Implementation of the Super-Systolic Array for Convolution

Jae-Jin Lee, Gi-Yong Song
Dept. of Computer Engineering
Chungbuk National University, Cheongju Chungbuk 361-763 Korea
E-mail : gysong@chungbuk.ac.kr

Abstract - High-performance computation on a large array of cells has been an important feature of systolic array. To achieve even higher degree of concurrency, it is desirable to make cells of systolic array themselves systolic array as well. The architecture of systolic array with its cells consisting of another systolic array is to be called super-systolic array.

In this paper we propose a scalable super-systolic array architecture which shows high-performance and can be adopted in the VLSI design including regular interconnection and functional primitives that are typical for a systolic architecture.

I. Introduction

VLSI has made implementation of system hardware or even highly parallel array processors economically feasible and technically realizable[1]. A systolic array[2-5] formed by interconnecting a set of identical data-processing cells in a uniform manner is a combination of an algorithm and a circuit that implements it, and is closely related conceptually to arithmetic pipeline. In a systolic array, data words flow from external memory in a rhythmic fashion, passing through many cells before the results emerge from the array's boundary cell and return to external memory. Upon receiving data words, each cell performs the same operation and transmits the intermediate results and data words to adjacent cells synchronously.

High-performance computation over a large array of cells has been an important feature of systolic array. To achieve even higher degree of concurrency, it is desirable to make cells of systolic array themselves systolic array as well. We will refer to this architecture of systolic array consisting of another systolic array in a hierarchical manner as a super-systolic array.

In this paper we propose a scalable super-systolic array architecture which consists of another systolic array, produces high -performance, and can be directly adopted in the VLSI design including regular and local interconnection and functional primitives that are typical for a systolic architecture. The cell of a systolic array derived through projection and scheduling upon the dependence graph, DG, [1] from the given behavior can be designed as another systolic array, and this systolization procedure of implementing cell as another systolic array could be applied repeatitively in a hierarchical manner until a cell having only primitive operators is obtained.

We choose a super-systolic array for convolution as an example to demonstrate the procedure for deriving a super-systolic array, and then check the improvement on performance because it is a simple problem with a variety of enlightening systolic solution, and more importantly, it is representative of a wide class of computation suited to systolic designs.

The systolic array for convolution can be thought of as a logical systolic array in a sense that the array assumes all operation to complete in a unit delay to maintain rhythmic data flow. However, difference in the required delay for different operators may not be negligible and forced selection of unit delay as the longest delay among operators ends up with a low-performance systolic array. To make the assumption reasonable, namely, to transform logical systolic array into a virtual systolic array, time-consuming operation such as multiplication should be effectively implemented with more primitive operations. The strategy mentioned above calls for implementation of another systolic array performing complex operation in a cell of the logical systolic array. We will refer to this virtual systolic array resulted from the above strategy as super-systolic array.

Derived super-systolic array for convolution is modeled and simulated in RT level using VHDL, then synthesized to a schematic and finally implemented using the cell library based on $0.35\,\mu m$ 1-poly 4-metal CMOS technology.

II. Super-Systolic Array for Convolution

A. Systolic Array for Convolution

The problem of convolution is defined as follows[1]: Given two sequences $u(i)$ and $w(i)$, i = 0, 1, …, N-1, the convolution of two sequence is

$$y(i) = \sum_{k=0}^{N-1} u(k)w(i-k)$$

The convolution problem can be viewed as a problem of combining two data streams, $w(i)$'s and $u(i)$'s, in a certain manner to form a resultant data stream of $y(i)$'s.

The rectangular shaped DG for convolution is shown in Fig. 1(a). Note that the $w(i)$ coefficients along the columns remains unchanged. This mean that the coefficient $w(i)$ may be a stored constant in the ith processor, as shown in Fig. 1(b). We first apply the systolization procedure to the convolution signal flow graph, SFG, [1] in Fig. 1(b). In Fig. 2(a) convolution SFG is shown along with the cut-sets. If we scale the delay by a factor of two, i.e., D $\rightarrow$ 2D′, then one delay can be transferred from the left-going edges to right-going edges in the cut-sets, leading to the systolic array for convolution shown in Fig. 2(b). The pipeline period, α , is two for this convolution array.

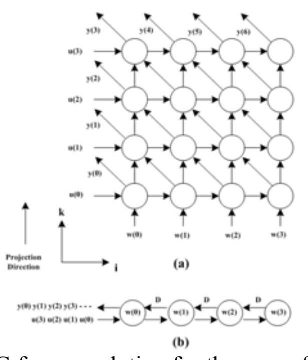


Fig. 1. (a) DG for convolution for the case of N=4 (b) SFG obtained by projection along [0 1]

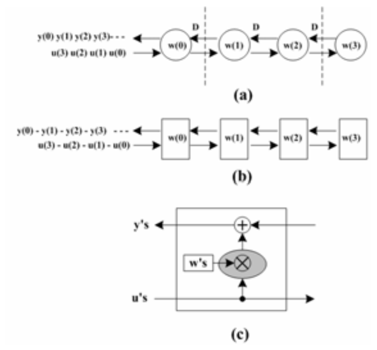


Fig. 2. (a) Convolution SFG (b) Systolic array for convolution (c) Systolic array cell

The systolic array consists of identical linearly-connected processing elements, or cells, as depicted in Fig. 2(b). The internal structure of each cell is shown in Fig. 2(c). Each cell contains a multiplier and an adder. In a high-complexity system, area restriction is very crucial, thus leads to a need for a systolic array-based implementation of the area -consuming operator such as multiplier[6-8]. The multiplier in each cell of the systolic array for convolution is a natural candidate for systolization and should be implemented using systolic array as is proposed in this paper.

B. *Systolic Array Multiplier*

In a high-complexity system, area restriction is very crucial and affects the final performance of the system. The systolic multiplier allows us to get high processing speed as well as limited resource consumption.

The DG for *M*-bit multiplier performing $p(i)=u(i)w(i)$ is shown in Fig. 3 for the case of $M=4$. The DG obtained can now be safely projected in the *ij*-direction, [1 1]. The default schedule is used. The data flow pattern divides the DG into upper and lower part as shown with dashed line in Fig. 3, each part resulting in systolic array with different interconnection. At the same time, output data are produced from every node, resulting in a large number of output port in the SFG generated by the *ij*-projection. To circumvent this problem, it is possible to extend the index space [9] of the DG, so the output occurs at points that will be mapped to the boundary nodes of the SFG only. The modified DG using procedure of index space extension is shown in Fig. 4. When the modified DG is projected along the *ij*-direction, the SFG with input and output port on the boundary node only is obtained. The systolic array for 4-bit multiplier with two ports, one for input and one for output each, is shown in Fig. 5. Output data emerge from the rightmost node and the array use *M* cells. Multiplicands stay in cells, multipliers and results move in the opposite direction.

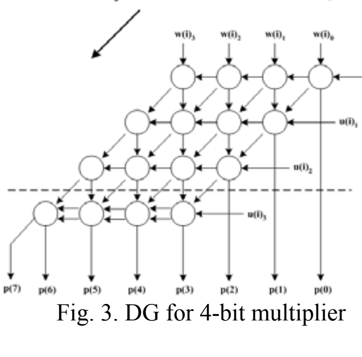


Fig. 3. DG for 4-bit multiplier

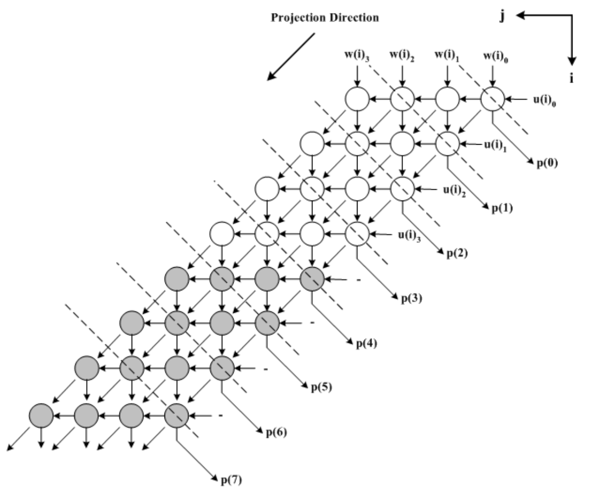


Fig. 4. Modified DG using index space extension

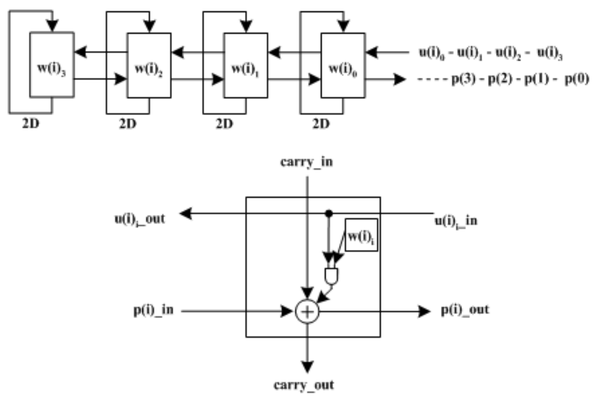


Fig. 5. Systolic array for 4-bit multiplier with two ports and internal structure of the cell

C. *Super-Systolic Array*

Making cells of systolic array themselves systolic array results in even higher degree of concurrency and even lower resource consumption, referring to the original systolic array as a super-systolic array.

An example of super-systolic array for convolution is depicted in detail in Fig. 6. Each cell of systolic array for convolution contains multiplier and adder. To get higher processing speed and area minimization, multiplier is designed again using systolic array, making systolic array for convolution a super-systolic array and the cell of systolic array for convolution a super-cell. The cell of a super-systolic array consisting of another systolic array is referred to as a super-cell. Internal structure of each cell is identical and is shown in Fig. 6.

To compare the performance of systolic array for convolution shown in Fig. 2 with that of super-systolic array for convolution shown in Fig. 6, we implement each of them on XCV200 with approximate Gate count 220,000, 2352 SLICEs[10]. The inputs, $u(i)$ and $w(i)$,are set to 16-bit each in this implementation. Fig. 7 and Fig.8 show implementations for each design and their implementation reports are listed in Table 1.

From the results, we can see that design using super-systolic array utilizes chip resource more efficiently and shows even higher performance than design using systolic array.

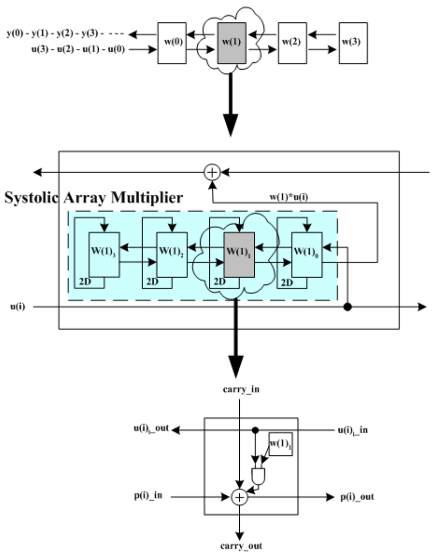
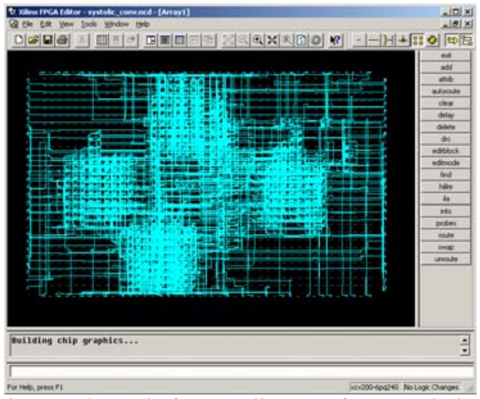


Fig. 6. Super-systolic array for convolution



```
Building chip graphics...
```

Fig. 7. Schematic for systolic array for convolution

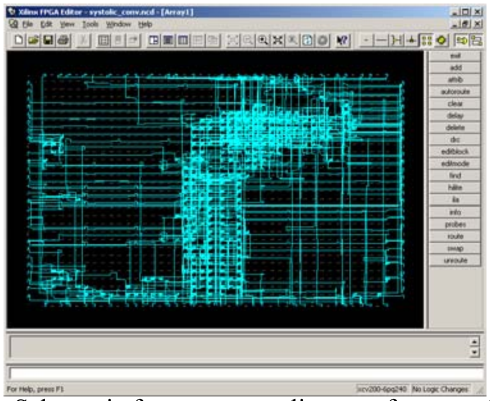


Fig. 8. Schematic for super-systolic array for convolution

TABLE 1.
Implementation reports

Implementation	SLICE	Average connection delay (ns)	Average connection delay on the 10 worst nets (ns)
Systolic Array	632	2.060	5.494
Super-Systolic Array	384	1.597	4.200

III. Simulation, Synthesis, and Implementation

Each of the systolic array multiplier and super-systolic array for convolution was modeled and simulated in RT level using VHDL[11], and synthesized to a schematic using Synopsys design compiler[12-13].

Simulation result using Synopsys VHDL simulation for 4-bit systolic multiplier is shown in Fig. 9.

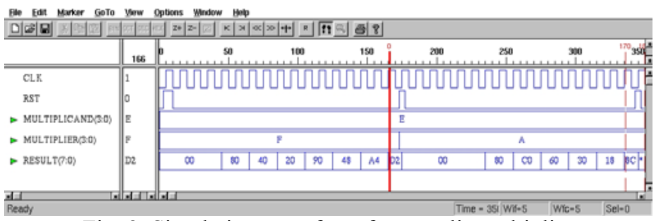

File Edit Marker GoTo View Options Window Help
166
CLK 1
RST 0
MULTIPLICAND(3:0) E
MULTIPLIER(3:0) F
RESULT(7:0) D2
E
F
A
00 80 40 20 90 48 A4 D2 00 80 C0 60 30 18 8C
0 50 100 150 200 250 300 350
Ready
Time = 351 Wlf=5 Wfc=5 Sel=0

Fig. 9. Simulation waveform for systolic multiplier

For two data streams $w(i)$'s -- 1, 8, C, and D -- and $u(i)$'s -- 2, 9, B, and F -- in hexadecimal each, the simulation result, i.e., the result of convolution, is shown in Fig. 10. Note that the successive output values 2, 19, 6B, ED, 171, 143, and C3 in hexadecimal occur periodically.

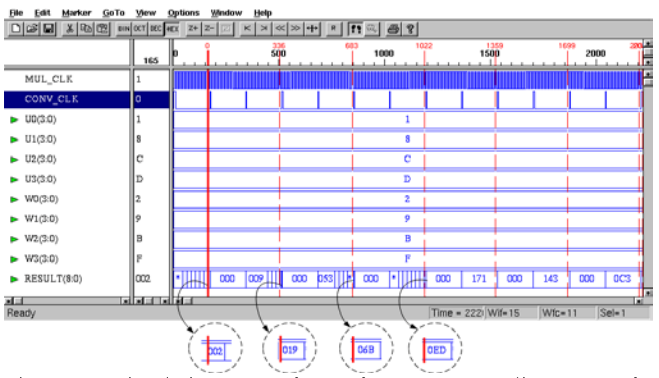


Fig. 10. Simulation waveform for super-systolic array for convolution

A synthesized schematic using Synopsys design compiler for super-systolic array for convolution is shown in Fig. 11 and internal structure of its cell, i.e., the super-cell is shown in Fig. 12. Note that super-cell of super-systolic array for convolution contains another systolic array, that is, systolic multiplier in this design. A schematic for systolic multiplier and internal structure of its cell is shown in Fig. 13 and Fig. 14.

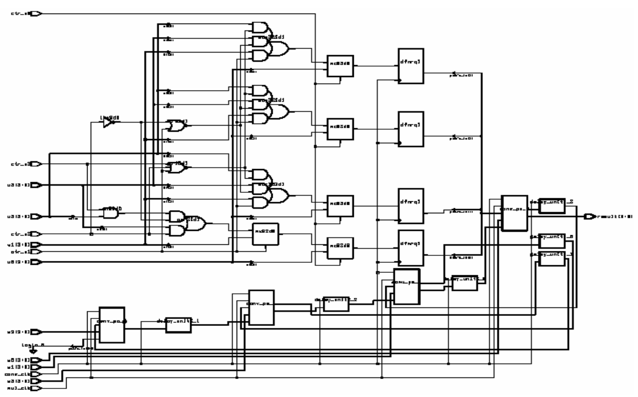


Fig. 11.Synthesized schematic for super-systolic convolution array

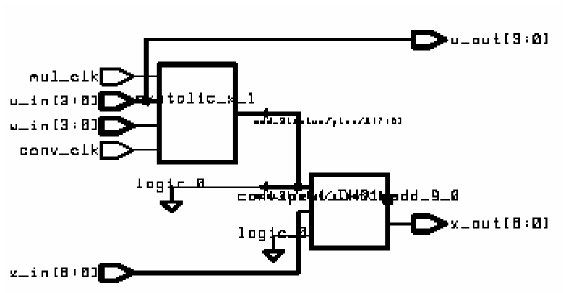


Fig. 12. Synthesized schematic for the cell of super-systolic convolution array

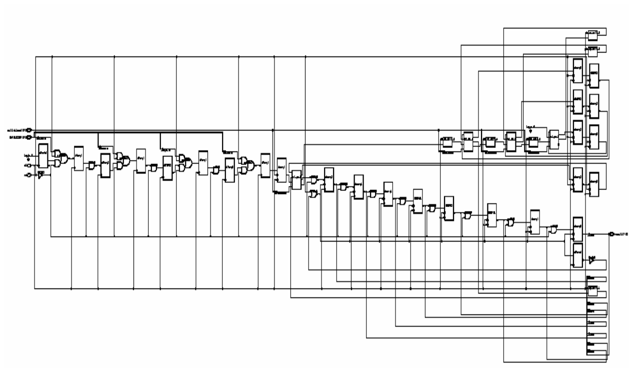


Fig. 13. Synthesized schematic for systolic multiplier

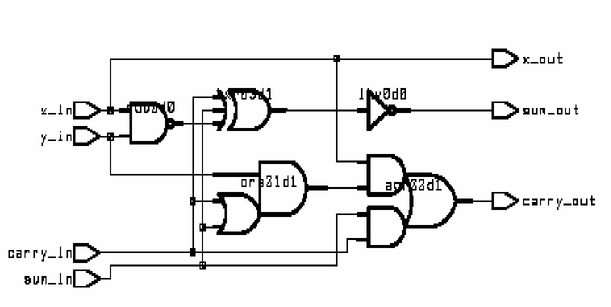


Fig. 14. Synthesized schematic for the cell of systolic multiplier

After being synthesized by Synopsys design compiler, super-systolic array for convolution was automatically implemented using Apollo tool provided by AVANT!. Used cell library is based on 0.35μm 1-poly 4-metal CMOS technology. The layout is shown in Fig. 15.

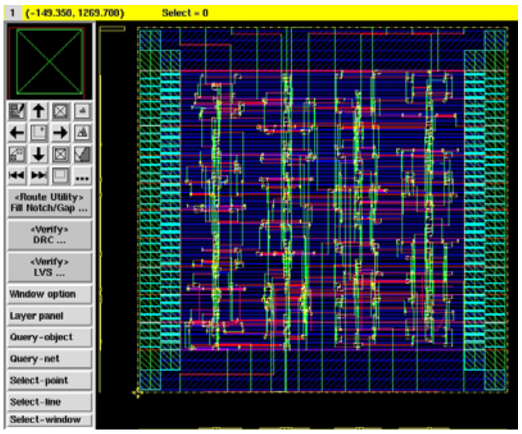


Fig. 15. Layout for super-systolic array for convolution

IV. Conclusions

This paper demonstrates a super-systolic array performing convolution. Making cells of systolic array themselves systolic array yields high-performance, bringing about high degree of concurrency. We refer to this architecture of systolic array consisting of another systolic array in a hierarchical manner as a super-systolic array.

High-performance real-time signal processing calls for the enhancement of concurrent computational capability. Systolic array offers a promising solution to this computational need and presages a technological breakthrough in signal/image processing applications. Super-systolic array approach has value in handling signal/image processing applications in which data rates are usually very high and the computational requirements are extremely demanding because fundamental DSP operations such as convolution/correlation, FFTs, FIR or IIR filters can be implemented using systolic array.

Designs based on super-systolic array architecture are simple, modular, expandable, and yield high-performance. Research on this architecture is particularly worthwhile in view of the fact that VLSI makes the implementation of systolic array chip feasible.

References

[1] S.Y.Kung, *VLSI Array Processors*, Prentice Hall, 1988.

[2] H.T.Kung, “Why Systolic Architectures?,” *Computer* Vol.15, No.1, pp.37-46, January 1982.

[3] H.T.Kung and C.E. Leiserson, "Systolic Arrays (for VLSI)," *Sparse Matrix Proc.*, Academic Press, Orland, fla., pp.256-282, 1979.

[4] S.Y.Kung, ”On Supercomputing with Systolic/Wavefront Array processors,” *IEEE*, vol72, no.7, July 1984.

[5] K.T.Johnson, A.R.Hurson, “General Purpose Systolic Arrays,*” IEEE Computers*, pp.20-31, Novemver 1993.

[6] H.I.Saleh, A.H.Khalil, M.A.Ashour, A.E.Salama, “Novel serial-parallel multipliers,” *IEEE Proc.*, vol.148, pp.183-189, August 2001.

[7] S.Sunder, F.El-Guibaly, A.Antoniou, “Area-efficient multipliers for digital signal processing applications,” *IEEE Trans. Circuit Syst.-II, Analog Digit. Signal Process.*, 43, (2), pp.90-95, 1996.

[8] K.M.Ellethy, M.A.Bayoumi, “Systolic Architecture for modulo multiplication,” *IEEE Trans. Circuit Syst.-II, Analog Digit. Signal Process.*, 42, (11), pp.725-729, 1995.

[9] S.K.Rao, *Regular Iterative algorithms and Their Implementation on Processor Arrays*, Ph.D. thesis, Stanford University, Stanford, California, 1985.

[10] *The Programmable Logic Data Book*. Xilinx, Inc. 1984.

[11] Y.C.Hsu, K.F.Tsai, J.T.Liu and E.S.Lin, *VHDL Modeling for Digital Design Synthesis*, Kluwer Academic Publishers, 1995.

[12] K.C. Chang, *Digital Systems Design with VHDL and Synthesis*, IEEE Computer Society Press, 1999.

[13] Weng Fook Lee, *VHDL Coding and Logic Synthesis with Synopsys*, Academic Press, 2000.