# An Efficient VLSI Architecture Parallel Prefix Counting With Domino Logic*

Rong Lin† Koji Nakano‡ Stephan Olariu§ Albert Y. Zomaya¶

## Abstract

*We propose an efficient reconfigurable parallel prefix counting network based on the recently-proposed technique of shift switching with domino logic, where the charge/discharge signals propagate along the switch chain producing semaphores results in a network that is fast and highly hardware-compact. The proposed architecture for prefix counting $N - 1$ bits features a total delay of $(4 \log N + \sqrt{N} - 2) * T_d$, where $T_d$ is the delay for charging or discharging a row of two prefix sum units of eight shift switches. Simulation results reveal that $T_d$ does not exceed 1ns under 0.8-micron CMOS technology. Our design is faster than any design known to us for $N \leq 2^{10}$. Yet another important and novel feature of the proposed architecture is that it requires very simple controls, partially driven by semaphores, reducing significantly the hardware complexity and fully utilizing the inherent speed of the process.*

**Index Terms:** *Special-purpose parallel architectures, digital signal processing, VLSI design, domino logic, computer arithmetic.*

## 1 Introduction

Due to the high degree of miniaturization possible today in VLSI technology, the size and complexity of designs that can be implemented in hardware has increased dramatically. This has made it technologically feasible and economically viable to develop high-speed applications-specific architectures featuring a spectacular performance increase over their general-purpose counterparts [1]. The main goal of this work is to present a special purpose architecture for fast and hardware-compact computation of a parallel binary prefix counter. Our design is based on the recently-proposed technique of shift switching with domino logic [6, 8].

Reconfigurable bus systems enhanced with shift switches have been recently proposed to solve a number of fundamental computational problems [4, 5, 6, 7, 8]. Such systems have the following features: (1) When special digital signals called state signals are propagating through a shift switch array, modulo operations can be performed directly, simplifying the traditional approach to basic arithmetic computation; and (2) During the process the state signals are inverted, alternatively, in two mutually inverted forms (n and p), minimizing the loads of transistors and maximizing the speeds of circuits.

By applying precharged CMOS domino logic techniques to pass-transistor-based shift switched buses we can obtain a domino charge/discharge chain featuring the interesting property that the charge/discharge signals can propagate along the chain and always produce a semaphore to indicate the end of the process [6]. This makes it possible to construct a network of processing elements to compute parallel binary prefix sums fast and in a highly hardware-compact fashion.

The problem of parallel binary prefix counting is fundamental in parallel processing, for its solution is the principle ingredient in arithmetic expression evaluation, storage and data compaction, processor assignment, and routing, among many others [10]. We first propose a parallel prefix counting network of I/O size $N - 1$ (that is, with $N - 1$ input bits, for $N = 2^{2k} = n * n$). This involves a two-level construction containing a total of $N + \sqrt{N}$ cascaded basic shift switches, with $N$ pass-transistor-based shift switch plus $\sqrt{N}$ trans-gate-based shift switches. Each pass-transistor-based switch is associated with a simple processing element (PE, for short): for each row of switch units there is a row processing element referred to as PE$_r$. In fact, both the PE's and the PE$_r$'s, are simple control units, receiving corresponding semaphores, and either sending control signals to tri-state drivers, or sending select signal to the MUX, or sending precharge/evaluation enable signals to start a precharge/evaluation process. Thus, the entire network can be perceived as an application-specific circuit.

We then go on to modify this architecture by replacing the PE's and the PE$_r$'s by a few simple combinational and sequential circuits, consisting of two registers and of two simple switches synchronized by the clock and the semaphore

*Work supported by NSF grants CCR-9522093 and MIP-9630870, by ONR grant N00014-97-1-0526, and by ARC grant 04-15-412-194.

†Department of Computer Science, SUNY at Geneseo, Geneseo, NY 14454, USA

‡Department of Electrical and Computer Engineering, Nagoya Institute of Technology, Showa-ku, Nagoya 466, JAPAN

§Department of Computer Science, Old Dominion University, Norfolk, VA 23529-0162, USA

¶Department of Electrical and Electronic Engineering, University of Western Australia, Perth, AUSTRALIA

in each node of the prefix counting network. We have simulated and tested the modified architecture. The SPICE circuit simulation (on 0.8-micron CMOS technology at a 5-V supply and 500MHz clock) has shown less than 1ns delay for each of the row precharge and row discharge operations.

Our prefix sum design achieves a total delay of $(4\log N + \sqrt{N} - 2) * T_d$, where $T_d$ is the delay required to charge or discharge a row of two prefix sum units (in our simulations $T_d \leq 1\text{ns}$). Since it is not realistic to compute the prefix sums of a large array of binary numbers, we assume that $N < 2^{14}$. Under this assumption, the proposed architecture is about 50% faster than any architecture known to us, including a tree of adders, or a processor with the same structure as ours but with each shift switch substituted by a half adder (half-adder-based processor, for short). Another important feature of the proposed design is that it has a compact VLSI area, requiring an area about $0.7 * (N + \sqrt{N}) * A_h$ which is almost linear in the input size. Here, $A_h$ is the area equivalent of a half adder. This area is significantly smaller than both the half-adder-based processors (about 50%), and the tree of half adders processors, which require an area of about $(N\log N - 1.5N + 1) * A_h$. The area for the control devices necessary in any such an architecture is omitted.

Yet another important feature of our architecture is that the $N$-prefix sums are computed and output row by row, with a simple control mechanism driven by semaphores produced at the end of the row's domino charging/discharging. This simplifies the hardware requirements, and the full inherent speed of the computation can be utilized.

## 2 Precharged CMOS shift switches and prefix sums units

Figure 1 illustrates a pass-transistor-based schematic of basic shift switch $S < 2, 1 >$. Once the switches are precharged (high) and preset by control $Y$, the discharging signals $X_{(2)}$ and $X_{(\overline{2})}$ from the shift_in port will pull-down the data path to yield, respectively, shift_out $R$ and $Q$.

To improve the efficiency of discharging, we cascade a small number of the n-switches – four, to be more precise – to form a prefix sums unit as illustrated in Figure 2. Here, the discharging process now yields the following (modulo 2) results: $u = (X + a) \bmod 2$, $v = (X + a + b) \bmod 2$, $w = (X + a + b + c) \bmod 2$, $z = (X + a + b + c + d) \bmod 2 = R$, $a' = \lfloor \frac{X+a}{2} \rfloor$, $b' = \lfloor \frac{X+a+b}{2} \rfloor$, $c = \lfloor \frac{X+a+b+c}{2} \rfloor$, $z' = \lfloor \frac{X+a+b+c+d}{2} \rfloor$. The complete process is composed of two phases that we describe next:

**A. The precharge phase.** Referring to Figure 2 that we use as illustration, the following should occur in the precharge phase:

1. E, a tri-state enable signal is set to 0, i.e. the output of each tri-state internal bus driver is in Hi-Z;

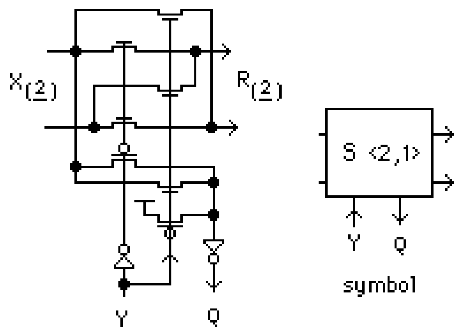


**Figure 1. Illustrating the nMOS pass-transistor-based schematic of the shift switch $S < 2, 1 >$: $(X_{(2)}, Y, R_{(2)}, \overline{Q})$.**

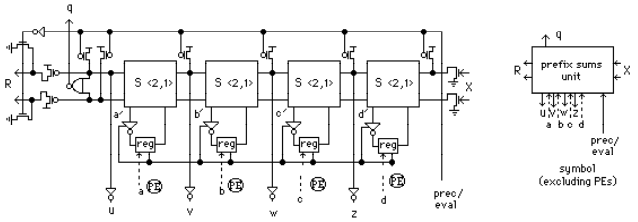


**Figure 2.** *Illustrating the prefix sums unit*

2. The input bit of each PE associated with the switch is loaded into the (state) register. This will preset each switch to the corresponding state;

3. The precharge and evaluation signal prec/eval is set to 0, which starts recharging all switches (outports including s) of the unit in parallel. When the precharge is done, the semaphores $q = 0$ and $R = 11$ are produced. If a row contains more than one switch unit, then the units are cascaded in a chain form.

**B. The evaluation phase.** In this phase the following should occur (s before, refer to Figure 2):

1. Set the precharge and evaluation signal prec/eval to 1;

2. When the discharging state signal $X = 01$ (or $X = 10$) arrives, its bit 0 pulls down the preset data path along the unit to produce for each unit, as described above, the bits $u,\ v,\ w,\ z$ as well as $a',\ b'\ c'\ d'$ and to produce the semaphores $q = 1$ and $R = 01$ (or $R = 10$);

3. If the signal E from $PE_r$ is 1, read the output bits $u,\ v,\ w,\ z$;

4. If the signal E from $PE_r$ is 1, each PE triggers a register-load operation to load the values $a',\ b'\ c'\ d'$ otherwise, there is no loading;

5. The precharge and evaluation signal prec/eval is set back to 0, i.e. start precharge again and keep in precharge phase until

the next evaluation begins. If a row contains more than one switch unit, the discharging process can propagate from one switch unit to another automatically.

Note that the PEs of each switch unit discussed here can be simply driven by the semaphore of the unit, in other words, all operations can be driven by the semaphore after initialization.

## 3 The parallel prefix counting architecture

Refer to Figure 3 for the block diagram of a complete prefix counting network with an input size of $N = 64$. The mesh consists of $n = 8$ rows, each featuring two cascaded switch units, an n-switch and a p-switch, as described in Section 2. To simplify the network control, we use an additional column of trans-gate-based shift switch array on the left part of the mesh.

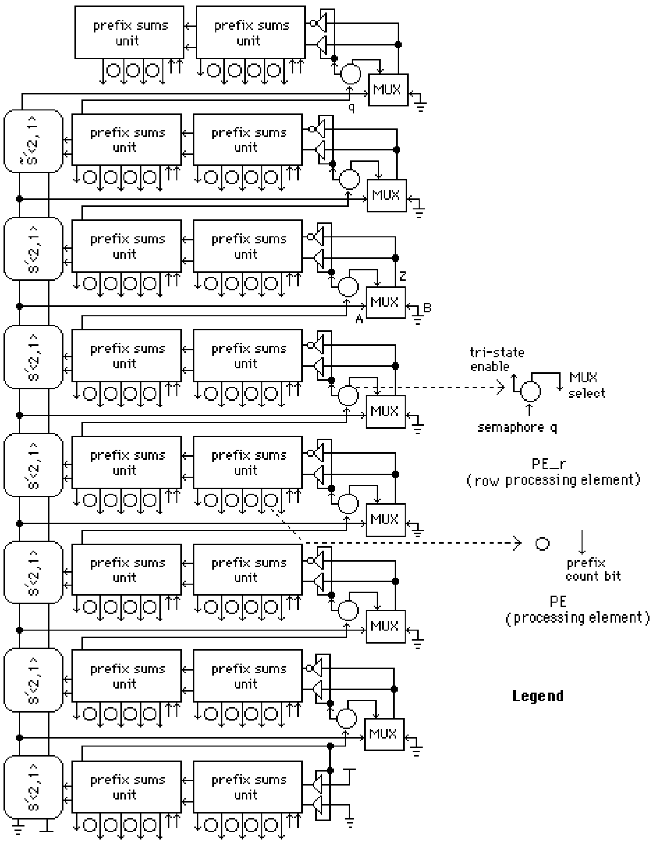


**Figure 3.** *The parallel prefix counting network.*

Let the input of the column switch array be a sequence of (p-type) state signals, $b1, b2, \ldots, b7$, and let the output be a sequence of regular signals, $p1, p2, \ldots, p7$. Clearly, we have $pi = (b1 + b2 + \cdots + bi) \bmod 2$. for all $1 \leq i \leq 7$. Note that this is slower than the precharged switch array and generates no semaphores. However, the computation does not require two phases. The beginning of each row consists of a row processing element $\mathrm{PE}_r$ which receives a semaphore from the previous row and controls a 2-input multiplexer and an input state signal generator consisting of two tri-state buffers. The algorithm is spelled out as follows:

**1. Initial stage.** {Computes and outputs the least significant bits of the prefix counts}

**Step 1.** All PEs load their input bits into their registers;

**Step 2.** Each prefix sum unit starts the initial precharge phase;

**Step 3.** $\mathrm{PE}_r$ sets select signal to 0, such that, the right input (i.e. 0) of each MUX is selected;

**Step 4.** $\mathrm{PE}_r$ sets $Er = 1$ and each row begins domino discharging;

**Step 5.** $\mathrm{PE}_r$ sets $E = 0$ (i.e. no output and register loading, refer to the evaluation phase in Section 2;

**Step 6.** When a semaphore value of 1 is received by the $i$-th $\mathrm{PE}_r$ $i$ times, it sets select signal to 1;

**Step 7.** The $i$-th $\mathrm{PE}_r$ sets $E = 1$ (i.e. to output and load register);

**2. Main stage** Consists of $\log N - 1$ iterations of the following

**Step 8.** $\mathrm{PE}_r$ sets select signal to 0 such that the right input (i.e. 0) of each MUX is selected;

**Step 9.** $\mathrm{PE}_r$ sets $Er = 1$, and each row begins domino discharging;

**Step 10.** $\mathrm{PE}_r$ sets $E = 0$ (i.e. no output and register loading);

**Step 11.** $\mathrm{PE}_r$ sets select signal to 1;

**Step 12.** $\mathrm{PE}_r$ sets $Er = 1$, each row begins domino discharging;

**Step 13.** $\mathrm{PE}_r$ sets $E = 1$ (i.e. output and register loading).

The algorithm can be interpreted as follows: In the initial stage (Steps 1 through 5) each row computes the least significant bit (LSB) of the sum of bits in the row. The results (called the parity-bits of the rows) then are prefix summed by the column switch array, which takes about $i$ steps of semaphore (row) propagation time (waiting time) to get the $i$-th prefix sum initially computed, the $i$-the row starts to compute the LSBs of the global prefix sums for the row (Steps 6 and 7).

When the processing of the initial stage is completed, each row begins executing the main stage. In the stage, the process is similar to that in the initial stage except that there is no waiting for the prefix sum of the parity-bits of the lower rows (the waiting takes place in the initial stage), the data items are always available and the computations are for the remaining bits of the prefix sums. In fact, the column switch array involves a pipelined process to produce the bits of the global prefix counts of the rows.

# 4 Simulations and tests performed

In our test implementations we use a modified prefix sum unit as shown in Figure 4. The PEs are removed, the precharge-discharge and I/O controls are performed correctly by the sequential circuit which consists of two registers and two simple switches synchronized by the clock and the semaphore (i.e. Cin/Cout). It is easy to see that the unit is functionally the same as the one shown in Figure 2. We also modify the overall prefix network architecture correspondingly as shown in Figure 5. The algorithm of the computation is the same as described in Section 3, except that all operations done by the PEs and by the $\mathrm{PE}_r$s are now being performed by simple combinational and sequential logic circuits plus reconfiguration switches. Their brief description is given below.

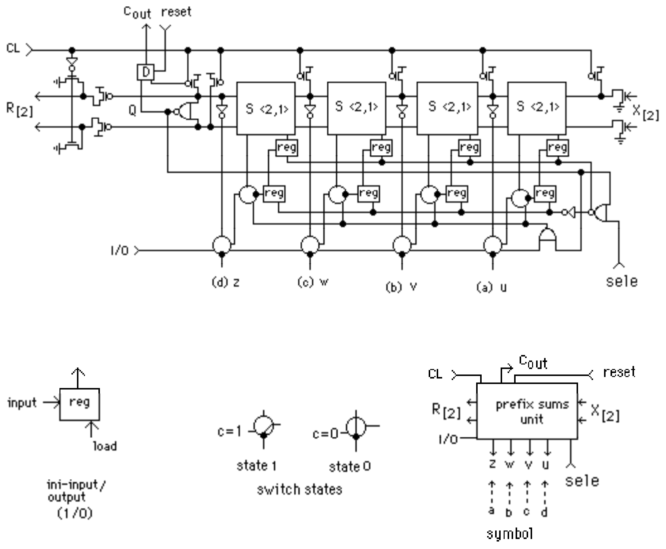


**Figure 4. The modified prefix sums unit.**

We have simulated the prefix sums unit. The SPICE circuit simulation (on 0.8-micron CMOS technology at a 5-V supply and 500MHz clock) has shown less than 1ns delay for each of row precharge and row discharge operations based on the circuit of Figure 4. We refer the reader to the analog trace shown in Figure 6. More simulations and tests are in progress. Based on the simulation results we can conclude that

1. The initial stage takes about $2Td + N\sqrt{N} * T_d$, where $T_d$ is the delay of charge or discharge of a row of two prefix sum units;

2. The main stage takes $(\log N - 1) * 4T_d$ time. Notice that $4T_d$ denotes two domino charge and discharge processes of a row. Note that the register loadings are overlapped with charge and discharge operations in all stages except the initial stage;

3. The total delay can be specified as $(4 \log N + N\sqrt{N} - 2) * T_d$.

It is worth noting that in our simulation, $T_d$ is no more than 1ns, and the total delay of the prefix computation is no more than 28ns. This total delay takes no more than 6 instruction cycles since under the VLSI technology we assumed, an instruction cycle is about 5 to 8 ns. Compared with the software computation of the prefix sums, which requires at least 63 instruction cycles, the speed-up of the proposed processor is significant.

Since the computation of the prefix sums of a large array of binary numbers, say $N \geq 2^{14}$ is not very realistic, we assume that $N < 2^{14}$. Under this assumption, our simulation results show that our processor is at least 50% faster than any processor known to us, including the tree of adders [3], or the processor with the same structure as ours but with each shift switch replaced by a half adder (half-adder-based processor for short).

Since each nMOS transistor-based shift switch is about 70% of a half-adder, the total area can be specified as $0.7 * (N + \sqrt{N}) * A_h$, where $A_h$ is a half-adder area equivalent (note that registers and basic controls devices are not counted because they are necessary in any scheme to accomplish the computation). This area is about 30% smaller than that of both half-adder-based processors, and of the tree of half-adders which require an area of $(N \log N - 1.5N + 1) * A_h$. Note that the half-adder-based processor requires a significantly larger number of control devices because it does not generate semaphores, even if it uses the traditional domino logic technique.

# 5 Concluding remarks

The main contribution of this paper was to propose a parallel prefix counting network of size $N - 1$ (i.e. with N-1 input bits, for $N = 2^{2k} = n * n$. Our design is a two-level architecture involving a total of $N + \sqrt{N}$ cascaded simple shift switches, with $N$ pass-transistor-based shift switches along with trans-gate-based shift switches. The resulting network achieves a delay of $(4 \log N + \sqrt{N} - 2) * T_d$, where where $T_d$ is the delay incurred in charging or discharging a row of two prefix sum units. Our simulation results show that under 0.8-micron CMOS technology $T_d$ is not more than 1ns.

Our architecture is also area-compact and is constructed by using CMOS, domino logic techniques on buses with shift switches. The processing elements require a very simple asynchronous control, being driven by semaphores produced at the end of each rows domino discharging process. This greatly simplifies the hardware requirements, and allows the full inherent speed of the computation to be utilized.

Finally, the application of the proposed binary prefix counter can be easily extended using a pipelined technique for larger binary counter. For example, with the available of a 63-bit prefix counter, for counting up to 127-bit, we may produce the prefix counts for the first set of 63 bits and then process in pipeline the second set of remaining 63 bits. We then send each processor (receiver) two results: The total of the previous set (i.e. the prefix count value of the last bit of the previous set, if there is any, otherwise 0) and the prefix count value of the corresponding bit. The sum of these two values, clearly is the prefix count of the corresponding bit.

## References


[1] K. Bondalapati, and V. K. Prasanna, Reconfigurable Meshes: Theory and Practice, *Proc. Reconfigurable Architecture Workshop, RAW'97*, Orlando, Florida, April 1997.

[2] I. S. Hwang, and A. L.. Fischer, Ultrafast compact 32-bit CMOS adders in multi-output domino logic, *IEEE Journal of Solid-State Circuits*. 24, (1989), 358–369.

[3] R. H. Krambeck, C. M. Lee, and H.S. Law, High-Speed Compact Circuits with CMOS, *IEEE Journal of Solid-State Circuits*, SC-17, No. 3, June, 1982.

[4] R. Lin, Reconfigurable Buses with Shift Switching - VLSI Radix Sort, *Proc. International Conference on Parallel Processing*, St. Charles, Illinois, 1992, Vol III, 2-9.

[5] R. Lin, and S. Olariu, Reconfigurable buses with shift switching –concept and applications, *IEEE Transactions on Parallel and Distributed Systems*, 6, (1995), 93-102.

[6] R. Lin, Shift switching and novel arithmetic schemes, *Proc. 29th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, November 1995.

[7] R. Lin and S. Olariu, Efficient VLSI architecture for Columnsort, *IEEE Trans. on VLSI Design*, 1999.

[8] R. Lin and S. Olariu, Reconfigurable shift switching parallel comparators, *VLSI Design*, in press, 1998.

[9] A. Mukherjee, *Introduction to nMOS & CMOS VLSI System Design*, Prentice-Hall, Englewood, NJ, 1986.

[10] E. E. Swartzlander, Jr., *Computer Arithmetic*, IEEE Press, Vol. 1 and 2, 1990.

[11] N. Weste and K. Eshraghian, *Principles oF CMOS VLSI design, a systems perspective*, Second Edition, Addison-Wesley, 1993.


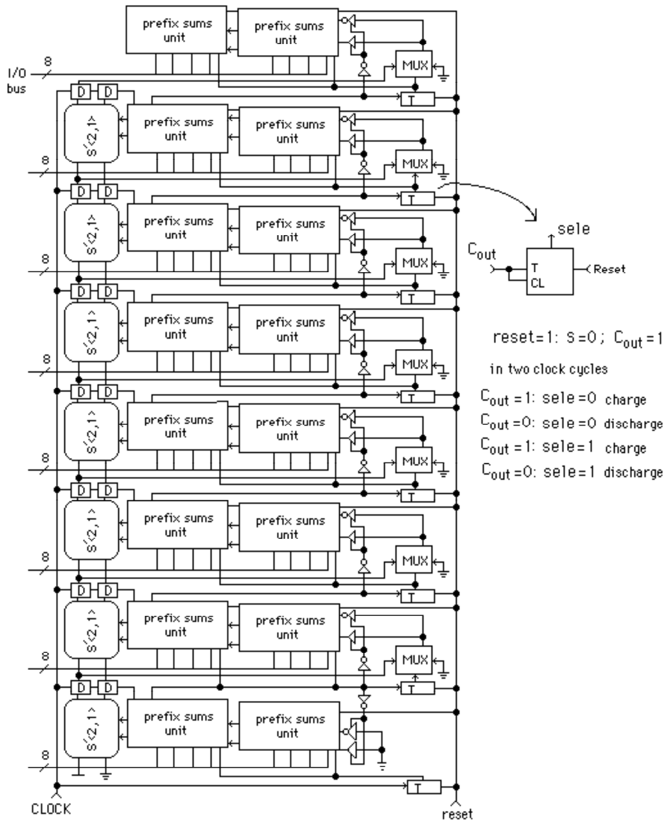


**Figure 5. The modified parallel prefix counting network.**

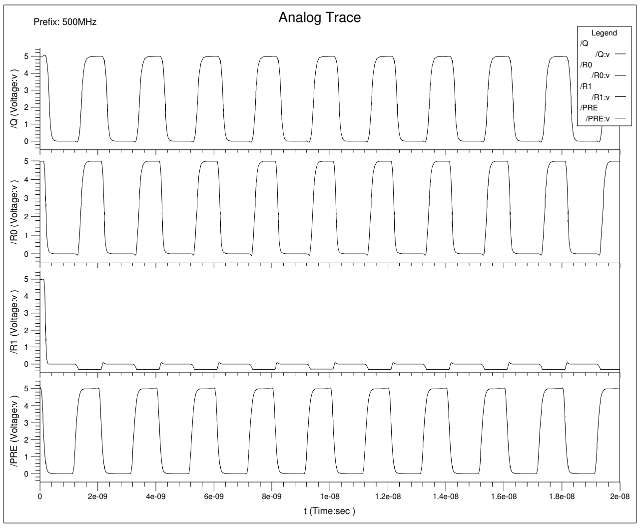


**Figure 6. The analog trace of the prefix sum circuit in Figure 5.**