M'88) was born in Tainan, Taiwan, Republic of China, on December 1, 1959. He received the B.S. degree in electronics engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1982, the M.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 1984, and the Ph.D. degree in electronics engineering from National Chiao Tung University in 1987.

He is currently an Associate Professor in the Department of Electrical Engineering at National Tsing Hua University, Hsinchu, Taiwan. His research interests include digital signal processing, VLSI, and digital speech processing.



Che-Ho Wei (M'77-SM'87) was born in Taiwan, Republic of China, in 1946. He received the B.S. and M.S. degrees in electronics engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1968 and 1970, respectively, and the Ph.D. degree from the University of Washington, Seattle, in 1976.

From 1976 to 1979, he was an Associate Professor at National Chiao Tung University, where he is now a Professor and Director of the Institute of Electronics. His present research interests include filter design, digital signal processing, and communication systems.



Sin-Horng Chen (S'80-M'84) received the B.S. degree in communication engineering and the M.S. degree in electronics engineering from National Chiao Tung University, Hsinchu, Taiwan, Republic of China, in 1976 and 1978, respectively, and the Ph.D. degree in electrical engineering from Texas Tech University, Lubbock, in 1983.

From 1978 to 1980, he was an Assistant Engineer for Telecommunication Labs., Taiwan. He is currently an Associate Professor in the Department of Communication Engineering at National Chiao Tung University. His research interests include digital signal processing and speech processing.