V3.3

A High Throughput Systolic Implementation of The Second Order Recursive Filter

Zhou B. B. and Brent R. P.
Computer Sciences Laboratory
Australian National University, Canberra, Australia

ABSTRACT

This paper introduces a high throughput systolic implementation of the direct-form second-order recursive filter. The systolic structure has the advantage of regularity over implementations of the block-state-variable form. Since communication is very expensive in VLSI implementations in terms of area, as well as time, our regular structure is better for VLSI than those based on block-state-variable filter descriptions.

1. Introduction

Many $2D$ (two-dimensional) VLSI structures have been introduced in the literature for $1D$ (one-dimensional) recursive digital filters with high throughput. The techniques applied for these implementations are mainly based on block-state-variable filter descriptions. This paper introduces a high throughput systolic implementation of direct-form second-order recursive filter. The number of multipliers required in the systolic structure is greater than in those structures based on the block-state-variable form. However, communication is very expensive in VLSI implementations in terms of area, as well as time[1]. Since our structure has the advantage of regularity, it is more suitable for VLSI than implementations of the block-state-variable form.

In Section 2, we derive a parallel algorithm for the second-order recursive filter. By modifying the original equation, our parallel algorithm allows L outputs to be computed simultaneously. The algorithm consists of two parts, namely recursive convolution and linear convolution. Thus Section 3 describes first two systolic sub-systems for these two convolutions and then shows that these two sub-systems can naturally be connected in a systolic way. In Section 4 the area complexity and the stability of the derived algorithm are discussed.

2. Algorithm

A second-order direct-form recursive filter can be expressed as

$$y_i = \sum_{j=0}^{2} w_j x_{i-j} + \sum_{j=1}^{2} r_j y_{i-j} \tag{1}$$

Because y_i depends on the availability of the immediately previous output y_{i-1}, the equation (1) can not be computed in parallel. To solve this problem, we make the following modification.

According to (1), we write y_{i-1} explicitly as

$$y_{i-1} = \sum_{j=0}^{2} w_j x_{i-1-j} + \sum_{j=1}^{2} r_j y_{i-1-j} \tag{2}$$

Substituting (2) into (1), we have

$$y_i = \sum_{j=0}^{3} w_j^{(1)} x_{i-j} + \sum_{j=2}^{3} r_j^{(1)} y_{i-j} \tag{3}$$

where $w_0^{(1)} = w_0$, $w_1^{(1)} = w_1 + r_1 w_0$, $w_2^{(1)} = w_2 + r_1 w_1$, $w_3^{(1)} = r_1 w_2$, $r_2^{(1)} = r_1^2 + r_2$ and $r_3^{(1)} = r_1 r_2$.

We see, from (3), that y_i does not depend on y_{i-1}. Thus y_i and y_{i-1} can be produced simultaneously. We can also write y_{i-2} explicitly from (1) and substitute it into equation (3). Then

$$y_i = \sum_{j=0}^{4} w_j^{(2)} x_{i-j} + \sum_{j=3}^{4} r_j^{(2)} y_{i-j} \tag{4}$$

where $w_0^{(2)} = w_0^{(1)}$, $w_1^{(2)} = w_1^{(1)}$, $w_2^{(2)} = w_2^{(1)} + r_2^{(1)} w_0$, $w_3^{(2)} = w_3^{(1)} + r_2^{(1)} w_1$, $w_4^{(2)} = r_2^{(1)} w_2$, $r_3^{(2)} = r_3^{(1)} + r_2^{(1)} r_1$ and $r_4^{(2)} = r_2^{(1)} r_2$.

It is clear that 3 outputs can be computed in parallel by using equation (4).

We can prove that, after $L-1$ iterations, y_i becomes

$$y_i = \sum_{j=0}^{L+1} w_j^{(L-1)} x_{i-j} + \sum_{j=L}^{L+1} r_j^{(L-1)} y_{i-j} \tag{5}$$

In the above equation, the coefficients $w_j^{(L-1)}$ and $r_j^{(L-1)}$ may be computed by the following iterative algorithm.

```
for 2 ≤ l ≤ L do
  begin
    for 0 ≤ j ≤ l+1 do
      begin {to compute w_j^(l-1)}
        if j < l-1 then
          w_j^(l-1) := w_j^(l-2)
        else if l-1 ≤ j < l+1 then
          w_j^(l-1) := w_j^(l-2) + r_{l-1}^(l-2) * w_{j-(l-1)}
        else w_j^(l-1) := r_{l-1}^{l-2} * w_2
      end; {end of computing w_j^(l-1)}
    {to compute r_j^(l-1)}
    r_l^(l-1) := r_l^(l-2) + r_{l-1}^(l-2) * r_1;
    r_{l+1}^(l-1) := r_{l-1}^(l-2) * r_2
    {end of computing r_j^(l-1)}
  end.
```

(6)

Although $L-1$ addditional multiplications have been introduced, y_i in (5) depends only on y_{i-L} and $y_{i-(L+1)}$ so L outputs can be computed simultaneously. Therefore, we may construct a high throughput parallel architecture for the second order recursive filter.

3. Architecture

The equation (5) can be split into two parts:

$$y_i = c_i + \sum_{j=L}^{L+1} r_j^{(L-1)} y_{i-j} \tag{7}$$

CH2561-9/88/0000-2053 $1.00 © 1988 IEEE

and

$$c_i = \sum_{j=0}^{L+1} w_j^{(L-1)} x_{i-j} \tag{8}$$

The equation (8) is just a linear convolution and (7) is called recursive convolution. Therefore, in this section we first derive a $2D$ systolic structure for the second order recursive convolution problem and then a $2D$ systolic ring structure for the linear convolution problem[5] is briefly described. Finally, because the output from the sub-system for the linear convolution is the input to that for the recursive convolution, we discuss the interconnection between these two structures.

3.1. Structure for recursive convolution

From (7), we express the second-order zero-input recursive filter as

$$y_i = r_L^{(L-1)} y_{i-L} + r_{L+1}^{(L-1)} y_{i-(L+1)} \tag{9}$$

Since L outputs can be computed in parallel, we arrange the output in (9) into L groups.

Setting $i = Lk + l$ for $0 \le l \le L-1$ and $k = 0, 1, 2, \cdots$, we have

$$\begin{aligned} y_{Lk+l} &= r_L^{(L-1)} y_{Lk+l-L} + r_{L+1}^{(L-1)} y_{Lk+l-(L+1)} \\ &= r_L^{(L-1)} y_{L(k-1)+l} + r_{L+1}^{(L-1)} y_{L(k-1)+l-1} \end{aligned} \tag{10}$$

In (10), y_{Lk+l} can be further divided into two parts:

$$y_{Lk+l} = y_{Lk+l}^{(0)} + y_{Lk+l}^{(1)} \tag{11}$$

where

$$\begin{cases} y_{Lk+l}^{(0)} = r_L^{(L-1)} y_{L(k-1)+l} \\ y_{Lk+l}^{(1)} = r_{L+1}^{(L-1)} y_{L(k-1)+l-1} \end{cases} \tag{12}$$

From (11) and (12), a parallel structure can be easily obtained. An example with $L = 5$ is depicted in Fig. 1.

The dashed lines in Fig. 1 denote zero-delay lines. One unit time for this system is the time for computing one multiplication plus two additions.

The derived structure is not systolic because there is a long wire directed from the left-most column to the right-most column. This long wire can be eliminated by a column permutation.

For simplicity, consider an example with $L = 5$. We number the 5 columns in Fig. 1 from left to right as

1 2 3 4 5

For the first two rows of the structure, we first fold the structure and then interleave, giving the permutation

1 5 2 4 3

For the third row we use this permutation followed by an odd-even interchange. The column indices after the above permutations become

$$\begin{array}{c} 1\ 5\ 2\ 4\ 3 \\ 1\ 5\ 2\ 4\ 3 \\ 5\ 1\ 4\ 2\ 3 \end{array} \tag{13}$$

Using these permutations, the systolic layout shown in Fig. 2 is derived from the structure of Fig. 1.

3.2. 2D systolic ring for linear convolution

Fig. 3 depicts a ring structure for solving the linear convolution problem. A detailed derivation of this structure can be found in [5]. With L columns, this structure allows L inputs to enter the system in parallel and produces L outputs simultaneously. Since $L = 5$ in Fig. 3, inputs and outputs have been arranged into 5 groups, respectively $\{x_{5k+l}\}$ and $\{y_{5k+l}\}$ for $0 \le l \le 4$. It has been proved that the most efficient $1D$ systolic array for solving this problem[3][4] is just a special case with $L = 1$. The corresponding systolic layout is depicted in Fig. 4. To transform Fig. 3 into Fig. 4, we use the following procedure.

We first number the columns in Fig. 3 as

1 2 3 4 5

We let the first row correspond to the permutation

1 5 2 4 3

For the remaining rows, we apply odd-even interchanges, giving $L+2$ rows as follows:

$$\begin{array}{c} 1\ 5\ 2\ 4\ 3 \\ 1\ 2\ 5\ 3\ 4 \\ 2\ 1\ 3\ 5\ 4 \\ 2\ 3\ 1\ 4\ 5 \\ 3\ 2\ 4\ 1\ 5 \\ 3\ 4\ 2\ 5\ 1 \\ 4\ 3\ 5\ 2\ 1 \end{array} \tag{14}$$

From (14) we obtain Fig. 4 by drawing the input and output lines explicitly.

3.3 Interconnection

In section 3.1, we assumed that the input c_i in (7) is zero when constructing the structure for recursive convolution. We should point out that the structure can also compute the recursive convolution with $c_i \neq 0$. When c_i is not zero, the equation for $y_{Lk+l}^{(0)}$ (or $y_{Lk+l}^{(1)}$) in (12) should be rewritten as

$$y_{Lk+l}^{(0)} = c_{Lk+l} + r_L^{(L-1)} y_{L(k-1)+l} \tag{15}$$

Since the basic processing element performs the computation of multiplication and accumulation, the inputs, c_{Lk+l} for $0 \le l \le L-1$, can enter the system from L input ports on the top of the structure. Therefore, we may put the structure for linear convolution on top of the structure for recursive convolution to obtain our desired result.

From (15), we see that the outputs produced by Fig. 3(or 4) should have the same subscripts as the inputs required by Fig. 1(or 2) in order to make the communication between the two structures correct. This means that columns with the same indices should be connected together. It is easy to see, from (13) and (14), that the two structures can be connected systolically if we reverse the order of the columns in Fig. 2. Our final result is depicted in Fig. 5.

4. Dicussion

For a second order recursive filter with L outputs in parallel, the number of multipliers used in our systolic structure is $L(L+4)$, which is greater than $L(L+1)/2+4L+3$, the number of multipliers in structures for the block-state-variable form, if $L \ge 4$. In VLSI implementation, however, communication is very expensive in terms of area, as well as time. Because of their irregularity, the structures for the block-state-variable form will occupy more area than our regular structure. For instance, a good systolic implementation of the block-state-variable form[4] takes at least $2L^2$ units of area for a second order recursive filter, if one unit of area is the area taken by one multiplier.

Suppose the original impulse response of a second order recursive filter in the z domain is $H(z)$. We can prove the impulse response after modification becomes

$$H'(z) = H(z)\frac{D(z)}{D(z)} \tag{16}$$

where $D(z) = 1 + \sum_{j=1}^{L-1} r_j^{(j-1)} z^{-j}$ and $r_j^{(j-1)}$ is computed by the iterative algorithm in (6).

Theoretically the impulse response after the modification is equivalent to the original one. However, the effect of the finite wordlength may cause instability if roots of $D(z)$ do not all lie in the unit circle. Recently, we have developed a new parallel algorithm[6]. This algorithm is guaranteed to be stable. Although the number of multipliers in the new structure is greater than the one described here, the data flow is faster in that structure. We have proved that, to achieve the same throughput, the new structure will take less area if the time for computing one multiplication is not greater than the time for computing two additions.

5. Conclusions

In this paper we have described a high throughput systolic implementation of the direct-form second-order recursive filter. Our implementation is better for VLSI than those based on the block-state-variable filter descriptions because it is regular and modular. However the large number of multipliers may limit its application. We hope that some stablized parallel algorithms with a reduced number of multipliers, but having a regular structure, will be developed in the future.

6. References

[1] Kung, H. T., Why systolic architectures? *IEEE Comput. Mag.*, Vol. 15, No. 1, Jan. 1982, pp.32–63.

[2] Kung, H. T. and Song, S. W., A systolic $2D$ convolution chip, Tech. Report, Dept. of Comput. Science, Carnegie-Mellon Univ., Pittsburgh, Pa. 15213, 1981.

[3] Li, G. J. and Wah, B. W., The design of optimal systolic arrays, *IEEE Trans. on Comput.*, Vol. C–34, No. 1, Jan. 1985, pp. 66-77.

[4] Lu, H. H., Lee, E. A. and Messerschmitt, D., Fast recursive filtering with multiple slow processing elements, *IEEE Trans. on Circuits and Systems*, Vol. CAS–32, No. 11, Nov. 1985, pp. 1119–1129.

[5] Zhou, B. B., $2D$ systolic ring structures for solving linear convolution probems, Tech. Report, Comput. Sci. Lab., Australian National Uni., Australia, Sept. 1985.

[6] Zhou, B. B. and Brent, R. P., A stablized parallel algorithm for the direct-form second-order recursive filter, Tech. Report, Comput. Sci. Lab., Australian National Uni., Australia, to appear.

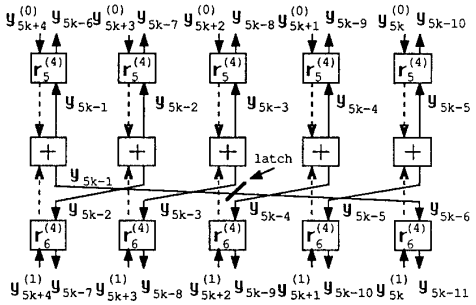

Fig. 1. Structure for recursive convolution (L=5)

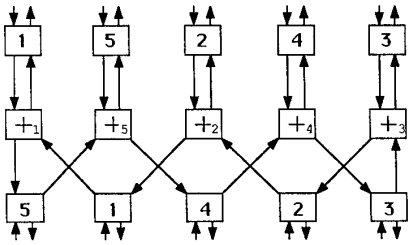

Fig. 2. Systolic layout of Fig. 1

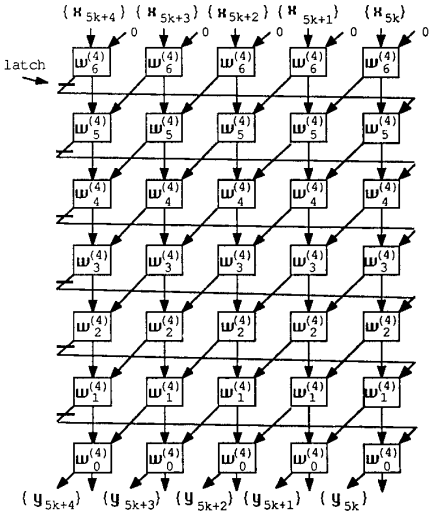

Fig. 3. Ring structure for linear convolution

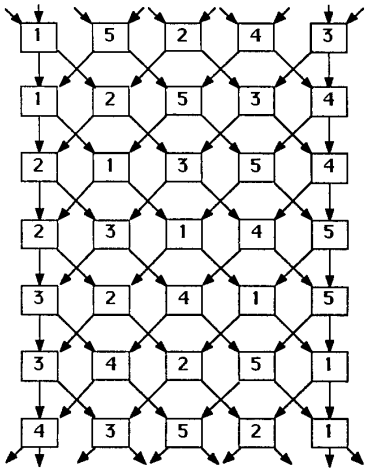

Fig. 4. Systolic layout of Fig. 3

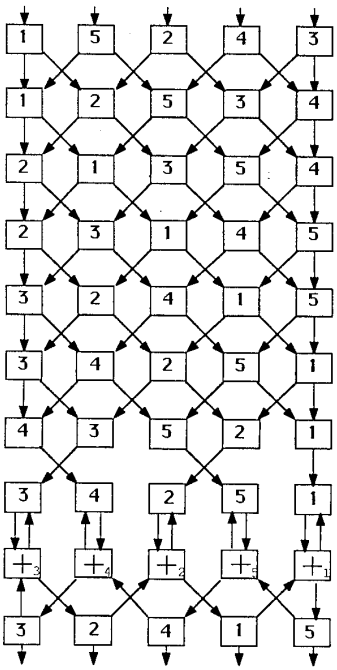

Fig. 5. Systolic implementation of direct-form second-order recursive filter