



Simulation of Priority-Concurrent Write Buses by Separable Buses with Fewer Switches

Maeba, Takashi

(Citation)

神戸大学海事科学部紀要, 3:63-69

(Issue Date)

2006-07-31

(Resource Type)

departmental bulletin paper

(Version)

Version of Record

(JaLCD0I)

<https://doi.org/10.24546/00517757>

(URL)

<https://hdl.handle.net/20.500.14094/00517757>



Simulation of Priority-Concurrent Write Buses by Separable Buses with Fewer Switches

Takashi MAEBA[†]

Abstract

Processor arrays with priority-concurrent write buses have been shown to be equal in computational power to processor arrays with separable buses. The separable bus can be dynamically segmented into smaller bus portions of any length by turning off some switches in the bus. The concurrent write bus permits two or more processors to send a datum simultaneously.

In this paper, we show that switches as asymptotically many as processors involved are required for the separable bus to ensure the equivalency in power. This indicates that propagation delays caused by the switches may increase gradually as the size of a processor array is enlarged. We, then, introduce a concurrent write rule, which permits two or more processors each to send the value 1 to an arbitrary bit of a bus simultaneously, into the separable bus in order to decrease the complexity of the bus. Our results show that a processor array with such separable buses can simulate a priority-concurrent write operation to the bus in constant time even though the number of switches involved is reduced to $O(1/\log^\alpha N)$, where N is the number of processors and α is a constant.

(Received April 1, 2006)

1. Introduction

Processor arrays can efficiently solve many different problems appearing in engineering and scientific fields [1, 2]. In addition, they can be area-efficiently embedded in VLSI circuits by virtue of their simplicity and regularity. For the problems that require data exchange between arbitrary processors, however, high-speed computations cannot be expected since the communication diameter of a processor array is significantly large. In order to resolve such difficulties with the diameter, processor arrays enhanced with various types of buses have been proposed and studied intently [3, 4, 5, 6, 7]. Furthermore, the difference in power of buses with such abilities as *exclusive/concurrent write*, *fusing*, and *segmenting* has been investigated thoroughly [8, 9].

We deal with processor arrays enhanced with two types of buses: a concurrent write bus operating under a *priority-conflict resolution* rule¹ (a priCW bus for short) and a *separable bus* that has exclusive write capability² and can be segmented into some bus portions of arbitrary length by turning off some switches in the bus [7]. Some of the results proved in [9] show that processor arrays with priCW buses

are equal in power³ to processor arrays with separable buses (termed *HV-RN*'s in [9], though) whenever at most $O(n^k)$ processors are available, where $k > 0$ is a constant and n is the size of a problem to be solved. In addition, the equivalency in power is valid for the separable bus if the distance between adjacent switches inserted into the bus is a constant. That is, if switches as asymptotically many as processors involved are available, then a priority-concurrent write operation to the bus (priCW operation) can be simulated on a processor array with separable buses (PAS_b) in constant time. Consequently, the complexity of the separable bus may increase due to a large number of switches. We, therefore, construct a constant-time algorithm for simulating the priCW buses by separable buses with fewer switches.

To begin with, we show that the distance d between adjacent switches in the separable bus must be a constant (independent of the number of processors) whenever the PAS_b simulates a priCW operation in constant time. Next, we introduce a concurrent write (CW) rule into the PAS_b, whose bus commonly runs under an exclusive write rule [7], in order to reduce the number of switches in the bus.

[†] The author is with the Faculty of Maritime Sciences, Kobe University.

¹ Each processor is assigned certain *priority* in advance; when two or more processors attempt to send a datum to the bus, the one with the highest priority succeeds (we assume, through the paper, that the highest priority is given to the processor with the lowest index).

² At a time unit, at most one processor can send a datum to the bus that is shared with the other processors.

³ Let $\mathcal{M}$ and $\mathcal{K}$ be two models of computation. Then, we say that $\mathcal{M}$ is equal in computational power to $\mathcal{K}$ if $\mathcal{M}$ can simulate any instruction of $\mathcal{K}$ in constant time, and vice versa.

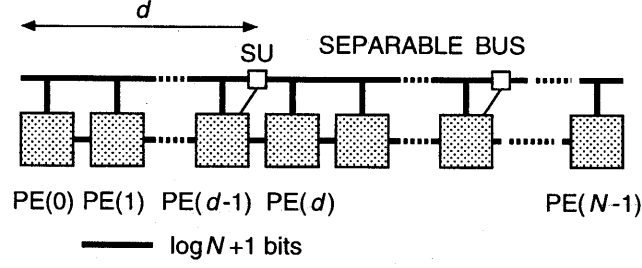


Figure 1. $N\text{-PAS}_b$.

The CW rule introduced permits two or more processors each to send only the value 1 to an arbitrary bit of a bus simultaneously, and is basically similar to the rule used in [10, 11, 12]. We prove that such a PAS_b can simulate a priCW operation without an asymptotic increase in time even though d is expanded from $O(1)$ to $O(\log^\alpha N)$, where N is the number of processors and $\alpha > 0$ is a constant. Separable buses operating under the CW rule mentioned above can be built using wired-logic mechanisms without substantial difficulty. We also show that a bus width of $O(\log N)$ -bit is sufficient for the PAS_b to simulate a priCW operation in constant time if the number of processors is at most N , as compared with a width of $O(N^c)$ -bit in [10, 12], where $0 < c \leq 1$. Finally, an algorithm for simulating a CRCW PRAM by a PAS_b is presented.

2. Definitions of Models

An $N\text{-PAS}_b$ is a linear array consisting of N processing elements (PE's) with a separable bus (see Fig. 1). Each bidirectional communication link between adjacent PE's has a width of $\log N + 1$ bits.⁴ A separable bus also consists of $\log N + 1$ wires, and involves a switching unit (SU) every d PE's. Each SU is controlled only by the PE directly connected. Suppose for simplicity that N/d is a natural number.

Each PE has a fixed number of registers consisting of $O(\log N)$ -bit and can perform each of basic instructions such as arithmetic and logical operations, a *send/receive* operation, an operation for turning 'off'/'on' an SU (if any), and a *load/store* operation, in $O(1)$ time. Each PE also holds a unique index j , $0 \leq j \leq N - 1$, in register ID, and is denoted by $\text{PE}(j)$.

The bus can be dynamically divided into some bus segments of adequate length by turning some SU's 'off', while adjacent bus segments can unite

by turning an SU 'on'. This allows the PAS_b to modify its bus configuration while a problem is being solved. A datum sent onto a bus (segment) can be received by all the PE's sharing the same bus (segment). Assume that a datum is transferred in constant time regardless of the length of the bus (segment). When at most one PE can send a datum (consisting of $\log N + 1$ -bit information) to the bus (segment) at a given time unit, we say that the bus operates under an exclusive write (EW) rule. We also introduce the following concurrent write (CW) rule:

Each PE can (separately from the other PE's) write the value 1 to an arbitrary bit out of $\log N + 1$ bits of the bus, but not the value 0, when more than one PE performs a *send* instruction to the bus at a given time unit. If two or more PE's write 1's to certain bits of the bus, then the corresponding bits hold 1's; while the remaining bits (to which no PE's write 1's) hold 0's, since we suppose that each bit of the bus is normally pulled down to the ground voltage. Note that two or more PE's each can concurrently send a datum in which only the k th bit is a 1 (k can differ by each PE) and the remainder are all 0's (e.g., $\underbrace{0 \cdots 0}_{\log N - k} 1 \underbrace{0 \cdots 0}_k = 2^k$, $0 \leq k \leq \log N - 1$)

to the bus, but cannot simultaneously send a value in which two or more bits are 1's (e.g., $011 \underbrace{0 \cdots 0}_{\log N - 2} =$

$\frac{3N}{4}$). We refer to this rule as an *extended concurrent write* (edCW) rule as stated in [12]. Obviously, any PE can *exclusively* send any value expressed with $\log N + 1$ bits to the edCW bus. Although edCW rules that permit several PE's to write 1's to multiple bits simultaneously may be considered, we do not deal with them here.

Two-dimensional $N_i \times N_j$ - PAS_b 's can be defined similarly.

⁴ All logarithms used in the paper are to the base 2, unless otherwise indicated.

3. Simulation Algorithms

3.1 A Lower Bound

We show a lower bound on the time for simulating priCW operations on a PAS_b operating under the EW rule. To begin with, a lower bound on the time for computing the logical OR of N inputs on the N - PAS_b is shown through a similar argument to [3]. We, then, derive a lower bound on the time for the simulation from the logical OR by *problem reduction*.

Lemma 1: When each PE holds $f_j \in \{0, 1\}$ for $0 \leq j \leq N-1$, $\Omega(\sqrt{d})$ time is required to compute the logical OR ($\vee$) of N inputs, $f_0 \vee f_1 \vee \dots \vee f_{N-1}$ on an N - PAS_b whose bus operates under the EW rule.

(Proof) Let $I(t)$ be the largest number of data (which are related to initial inputs f_j) that an arbitrary PE can receive by time t . Then, $2t+1$ data are acquired via communication links and the information of $\sum_{s=0}^{t-1} (2s+1)$ data is transferred through any bus segment. Since the bus can be divided into at most N/d segments and each bus segment can transmit the information exclusively sent by a PE, we have

$$I(t) = 2t + 1 + O(N/d) \sum_{s=0}^{t-1} (2s + 1).$$

Now assume that the computation has finished at time T . Then, at least N data related to the initial inputs must be acquired by the PE that holds the result of the logical OR. Otherwise, there exists more than or equal to one datum that will not affect the final result. Hence, from the inequality

$$I(T) \geq N,$$

we have $T = \Omega(\sqrt{d})$. $\square$

Next, we derive a lower bound on the time for solving a leftmost one problem from Lemma 1. When each PE(j) holds $f_j \in \{0, 1\}$ for $0 \leq j \leq N-1$, the leftmost one problem is a problem in which the lowest index ID' of all the PE's with $f_j = 1$ is searched out (i.e., $ID' = \min\{j | f_j = 1\}$). Suppose that the value -1 is returned when no PE holds $f_j = 1$. We call PE(ID') the *leftmost-one PE*.

Lemma 2: When each PE holds $f_j \in \{0, 1\}$ for $0 \leq j \leq N-1$, $\Omega(\sqrt{d})$ time is required to solve the leftmost one problem on an N - PAS_b whose bus operates under the EW rule.

(Proof) Given an algorithm $\mathcal{A}$ that solves the leftmost one problem on an N - PAS_b whose bus operates under the EW rule, the logical OR of N inputs

can be computed by $\mathcal{A}$ as follows:

$$\bigvee_{j=0}^{N-1} f_j = \begin{cases} 0 & \text{if there is no leftmost-one PE} \\ & \text{(i.e., the result is } -1\text{);} \\ 1 & \text{otherwise.} \end{cases}$$

Thus, Lemma 1 applies to the leftmost one problem. $\square$

Lemma 2 leads to the following result on the simulation of a priCW operation on the N - PAS_b .

Theorem 1: An N - PAS_b whose bus operates under the EW rule requires $\Omega(\sqrt{d})$ time to simulate a priCW operation.

(Proof) Let $\mathcal{B}$ be an algorithm for simulating a priCW operation on an N - PAS_b under the EW rule. Then, $\mathcal{B}$ solves the leftmost one problem as follows:

Each PE with $f_j = 1$ concurrently sends $ID + 1$ to the bus through $\mathcal{B}$. Then, $ID' + 1$ appears on the bus, where ID' holds the lowest index of all the PE's with $f_j = 1$ (i.e., the index of the leftmost-one PE). Now all PE's know ID of the leftmost-one PE by receiving the datum on the bus and then subtracting 1 from the received. If no leftmost-one PE exists, all PE's get -1 because the bus holds 0.

Since $\Omega(\sqrt{d})$ time is required to solve the leftmost one problem on the N - PAS_b under the EW rule, the theorem follows. $\square$

Theorem 1 implies that $d = O(1)$ (hence, $O(N)$ -SU) is necessary (and is sufficient [9]) to simulate a priCW operation on the N - PAS_b in $O(1)$ time as long as the bus operates under the EW rule. In the next section, we demonstrate that d can be expanded to $O(\log^\alpha N)$ with a constant *slowdown* when the PAS_b employs the edCW rule defined in Sect.2. for reduction in the number of SU's, where α is a constant.

3.2 Simulation of PriCW Buses

As preliminaries to constructing an algorithm for simulating priCW operations on a PAS_b , we provide an algorithm for solving the leftmost one problem first. In this section, the edCW rule described in Sect.2. is applied to the bus of the PAS_b . Suppose that $d = \log^\alpha N$ and d is a natural number for simplicity, where $\alpha \leq \frac{\log N}{\log \log N}$ is a natural number. A datum consists of $\log N + 1$ bits; a negative is expressed using the 2's complement representation.

Initially, $f_j, 0 \leq j \leq N-1$, is stored in register f of PE(j). The value of $\log^\alpha N$ is also stored in register d of PE(j). At the end of the computation, register lm of every PE retains the index of the leftmost-one PE if it exists and retains -1 otherwise.

Algorithm 1: LMO;

0° Each PE(j) for $0 \leq j \leq N - 1$ performs $lm \leftarrow -1$.

1° {In each block consisting of d PE's sharing the same bus segment, it is examined whether there exist PE's with $f = 1$.}

1-1° Each PE($u \cdot d - 1$) turns 'off' its SU in the bus, where $1 \leq u < \frac{N}{d}$.

1-2° Each PE with $f = 1$ sends 1 ($= 2^0$) to the 0th bit (i.e., the LSB) of the bus, while each PE($u \cdot d - 1$) receives the datum on the bus and then stores it into register y , where $1 \leq u \leq \frac{N}{d}$.

2° {Find the block including the leftmost-one PE.}

2-1° Each PE($u \cdot d - 1$) with $y = 0$ turns 'on' its SU, where $1 \leq u < \frac{N}{d}$.

2-2° Each PE($u \cdot d - 1$) for $1 \leq u < \frac{N}{d}$ sends the contents in ID to the bus if $y = 1$, while PE($N - 1$) sends the contents in ID to the bus if $y = 1$ and sends -1 to the bus otherwise. PE(0) receives the datum on the bus, then stores it into B.

2-3° PE($u \cdot d - 1$) turns 'on' its SU, where $1 \leq u < \frac{N}{d}$.

2-4° PE(0) sends the contents into B to the bus, while each PE(j) receives the datum on the bus and then stores it into B, where $1 \leq j \leq N - 1$.

2-5° Each PE(j), $0 \leq j \leq N - 1$, performs $c \leftarrow f$ if $ID \geq B - (d - 1)$ and $ID \leq B$, and performs $c \leftarrow 0$ otherwise.

3° {Find the leftmost-one PE within the block, which includes some PE's with $c = 1$.}

3-1° Each PE with $c = 1$ performs $mp \leftarrow d - ((ID \bmod d) + 1)$.

3-2° For $s = 0, 1, \dots, \alpha - 1$, each PE with $c = 1$ executes the following instructions sequentially:

$mn \leftarrow \lfloor mp / \log N \rfloor$; $g[s] \leftarrow mp - mn \cdot \log N$; $mp \leftarrow mn$.

3-3° For $s = \alpha - 1, \alpha - 2, \dots, 0$, each PE performs the following Steps 3-3-1° and 3-3-2° in sequence:

3-3-1° Each PE with $c = 1$ sends 1 to the $g[s]$ th bit of the bus (hence, $2^{g[s]}$), and each PE with $c = 1$ receives the datum on the bus, then stores it into y .

3-3-2° Each PE with $c = 1$ performs $c \leftarrow 0$ if $2^{g[s]} \leq y/2$ (i.e., at least one bit b_k in register y contains a 1, where $k > g[s]$).

4° {Broadcast ID of the leftmost-one PE.}

4-1° Each PE($u \cdot d - 1$), $1 \leq u < \frac{N}{d}$, turns 'on' its SU.

4-2° (The leftmost-one) PE with $c = 1$ performs $lm \leftarrow ID$.

4-3° (The leftmost-one) PE with $c = 1$ sends the contents in lm to the bus, while each PE with $c = 0$ receives the datum on the bus and then stores it

into lm if $B \neq -1$. $\square$

Algorithm **LMO** finds out the block including the leftmost-one PE in Steps 1° and 2°, then determines the leftmost-one PE within the block in Step 3°. Step 3° selects the PE with the lowest ID from among the PE's, which hold $f_j = 1$, in the block using a radix r -number system approach proposed in [10, 12]. We substitute finding the lowest ID for searching out the highest $\widetilde{ID}$ here. $\widetilde{ID}$ gives the relative location of a PE within a block consisting of d PE's, where the leftmost PE has the maximum and the rightmost the minimum (i.e., $\widetilde{ID}(j) = d - ((j \bmod d) + 1)$). Since $\widetilde{ID}(j)$ can be expressed in the form

$$\widetilde{ID}(j) = \sum_{s=0}^{\alpha-1} g_s(j) \cdot \log^s N,$$

we obtain the highest $\widetilde{ID}(j)$ (hence, the lowest ID) of all the PE's with $f_j = 1$ by sequentially examining g_s from the most significant digit ($s = \alpha - 1$) to the least one ($s = 0$), where $r = \log N$ and $0 \leq g_s(j) \leq \log N - 1$. In Step 3-3-1°, as each PE(j) with $c = 1$ (i.e., PE with $f_j = 1$ within the block found in 2°) simultaneously writes 1 to the $g_s(j)$ th bit of the bus, the value $y = \sum_{p=0}^{\log N - 1} b_p \cdot 2^p$ appears on the bus, where $b_p = 1$ if there exists at least one PE(j) such that $g_s(j) = p$; $b_p = 0$ otherwise. Each PE(j) with $c = 1$, then, knows whether its own $g_s(j)$ is larger than the others or not by comparing y with $2^{g_s(j)}$ (in 3-3-2°). Consequently, the highest $\widetilde{ID}(j)$ satisfies $2^{g_s(j)} > y/2$ for every iteration s , $\alpha - 1 \geq s \geq 0$. Each step of **LMO** consists of only basic instructions; **LMO** runs in $O(\alpha)$ time, since Steps 3-2° and 3-3° are repeatedly executed α times. Hence, the following result is obtained.

Lemma 3: An N -PAS_b whose bus operates under the edCW rule can solve the leftmost one problem in $O(1)$ time, where $d = \log^\alpha N$ and α is a constant (natural number). $\square$

Next, we provide an algorithm for simulating priCW operations on the N -PAS_b through **LMO**. Assume that data to be sent are stored in registers x of some PE's before the execution of the algorithm.

Algorithm 2: PriCW;

0° Each PE that intends to execute a *send* instruction (to the bus) performs $f \leftarrow 1$, while the other PE's perform $f \leftarrow 0$.

1° Execute **LMO**.

2° PE(lm) sends the contents in its x to the bus. $\square$

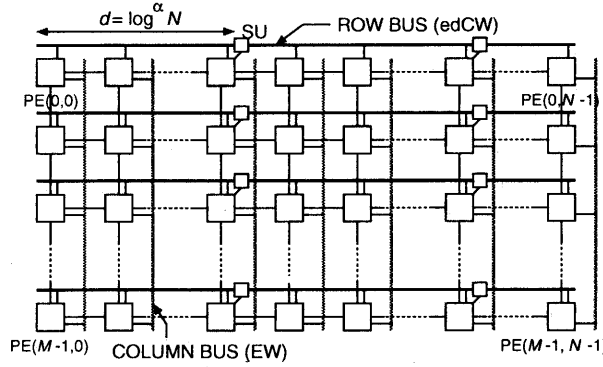


Figure 2. $M \times N$ -PAS_b simulating (N, M) -PRI CRCW PRAM.

Steps 0° and 2° of algorithm **PriCW** consist of only basic instructions; Step 1° runs in $O(1)$ time as shown in Lemma 3. We, therefore, obtain the following result.

Theorem 2: A priCW operation can be simulated in $O(1)$ time on an N -PAS_b operating under the edCW rule, where $d = \log^\alpha N$ and α is a constant (natural number). $\square$

Theorem 2 states that introducing the edCW rule, which is relatively simple to implement in hardware, allows the PAS_b to expand d from a constant [9] to $O(\log^\alpha N)$ without asymptotically increasing the running time for the simulation. Consequently, reduction in the number of SU's from $O(N)$ to $O(N/\log^\alpha N)$ is achievable, and makes possible the curtailment of propagation delays (throughout the paper, we assume a fixed propagation delay regardless of the number of SU's, though).

3.3 Simulation of Priority-CRCW PRAM

Let (N, M) -PRI CRCW PRAM denote a PRAM that consists of N processors ($P_j, 0 \leq j \leq N-1$) and M shared memory-cells ($S_i, 0 \leq i \leq M-1$), and has the capability of *concurrent read* and *concurrent write* resolved by the *priority* rule. As an application of algorithm **PriCW**, we present an algorithm for simulating the (N, M) -PRI CRCW PRAM by an $N \times M$ -PAS_b using the method as shown in [13]. Note that the $N \times M$ -PAS_b employed here has EW column buses with $d = M$ (i.e., no SU's exist in the buses) and edCW row buses with $d = \log^\alpha N$ (see Fig. 2). This arrangement suggests that the application of the edCW rule to *reconfigurable bus systems* [13] makes possible reducing the number of switches used for this simulation.

Since simulation of each of basic instructions such

as arithmetic-logic operations performed in processors is clearly trivial, we focus our attention on the simulation of a *read* and a *write* instruction to the shared memory of the PRI CRCW PRAM. Suppose that P_j is simulated by $PE(0, j)$ and S_i is simulated through register sm of $PE(i, 0)$, where $0 \leq i \leq M-1$ and $0 \leq j \leq N-1$.

Algorithm 3: PriCRCW-SM;

write(A, x):

- 1° Each $PE(0, j)$ that intends to execute a *write* instruction performs $f \leftarrow 1$, while the others $PE(0, j)$ perform $f \leftarrow 0$, where $0 \leq j \leq N-1$.
- 2° Each $PE(0, j)$ sends the contents in f to the column bus, while each $PE(i, j)$ receives the datum on the column bus and then stores it into f , where $1 \leq i \leq M-1$ and $0 \leq j \leq N-1$.
- 3° Each $PE(0, j)$ with $f = 1$, for $0 \leq j \leq N-1$, sends the address (the contents in A) of the memory-cell, into which a datum should be written, and the datum itself (the contents in x) to the column bus in sequence, while each $PE(i, j)$ with $f = 1$, for $1 \leq i \leq M-1$ and $0 \leq j \leq N-1$, receives two data on the column bus in sequence and then stores them into A and x , respectively.
- 4° Each $PE(i, j)$ with $f = 1$ performs $f \leftarrow 0$ if $A \neq i$, where $0 \leq i \leq M-1$ and $0 \leq j \leq N-1$.
- 5° All the PE's in each row $i, 0 \leq i \leq M-1$, execute Steps 1° and 2° of **PriCW**. Then each $PE(i, 0)$ with $lm \neq -1$ receives the datum on the row bus, and stores it into sm , where $0 \leq i \leq M-1$.

read(A, r):

- 0° Each $PE(i, u \cdot d - 1)$ turns 'on' its SU in the row bus, where $0 \leq i \leq M-1$ and $1 \leq u < \frac{N}{d}$.
- 1° Each $PE(0, j)$ that intends to execute a *read* instruction performs $f \leftarrow 1$, while the others $PE(0, j)$ perform $f \leftarrow 0$, where $0 \leq j \leq N-1$.

2° Each $PE(0, j)$ sends the address (the contents in A) of the memory-cell, from which a datum should be read, to the column bus if $f = 1$ and sends -1 otherwise, while each $PE(i, j)$ receives the datum on the column bus and then stores it into A , where $1 \leq i \leq M - 1$ and $0 \leq j \leq N - 1$.

3° Each $PE(i, 0)$ sends the contents in sm to the row bus, while each $PE(i, j)$ receives the datum on the row bus and then stores it into sm if $A = i$, where $0 \leq i \leq M - 1$ and $1 \leq j \leq N - 1$.

4° Each $PE(i, j)$ sends the contents in sm to the column bus if $A = i$, while each $PE(0, j)$ with $f = 1$ receives the datum on the column bus and then stores it into r , where $0 \leq i \leq M - 1$ and $0 \leq j \leq N - 1$. $\square$

In **PriCRCW-SM**, Step 5° of **write** runs in $O(1)$ time as shown in Theorem 2; the rest of the steps consist of only basic instructions. All the steps of **read** also consist of only basic instructions. **PriCRCW-SM**, therefore, runs in $O(1)$ time. This implies that any algorithm running on the (N, M) -PRI CRCW PRAM can be executed on the $N \times M$ -PAS_b (as illustrated in Fig. 2) with a constant slowdown.

Theorem 3: Any algorithm running on an (N, M) -PRI CRCW PRAM in time T can be executed in $O(T)$ time on an $M \times N$ -PAS_b whose column buses with $d = M$ operate under the EW rule and row buses with $d = \log^\alpha N$ operate under the edCW, where α is a constant. $\square$

A *hyper-broadcast bus network* (HBBN) [12] whose buses operate under the edCW rule, as well as the PAS_b, can simulate any one step of the (N, M) -PRI CRCW PRAM in $O(1)$ time with $O(MN)$ processors. The HBBN, however, employs a bus width of $O(N^c)$ -bit, where $0 < c \leq 1$ is a constant, whereas a width of $O(\log N)$ -bit is sufficient for the PAS_b. If the bus width is limited to $O(\log N)$ -bit, the HBBN requires $O(N^c)$ [12] times as many processors as the PAS_b does, namely $O(MN^{1+c})$ processors. This indicates that enhancing the edCW buses (of HBBN) with $O(MN/\log^\alpha N)$ switches results in thinning down the bus width from $O(N^c)$ to $O(\log N)$ bits for that simulation.

4. Conclusion

In this paper, we have introduced the edCW capability into the PAS_b to decrease the complexity of the bus. Our results have shown that the N -PAS_b with such capability can simulate a priCW operation in $O(1)$ time even though the number of switching units in the bus is reduced to $O(1/\log^\alpha N)$, where α is a constant. We also

mentioned that the application of our results to the HBBN makes possible thinning down its bus width from $O(N^c)$ to $O(\log N)$ bits, where $c \leq 1$ is a constant.

References

- [1] F. T. Leighton, "Introduction to parallel algorithms and architectures: Arrays· Trees· Hypercubes," Morgan Kaufmann Pub., San Mateo, CA, 1992.
- [2] R. Miller and Q.F. Stout, "Parallel algorithms for regular architectures," MIT Press, Cambridge, MA, 1996.
- [3] Q.F. Stout, "Mesh-connected computers with broadcasting," IEEE Trans. Comput., pp.826-830, vol.C-32, no.9, Sept. 1983.
- [4] C.S. Raghavendra, "HMESH: A VLSI architecture for parallel processing," Proc. Conf. Algorithms and Hardware for Parallel Processing, pp.76-83, Aachen, Germany, Sept. 1986.
- [5] V.K. Prasanna-Kumar and C.S. Raghavendra, "Array processor with multiple broadcasting," J. Parallel and Dist. Computing, vol.4, no.2, pp.173-190, April 1987.
- [6] R. Miller, V.K. Prasanna, D.I. Reisis, and Q.F. Stout, "Data movement operations and applications on reconfigurable VLSI arrays," Proc. International Conf. Parallel Processing, vol.1, pp.205-208, Aug. 1988.
- [7] T. Maeba, S. Tatsumi, and M. Sugaya, "Algorithms for finding maximum and selecting median on a processor array with separable global buses," Electronics and Communications in Japan, part 3, vol.J73, no.6, pp.39-47, 1990.
- [8] Y. Ben-Asher, D. Peleg, P. Ramaswami, and A. Schuster, "The power of reconfiguration," J. Parallel and Dist. Computing, vol.13, no.2, pp.139-153, 1991.
- [9] J.L. Trahan, R. Vaidyanathan, and R.K. Thiruchelvan, "On the power of segmenting and fusing buses," J. Parallel and Dist. Computing, vol.34, no.1, pp.82-94, 1996.
- [10] T.-W. Kao and S.-J. Horng, "Efficient algorithms for computing two nearest-neighbor problems on a RAP," Pattern Recognition, vol.27, no.12, pp.1707-1716, 1994.

- [11] H.M. Alnuweiri, "Parallel constant-time connectivity algorithms on a reconfigurable network of processors," IEEE Trans. Parallel and Dist. Systems, vol.6, no.1, pp.105-110, Jan. 1995.
- [12] H.-R. Tsai, S.-J. Horng, S.-S. Tsai, T.-W. Kao, and S.-S. Lee, "Solving an algebraic path problem and some related graph problems on a hyper-bus broadcast network," IEEE Trans. Parallel and Dist. Systems, vol.8, no.12, pp.1226-1235, Dec. 1997.
- [13] B. -F. Wang and G. -H. Chen, "Two-dimensional processor array with a reconfigurable bus system is at least as powerful as CRCW PRAM," Info. Process. Letters, vol.36, no.1, pp.31-36, Oct. 1990.