ISSPA 87, signal processing, theories, Implementations and Applications.
Editor: B. Boashash, Brisbane, Australia
24-28 August, 1987.

An Efficient Architecture for Solving the Recursive Convolution Equation with High Throughput

B. B. ZHOU and R. P. BRENT
Computer Sciences Laboratory, Department of Engineering Physics, Australian National University, Canberra

1. Introduction

The recursive convolution equation can be expressed as

$$y_i = \sum_{j=1}^{N} r_j y_{i-j} \tag{1}$$

It plays a very important role in digital signal processing. It is usually considered that recursion implies sequential execution. Thus only $1D$ (one-dimensional) architectures were taken into account in VLSI implementation of this kind of problem. A good example is the two-slow[5] $1D$ systolic array derived by Kung[2]. Recently, Robert et. al.[6] modified this $1D$ systolic array by using the divide-and-conquer technique[1]. With about the same area, the modified array has twice the throughput of the original one.

In this paper, we shall derive a $2D$ systolic architecture for solving the recursive problem (1). Our solution has about twice the area, but twice the throughput of Robert's array.

Section 2 briefly describes Kung's $1D$ two-slow systolic array and gives the idea of deriving Robert's array. Section 3 gives a detailed discussion of our $2D$ architecture with a throughput rate of two. An example of the use of our efficient architecture to solve an IIR problem is given in section 4.

2. $1D$ Systolic Arrays

Fig. 1 depicts the two-slow systolic array for the recursive convolution problem. In this array, coefficients r_i are prestored in cells. The output y_i travels from right to left to accumulate its terms. Then it is fed back from the left end of the array as an input for the computation of other outputs. In order to produce the correct results, consecutive y_i's are separated by two time units, so the throughput rate is only $1/2$.

If the coefficients with odd (or even) subscripts are all zeros, y_i needs only to meet $y_{i-2}, y_{i-4}, y_{i-6}, \cdots$ (or $y_{i-1}, y_{i-3}, y_{i-5}, \cdots$). In this case, the array described above can be used with a throughput rate of one, as shown in Fig. 2.

It may be possible to divide the general problem into two sub-problems, one containing the odd subscript coefficients and the other containing the even subscript coefficients. These two sub-problems can be implemented on the arrays in Fig. 2 with a throughput rate of one. Then we may combine the two partial results into a result for the original problem, obtaining an array with twice the throughput of the two-slow array. Unfortunately we can not use this procedure for the equation in (1) if $r_1 <> 0$. We see, from Fig. 2(b), that the output coming out from the system will be fed back into the system immediately at the next time unit. However this output is now a partial result only and it has to be combined with another output to form a complete result for the original problem. Therefore the complete result must be fed back into the system before it is produced. To solve this problem, the following modification is required.

From (1), we obtain

$$y_{i-1} = \sum_{j=1}^{N} r_j y_{i-1-j} \tag{2}$$

Substituting (2) into (1), we have

$$\begin{aligned} y_i &= r_1 y_{i-1} + \sum_{j=2}^{N} r_i y_{i-j} \\ &= r_1 \sum_{j=1}^{N} r_j y_{i-1-j} + \sum_{j=2}^{N} r_j y_{i-j} \\ &= \sum_{j=2}^{N+1} r_j^{(1)} y_{i-j} \end{aligned} \tag{3}$$

where $r_j^{(1)} = r_j + r_1 r_{j-1}$, for $2 \le j \le N$ and $r_{N+1}^{(1)} = r_1 r_N$. This modification does not change the total number of coefficients.

We now divide (3) into two sub-equations:

$$\begin{cases} y_i^0 = \sum_{m=1}^{N/2} r_{2m}^{(1)} y_{i-2m}; \\ y_i^1 = \sum_{m=1}^{N/2} r_{2m+1}^{(1)} y_{i-(2m+1)}. \end{cases} \tag{4}$$

In (4), we assume that N is even without loss of generality. Fig. 3 depicts two sub-systems for solving these two sub-problems with a throughput rate of one.

From Fig. 3 we see that we have extra one time unit to accumulate the two partial results so that the complete result is formed before it is required by the system. Therefore, we can easily combine the two sub-systems into a system for solving the recursive convolution problem with a throughput rate of one, as shown in Fig. 4.

The architecture in Fig. 4 is slightly different from the one derived by Robert, et. al, but both systems work equally well.

3. An Efficient 2D Systolic Architecture

We see, from (1), that y_i can not be produced before y_{i-1} is available. However, in (3), y_i does not depend on the availability of y_{i-1}. Therefore y_i and y_{i-1} can be produced simultaneously. The throughput rate is further increased. To achieve this, we use the following procedure.

First we arrange the outputs in (3) into two groups, one containing the outputs with even subscripts and the other those with odd subscripts:

$$\begin{cases} y_{2k} = \sum_{j=2}^{N+1} r_j^{(1)} y_{2k-j}; \\ y_{2k-1} = \sum_{j=2}^{N+1} r_j^{(1)} y_{2k-1-j}, \end{cases} \tag{5}$$

where $k = 1, 2, 3, \cdots$.

Each group above can be further divided into two parts, that is,

$$\begin{cases} y_{2k} = y_{2k}^0 + y_{2k}^1; \\ y_{2k-1} = y_{2k-1}^0 + y_{2k-1}^1. \end{cases} \tag{6}$$

In the above equations,

$$\begin{cases} y_{2k}^0 = \sum_{m=1}^{N/2} r_{2m}^{(1)} y_{2(k-m)}; \\ y_{2k}^1 = \sum_{m=1}^{N/2} r_{2m+1}^{(1)} y_{2(k-m)-1} \end{cases} \tag{7}$$

and

$$\begin{cases} y_{2k-1}^1 = \sum_{m=1}^{N/2} r_{2m}^{(1)} y_{2(k-m)-1}; \\ y_{2k-1}^0 = \sum_{m=1}^{N/2} r_{2m+1}^{(1)} y_{2(k-m)-2}. \end{cases} \tag{8}$$

The four sub-equations in (7) and (8) are simply convolution problems. They can be implemented on the two-slow systolic array described in section 2. Then we combine the partial results to form a result for the original problem, as shown in Fig. 5.

Since the sub-problems are implemented on the two-slow systolic arrays, the method described in section 2 can be used directly in Fig. 5 to double the throughput rate.

The coefficients in Fig. 6 are pre-computed as follows: For $1 \le l \le 3$

$$\begin{cases} r_j^{(l)} = r_j^{(l-1)} + r_l^{(l-1)} r_{j-l}, & l+1 \le j \le N+l-1; \\ r_{N+l}^{(l)} = r_l^{(l-1)} r_N, \end{cases} \tag{9}$$

where $r_j^{(0)} = r_j$.

The dash-lines between the adders in Fig. 6 are zero-delay lines. Assuming that the time for performing three additions is less than that for one multiplication and one addition, they can be computed in one unit of time. Our final architecture is given in Fig. 7.

4. An IIR Problem

An IIR digital filter can be expressed as

$$y_i = \sum_{j=0}^{M} w_j x_{i-j} + \sum_{j=1}^{N} r_j y_{i-j} \tag{10}$$

The second sum of the above equation is a recursive convolution problem. It can be implemented on the array described in the previous section. The first sum is a linear convolution problem. To solve this problem, we first divide it into eight sub-problems, in a similar way to the recursive convolution problem, so that the sub-problems can be implemented on two-slow arrays with a throughput rate of one. Instead of connecting the partial results together to form a result of the linear convolution problem, we use these partial results as eight inputs to the array for the recursive convolution problem. We then obtain a system for solving IIR problems, as shown in Fig. 8.

The system in Fig. 8 consists of three parts. Part 2 is the architecture for recursive convolution problems. The eight sub-arrays for the linear convolution problem are divided into two parts (part 1 and part 3), one placed on the top of part 2 and one underneath part 2. In the following, we discuss the communication between part 1 and part 2. The communication between part 2 and part 3 is similar.

It can be seen, from (10), that the outputs produced by part 1 should have the same subscripts as the inputs required by part 2 in order to make the communication correct. Fig. 9 depicts an example to show that the inputs and outputs in part 1 are organized to match this communication requirement to the array in Fig. 7. Because we are considering the communication problems, Fig. 9 shows only the first cells of the four sub-arrays in part 1.

The drawback of Fig. 9 is that global propagation of the input is required. We can apply the cut theorem[3][4] to Fig. 9 to eliminate the global interconnection. The cut set is given in Fig. 9 and the resulting structure with local interconnection is depicted in Fig. 10.

5. Concluding Remarks

In this paper, we derive an efficient 2D systolic architecture for solving the recusive concolution equation. With about twice the area, this system has twice the throughput rate of Robert's array. We also give an example to show that, by using our systolic architecture, IIR problems can be computed systolically with a throughput rate of two. The idea described in this paper can be extended further to achieve even higher throughput. However it seems that global communication can not be avoided if we want to obtain a throughput rate higher than two.

6. References

[1] Horowitz, E. and Zorat, A., Divide-and-Conquer for parallel procesing, *IEEE Trans. on Computers*, Vol.C-32, No. 6, June, 1983, pp. 582-585.

[2] Kung, H. T., Special-purpose devices for signal and image processing: an opportunity in VLSI, Tech. Report, Dept. of Comput. Science, Carnegie-Mellon Univ., Pittsburgh, Pa. 15213, 1980.

[3] Kung, H. T. and Lam, M. S., Wafer-scale integration and two-level pipelined implementations of systolic arrays, *Journal of Parallel and Distributed Computing*, Vol. 1, 1984, pp. 32-63.

[4] Kung, S. Y., On supercomputing with systolic/wavefront array processors, *Proceedings of the IEEE*, Vol. 72, No. 7, July 1984, pp. 867-884.

[5] Leiserson, C. E., Area-efficient VLSI computation, PhD dissertation, Carnegie-Mellon University, 1981. (Published by MIT Press, Cambridge, Mass., 1983.)

[6] Robert, Y. and Tchuente, M., An efficient systolic array for the 1*D* recursive convolution problem, *Journal of VLSI and Computer Systems*, Vol. 1, No. 4, pp. 398-407.

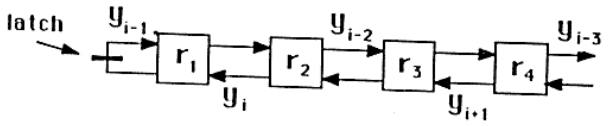

Fig. 1. A two-slow systolic array

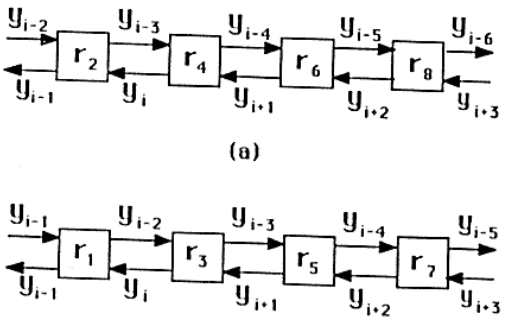

Fig. 2 Implementation of recursive problems with (a) odd subscript coefficients = 0 and (b) even subscript coefficients = 0

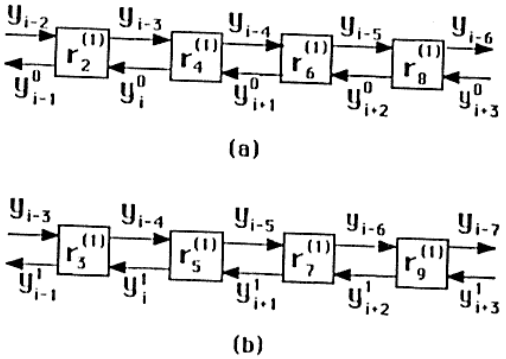

Fig. 3 Implementation of two sub-equations in (4)

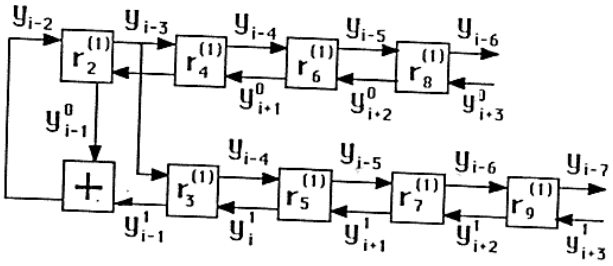

Fig. 4 A systolic array for the recursive convolution with a throughput rate of one

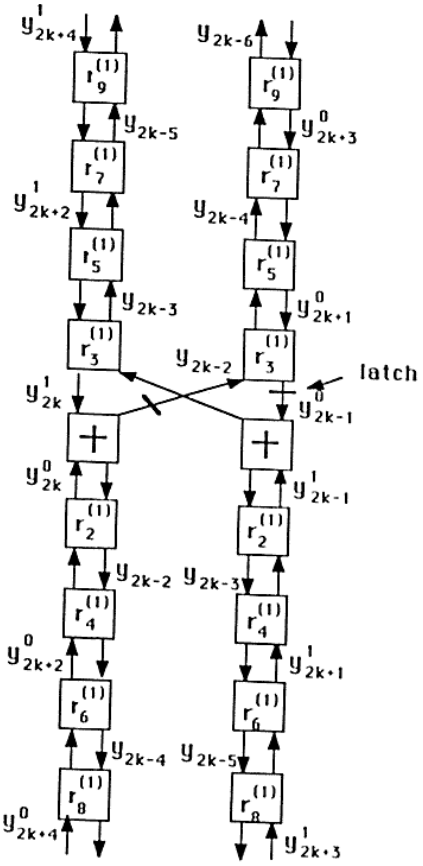

Fig. 5 An architecture producing two outputs simultaneously

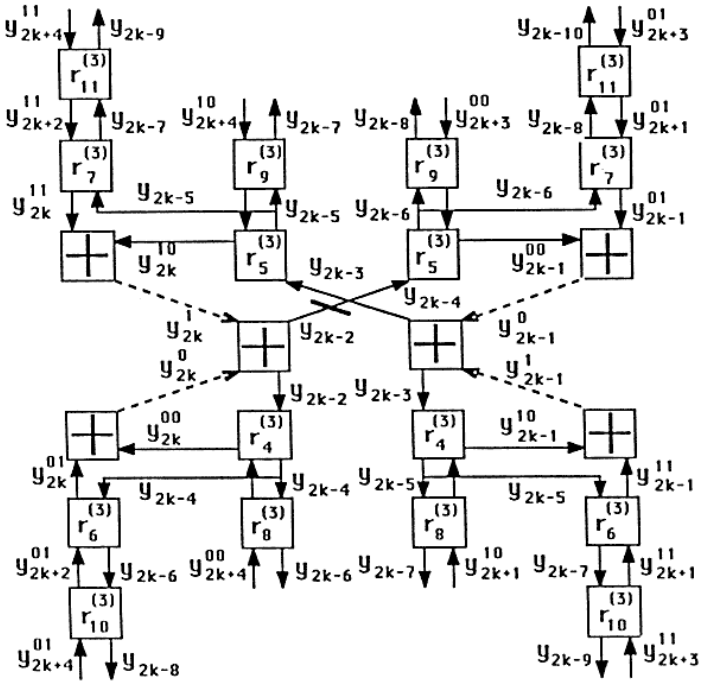

Fig. 6 An architecture with a throughput rate of two

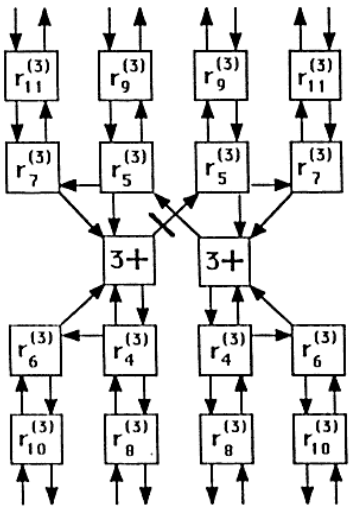

Fig. 7 The final systolic architecture

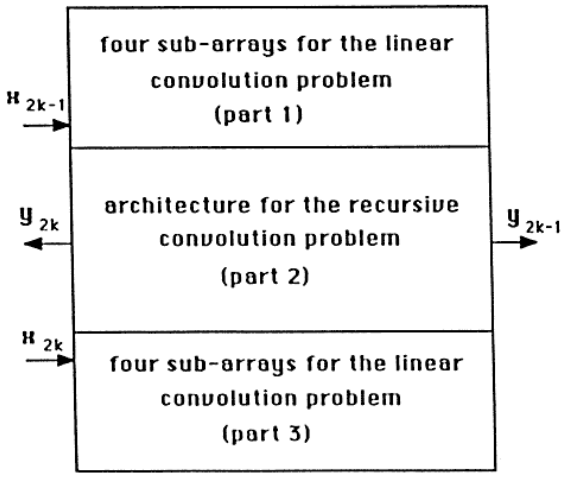

Fig. 8 An schematic structure for solving IIR problems

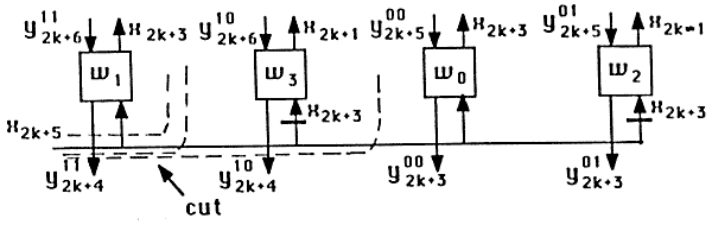

Fig. 9 The input/output arrangement in part 1

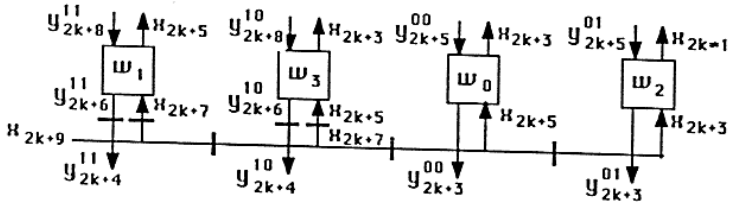

Fig. 10 Systolic input/output arrangement in part 1