A New Buses Scheme for Fast Inner-Product Computation

R. Lin
Department of Computer Science
SUNY at Geneseo
Geneseo, NY 14454

S. Olariu
Department of Computer Science
Old Dominion University
Norfolk, VA 23529

Abstract In this paper we adopt and modify the shift switching mechanism to propose a novel VLSI inner product processor architecture involving broadcasting on short buses, i.e. buses with no more than 8 switches each. The computation of the inner product of two positive vectors of N items each consisting of m bits takes only $\lceil \log_3(\frac{mN}{7}) \rceil + 1$ broadcasts, plus a carry-save and a carry-propagate additions.

1. Introduction

Recently the authors have proposed a new switching mechanism, that we call shift switching, which allows switches to cyclically permute incoming signals on buses. We have shown that this is a simple and powerful approach to improving the efficiency of a bus system [7-12].

The main purpose of this paper is to propose a novel scheme using "short" broadcasting on buses consisting no more than 8 switches to construct fast inner product processors. The processors not only improve the theoretical result of [9], where broadcasts on buses of N switches are required for an inner product processor of size N, but also are of practical relevance.

The computation of the inner product of two positive vectors of N items each consisting of m bits takes $\lceil \log_3(\frac{mN}{7}) \rceil + 1$ broadcasts, each over 8 switches plus a carry-save addition and a final-step carry-propagate addition. Our estimates show that the processor delay is about $T_1 = 2\frac{\log N + \log m}{\log 3} T_{(3,2)} + T_{pa} = T_1' + T_{pa}$, where $T_{(3,2)}$ is the delay of a 1-bit full adder, T_{pa} is the delay of a carry-propagate adder of less than 2m+logN bits. Compared with the well known inner product processor of Smith and Torng [16], whose delay for the same task amounts to $T_2 = 2(\log N + \log m - 1)T_{(3,2)} + T_{pa} = T_2' + T_{pa}$, the reduction time gain $\frac{T_1'}{T_2'}$ is about $\frac{1}{\log 3}$ (or ranging from 63% to 68%) where the time gain $\frac{T_1}{T_2}$ is about 69% to 73%. In addition, our processor is also competitive in terms of the required VLSI area.

The paper is organized as follows: Section 2 gives a brief review on the shift switches, shift switch counters and compressors. Sections 3, presents the inner product processor architecture, and the partial products reduction scheme; Section 4 estimates time and area of the processor. Section 5 concludes the paper.

The basic concept of shift switching has been introduced in [7-12]. In this section we offer a brief overview of shift switches with a focus on 2-bit shift switches and the corresponding small counters and compressors which are building-blocks for the construction of our inner product processors.

We shall use the notation S(w, d) to stand for a shift switch of w *switching elements*, with the switch states controlled by a group of d input bits. It is also called d-bit shift switch if it can shift a signal up to d bits. A d-bit controlled switch can have up to 2^d switch states. For our purposes, however, we only use d+1 states and also only consider the cases of $2 \le w \le 4$. We summarize a d-bit shift switch as follows (see Figures 1 and 2):

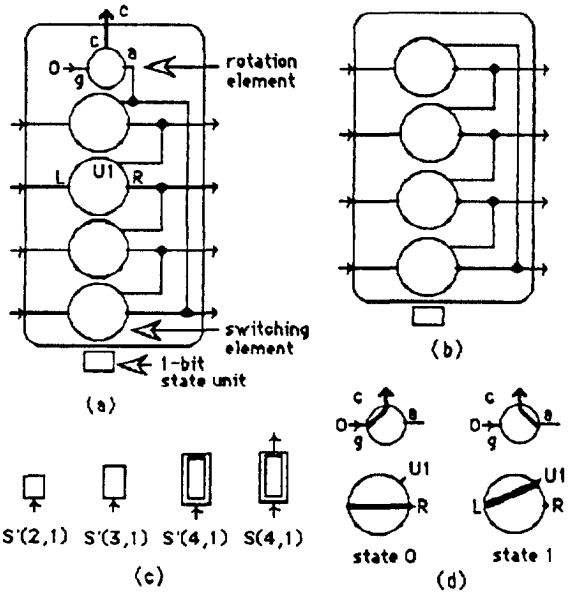

Figure 1. (a) shift switch S(4, 1); (b) shift switch S'(4, 1); (c) the switch graphs; (d) the switch states.

The features of d-bit shift switches (d=1, 2) can be summarized as follows:

(1) Each switching element consists of d+1 *switch cells* (or pass transistors), each controlled by a bit control-signal.

(2) An S(w, d) has d additional switching elements called *rotation elements*. The variant of S(w, d) which has no rotation element in it is denoted by S'(w, d) .

(3) We say that a switch is in state k, if the number of 1s in the group of input bits is k, (k=0, 1 or 2).

(4) Each switching element of S(w, d) has d+2 contacts, L (left), R (right), U1 (up 1 bit), U2 (up 2 bits for d=2). Each rotation element has three contacts, c (rotation bit), a (shifting signal bit), and g (ground).

(5) Each shift switch has a logic unit called (switch) state

The work was supported, in part, by National Science Foundation under grants MIP-9307664 and CCR-9407180.

unit which converts d input bits into the control signals. Under the controlof these signals the following switch cell contacts are connected: When a switch is in state:
0, L and R in each switching element; c and g in each rotation element;
1, L and U1, in each switching element; a and c in each rotation element, if the switch is S(w, 1); a and c in each left rotation element and g, c in each right rotation element, if the switch is S(w, 2);
2, L and U2, in each switching element, and a and c in both the right and the left rotation elements.

It is easy to confirm that being in state k, a shift switch is capable of shifting the incoming signal by k bits (note in this paper shifting also means rotating).

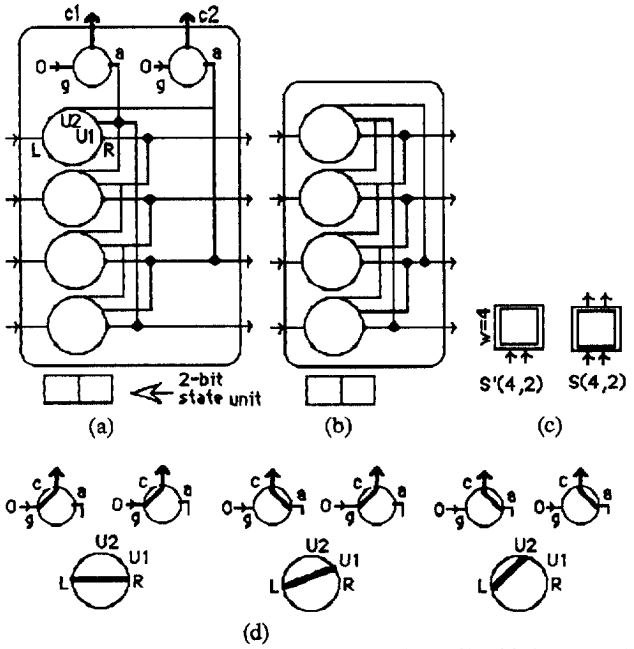

Figure 2. (a, b, c) shift switch S(4, 2), S'(4, 1) and their graphs; (d) their three states.

In this paper two types of shift switching devices: $(k, 3)_4$ $(4\le k\le 7)$ counters and $(k, 2, \lfloor k/4 \rfloor)_4$ $(8\le k\le 15)$ compressors are considered for counting and compressing the number of 1s of an input bit sequence respectively. Note that imposing the restriction of bus width of 4 is only for the simplicity of analysis, the optimal bus width (4, 8, or 16 etc.) should be determined based on hardware parameters.

Figure 3 shows (3, 2), (2, 2) and $(k, 3)_4$ $(4\le k\le 7)$ counters and their (Dadda style) graphic representations [3]. A $(k, 3)_4$ counter is built on a bus of 4 bits consisting of an S'(2, 1) switch, an S'(3, 1) switch and k-2 S(4, 1) switches. The k input bits are converted into signals to set up the shift switches. One end of the bus receives a broadcasting signal called standard 4-bit shifting signal, 0001. The other end of the bus produces the least significant two bits to the count (the output of the encoder). An OR gate is used (for $5\le k\le 7$) to collect rotation bits (note that among the four rotation bits there is at most one 1), producing the most significant bit of the count. Each counter $(k, 3)_4$ can be represented by a graph of two parts separated by a horizontal line. The upper part consists of a number of dots in a column representing the number of input bits of the counter. The lower part consists of three output dots in three columns (along a slanted line), representing a 3-bit binary number. The output slanted line consists of two line segments. Cross-bars may be added on the segments to indicate the decrements of the number of the input dots. The output slanted line with no cross-bar represent $(k, 3)_4$. for k=7, while the line segment with a cross-bar on the first (second, or each) segment represents $(k, 3)_4$ for k=6 (5, or 4). By a compressor $(k, 2, \lfloor k/4 \rfloor)_4$ we mean a digital filter with an input of k bits and an output of a 2-bit number r plus $\lfloor k/4 \rfloor$ bits where the the sum of the input bits should have r as its least significant bits and should have the sum of the $\lfloor k/4 \rfloor$ bits as its most significant bits. Figure 4 shows $(k, 2, \lfloor k/4 \rfloor)_4$ compressors and/or their graphic representations for $(8\le k\le 15)$. They are built on a 4-bit bus consisting of an S'(2, 1), an S'(4, 2), $\lfloor (k-3)/4 \rfloor$ S(4, 2) switches, and MOD(k-1, 2) of S(4, 1) switches. The k input bits are converted into signals (gates) to set up the shift switches. One end of the bus receives a 4-bit shifting signal, 0001. The other end of the bus provides the least significant output bits of the count by a 4-to-2 encoder. Several 4-input OR gates are used to collect rotation bits, producing $\lfloor k/4 \rfloor$ output bits, whose sum (not computed here) provides the most significant two bits of the count (so a compressor does not count the input bits completely). Similarly, each compressor is represented by a graph of two parts separated by a bar. However, five (or four) dots distributed in three columns in the lower part are connected in a tree shape. Cross-bars may be added on the tree's second-level branches to indicate the decrements of its input dots. A tree with three second-level branches and without a cross-bar represents the compressor $(k, 2, 3)_4$ for k=15. While a tree with two second-level branches and without a cross-bar represents the compressor $(k, 2, 3)_4$ for k=11. A cross-bar on the first, (second, or third, in top-down

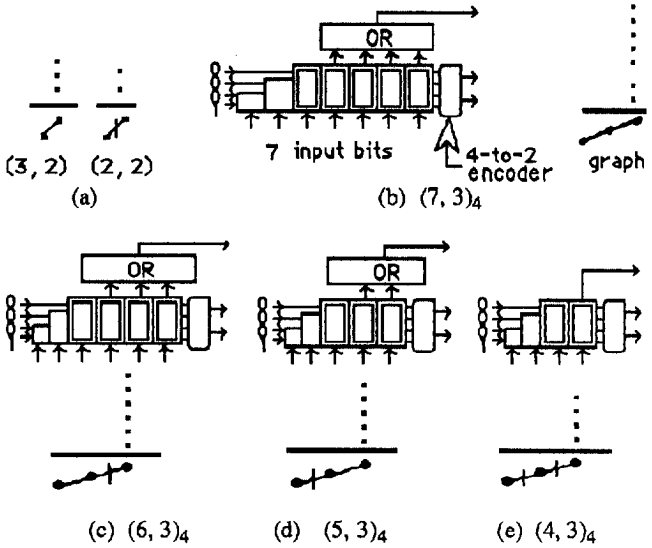

Figure 3. (a) Dadda's representation of counters (3, 2) and (2, 2); **(b, c, d and e)** counter $(k, 3)_4$ for k=7, 6, 5 and 4 and the graphs.

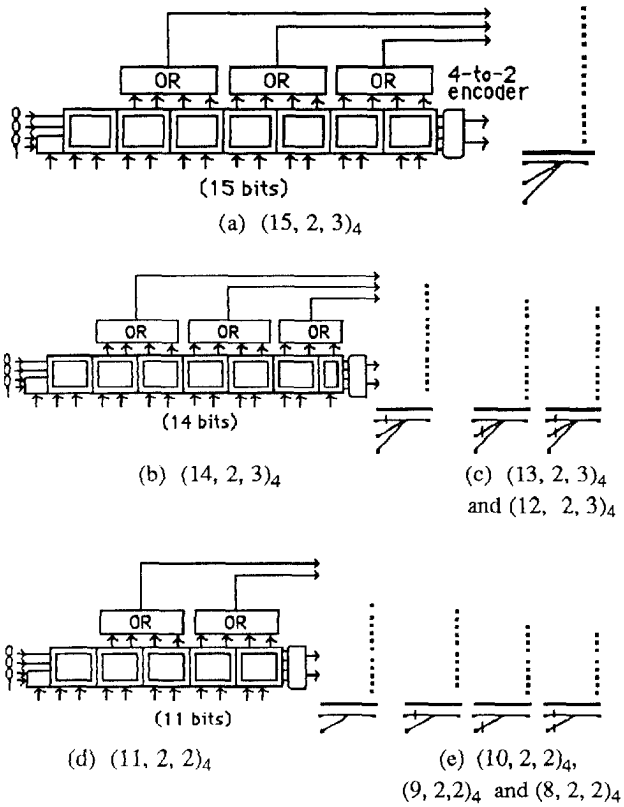

Figure 4. Compressors and/or their graphs for $(k, 2, 3)_4$ (k= 15, 14, 13, 12) and $(k, 2, 2)_4$ (k=11, 10, 9, 8).

order) branch indicates the decrementing k by 1 (2 or 4).

The correctness of the counting (or compressing) on the above devices can be justified by the following simple observation: Let BSUM denote the sum of the input bits of the counter (or compressor). The remainder of BSUM over 4 can be represented by the output of the encoder (the two dots of the first two columns on the right), and the quotient of BSUM over 4 can be represented by the output from the top of the counter or compressor (the dots of the third column).

3. The inner product processor architecture

In this section we propose a novel inner product processor architecture, obtained by applying a reduction scheme to reduce the initial partial product matrices to two numbers.

For simplicity let the input consist of two positive integer vectors: $A=(a_{N-1} .. a_i .. a_0)$, $B=(b_{N-1} .. b_i .. b_0)$. The desired inner product is written as $A \cdot B = \sum_{i=0}^{N-1} a_i \cdot b_i$, where a_i and b_i are m bit positive integers.

We line up the corresponding columns of N partial product matrices to make one matrix, which we call the inner product matrix, with 2m-1 columns and Nm rows as show in Figure 5.

We use shift switching counter $(k, 3)_4$ $(4 \le k \le 7)$ and compressor $(k, 2, \lfloor k/4 \rfloor)_4$ $(8 \le k \le 15)$ plus necessary (3, 2) and (2, 2) counters to achieve the fast reduction (see Figure 5). Similar to Dadda's reduction scheme [3], our reduction

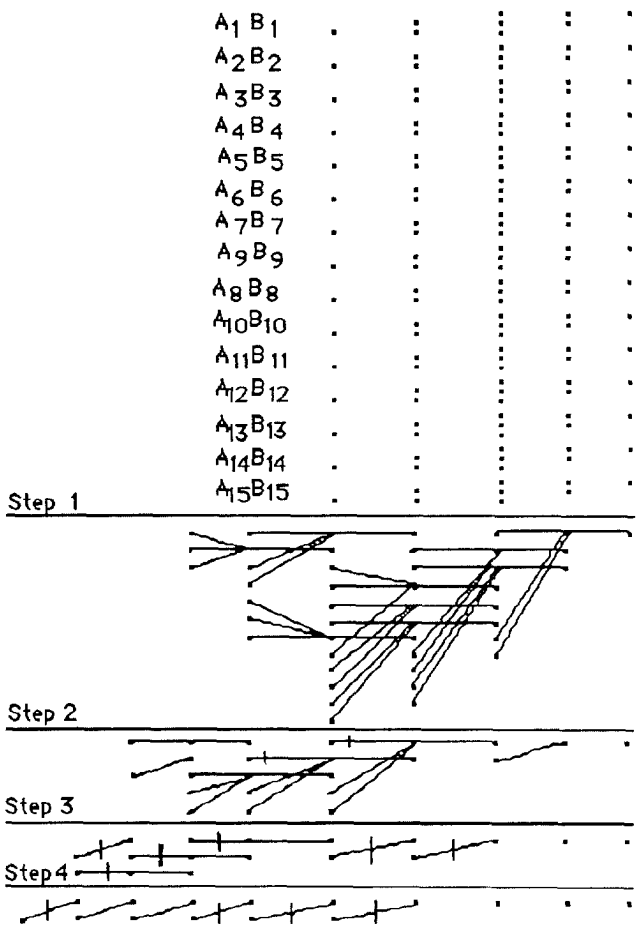

Figure 5. A reduction network for inner product of A·B with N=15 and m=3.

produces a sequence of intermediate matrix heights that minimizes the number of steps taken. The sequence is determined by the following rules: (1) Working back from the final matrix of two rows, limits the height of the second final matrix to 3; (2) for remaining matrices, limit the height H_i of the i-th matrix to $15\lfloor H_{i+1}/5 \rfloor + t(r)$, for $r=H_{i+1}$ MOD 5, here t(r)=1 if r=1, t(r)=3 if r=2, t(r)=7 if r=3, t(r)=11 if r=4, and t(r)=0 if r=0.

Table 1: Number of reduction stages

H (=Nm)	number of stages
3	1
7	2
18	3
52	4
153	5
457	6
1368	7
4102	8
12303	9
36907	10

Table 1 lists the minimum number of steps needed for the reduction of an inner product matrix with the input vector size N and the array element word size m. It is easy to verify that if it takes i steps to reduce the initial inner product matrix of H_i rows into two rows, then it takes i+1 steps to reduce an initial inner product matrix of H_{i+1} rows into two rows, with $H_{i+1} = 15\lfloor H_i/5 \rfloor + 3 + (i \text{ MOD } 2)*4$ for $i \geq 1$ and $H_1=3$.

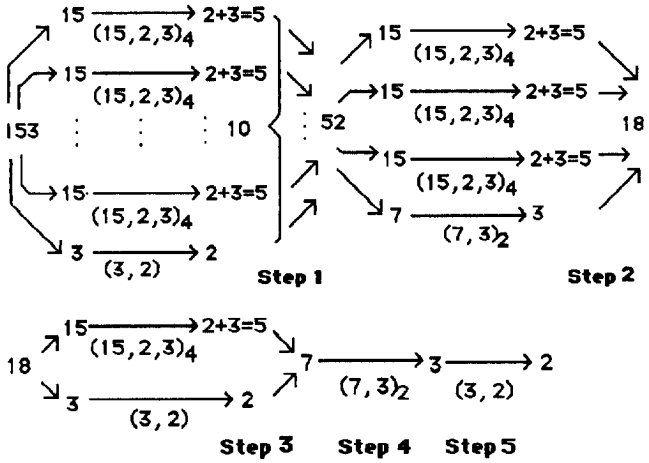

Figure 6. The reduction of (the matrix heights of) 153 bits into 2 bits.

Based on the above scheme we see that: we have a reduction rate of about 3 (i.e. the height of the inner product matrix is reduced by a factor of 3 in each step) for all steps except the last two steps (which have reduction rates of 3/2 and 7/3 respectively). Thus the number of reduction steps required is about $\log_3(\frac{mN}{7}) + 2$. Figure 6 illustrates the five step reduction process for H=Nm=153.

Typically, consider the case of m=31, N=1023. Since H=Nm=31713, ten steps are required to reduce the inner product matrix into 2 numbers. In fact, It consists of eight broadcasts (each over 8 switches) involving $(15, 2, 3)_4$ compressors, one broadcast involving $(7, 3)_4$ counters, and one reduction using (3, 2) and (2, 2) counters.

4. Time and area estimates

To implement counters $(k, 3)_4$ $(4 \leq k \leq 7)$ and compressors $(k, 2, \lfloor k/4 \rfloor)_4$ $(8 \leq k \leq 15)$, three types of bus broadcast technique, active, passive, and pre-charged [6, 17, 22], could be chosen. In this paper we consider active broadcast, where three-state internal buffers are used to handle bus outputs. We first introduce the following notation:

T_{cell} --- the delay of setting a switch cell (a pass transistor) according to the available state signal;

TB_N --- the delay for a broadcast signal to propagate through N switch cells;

T_{buffer} --- the delay of a three-state internal buffer;

T_{gate} --- the delay of a normal gate (i.e. invertor, 2-input and 3-input NAND or NOR gate);

T_{cgate} --- the delay of a complex gate;

T_{egate} --- the delay of an exclusive-OR gate;

$T_{(3, 2)}$ --- the delay of a 1-bit full adder, or the delay of (3, 2) counter.

According to [1, 2 13, 22] we may assume that $T_{cell} = 0.5 T_{gate}$, $TB_{16} = 2TB_8 = 4TB_4 = 4T_{gate}$, $T_{buffer} = 2 T_{gate}$, $T_{(3, 2)} = 4T_{gate}$ and $T_{cgate} = 1.5 T_{gate}$, $T_{egate} = 2T_{gate}$. Note that we also assumed that the broadcast delay on a short VLSI bus is linear on the number of switches the bus contains.

The delay of $(7, 3)_4$ counter can be expressed as:

$$T_{(7, 3)_4} = T_{gate} + T_{cell} + TB_8 + T_{buffer} + T_{cgate} = 7(T_{gate}) \leq 2T_{(3, 2)}.$$

The delay of $(15, 2, 3)_4$ compressor can be expressed as:

$$T_{(15, 2, 3)_4} = T_{egate} + T_{cell} + TB_8 + T_{buffer} + T_{cgate} = 8(T_{gate}) \leq 2T_{(3, 2)}.$$

The carry propagation adder is assumed to be implemented as a carry look-ahead adder (CLA) whose delay is denoted by T_{pa}.

The number of reduction steps required in the process is

$$\log_3(\frac{mN}{7}) + 2 = \frac{\log N + \log m - \log 7}{\log 3} + 2$$

Each step has a broadcast delay of $T_{(15, 2, 3)_4}$ except the last step of addition delay $T_{(3, 2)}$ and last second step of broadcast delay $T_{(7, 3)_4}$. Thus the total processor delay is

$$T_1 = \frac{\log N + \log m - \log 7}{\log 3} 2T_{(3, 2)} + 4T_{(3, 2)} + T_{pa}$$

$$\doteq 2\frac{\log N + \log m}{\log 3} T_{(3, 2)} + T_{pa} = T_1' + T_{pa}, \text{ where } T_1'$$

representing the reduction delay.

We compare the estimated delays of our inner product processor with the delays of Smith and Torng's processor. The total delay of the latter is $T_2 = 2(\log N + \log m - 1)T_{(3, 2)} + T_{pa} = T_2' + T_{pa}$ [16]. Thus our reduction time gain $\frac{T_1'}{T_2'}$ is about $\frac{1}{\log 3}$, or more precisely, is in between 0.63 to 0.68. In general, T_{pa} is relatively small compared with the total delay, so we may assume $T_{pa} = 0.2T_2'$. Then, the time gain $\frac{T_1}{T_2}$ is ranging 69 to 73%.

Consider again the example of N=1023, m=32. The reduction delay of our processor is:

$$8*T_{(15, 2, 3)_4} + T_{(7, 3)_4} + T_{(3, 2)} = 75\ T_{gate}.$$

The reduction delay of Smith and Torng's processor is $2(10+5-1)*4\ T_{gate} = 112T_{gate}$.

Thus our processor has a reduction time gain about 67%.

The relative area of our processor is roughly estimated as follows:

The number of input dots in our reduction network is estimated as: $N*m^2$ in the initial inner product matrix (ref. to as dots in step 1), and about $N*m^2/3$ in step 2, and about $N*m^2/3^2$ in step 3, and so on, except the last two steps where the dot-reduction rates are about 7/3 and 3/2 respectively. The number of reduction steps is about $\log_3(\frac{mN}{7}) + 2$. It is easy to see that the total number of dots of the reduction network (excluding the output of the last step, i.e. the two row matrix) is between $1.5N*m^2$ and $3N*m^2$.

Notice that each dot (being an input bit of a shift switch) in the network requires about a half amount of hardware for a S(4, 2) shift switch in general, and an S(4, 2) is composed of an exclusive-OR gate, two complex gates and 16 switch cells. For simplicity, the area of the encoders, the OR gates on the top of buses is considered to be the same as area saved by the use of switches of S(4, 1), S'(4, 2), S'(4, 1) and S'(2, 1) in each bus. Based on the data from LSI book [13], we estimate the area of an S(4, 2) equal to the area of 8 gates or the area of a 1-bit full adder (A_{csa} for short). We now have the area of our processor is in between $0.75N*m^2 A_{csa} + A_{fa}$ and $1.5N*m^2 A_{csa} + A_{fa}$ (A_{fa} is the area of the carry propagate adder). While the area of Smith and Torng's processor has an area of $2(Nm(m+2) + (2m-1)\log N + m(\log m - 3))A_{csa} + A_{fa}$ [16], which is about $2N*m^2 A_{csa} + A_{fa}$ or the same as that of ours.

5. Concluding remarks

In this paper we have adopted the shift switching mechanism to obtain a novel VLSI inner product processor architecture involving broadcasting over short buses, i.e. buses with no more than 8 switches. The computation of the inner product of two positive vectors of N items, each consisting of m bits, takes $\log_3(\frac{mN}{7}) + 1$ broadcasts, plus a carry-save addition and a carry-propagate addition. Recent VLSI implementations have demonstrated that the broadcast delay incurred in traversing a switch is quite small though not zero. For example, the switch delay obtained on the YUPPIE chip is about $\frac{1}{16}$ns to 1ns (see [6] pp. 238). These experiments confirm the feasibility and potential benefits of the architectures. Based on our estimate the processor delay is about $T_1 = 2\frac{\log N + \log m}{\log 3} T_{(3,2)} + T_{pa} = T_1' + T_{pa}$ (here $T_{(3,2)}$ is the delay of a 1-bit full adder, T_{pa} is the delay of a carry propagate adder of less than 2m+logN bits). Compared with the well known inner product processor of Smith and Torng [16], whose running time for the same task amounts to $T_2 = 2(\log N + \log m - 1)T_{(3,2)} + T_{pa} = T_2' + T_{pa}$. Our process reduction (reducing input bits into two numbers) time gain $\frac{T_1'}{T_2'}$ is about $\frac{1}{\log 3}$ (or 66%) and the total time gain $\frac{T_1}{T_2}$ is ranging 70%. In additional to the time performance, our processor is also competitive in terms of the required VLSI area (about the same as that of Smith and Torng's process).

REFERENCES

[1] K. Bickerstaff, M. Schulte, and E. Swartzlander, Reduced area multipliers, in *Proc. of International Conference on Application-Specific Array Processors (ASAP)*, Venice, Italy, Oct 1993. 478-489.

[2] G. Bilardi, M. Pracchi, and F. P. Preparata, A Critique of network speed in VLSI models of computation, IEEE Journal of Solid-State Circuit, Vol. SC-17, NO. 4, august 1982.

[3] L. Dadda, On parallel digital multipliers, Alta Freq. 45, 1976, 574-580.

[4] J. Jang, H. Park and V. K. Prasanna, An optimal multiplication algorithm on reconfigurable mesh, *Proc. of IEEE Symp. on Parallel and Distributed Processing* 1992.

[5] H. T. Kung and C. E. Leiserson, Algorithms for VLSI Processor Arrays, *Introduction to VLSI Systems*,C. Mead and L. Conway, Reading MA: Addison-Wesley, 1980.

[6] H. Li and M Maresca Polymorphic-torus architecture for computer vision, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 11, no. 3, 1989, 233-243.

[7] R. Lin , Reconfigurable buses with shift switching -- VLSI radix sort, in *Proc. of International Conference on Parallel Processing (ICPP-92)*, St. Charles, IL, Aug. 1992, Vol III, 2-9.

[8] R. Lin, A novel parallel counter scheme, submitted.

[9] R. Lin and S. Olariu, Computing the inner product on reconfigurable buses with shift switching, in *Proc. of Joint Conference on Vector and Parallel Processing (CONPAR 92)*,Lyon, France, Sep. 1992, 181-192.

[10] R. Lin and S. Olariu, Reconfigurable buses with shift switching --Concept and applications to appear in *IEEE Trans. on Parallel And Distributed System*. The preliminary version of the paper was published *inProc. of International Phoenix Conference on Computers and Communications (IPCCC-93)*, 23-29.

[11] R. Lin and S. Olariu, A practical constant time sorting network, in *Proc. of International Conference on Application-Specific Array Processors (ASAP)*, Venice, Italy, Oct 1993.

[12] R. Lin and S. Olariu, Short reconfigurable buese for computer arithmetic, Proc. of the Workshop on Reconfigurable Architectures, the 8th International Parallel Processing Symposium, Cancun, Maxico, 1994.

[13] *LSI logic 1.0 micron cell-based products data book, LSI logic corporation*, Milpitas, Califoria, 1991.

[14] M. Mehta, V. Parmar and E. Swartzlander, Jr., High-speed multiplier design using multi-input counter and compressor circuits, in *Proc. of IEEE 10th Symposium on Computer Arithmetic*, Grenoble, France, 1991.

[15] R. Miller, V. K. Prasanna Kumar, D. Reisis and Q.F. Stout, Parallel computations on reconfigurable meshes, *IEEE Trans. Computers*, 42(6) , June 1993, pp. 678-692.

[16] S. P. Smith and H. C. Torng, Design of a Fast Inner Product Processor, in *Proc. of IEEE 7th Symposium on Computer Arithmetic*, 1985.

[17] D. B. Shu and J. G. Nash, The gated interconnection network for dynamic programming, S. K. Tewsburg et al. (ed), *Concurrent Computations*, Plenum Publishing Corp., 1988.

[18] E. E. Swartzlander, Jr., *Computer Arithmetic Vol. 1*, 2 (IEEE CSP, CA, 1990).

[19] E. E. Swartzlander, Jr., Parallel Counters, *IEEE Trans. Computers*, C-22 , November 1973, pp. 1021-1024.

[20] E. E. Swartzlander, Jr., Barry K. Gilbert, and Irving S. Reed, Inner Product Computers, *IEEE Trans. Computers*, C-27, 1, Jan. 1978, pp. 21-31.

[21] J. D. Ullman, *Computational Aspect of VLSI*, (Computer Science Press, MD, 1983).

[22] J. F. Wakerly, Digital Design: Principles and practices, Prentice-Hall, Englewood Cliffs, New Jersey, 1990.