BIT-LEVEL SYSTOLIC IMPLEMENTATION OF UNIVERSAL SECOND-ORDER RECURSIVE DIGITAL FILTER

Hon Keung Kwan and Pang Chung Tsang
Department of Electrical Engineering
University of Windsor
Windsor, Ontario
Canada N9B 3P4

Abstract
In this paper, we present our results on the bit-level systolic array implementation of two universal-output second-order recursive digital filters. With the change of a sign and a parameter in each resultant array, the type of the input to output transfer function can be switched to any one of low-pass, high-pass, band-pass, or band-stop Butterworth, Chebyshev or elliptic second-order transfer function.

Introduction
Multi-output passive second-order digital filter structures have been a subject of current interest [1]-[2]. The major features of this type of filters include capability of getting rid of zero-input and constant-input oscillations using magnitude truncations at the two outputs of the passive two-port used [3], and availability of all types of transfer function outputs with a minimum number of delays, multipliers, and adders [1]-[3]. In this paper, we consider the problem of implementation of two of our previous filter structures in the form of systolic array. To reduce the degree of complexity of the resultant systolic array, we have combined the low-pass, high-pass, band-pass, and bandstop outputs of a filter structure into a single universal output. With a change of a sign and the value of a parameter, the universal output can be changed to any type of desired transfer function.

Types of Transfer Function
In general, low-pass (LP), high-pass (HP), band-pass (BP) and band-stop (BS) second-order Butterworth and Chebyshev digital filter transfer functions can be expressed as [2]

\[ T_{LP}(z^{-1}) = a (1 + z^{-1})^2 / D(z^{-1}) \]  \hspace{1cm} (1)
\[ T_{HP}(z^{-1}) = a (1 - z^{-1})^2 / D(z^{-1}) \]  \hspace{1cm} (2)
\[ T_{BP}(z^{-1}) = a (1 - z^{-2}) / D(z^{-1}) \]  \hspace{1cm} (3)
\[ T_{BS}(z^{-1}) = a (1 + az^{-1} + z^{-2}) / D(z^{-1}) \]  \hspace{1cm} (4)

where
\[ D(z^{-1}) = 1 + b_1 z^{-1} + b_2 z^{-2} \]  \hspace{1cm} (5)

Similarly, for elliptic digital filters, the four types of filter transfer functions can be expressed as [2]

\[ T'(z^{-1}) = a (1 + az^{-1} + z^{-2}) / D'(z^{-1}) \]  \hspace{1cm} (6)

Multi-Output Second-Order Digital Filter
In general, (1)-(6) can be generalized as

\[ T(z^{-1}) = a (1 + az^{-1} + z^{-2}) / D(z^{-1}) \]  \hspace{1cm} (7)

For Butterworth and Chebyshev LP, \(a = 2\) and +ve sign for \(z^{-2}\) is chosen; for Butterworth and Chebyshev HP, \(a = -2\) and +ve sign for \(z^{-2}\) is chosen; for Butterworth and Chebyshev BP, \(a = 0\) and +ve sign for \(z^{-2}\) is chosen; and for Butterworth and Chebyshev BS, \(a\) is a real value and +ve sign for \(z^{-2}\) is chosen.

To implement (7), a second-order digital filter structure, as shown in Fig. 1, is used. The structure of the lossless digital two-port \((g)\) can be derived using the passivity and stability conditions [1].

Using the first two lossless digital two-ports derived in Table 1 of [1], two filter structures are formed as shown in Figs. 2-3, each of them consists of two delays, three multipliers, and ten adders. The actual digital transfer function of each of the two filters can be obtained by taking the \(z\)-transform of the output discrete-time signal over the input discrete-time signal as

\[ T(z^{-1}) = \frac{V(z^{-1})}{U(z^{-1})} \]  \hspace{1cm} (8)

Fig. 1 A second order 1-D passive state-space digital filter

\[ T(z^{-1}) = \frac{V(z^{-1})}{U(z^{-1})} \]  \hspace{1cm} (8)

Fig. 2 1-D universal second-order recursive digital filter (Configuration 1)
For the +ve sign adder input,
\[
T(z^{-1}) = \frac{2 \{(a_1-a_2-1)[1 + z^{-2}] + (B-1)(a_1-1) + 2a_2 \} z^{-1}}{1 - (a_1+a_2) z^{-1} + (1-a_1+a_2) z^{-2}}
\]
(9)

For the -ve sign adder input,
\[
T(z^{-1}) = \frac{2 \{(a_1-a_2-1)[1 - z^{-2}] + (B-1)(a_1-1) \} z^{-1}}{1 - (a_1+a_2) z^{-1} + (1+a_1-a_2) z^{-2}}
\]
(10)

Similarly for the Configuration 2 as shown in Fig. 3, two universal transfer functions can also be obtained.

For the +ve sign input,
\[
T(z^{-1}) = \frac{2 \{(a_1-a_2+1)[1 + z^{-2}] + (B-1)a_2 + 2(a_2-1) \} z^{-1}}{1 - (a_1+a_2) z^{-1} - (1+a_1-a_2) z^{-2}}
\]
(11)

For the -ve sign input,
\[
T(z^{-1}) = \frac{2 \{(a_1-a_2+1)[1 - z^{-2}] + (B-1)a_2 \} z^{-1}}{1 - (a_1+a_2) z^{-1} - (1+a_1-a_2) z^{-2}}
\]
(12)

**Systolic Implementation of Configuration 1**

For the digital filter shown in Fig. 2, the actual implementation can be carried out by implementing the following three difference equations.
\[
x(k+1) = [a_1(x(k)+y(k)+2u(k))-y(k)-u(k)] + u(k)
\]
(13)
\[
y(k+1) = [a_2(x(k)+y(k)+2u(k)) + x(k)+u(k)] + u(k)
\]
(14)
\[
v(k) = Bx(k)yz(k)x(k+1)-y(k+1)
\]
(15)

where \(x(k)\) and \(y(k)\) are the two state variables of the filter; \(u(k)\) is the input of the filter; \(v(k)\) is the output of the filter; \(a_1\), \(a_2\) and \(B\) are the multipliers' values, which can be found out by direct matching with the coefficients of the desired filter transfer function; and \([\cdot]/\cdot\) represents the operation of magnitude truncation.

In order to simplify the resultant systolic structure of the filter, the following substitutions are used.
\[
x'(k) = x(k) + u(k)
\]
(16)
\[
y'(k) = y(k) + u(k)
\]
(17)

On substituting (16)-(17) into (13)-(14), we obtain
\[
x(k+1) = [a_1(x'(k)+y'(k)) - y'(k)] + u(k)
\]
(18)
\[
y(k+1) = [a_2(x'(k)+y'(k)) + x'(k)] + u(k)
\]
(19)

Fig. 4 shows the block diagram representation of the systolic implementation of the Configuration 1 filter (eqns. (15),(18)-(19)). In Fig. 4, the two multiplexers which are denoted by 'MUX' are used to feed in the initial zero conditions at delays (i.e., \(x(0)\) and \(y(0)\)) at the start of computation. Immediately after the start, each MUX will act as a direct path for the signals \(x(k)\) and \(y(k)\). Besides, the two MUXes, four bit-level systolic arrays, namely A1, A2, B1 and B2, are used in the implementation. Figs. 5-8 show the internal circuits of the four arrays, respectively, of which the definition of the basic cells of the four arrays are shown in Fig. 9. The arrays A2 and B2 implement the eqns. (18)-(19). The array A1 is actually a bit-level pipelined multiplier and the array B1 is a bit-level pipelined adder. These arrays implement eqn. (15). The array A2 carries out the multiply and add operations before magnitude truncations of (18)-(19). The input (upper) array B2 (i.e., the one above the array A2) computes eqns. (16)-(17). The output (lower) array B2 (i.e., the one below the array A2) computes the subtraction of \(u(k)\) in (18)-(19). Similarly, the array A1 computes the multiply and add operation of \([Bx(k)yz(k)]\) of (15). The array B1 computes the addition of \([x(k+1)\) and \(-y(k+1)]\) to \([Bx(k)yz(k)]\). Actually, the arrays A1 and B1 are the degenerations of the arrays.
A2 and B2. For proper synchronization of signals for computation, each bit of \( u(k) \) is delayed by \( (8 + 3b) \) clock cycles (where \( b \) is the number of fraction bits used in the implementation). Before feeding \( y(k) \) into the array A1 for computation, it will pass through an inverter (i.e., \( \text{INV} \) as shown in Fig. 4). This \( \text{INV} \) calculates the 2's complement of \( y(k) \) if minus sign in (15) is selected. On the other hand, if plus sign in (15) is selected, \( \text{INV} \) acts as a direct path for the signal \( y(k) \).

Fig.5 Internal circuit for bit-level systolic array B1

![Internal circuit for bit-level systolic array B1](image)

Fig.6 Internal circuit for bit-level systolic array A1

![Internal circuit for bit-level systolic array A1](image)

c) Input array B2

![Input array B2](image)

b) Output array B2

![Output array B2](image)

Fig.7 Internal circuit for bit-level systolic array B2

Restricting the dynamic range of the signal to be \( \pm 1, 3 \) integer bits and \( b \) fractional bits are used for signal representation. For illustration, a particular case of \( b=1 \) is shown in Figs. 5-8. Similar to (4)-[8], a bit of signal of \( x'(k) \) of Figs. 5-8 is represented by \( x'(k-m) \), where \( m \) denotes the number of cycle delay to indicate the timing and \( j \) denotes the position of a bit having a bit value of 2. Similar notations are used for all other signals, \( y'(k-m) \) and \( u(k-m) \), and the multiplier values, \( a_t(k-m) \) and \( b_t(k-m) \). The operation of MT is carried out by applying the logical 'AND' value of the msb after multiplication (integer bit of 2^1) and the rounding bit (fractional bit of 2^-1) as the sum carry bit to the output array B2. In the first clock cycle, the lab of \( x(k) \) (i.e., \( x(k)-e \) ), \( y(k) \) (i.e., \( y(k)-u \) ) and \( u(k) \) (i.e., \( u(k)-e \) ) appear at the rightmost elements of the input array B2 to perform the addition of (16) and (17) and then pass the carry bits to the next left cell. The remaining bits of the signals \( x(k) \), \( y(k) \) and \( u(k) \) are fed into the next left cell of the input array B2 sequentially. The output \( x'(k) \) and \( y'(k) \) of the input array B2 will be obtained bit by bit in
the subsequent clock cycles. These outputs of B2 then pass through two delays to get proper synchronization of \( x'(k-3).b \) (or \( y'(k-3).b \)) and \( x'(k-3).a \) (or \( y'(k-3).a \)) which should appear at the top right cell of the array A2 at the same time. Together with the map of the multiplier coefficients, \( a(k-3)z \) and \( a(k-3)z \), the basic cell of the array A2 will perform the bit-level multiplication and addition.

In the next clock cycle, every basic cell passes its carry bits and the multiplier bit (\( a(k-m_3) \)) to the next left cell and the signal bits (\( x'(k-m_2) \) and \( y'(k-m_2) \)) to the cell below. In the next cycle, the second top right cell accepts its input bits (\( x'(k-4).a \) and \( y'(k-4).a \)) at the top; \( x'(k-4).b \) and \( y'(k-4).b \) in the diagonal direction). Together with the multiplier bits and carry bits, the bit-level multiplication and addition is then carried out. Similar operation occurs at the second cell from the top in the right boundary of the array A2. The whole process is then repeated for the next cycle.

After \((2\times2b)\) clock cycles, \( x'(k-5.2b).a \) and \( y'(k-5.2b).a \) are obtained at the outputs of the array A2. These outputs then pass through \((3b)\) delays to obtain proper synchronization with the msb after multiplication (\( x'(k-6.3b) \) and \( y'(k-6.3b) \)).

Fig. 10: Internal circuit for bit-level systolic array A2 (Configuration 2)

**Concluding Remarks**

In this paper, bit-level systolic arrays for the implementation of two universal second-order passive recursive digital filters have been presented. With a control of the sign of the right input of the left-most adder and the value of \( a \), any type of filter transfer function can be obtained through the same output point. As the form of an all-pass transfer function is quite different from the others, it has not been considered for implementation in the present work.

**Acknowledgment**

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada.

**References**


