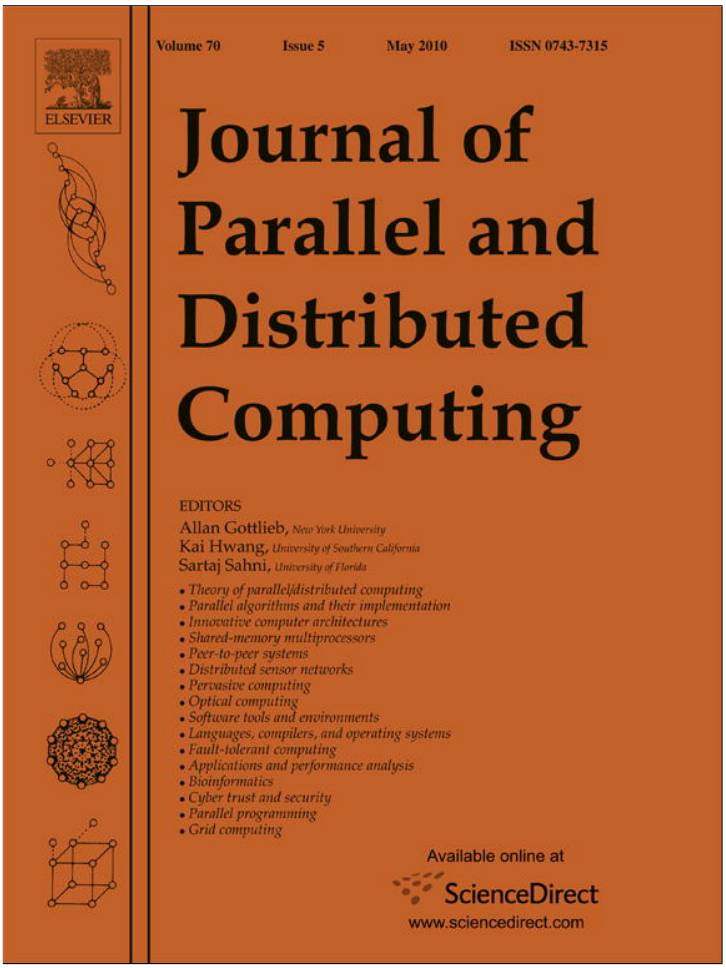

J. Parallel Distrib. Comput. 70 (2010) 458–466

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Relating the power of the Multiple Associative Computing (MASC) model to that of reconfigurable bus-based models☆

Jerry L. Trahan [a], Mingxian Jin [b,*], Wittaya Chantamas [c], Johnnie W. Baker [c]

[a] Department of Electrical and Computer Engineering, Louisiana State University, Baton Rouge, LA 70803, United States
[b] Department of Mathematics and Computer Science, Fayetteville State University, Fayetteville, NC 28301, United States
[c] Department of Computer Science, Kent State University, Kent, OH 44242, United States

ARTICLE INFO

Article history:
Received 8 January 2009
Received in revised form
23 April 2009
Accepted 15 August 2009
Available online 21 August 2009

Keywords:
Parallel computational model
Associative computing
Multiple SIMD
Simulation
Reconfigurable buses
Reconfiguration

ABSTRACT

The MASC (Multiple ASsociative Computing) model is a multi-SIMD model that uses control parallelism to coordinate the interaction of data parallel threads and supports associative SIMD computing on each of its threads. There have been a wide range of algorithms developed for this model. Research on using this model in real-time system applications and building a scalable MASC architecture is currently quite active. In this paper, we present simulations between MASC and reconfigurable bus-based models, e.g., various versions of the Reconfigurable Multiple Bus Machine (RMBM). Constant time simulations of the basic RMBM by MASC and vice versa are obtained. Simulations of the segmenting RMBM, fusing RMBM, and extended RMBM by MASC in non-constant time are also discussed. By taking advantage of previously established relationships between RMBM and two other popular parallel computational models, namely, the Reconfigurable Mesh (RM) and the Parallel Random Access Machine (PRAM), we extend our simulation results to further categorize the power of the MASC model in relation to RM and PRAM.

1. Introduction

The ASsociative Computing (ASC) model is based on the STARAN associative SIMD computer designed by Kenneth Batcher at Goodyear Aerospace in the early 1970s and its heavily Navy-utilized successor, the ASPRO. These architectures easily supported a wide range of applications and a simple style of programming much closer to sequential programming than to traditional multiprocessor programming. The MASC (for *Multiple ASsociative Computing*) model is a generalization of the associative computing model designed to support multiple ASC threads by using control parallelism to substantially improve the low processor utilization often criticized in SIMDs. It was proposed by Potter et al. [18] in 1994. Since the MASC model is a strictly synchronous model that supports SIMD computation, it is sometimes called a multi-SIMD or MSIMD model (i.e., a SIMD enhanced with multiple instruction streams).

In contrast to a number of other parallel models, the MASC model possesses certain constant time global properties, when the word size is assumed to be a constant, such as constant time broadcasting, constant time global reduction operations, and constant time associative search. These properties have enabled MASC not only to solve a wide range of problems effectively [2,7] but also problems in special areas such as real-time air traffic control in an extremely efficient manner, using worst case analysis to ensure that all deadlines are met [13]. A standard associative language that supports the one IS version of MASC (also called ASC) has been implemented on a number of SIMD platforms [17,18]. Possible techniques for implementing the MASC model have been explored and related research work is still ongoing [5,20,26].

The power of a computational model is indicated by the efficiency with which it can simulate other computational models and algorithms it supports. To evaluate the power of the MASC model, simulation and comparison with other popular parallel models have been studied previously. If a constant word size is assumed, a constant time simulation with high probability of PRAM (*Parallel Random Access Machine*) using MASC and a constant time simulation of MMB (*Mesh with Multiple Buses*) using MASC have been established [3,23]. In this paper, we present simulations between MASC and reconfigurable bus-based models RMBM (*Reconfigurable Multiple Bus Machine*). Our goal is to compare the power of the MASC model to that of the RM (*Reconfigurable Mesh*) model, as RM has been widely accepted as an extremely powerful model. To do so, we use RMBM as a bridge model and discuss simulations between RMBM and MASC first. Then, we extend our simulation

☆ The preliminary version of this work has been published in Jin and Baker (2007) [11].

* Corresponding author.
E-mail addresses: trahan@ece.lsu.edu (J.L. Trahan), mjin@uncfsu.edu (M. Jin), wchantam@cs.kent.edu (W. Chantamas), jbaker@cs.kent.edu (J.W. Baker).

0743-7315/$ – see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2009.08.001

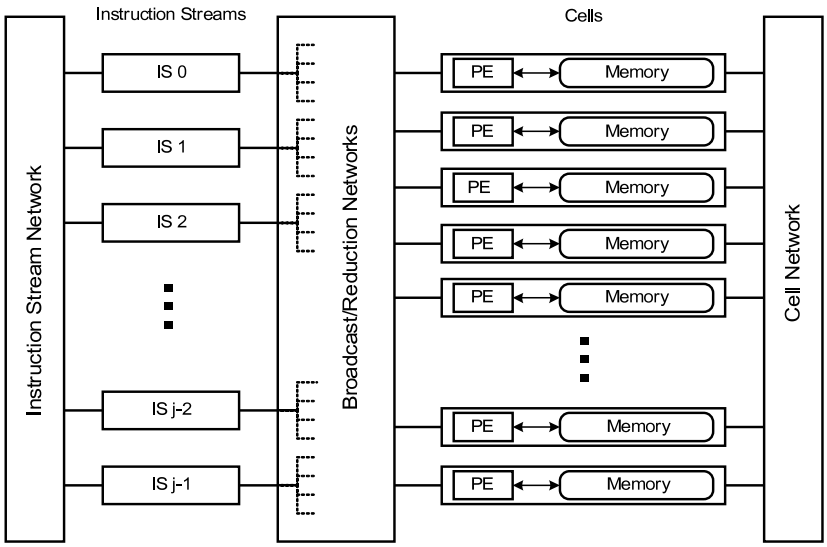

Fig. 1. The MASC model.

results to establish further relationships among MASC, RM, and PRAM, as the strongest version of RMBM has the same power as RM and its weakest version has the same power as PRAM. Our work provides a better understanding of the MASC model and useful information concerning its power.

This paper is organized into eight sections. Section 2 summarizes properties of MASC, RMBM, and other models used in this paper. Sections 3–5 present simulations between MASC and different versions of RMBM, respectively. Section 6 compares two resulting MASC models when different word sizes are assumed. Section 7 extends the simulation results to categorize the power of MASC in relation to those of RM and PRAM. Section 8 gives concluding remarks and identifies some open problems.

2. Related parallel models

In this section, we provide a short summary of the parallel models used in this paper, i.e., the MASC, the RMBM, the RM, and the PRAM.

2.1. MASC

As shown in Fig. 1, MASC consists of an array of processing elements (PEs), and one or more instruction streams (ISs). A MASC with n PEs and j ISs is denoted by MASC (n, j). If S_i denotes the set of PEs listening to IS_i, then the set S_i normally changes during execution. Moreover, if IS_i and IS_j are distinct, then at any point during the execution, the sets S_i and S_j are disjoint.

Each PE, paired with its local memory, is called a *cell*. In this paper, we use the terms PEs and cells interchangeably. Instructed by its assigned IS, a PE performs arithmetic and logic operations as a conventional computer. However, a PE neither stores a copy of the program nor participates in decoding this program.

The number of ISs is expected to be considerably smaller than the number of PEs. Both ISs and PEs have unique numbers and each knows its number. To store a processor identification number, a non-constant word size of $\Theta(\log n)$ is required for a parallel model with n PEs, so $\Theta(\log n)$ is a frequent assumption for word size and bus widths in parallel models. Associative operations on a MASC employ a broadcast/reduction network that can have a different, smaller word size v for associative (reduction) operations. The preceding paragraph assumed that v was constant. Let MASC_v denote a MASC with word size v for associative operations, so MASC_c (or $\text{MASC}_{\log n}$) denotes a MASC with constant c (or $\log n$) associative operation word length. In Section 6, we will present a constant time simulation between MASC_c and $\text{MASC}_{\log n}$ derived from a cycle of simulation results.

The MASC model may have three networks, namely, a cell network for cell communications, an IS network for IS communications, and a broadcast/reduction network for communication between an IS and its set of cells. Both cell network and IS network may be implemented by a bus, a tree, or other simple network to facilitate the communications among cells and among ISs. They are optional for the model, as it is shown in Section 7 that with or without them, the power of the MASC model does not change. The broadcast/reduction network is essential, however. It may be implemented using separate network circuits or sharing the same network circuit for both broadcasting and global reduction operations. In practice, the network can be constructed as a tree-structured set of resolver circuits. Each leaf represents a PE and all PEs is divided into groups of 4 to make a basic 4-PE resolver as a building block. A resolver with n PEs can be built as a $\lceil \log_4 n \rceil$ level tree of 4-PE resolvers. When a broadcast or a reduction operation is performed, a bit takes at most $2\lceil \log_4 n \rceil - 1$ gate delays from any input to any output. This contrasts to a broadcast on a bus-based architecture that takes a linear order of gate delays with respect to the bus length and number of processors. Considering that the linear gate delay bus broadcast is generally assumed to be a constant time operation, the logarithmic gate delay MASC broadcast can certainly be treated as a constant time operation, even for impractically large numbers of processors n [10].

With its broadcast/reduction network, the MASC model supports global and broadcast operations between an IS and its active cells in constant time (assuming the word size for these operations is a constant; Sections 3–6 explore relaxing this assumption). These operations include the global reduction operations of OR and AND of binary values, the maximum and minimum of integer or real values, and an associative search by a given search pattern. Following an associative search, an IS can determine if any PE matches the search pattern ("any-responders") and select an arbitrary PE from its responders (or "pick-one") in constant time. These constant time operations are called associative operations and support highly efficient database operations on records stored in identical locations across the PEs. The feasibility of these assumptions has been justified by Jin et al. [10].

PEs grouped with the same IS can read a value from or write to that IS. The IS activates processors who need to read. The data item

to be read is broadcast to those active PEs to first read and then store the broadcast item. PEs grouped with the same IS can write to that IS where the IS uses associative operations to determine what is written. A typical occurrence is for an IS to first have the active PEs make a search. If there is at least one responder, PickOne selects one of the responding PEs and the IS instructs this PE to put a data item on the broadcast bus. Another method is for active PEs for an IS to do a maximum or minimum (AND or OR) reduction over a value stored in a word (bit, respectively). The reduction network should connect to the IS so that it automatically receives the reduction value. If some of the PEs need this reduction value, the IS can broadcast it to them.

Earlier simulations between MASC and an enhanced meshes model (MMB) [3] allow MMB algorithms to be executed by MASC using the same number of processors. A MASC(n, j) with a 2D mesh cell network can simulate a $\sqrt{n} \times \sqrt{n}$ MMB in $O(1)$ time when $j = \Omega(\sqrt{n})$, but simulation of MASC(n, j) with a $\sqrt{n} \times \sqrt{n}$ MMB takes $O(jn^{\frac{1}{6}})$ time. This is in line with the finding that MASC(n, j) with a 2D mesh is more powerful than a $\sqrt{n} \times \sqrt{n}$ MMB when $j = \Omega(\sqrt{n})$ [3].

2.2. Reconfigurable mesh

A 2D *Reconfigurable Mesh* (RM) is a basic mesh model enhanced by reconfigurable buses. Each processor has four ports, referred to as N, S, E, and W, that can be controlled locally, allowing disjoint buses to be established dynamically. With the ability of reconfiguring bus connections during algorithm execution, an RM can create different communication patterns based on the algorithm needs.

Local connections of a processor can be restricted in different ways to obtain variants of RM. For example, if a processor is allowed to fuse at most one pair of the ports {EW} or {NS}, the restricted RM is called a BRM (*Basic RM*). If a processor is allowed to fuse any pairs of ports, the restricted RM is called an LRM (*Linear RM*). If all processors connect their ports as {EW} and {NS}, the restricted RM is equivalent to an MMB.

The RM model has been widely accepted as an extremely powerful model. A number of constant time algorithms have been discovered for this model [1,4,15,16] (assuming a constant time bus broadcast), while they require non-constant time on other models [1,16,25]. Due to the power of RM and the fact that it is a well-known model, we wish to establish a relationship between MASC and RM. This relationship could be a useful tool in evaluating the power of the MASC model. However, dissimilarities between the MASC and a general RM make direct simulations difficult. Instead, we consider a bridge model – the RMBM that has been shown to be equally powerful to RM – to simulate, with the intent of establishing a relationship between MASC and RM.

2.3. Parallel random access machine

The *Parallel Random Access Machine* (PRAM) model is a fundamental parallel model and continues to be an important model for parallel computation. A PRAM comprises a collection of identical processors and a shared memory [1,9]. Each processor also has a local memory. Generally, each processor executes the same step in a program synchronously. Processors cycle through three phases synchronously—read a memory element, execute a program step, and write to a memory element. Exclusive read (ER) or concurrent read (CR) of memory elements is possible. Likewise, exclusive write (EW) or concurrent write (CW) is possible. For CW, some predefined rule resolves write conflicts. Examples include Common (all processors writing to the same location write the same value), Collision (if multiple processors write to the same location, then a special collision symbol appears), Collision+(if all concurrently writing processors write the same value, then it behaves like Common; otherwise it behaves like Collision), Arbitrary (an arbitrary writing processor succeeds in its write attempt), and Priority (the highest priority writing processor succeeds) [1,9].

Ulm and Baker [23] studied simulations between MASC and PRAM. A MASC without a cell network can simulate a Priority CRCW PRAM in $O(1)$ time with high probability, and a PRAM can simulate each step of a j IS MASC in $O(j)$ time with high probability [23]. In this paper, we are not directly focused on giving a MASC-PRAM simulation. Our results in this paper do, however, establish $O(1)$ time deterministic simulations between MASC and PRAM (Section 6), from which we obtain the fact that MASC and PRAM are actually in the same category in terms of their computation power.

2.4. Reconfigurable multiple bus machine

The *Reconfigurable Multiple Bus Machine* (RMBM) is a reconfigurable bus-based model proposed by Trahan et al. [22,24]. The RMBM model consists of a set of processors and a set of buses that are used for processor communication. Through the use of its local settings, a processor can manipulate its read and write connections to buses as well as manipulate bus structure. A processor can split a bus into segments or connect (fuse) one bus to another. Each processor controls one pack of switches for each bus, with up to five switches in each pack, and a fuse line that crosses all buses. (See Fig. 2 which shows an RMBM model but omits fuse lines and switches.) Two switches control the connection of the bus to the processor's reading port and writing port, respectively. Two switches are used to segment the bus at the points where they are located. One is located to the left of the reading port and the other is between the reading and writing ports. The last switch is used to connect the processor fuse line to the bus. The processor fuse line can connect multiple (not necessarily adjacent) buses together.

Initially, all switches are reset, that is, buses are neither segmented nor fused and processor read and write ports are not connected to a bus. A processor can set or reset a switch in one time step but can only set or reset one switch at a time. A processor may connect its reading and writing ports to different buses. At any point of time, however, a processor can connect each port to only one bus. A value written on a bus reaches the other processors connected to the bus in the same step. We consider the concurrent read, concurrent write (CRCW) RMBM in this paper.

Depending on switches available in the processors, there are four versions of RMBM [22,24]:

B-RMBM: A processor only has switches to connect reading and writing ports to a bus.
S-RMBM: In addition to read/write switches, a processor can segment the bus.
F-RMBM: In addition to read/write switches, a processor can fuse buses using its fuse line.
E-RMBM: All five switches are functional for a processor, namely, a processor can connect to a bus for read/write, split a bus into segments, and fuse a bus to other buses via its fuse line. This is the strongest version of RMBM.

Relationships among different versions of RMBM, RM, and PRAM have been established [22,24]. Considering the similarities of RMBM and MASC models and taking advantage of the established relationships, we develop simulations between MASC and RMBM first, and then extend the simulation results to further relate MASC and RM. These imply relationships between MASC and PRAM as well.

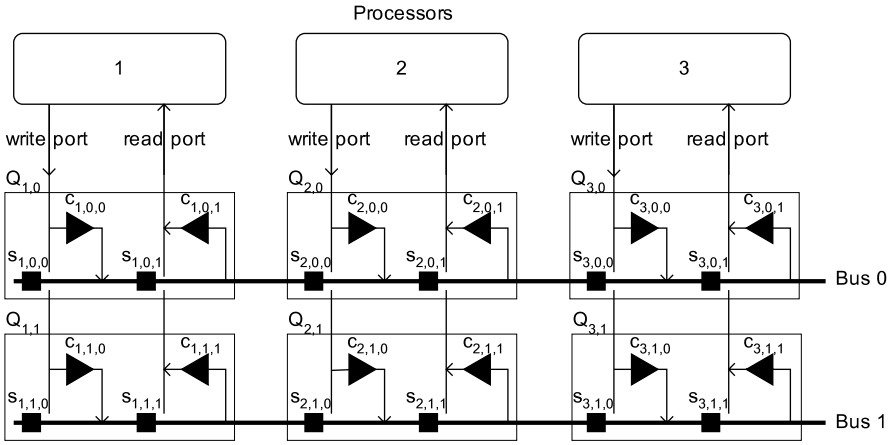

Fig. 2. The S-RMBM model with three processors.

The next three sections present simulations between the MASC model and different versions of the RMBM model. We start with the weakest version—B-RMBM. To avoid any confusion on the word size assumption on the MASC model, throughout Sections 3–5, we will assume MASC can perform constant time associative operations on v-bit words, where v can be a constant or a function of n, e.g., $O(\log n)$.

3. Simulations between MASC and B-RMBM

Simulating MASC with B-RMBM is fairly straightforward, as there are some similarities between the structures of the two models. Different B-RMBM resources, however, lead to slightly different results. Let $\text{MASC}_v(n, j)$ denote a MASC with n PEs and j ISs that can perform constant time associative operations (i.e., a global OR/AND operation, a maximum/minimum finding operation, or an associative search) on v-bit wide data. Let B-RMBM(n, m) denote a B-RMBM with n processors and m buses. To simulate each step of a $\text{MASC}_v(n, j)$, we start with a base B-RMBM with n processors and j buses using Common concurrent write (CW). We will examine how different combinations of available processors, buses, and write resolution rules affect the simulation. Assume that $j \leq n$. Let each of the first j processors, p_i, on the B-RMBM simulate a unique MASC IS, IS_i. The read/write ports of each p_i use bus i for IS_i communications. Each of the overall n processors simulates one of the n MASC PEs. For each of these n processors, the bus to which its read/write ports connect is determined by the IS to which the corresponding simulated MASC PE listens.

Each processor of the B-RMBM can simulate the local operations of the corresponding processor of the MASC, so the MASC operations to examine closely are those involving the broadcast/reduction networks. These are OR/AND, maximum/minimum, associative search, AnyResponder, and PickOne.

- bitwise OR/AND of up to n v-bit values—A B-RMBM can compute the OR or AND of n bits on one bus in constant time, so the base B-RMBM can execute this of one bit position at a time to get the result in $O(v)$ time. Alternatively, a B-RMBM(n^2, $\max\{n, jv\}$), where $v \leq n$, can compute this operation in $O(\log v)$ time as follows. Let IS_i with set S_i of PEs compute a global OR. For each processor p_k of the corresponding set S_i' on the B-RMBM, p_k first distributes v_i copies of its data to v other processors $p_{k,m}$, for $1 \leq m \leq v$. Each $p_{k,m}$ extracts bit m of the data, then over all p_k in S_i', processors $p_{k,m}$ compute the OR of their bits. Next, $\log v$ levels of pairwise merges of results combines the v individual ORs into a final v-bit result.
- maximum/minimum of up to n v-bit values—The base B-RMBM can compute a maximum or minimum in a bit-serial manner by iterating broadcasting and concurrent write steps via the bus for each bit of the field in the order of the left (most significant) bit to the right (least significant) bit. This runs in $O(v)$ time. This algorithm is based on a parallel search algorithm by Falkoff [8]. Alternatively, use a B-RMBM(n^2, n) to simulate a step of finding a maximum in $O(1)$ time, where $j \leq n$, following a technique used by PRAMs [12,21] and RMs [15]. The idea is to compare all pairs of elements for an IS computing maximum and use AND operations to identify as the maximum value one that is greater than or equal to all other elements. Specifically, let $p_{g,h}$, where $1 \leq g, h \leq n$, denote a processor of the B-RMBM. Broadcast all values being written and their corresponding IS indices, and let each $p_{g,h}$ read value r_g and IS i_g as well as r_h and IS i_h. If $i_g = i_h$, then proceed; otherwise, do nothing. On bus g, for all $p_{g,h}$ with $i_g = i_h$, AND the outcome of whether $r_g \geq r_h$.
- associative search—When IS_i performs an associative search with value t and set S_i of PEs, on the base B-RMBM, p_i writes t to bus i, and the processors corresponding to S_i read and compare t to their local values.
- AnyResponder—After an associative search for IS_i on the base B-RMBM, any processor p_k with a match can write a 1 on bus i. Recall that this B-RMBM uses Common CW. Processor p_i reads bus i to determine whether any of its processors had a match.
- PickOne—The base B-RMBM can compute a maximum (or minimum) on the IDs of matching processors in $O(\log n)$ time to select one matching processor. As one alternative, a B-RMBM(n^2, n) can compute a maximum (or minimum) in $O(1)$ time as described above. As a second alternative, a B-RMBM(n, j) with Arbitrary or Priority CW can select one matching processor in $O(1)$ time. As a third alternative, a B-RMBM($n \log n$, $O(n)$) using Common, Collision, or Collision+ CW can select one matching processor in $O(1)$ time, using a PRAM technique due to Ragde [19].

Theorem 1 compiles these alternatives into a range of simulations of a MASC_v by a B-RMBM.

Theorem 1. *Each step of a* $\text{MASC}_v(n, j)$ *with constant time associative operations on* v*-bit data, can be simulated by:*

(i) *an* Arbitrary *or* Priority *CW B-RMBM(n, j) in* $O(v)$ *time,*
(ii) *a* Common, Collision, *or* Collision+ *CW B-RMBM(*$n \log n$, $O(n)$*) in* $O(v)$ *time,*
(iii) *a* Common, Collision, *or* Collision+ *CW B-RMBM(n, j) in* $O(v + \log n)$ *time, or*
(iv) *a B-RMBM(*n^2, $\max\{n, jv\}$*), where* $v \leq n$*, in* $O(\log v)$ *time.*

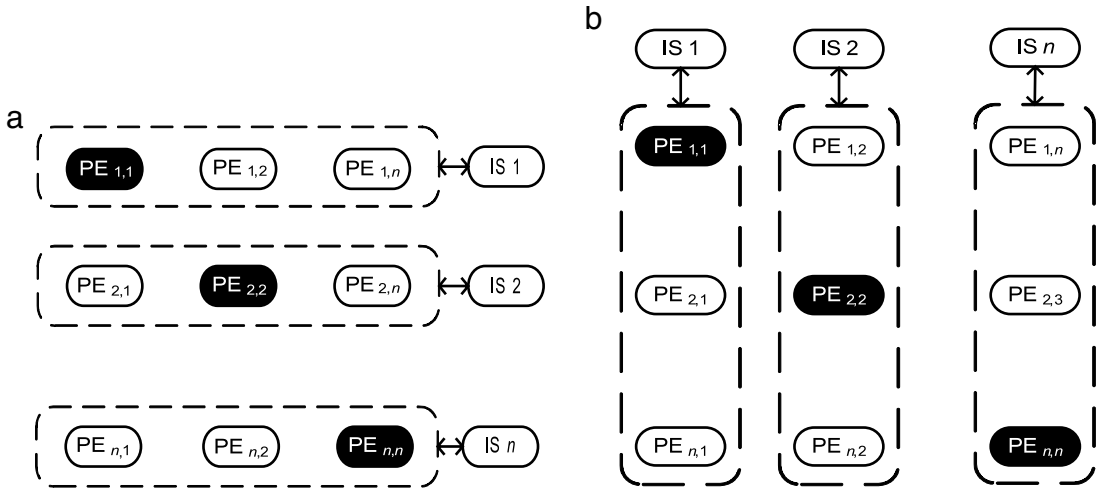

Fig. 3. PE grouping on the MASC to simulate a bus-access operation of the S-RMBM: (a) Row grouping; (b) Column grouping.

Remark: For constant v, the simulations of Theorem 1(i, iii, iv) run in $O(1)$ time.

Next, we consider simulating a B-RMBM using a MASC, specifically, a B-RMBM(n, m) with n processors and m buses on a MASC$_v(n, j)$ with n PEs and jISs. If $j \geq m$, let each MASC PE simulate one of the n B-RMBM processors and let MASC IS$_i$ simulate the ith B-RMBM bus for $1 \leq i \leq m$. Depending on how a B-RMBM processor sets the read (write) port to a bus in a particular step, a MASC PE listens (writes) to the corresponding simulating MASC IS. The MASC emulates concurrent reads from an RMBM bus by a broadcast by an IS. The MASC emulates concurrent writes to an RMBM bus by associative operations and the broadcast/reduction network. For example, for COLLISION CW on an RMBM bus, a MASC IS uses "AnyResponders" to make sure the set of PEs wishing to write is non-empty. If non-empty, use PickOne to select one PE from the set of PEs wishing to write and temporarily deactivate this PE. Use PickOne again to see if there is a second PE wishing to write. If a second PE is found, a special collision symbol will be broadcast to PEs wishing to read. If there is only one PE wishing to write, the data this PE holds will be broadcast to the PEs that wish to read. The MASC can also emulate COMMON, COLLISION+, and ARBITRARY CW in constant time. If the B-RMBM uses PRIORITY CW, however, it takes an extra $O(\lceil(\log n)/v\rceil)$ time to locate the particular PE ID number required by the priority rule.

If $j < m$, let IS$_1$ simulate B-RMBM buses $b_1, b_{j+1}, b_{2j+1}, \ldots$; IS$_2$ simulate B-RMBM buses $b_2, b_{j+2}, b_{2j+2}, \ldots$; and IS$_j$ simulate B-RMBM buses $b_j, b_{2j}, b_{3j}, \ldots$. Since each IS simulates at most $\lceil m/j\rceil$ buses, the MASC executes the simulation described above (for the case $j \geq m$) $\lceil m/j\rceil$ times. Thus, we have Theorem 2.

Theorem 2. *Each step of a B-RMBM(n, m) can be simulated by a* MASC$_v(n, j)$ *with constant time associative operations on v-bit data in:*

(i) $O(m/j)$ *time, if the simulated B-RMBM uses* COMMON, COLLISION, COLLISION+, *or* ARBITRARY *CW, or*

(ii) $O\left(\frac{m}{j} \cdot \left\lceil \frac{\log n}{v} \right\rceil\right)$ *time, if the simulated B-RMBM uses* PRIORITY *CW.*

Clearly, when $j = \Omega(m)$, it is $O(1)$ time for (i). *When $j = \Omega(m)$ and v is $\Omega(\log n)$, it is $O(1)$ time for* (ii).

4. Simulations between MASC and S-RMBM

Since an S-RMBM is a B-RMBM enhanced by adding to processors the ability of splitting a bus into bus segments, an S-RMBM with one bus can perform a PickOne operation with exclusive writes. Consequently, an $O(v)$ time simulation of MASC using S-RMBM(n, j) follows from Theorem 1.

We consider simulation of an S-RMBM using a MASC with v-bit associative operations. For an S-RMBM(n, m) with n processors and m buses, the simulation below takes $O((\log n)/v)$ time. When v is $\Omega(\log n)$, this is constant time.

The idea behind the simulation is that, since n S-RMBM processors can read in one step, regardless of the number of distinct bus segments, the MASC simulates only the segments read by the S-RMBM processors. Let S denote S-RMBM(n, m) with n processors and m buses. We simulate S using Q, a MASC(n^2, n) consisting of n^2 PEs and n ISs. Let p_i denote processor i of S, and let PE$_{i,j}$, for $1 \leq i, j \leq n$, denote a PE of Q, viewing the PEs as arranged in an $n \times n$ grid.

Q will use two groupings to assign PEs to the ISs. Under the *column grouping*, Q assigns the n PEs in the jth column to IS$_j$ (i.e., PE$_{i,j}$ for $1 \leq i \leq n$) and under the *row grouping*, Q assigns the n PEs in the ith row to IS$_i$ (i.e., PE$_{i,j}$ for $1 \leq j \leq n$) (See Fig. 3.) The blackened PEs are main PEs in their groups. The idea behind the column grouping is to use an IS for each processor p_j of S to broadcast any actions and value written by p_j. The idea behind the row grouping is to use an IS for each processor p_i of S to reduce the values written to the bus from which p_i reads and determine the specific value read by p_i. Observe PE$_{i,i}$ is in IS$_i$ under both groupings and is the main PE for simulating p_i. In particular, PE$_{i,i}$ stores p_i's data and executes its operations.

Each PE$_{i,j}$ in column j holds two m-bit arrays $L_{i,j}$ and $R_{i,j}$ in which entry k of $L_{i,j}$ ($R_{i,j}$) indicates whether p_j has set its left (right) segment switch on bus k. (We assume that each Q cell has at least $O(m)$ bits of memory.) Initially, each bit of these arrays is 0, indicating that no S segment switch is set.

In each step, each processor of S performs any of the following operations: set or reset one segment switch on a bus; move its read or write connection from one bus to another; write; read; and perform local computation. The key for Q is simulating reads. The idea here is that, for each reading processor p_i of S, an IS and its PEs in Q use arrays $L_{i,j}$ and $R_{i,j}$ to find the boundaries of the bus segment from which p_i reads and then evaluate writes only on that bus segment. Overall, Q simulates each operation in a step of S as follows.

Step 1. *Segment switch change*—Assign PEs to ISs using the column grouping. For each j, if p_j sets (resets) a switch on bus k, then IS$_j$ broadcasts the following to its PEs: the bus ID k, whether the action was to the left or to the right switch, and whether the action was to set or reset. Each PE$_{i,j}$ reads, then writes 1 (0) to bit k of array $L_{i,j}$ or $R_{i,j}$, as appropriate.

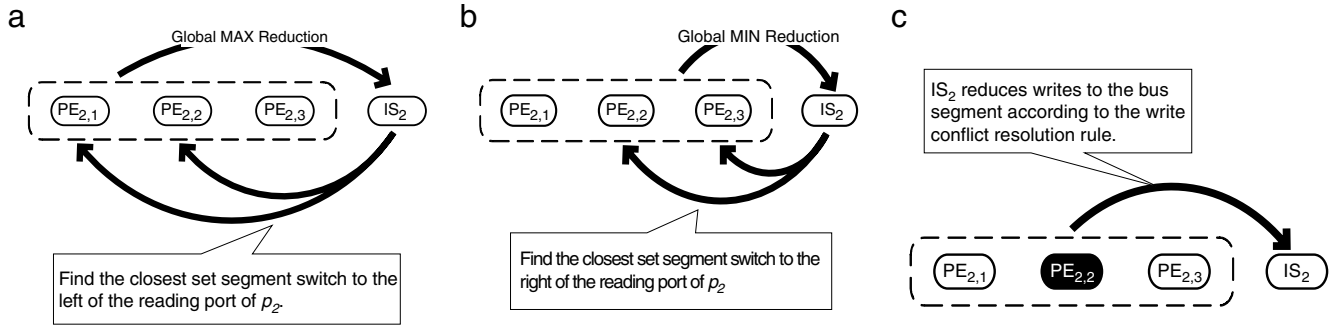

Fig. 4. Read step of simulating an S-RMBM on a MASC.

Step 2. *Move read or write connection*—For each j, if p_j moves its read connection from bus k to bus g, then IS_j broadcasts this information to its PEs (i.e., $PE_{i,j}$, for all i). Q acts similarly for moving write connections.

Step 3. *Write*—For each p_j that writes, IS_j broadcasts to its column of PEs the value from $PE_{j,j}$ that p_j writes.

Step 4. *Read*—Assign PEs to ISs using the row grouping. If p_i reads from bus g, then determine the location of the closest set segment switch to the left of the reading port of p_i as follows. IS_i broadcasts the bus ID g to $PE_{i,j}$, for all j. Next, for each $k \leq i$, if p_k has set a segment switch on bus g (that is, if entry g of $L_{i,k}$ is 1), then $PE_{i,k}$ sends k to IS_i, which stores the maximum value written (Fig. 4(a)). Determine similarly the location of the closest set segment switch to the right of the reading port of p_i (Fig. 4(b)). Next, for each p_k that writes on bus g between the set segment switches, $PE_{i,k}$ sends to IS_i the value written by p_k (Fig. 4(c)). In the event of concurrent writes on S, IS_i computes the bus data using associative operations and PickOne appropriate to the write conflict resolution rule of S.

Fig. 4 illustrates this step for the case of p_2 reading from bus g.

Step 5. *Local computation*—Each IS_i performs the local computation of p_i.

Assuming bus width $O(\max\{\log m, \log n\})$, Steps 1 and 2 take constant time. Steps 3 and 5 take constant time. Using MASC associative operations to obtain the max/min value of processor IDs and resolve write conflicts in Step 4 takes $O((\log n)/v)$ time. Therefore, the total time for the simulation is $O((\log n)/v)$.

Notice that we can use the same constructed MASC to simulate a B-RMBM(n, m), since the B-RMBM is a special case of the S-RMBM without any segment switch set on processors. Thus, we have Theorem 3 and Corollary 4 as follow.

Theorem 3. *An S-RMBM(n, m) can be simulated by a* $\text{MASC}_v(n^2, n)$ *in* $O((\log n)/v)$ *time if the MASC can perform* v*-bit width associative operations in constant time.*

Corollary 4. *A B-RMBM(n, m) can be simulated by a* $\text{MASC}_v(n^2, n)$ *in* $O((\log n)/v)$ *time if the MASC can perform* v*-bit width associative operations in constant time.*

5. Simulations between MASC and F-RMBM, E-RMBM

Since both F-RMBM and E-RMBM have more capabilities than the B-RMBM, obviously either F-RMBM(n, j) or E-RMBM(n, j) can simulate a step of a $\text{MASC}_v(n, j)$ in $O(v)$ time, without setting fuse switches or segment switches.

We now discuss simulating an F-RMBM using a MASC. The idea of the algorithm is for the MASC to compute the buses that can reach each fuse line by an approach similar to transitive closure by recursive doubling. Let Z denote an F-RMBM(n, m) with n processors and m buses and Q denote a $\text{MASC}(n^2, n)$ with n^2 PEs and n ISs. We use the same row and column groupings as in Section 4 to assign PEs to ISs in Q.

Each $PE_{i,j}$ holds two m-bit strings $F_{i,j}$ and $C_{i,j}$ in which each bit f of $F_{i,j}$ indicates whether p_j has set its fuse switch on bus f and each bit c of $C_{i,j}$ indicates whether buses that p_i has fused are connected to bus c. Initially, each bit of $F_{i,j}$ and $C_{i,j}$ is 0 indicating no switches are set. Note that a processor of Z can have multiple fuse switches in the set position at the same time but can have at most one read or write switch set to one bus at the same time.

In each step, each Z processor performs the same operations as stated in Section 4. Q simulates each of these operations as follows (since Steps 2, 3, and 5 are identical to those in Section 4, the details are omitted here).

Step 1. *Set/reset a fuse switch*—Assign PEs to ISs using the column grouping. For each j, if p_j sets or resets the fuse switch on bus h, then IS_j broadcasts to its PEs (i.e., $PE_{i,j}$, for all i) the bus ID h and whether the action was set or reset. Each $PE_{i,j}$ reads and updates string $F_{i,j}$.

Step 2. *Move read or write connection*

Step 3. *Write*

Step 4. *Read*—Assign PEs to ISs using the row grouping. If p_i reads from bus g, then determine the group of fused buses that includes bus g. Q performs the following steps to check common fused buses. All ISs have their PEs copy $F_{i,j}$ to $C_{i,j}$. Execute Steps 4.1–4.3 for $\log m$ iterations to flag in C_i the buses connected to buses fused by p_i.

Step 4.1. Reassign PEs to ISs using the row grouping. Each PE sets $flag_{i,j}$ to 0. Each IS_i broadcasts $C_{i,i}$ to $PE_{i,j}$. Each $PE_{i,j}$ computes $C_{i,i}$ AND $C_{i,j}$. If at least one bit of the result is 1, then $PE_{i,j}$ sets $flag_{i,j} = 1$. If $flag_{i,j} = 0$, then deactivate $PE_{i,j}$.

Step 4.2. For $k = 1$ to $\lceil m/v \rceil$, repeat the following. IS_j computes the global OR of the kth block of v bits of $C_{i,j}$ of all active $PE_{i,j}$, and each $PE_{i,i}$ stores the result in the kth block of v bits of $C_{i,i}$.

Step 4.3. ISs activate all PEs deactivated in Step 4.1, if any. Reassign PEs to ISs using the column grouping. Each IS_j broadcasts $C_{j,j}$ to $PE_{i,j}$ and $PE_{i,j}$ copies this data to $C_{i,j}$.

Next, execute Step 4.4 to use the bus connectivity information generated in Steps 4.1–4.3 to collect and reduce data written to the buses connected to bus g from which a processor p_i reads.

Step 4.4. Reassign PEs to ISs using the row grouping. IS_i broadcasts g, the index of the bus from which p_i wishes to read, to all $PE_{i,j}$. Each $PE_{i,j}$ checks if bit g of $C_{i,j}$ is 1. Use a PickOne operation to select one $PE_{i,h}$ satisfying this condition. IS_i broadcasts $C_{i,h}$ from this PE to all $PE_{i,j}$. For $PE_{i,j}$, if p_j writes to bus e and bit e of $C_{i,h}$ is 1, then $PE_{i,j}$ is active; otherwise inactive. (If no $PE_{i,h}$ has bit g of $C_{i,h}$ as 1, then IS_i broadcasts this result to all $PE_{i,j}$. For $PE_{i,j}$, if p_j writes to bus g, then $PE_{i,j}$ is active; otherwise inactive.) Active processors write and, in the event of concurrent writes on Z, IS_i computes the bus data using associative operations and PickOne appropriate to the write conflict resolution rule of Z.

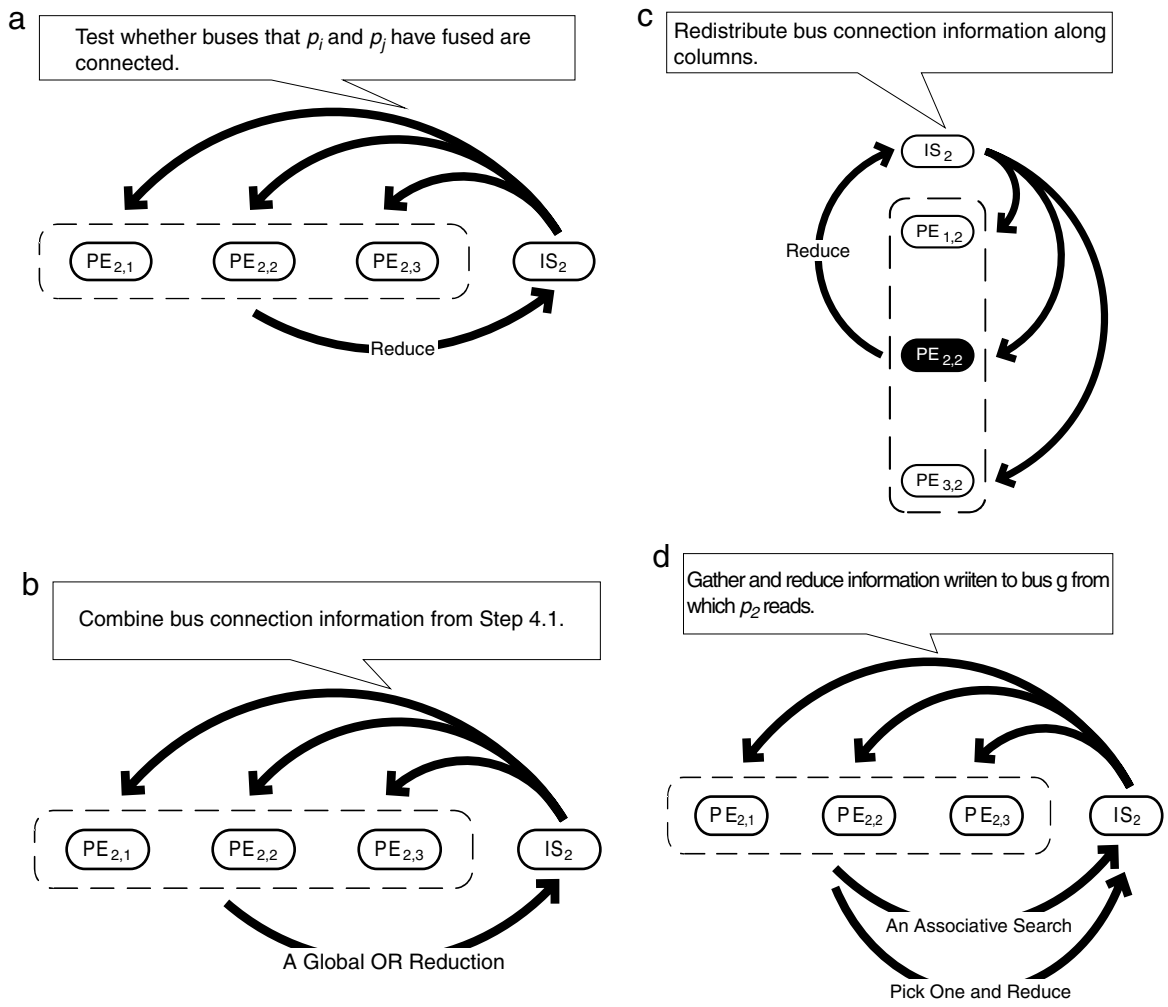

Fig. 5. Illustration of Step 4—(a) Step 4.1 (b) Step 4.2 (c) Step 4.3 (d) Step 4.4.

Step 5. *Local Computation*

Fig. 5 illustrates the operations in Step 4 for a situation in which p_2 reads from bus g and $m = 2$.

Thus, we have the following theorem.

Theorem 5. *Each step of an F-RMBM(n, m) can be simulated by a MASC(n^2, n) in $O(\lceil m/v \rceil \log m)$ time, if the MASC can perform v-bit width associative operations in constant time.*

Proof. At the start of each simulated step, each $PE_{i,i}$ holds $F_{i,i}$ with all the fuse switches set by p_i. Steps 1–3 distribute information on set fuse switches, the positions of read and write connections, and the value written by p_i to all $PE_{i,j}$. Steps 4.1–4.3, over $\log m$ iterations, calculate in $C_{i,i}$ the buses that can reach the fuse line of p_i in a manner much like a recursive doubling approach to transitive closure. A $PE_{i,j}$ is active after Step 4.1 if its $C_{i,j}$ and $C_{i,i}$ have a 1 in a common position, that is, if the set of buses known to reach the fuse line of p_i and the set of buses known to reach the fuse line of p_j have a common element. If they have a common element, then Step 4.2 combines the two sets of buses because they can all reach each other. Step 4.3 distributes the updated information to be in the proper places for the next iteration. At the start of iteration q, each $C_{i,i}$ holds the set of buses that can reach the fuse line of p_i by paths via up to 2^{q-1} fuse lines, and at the end of iteration q, it holds the buses that can reach the fuse line of p_i by paths via up to 2^q fuse lines. At the start of Step 4.4, $C_{i,i}$ holds connectivity information stored according to fuse line index. Step 4.4 extracts connectivity information according to bus index, handles multiple writes, and delivers data that processors of Z would read.

Steps 1, 2, 3, and 5 run in constant time. The time complexity of the algorithm depends on Step 4. Let v denote the data width on which an IS can perform an associative operation in constant time, and let $b \geq v$ denote the bus width. Assume b is at least $\max\{\log m, \log n\}$. Each iteration of Steps 4.1 and 4.3 runs in $O(m/b)$ time, and each iteration of Step 4.2 runs in $O(\lceil m/v \rceil)$ time. The one execution of Step 4.4 runs in $O(m/b)$ time. Since $b \geq v$, Step 4 runs in $O(\lceil m/v \rceil \log m)$ time, and this is also the time complexity of the overall algorithm. □

Note: If the F-RMBM uses PRIORITY CW, then each IS computes bus data for Step 4.4 in $O((\log n)/v)$ time; otherwise, each IS computes bus data in constant time.

To simulate the E-RMBM, we use a simulation of an E-RMBM on an F-RMBM.

Theorem 6. *Each step of an E-RMBM(n, m) can be simulated by a MASC($(n + 2m)^2$, $4nm$) in $O(\lceil m/v \rceil \log m)$ time, if the MASC can perform v-bit width associative operations in $O(1)$ time.*

Proof. An F-RMBM($n + 2m$, $4nm$) can simulate each step of an E-RMBM(n, m) in $O(1)$ time [22]. The result follows from Theorem 5. □

6. Relating $MASC_c$ and $MASC_{\log n}$

So far, simulations in Sections 3–5 have specified that the MASC can perform constant time associative operations on v-bit wide data, where v is a variable. If the MASC processes data in a bit-serial manner [10], then constant time operations would imply a constant value for v. When v is a function of n, for example, $\Theta(\log n)$ in Theorems 2 and 3, then it appears that the assumption of constant time associative operations for a $MASC_{\log n}$ is overly generous. In this section, however, we establish that a constant time simulation of $MASC_{\log n}$ by $MASC_c$ exists when the number of PEs increases by a polynomial factor. Consequently, associative operations can be performed in constant time on $MASC_{\log n}$ as well.

We will present a cycle of simulations linking $MASC_c$ and $MASC_{\log n}$. To start, Lemma 7 describes a constant time simulation of a PRAM by a B-RMBM.

Lemma 7 ([22]). *Each step of a PRAM(n, s) can be simulated in $O(1)$ time on a B-RMBM($n + s$, s), where the PRAM and B-RMBM have the same concurrent or exclusive read and write capabilities.*

Next, recall Theorem 2(i)with an $O(m/j)$ time simulation of a B-RMBM by a $MASC_c$, where m is the number of buses on the B-RMBM and j is the number of ISs on the $MASC_c$. When $j = \Theta(m)$, this is constant time. Clearly, a $MASC_c$ can be simulated on a $MASC_{\log n}$. Stringing these results together, a PRAM can be simulated on a B-RMBM which can be simulated on a $MASC_c$ which can be simulated on a $MASC_{\log n}$. The next result establishes that a $MASC_{\log n}$ can be simulated on a PRAM, closing the cycle.

Lemma 8. *Each step of a $MASC_{\log n}(n, j)$ can be simulated in $O(1)$ time on a* Common *CRCW PRAM(n^2, $n + j$).*

Proof. Let S denote a $MASC_{\log n}(n, j)$. We will construct a Common CRCW PRAM(n^2, $n + j$) R that will simulate each step of S in constant time. S can perform constant time associative operations on $\log n$-bit wide data; let the data width of R be $\Theta(\log n)$ bits, a standard assumption for PRAMs. Each processor of R can perform the local processor actions of the corresponding processor of S, so we just need to focus on the associative operations.

Let $p_{g,h}$ denote a processor of R, where $0 \leq g, h < n$. For IS_i with k PEs in the set $\{i_1, i_2, \ldots, i_k\}$, let $T(i)$ denote the *team* of k^2 processors of R that will simulate the associative operation of IS_i; this team comprises processors $p_{g,h}$ such that $g, h \in \{i_1, i_2, \ldots, i_k\}$. Let $TS(i)$ denote the *small team* of k processors of R that will simulate some associative operations of IS_i; this team is the subset of $T(i)$ comprising processors $p_{g,g}$ such that $g \in \{i_1, i_2, \ldots, i_k\}$. R has one shared memory cell per PE, denoted cp_g, for $0 \leq g < n$, and one shared memory cell per IS, denoted cis_i, for $0 \leq i < j$.

Each IS of S may simultaneously perform any associative or computational operation independent of other ISs in the same step. We now describe the simulation of the associative operations of IS_i with $k \leq n$ PEs where each PE holds an operand.

- OR, AND—$TS(i)$ can compute the OR or AND of k binary values in constant time using shared memory cell cis_i with Common CW.
- Maximum, Minimum—$T(i)$ can compute the maximum or minimum of k values, each up to $\log n$ bits wide, in constant time using Common CW. As in the proof of Theorem 1, the idea for maximum is to compare all pairs of elements and use AND operations to identify the maximum value [15,21].
- Associative search—$TS(i)$ can perform an associative search in constant time using shared memory cell cis_i with CREW. Write the search value in cis_i, then each processor in $TS(i)$ can read IS_i's value and compare that to its PE's value.
- Any Responder—$TS(i)$ can perform an AnyResponder operation in constant time using shared memory cell cis_i with Common CW. $TS(i)$ computes an OR where those processors that match (do not match) the search contribute 1 (0).
- Pick One—$T(i)$ can perform a PickOne operation in constant time by computing maximum or minimum among IDs of PEs that matched an associative search.

Observe that for any two ISs, IS_i and IS_f, $T(i)$ and $T(f)$ are disjoint and the shared memory cells that they use are disjoint, so R can independently and simultaneously perform the associative operations for all ISs. □

From this cycle and other previously known results, we have the following.

Theorem 9. *Each of the following models can simulate a step of any other in constant time with at most a polynomial factor increase in hardware resources:* PRAM, S-RMBM, B-RMBM, $MASC_c$, *and* $MASC_{\log n}$.

Based on this theorem, we will be able to use the terms $MASC_c$ and $MASC_{\log n}$ interchangeably in most cases if no confusion is caused.

7. Extension of relationships among MASC, PRAM, and RM

In [6,22,24], relative powers of the PRAM, RMBM, and RM models have been well studied. Related models can be placed into two groups, i.e.,

Group 1: B-RMBM, S-RMBM, PRAM

Group 2: F-RMBM, E-RMBM, RM, LRM.

All models in the same group have the same power. Any model in Group 2 is more powerful that any model in Group 1.

Based on these relationships and the simulation results that we have presented, we have the following observations. Note that all cases are CRCW.

(1) Since constant time simulations between MASC and B-RMBM and constant time simulations between MASC and S-RMBM exist, MASC goes into the first group in terms of its power. MASC has the same power as PRAM.
(2) The MASC used throughout the paper is a MASC without cell and IS networks. Akl [1] found that a PRAM can simulate any interconnection network as well as any combinational circuit in constant time. From the observation (1) and the fact that cell and IS networks can be simulated by a Common CRCW PRAM in constant time, this MASC model is equivalent in power to a general MASC model that has both cell and IS networks.
(3) When applying a special case of Heide and Pham's model [14] to an MMB, regardless of relative values of number n of processors and m of buses, MMB needs $\Omega(\log \log n)$ time to compute the maximum. As MASC can do this in constant time, MMB does not belong in Group 1.
(4) Since MASC has the same power as the models in Group 1, it is less powerful than RM, LRM, E-RMBM, and F-RMBM.
(5) It can be shown that S-RMBM has the same power as BRM, a basic RM (the proof is omitted). Although our previous BRM simulation of MASC [3] takes non-constant time, a constant time BRM simulation using MASC is possible.

Therefore, our modified grouping to the above relationships is as follows.

Group 1: MASC, B-RMBM, S-RMBM, PRAM, BRM

Group 2: F-RMBM, E-RMBM, RM, LRM

8. Conclusion

We have presented simulations between MASC and versions of RMBM—reconfigurable bus-based models. The MASC and the basic RMBM can simulate each other in constant time with appropriate assumptions. Also, MASC can simulate the segmenting RMBM in $O((\log n)/v)$ time and the fusing RMBM and extended RMBM in $O(\lceil m/v \rceil \log m)$ time. By taking advantage of established

relationships between RMBM, RM, and PRAM, we have categorized the power of the MASC model in relation to those of RM and CRCW PRAM. The power of the MASC model is comparable to that of CRCW PRAM but less than that of RM. For some special cases of RM with restrictions, however, a MASC can perform a constant time simulation (e.g., simulations of MMB and BRM). Both RM and PRAM are very popular and important models for parallel computation; especially RM has been widely accepted as an extremely powerful parallel computation model. Our research results provide a better understanding of the MASC model concerning its power.

Acknowledgment

This work was supported in part by the National Science Foundation under grant number CCR-0310916.

References

[1] S.G. Akl, Parallel Comput.: Models and Methods, Prentice Hall, New York, 1997.
[2] M. Atwah, J. Baker, S. Akl, An associative implementation of classical convex hull algorithms, in: Proc. of the 8th IASTED on Parallel and Distributed Computing Systems, PDCS, 1996, pp. 435–438.
[3] J. Baker, M. Jin, Simulation of enhanced meshes with MASC, a MSIMD model, in: Proc. of the 11th IASTED PDCS, 1999, pp. 511–516.
[4] Y. Ben-Asher, D. Peleg, R. Ramaswami, A. Schuster, The power of reconfiguration, J. Parallel Distrib. Comput. 13 (1991) 139–153.
[5] W. Chantamas, J. Baker, M. Scherger, Compiler extension of the ASC language to support multiple instruction streams in the MASC model using the manager-worker paradigm, in: Proc of the PDPTA 2006, Las Vegas, June 2006.
[6] C.A. Córdova-Flores, J.A. Fernández-Zepeda, A.G. Bourgeois, Constant time simulation of an R-mesh on an LR-mesh, in: Proc. of the 21st IPDPS (Workshop on APDCM), CD-ROM, March 2007.
[7] M. Esenwein, J. Baker, VLCD string matching for associative computing and multiple broadcast mesh, in: Proc. of the 9th PDCS, 1997, pp. 69–74.
[8] A.D. Falkoff, Algorithms for parallel-search memories, J. ACM (JACM) 9 (4) (October 1962) 488–511.
[9] J. JaJa, An Introduction to Parallel Algorithms, Addison Wesley, 1992.
[10] M. Jin, J. Baker, K. Batcher, Timings of associative operations on the MASC model, in: Proc. of the 15th IPDPS (WMPP), April 2001.
[11] M. Jin, J. Baker, On the power of the Multiple Associative Computing (MASC) Model related to that of reconfigurable bus-based models, in: Proc. of the 21st IPDPS (Workshop on APDCM), CD-ROM, Long Beach, California, March 2007.
[12] L. Kucera, Parallel computation and conflicts in memory access, Inform. Process. Lett. 14 (1982) 93–96.
[13] W.C. Meilander, J.W. Baker, M. Jin, Importance of SIMD computation reconsidered, in: Proc. of the 17th IPDPS (WMPP), April 2003.
[14] F.M. Heide, H.T. Pham, On the performance of networks with multiple busses, in: Proc. of Symposium On Theoretical Aspects of Computer Science (STACS '92), in: Lecture Notes in Comput. Sci., vol. 577, Springer, Berlin, 1992, pp. 97–108.
[15] R. Miller, V.K. Prasanna-Kumar, D. Reisis, Q. Stout, Parallel computations on reconfigurable meshes, IEEE Trans. Comput. 42 (1993) 678–692.
[16] S. Olariu, J. Schwing, J. Zhang, On the power of two-dimensional processor arrays with reconfigurable bus systems, Parallel Process. Lett. 1 (1) (1991) 29–34.
[17] J. Potter, Associative Computing: A Programming Paradigm for Massively Parallel Computers, Plenum Press, New York, 1992.
[18] J. Potter, J. Baker, S. Scott, A. Bansal, C. Leangsuksun, C. Asthagiri, ASC: An associative-computing paradigm, Computer 27 (11) (1994) 19–25.
[19] P. Ragde, Processor-time tradeoffs in PRAM simulations, J. Comput. Syst. Sci. 44 (1992) 103–113.
[20] K. Schaffer, R.A. Walker, A prototype multithreaded associative SIMD processor, in: Proc. of the 21st IPDPS (Workshop on APDCM), Long Beach, California, March 2007.
[21] Y. Shiloach, U. Vishkin, Finding the maximum, merging, and sorting in a parallel computation model, J. Algorithms 2 (1981) 88–102.
[22] J.L. Trahan, R. Vaidyanathan, R.K. Thiruchelvan, On the power of segmenting and fusing buses, J. Parallel and Distributed Computing 34 (1) (1996) 82–94.
[23] D. Ulm, J. Baker, Simulating PRAM with a MSIMD model (ASC), in: Proc. of the Int'l. Conf. on Parallel Processing, 1998, pp. 3–10.
[24] R. Vaidyanathan, J.L. Trahan, Dynamic Reconfiguration: Architectures and Algorithms, Kluwer Academic/Plenum Publishers, New York, 2004.
[25] B.-F. Wang, G. Chen, Two-dimensional processor array with reconfigurable bus system is at least as powerful as CRCW model, Inform. Process. Lett. 36 (1990) 31–36.
[26] H. Wang, R.A. Walker, Implementing a multiple-instruction stream associative MASC processor, in: Proc. of the 18th IASTED PDCS, Dallas, Texas, 2006, pp. 460–465.

Jerry L. Trahan received his B.S. from Louisiana State University in 1983 and his M.S. and Ph.D. from the University of Illinois at Urbana-Champaign in 1986 and 1988, respectively. Since 1988, he has been a faculty member in the Department of Electrical and Computer Engineering, Louisiana State University, where he is currently an Associate Professor. His research interests include reconfigurable computing, theory of computation, and models of parallel computation. He has served as a reviewer for numerous journals, and as a program committee member of several workshops and conferences.

Mingxian Jin is currently an Assistant Professor in the Department of Mathematics and Computer Science at Fayetteville State University, North Carolina. She received her Ph.D. degree in computer science from Kent State University in 2004. She also received her B.S. in computer science from Wuhan University, China and M. Eng. in computer engineering from North China Institute of Computing Technology, Beijing, China. Her research interests include parallel processing, associative SIMD computing, parallel computational models and algorithms, real-time systems.

Wittaya Chantamas is currently a Ph.D. candidate in computer science at Kent State University. His research interests include parallel and distributed algorithms, modeling, and simulation; associative SIMD and multi-SIMD computing; and massively parallel architectures. Chantamas received an MS in computer science in 1999 from Syracuse University and a BS in Mathematics in 1996 from Kasetsart University in Thailand. He is a member of the IEEE.

Johnnie W. Baker received his B.A. degree from Hardin-Simmons University and his M.S. and Ph.D. degree from the University of Texas at Austin. He was a faculty member at Florida State University prior to joining the faculty at Kent State University in 1973. In addition to publications in computer science, he has published in mathematics, computational chemistry, and bioinformatics. Baker's current research interests in parallel computing include parallel algorithms, parallel modeling and simulation, associative SIMD and multi-SIMD computing, massively parallel architectures, real-time systems, and SIMD real-time air traffic control. He has refereed for numerous conferences and journals, and served as an editor for Parallel Processing Letters for over 15 years. He is a member of both ACM and IEEE Computer Society.