Relating Two-Dimensional Reconfigurable Meshes with Optically Pipelined Buses

Anu G. Bourgeois and Jerry L. Trahan
Department of Electrical & Computer Engineering
Louisiana State University, Baton Rouge, Louisiana, 70803
{anu, trahan}@ee.lsu.edu

Abstract

Recently, many models using reconfigurable optically pipelined buses have been proposed in the literature. We present simulations for a number of these models and establish that they possess the same complexity, so that any of these models can simulate a step of one of the other models in constant time with a polynomial increase in size. Specifically, we determine the complexity of three optical models (the PR-Mesh, APPBS, and AROB) to be the same as the well known LR-Mesh and the cycle-free LR-Mesh.

1. Introduction

A system with an optically pipelined bus uses optical waveguides of unidirectional propagation and predictable delays instead of electrical buses to transfer information among processors. These two properties enable synchronized concurrent access to an optical bus in a pipelined fashion [10]. Combined with the abilities of the bus structure to broadcast and multicast, this architecture suits many communication-intensive applications.

Several similar models exist with "optically pipelined buses," including the *Linear Array with a Reconfigurable Pipelined Bus System* (LARPBS) [7], the *Linear Pipelined Bus* (LPB) [6], the *Pipelined Optical Bus* (POB) [14], the *Linear Array with Pipelined Optical Buses* (LAPOB) [2], the *Pipelined Reconfigurable Mesh* (PR-Mesh) [12], the*Array with Reconfigurable Optical Buses* (AROB) [8], *Array Processors with Pipelined Buses* (APPB) [5], the *Array Processors with Pipelined Buses using Switches* (APPBS) [3], the *Array with Synchronous Optical Switches* (ASOS) [10], and the *Reconfigurable Array with Spanning Optical Buses* (RASOB) [9]. Section 2 describes the PR-Mesh and LARPBS as representatives of these models.

Many of the optically pipelined models proposed have different features, making it difficult to relate results from one model to another. It is a useful endeavor, therefore, to unify these models in order to increase understanding of which features are essential and to be able to translate algorithms from one model to another. In an earlier paper [11], we determined the equivalence of three one-dimensional reconfigurable optical models: the LARPBS, LPB, and POB. This result implies an automatic translation of algorithms (without loss of speed or efficiency) among these models. In this paper we consider two-dimensional models. This presents obstacles not present when analyzing linear arrays, such as the larger number of configurations possible due to the multiple dimensions. To account for this, we establish their equivalence in a slightly different context; here we consider their complexity by relating their time to within a constant factor and the number of processors to within a polynomial factor.

We have established that the PR-Mesh has the same complexity as the cycle-free *Linear Reconfigurable Network* (LR-Mesh) [12]. In this paper we prove that in constant time using a polynomial number of processors the cycle-free LR-Mesh can solve the same class of problems as the LR-Mesh (Section 3.1). This result implies that the PR-Mesh can solve the same class of problems within the same order of steps using polynomial processors. We extend this complexity class to include two other optical models, namely the AROB and APPBS in Section 3. Our results are some of the first to unify reconfigurable optical models and relate them to other more widely known models.

2. PR-Mesh description

Let an optically pipelined bus have the same length of fiber between consecutive processors, so propagation delays between consecutive processors are the same; we

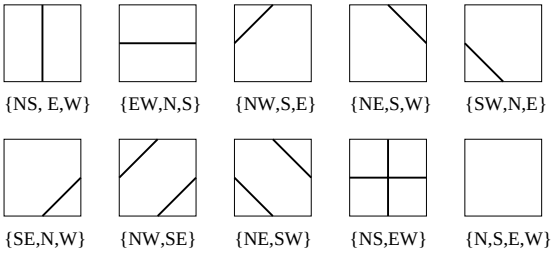

Figure 1. Internal port connections

refer to this delay as one *petit cycle*. Let a *bus cycle* be the end-to-end propagation delay on the bus. We specify time complexity in terms of a step comprising one bus cycle and one local computation. For more details on the time complexity issue, see Guo *et al.* [4] and Pan and Li [7].

The *Pipelined Reconfigurable Mesh* (PR-Mesh) [12] is a k-dimensional mesh of processors in which each processor has $2k$ ports. Although we will use 3-dimensional models, we can map the models to two dimensions and maintain the number of processors to be polynomial in N [13]. Each processor can locally configure its port connections so that each port can fuse with at most one other port, so that all buses formed are linear. Figure 1 depicts the ten possible port partitions for a two-dimensional PR-Mesh. Local fusing creates buses that run through their fused ports to adjacent processors, then through their fused ports, and so on. Each such linear bus corresponds to an LARPBS [7], a one-dimensional version of the PR-Mesh, which we now describe.

2.1. Structure of a PR-Mesh linear bus

In the LARPBS, as described by Pan and Li [7], the optical bus is composed of three waveguides, one for carrying data (the *data waveguide*) and the other two (the *reference* and *select waveguides*) for carrying address information (see Figure 2). (For simplicity, the figure omits the data waveguide, as it resembles the reference waveguide.) Each processor connects to the bus through two directional couplers, one for transmitting and the other for receiving [4, 10]. The receiving segments of the reference and data waveguides contain an extra segment of fiber of one unit pulse-length, Δ, between each pair of consecutive processors (shown as a delay loop in Figure 2). The transmitting segment of the select waveguide also has a switch-controlled conditional delay loop of length Δ between processors R_i and R_{i+1}, for each $0 \leq i \leq N-2$ (Figure 2).

To allow segmenting, the LARPBS has optical switches on the transmitting and receiving segments

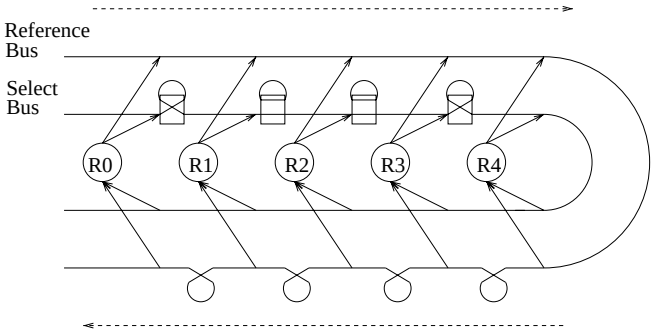

Figure 2. Structure of an LARPBS

of each bus for each processor. With all switches set to *straight*, the bus system operates as a regular pipelined bus system. Setting the switches at R_i to *cross* segments the whole bus system into two separate pipelined bus systems, one consisting of processors $R_0, R_1, \cdots, R_i$ and the other consisting of R_{i+1}, $R_{i+2}, \cdots, R_{N-1}$.

2.2. Addressing techniques

The LARPBS uses the *coincident pulse technique* [10] to route messages by manipulating the relative time delay of *select* and *reference* pulses on separate buses so that they will coincide only at the desired receiver. Each processor has a *select frame* of N bits (*slots*), of which it can inject a pulse into a subset of the N slots. The coincident pulse technique admits broadcasting and multicasting of a single message by appropriately introducing multiple select pulses within a select frame.

When multiple messages arrive at the same processor in the same bus cycle, it receives only the first message and disregards subsequent messages that have coinciding pulses at the processor. (This corresponds to priority concurrent write.)

3. Relating two-dimensional optical models

Given the number of algorithms developed on reconfigurable models and the growing body of research on them, it is important to relate these models to each other and to more widely known models. We first define some terminology prior to presenting the results.

For model Z, let $Z(T, \text{poly}(N))$ denote the class of languages accepted by model Z in $O(T)$ steps with polynomial in N processors. The class L is the class of languages accepted by deterministic Turing machines with work space bounded by $\log N$.

3.1. Complexity of the PR-Mesh

The *Linear Reconfigurable Network* (LR-Mesh) [1] has a mesh structure and each processor can locally configure its port connections as in a PR-Mesh. The difference is that it uses electronic buses instead of optical buses. Thus, it is not able to pipeline messages. However, a value written on a port reaches all ports connected to the same bus in one time step.

In a prior paper, we established that the PR-Mesh has the same complexity as the *cycle-free* LR-Mesh that is, all buses are linear and without cycles [12]. Due to the U-turn structure of the PR-Mesh buses, cycles are not allowed; it is necessary to separate the transmitting segment from the receiving segment.

Ben-Asher *et al.* [1] established $L = \textit{LR-Mesh}(1, \text{poly}(N))$ using an LR-Mesh that allows cycles. They used the decision problem CYCLE, which is complete for L with respect to NC^1 reductions.

Definition 1 *[1]* CYCLE *is the following decision problem. The input is a permutation on N vertices, that is, a directed graph of out-degree 1 (given by its adjacency matrix), with two special vertices u and v. The answer is '1' if u and v are on the same cycle.*

To solve the CYCLE problem, Ben-Asher *et al.* devised the following algorithm. Let each processor of an $N \times N$ LR-Mesh hold one bit of the input adjacency matrix. Assume that vertex i maps to j and j maps to vertex k. After a series of communication steps, all processors in column j hold the IDs of predecessor i and successor k. Processors then create a linear bus between adjacent vertices, so that each cycle in the input permutation induces a cycle in the LR-Mesh. Processor u writes a message on its cycle, and v receives the message if the two are on the same cycle.

This gives the following LR-Mesh solution to any problem Π in L: simulate the NC^1 circuit transforming the instance of Π to an instance of CYCLE, then solve the resulting instance of CYCLE. Ben-Asher *et al.* also developed a simulation of the NC^1 circuit (without the use of cycles), establishing $L \subseteq \textit{LR-Mesh}(1, \text{poly}(N))$. They further proved that $\textit{LR-Mesh}(1, \text{poly}(N)) \subseteq L$, thereby obtaining $L = \textit{LR-Mesh}(1, \text{poly}(N))$.

We aim to prove that *cycle-free* $\textit{LR-Mesh}(1, \text{poly}(N)) = L$. We use an $O(N) \times O(N) \times O(N)$ cycle-free LR-Mesh to solve CYCLE, and thus establish the same complexity. The approach we take is similar to that of Ben-Asher *et al.*, mapping the given adjacency matrix to the bottom layer $O(N) \times O(N)$ LR-Mesh and after a series of communication steps, all processors in the j^{th} column hold the IDs of the vertices immediately before and after vertex j in the permutation. Embedding permutation graph edges uses the third dimension of the cycle-free LR-Mesh, as described below.

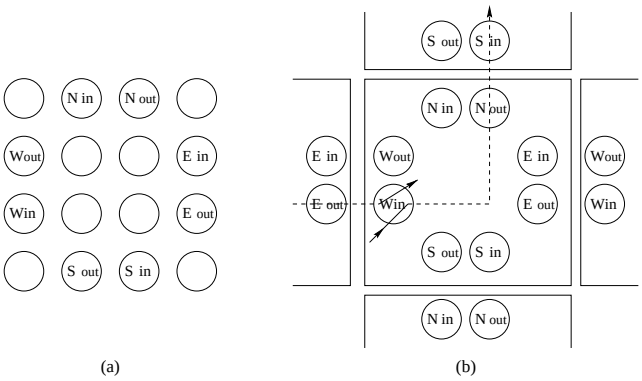

Figure 3. Block of 4×4 processors for simulations: (a) labeling of processors within blocks; (b) arrangement of blocks and connections.

The LR-Mesh has N layers of $O(N) \times O(N)$ processors, where each layer can be broken down into 4×4 blocks of processors, as shown in Figure 3. Label eight of the processors within each block as "in" or "out" to represent the direction of the permutation mapping, although the cycle-free LR-Mesh is undirected. Let $block(i, j)$ denote the block in the i^{th} row and j^{th} column of blocks, where $0 \leq i, j < N$. The blocks on the diagonal represent the vertices.

We create linear buses, one bus corresponding to each vertex, such that the buses extend up the layers of the mesh. Bus connections are identical in each layer, and depend on the permutation. For each vertex j with successor vertex k, in each layer, a bus connects $block(j, j)$ via $block(k, j)$ to $block(k, k)$ within the layer, then steps up to $block(k, k)$ in the next layer. This bus exits $block(j, j)$ from N_{out} or S_{out} and enters $block(k, k)$ from E_{in} or W_{in}, depending on the relative values of j and k. The "in" port also routes this connection up to $block(k, k)$ in the layer above. The bus coming from the layer below also enters at the same "in" port processor, and is configured to connect to the vertical bus leaving $block(k, k)$. Figure 3(b) shows the connections for a block whose predecessor reaches it via a block from its left, and successor corresponds to some row above. (Connections shown as dashed lines are all within the same layer. Connections shown as solid lines run either to the layer above or from the layer below.)

To determine if vertices u and v are on the same cycle, let $block(u, u)$ in layer 0 write on its bus. If v is on a cycle with u, then $block(v, v)$ on some layer will

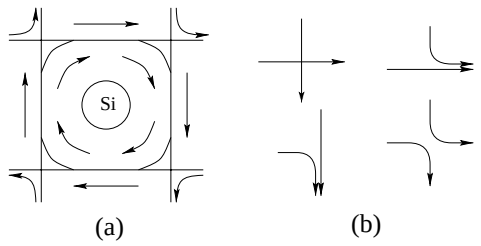

Figure 4. APPBS processor with switches: a) switch connections at each APPBS processor; b) switch configurations of top right switch at each APPBS processor.

receive the message from $block(u,u)$, indicating a '1' answer to the CYCLE decision problem.

Theorem 1 *Cycle-free LR-Mesh*$(1, \text{poly}(N)) = L$.

Corollary 2 *PR-Mesh* $\left(\log^j N, \text{poly}\,(N)\right)$
$=$ *cycle-free LR-Mesh* $\left(\log^j N, \text{poly}\,(N)\right)$
$=$ *LR-Mesh* $\left(\log^j N, \text{poly}\,(N)\right)$.

3.2. Complexity of the APPBS

The *Array of Processors with Pipelined Buses using Switches* (APPBS) [3] is another reconfigurable model that uses pipelined optical buses. Unlike the PR-Mesh, the model uses four switches at each processor to connect to each of the adjacent buses. Four configurations are available to each switch (Figure 4). Each processor locally controls its switches, and can change its configuration once or twice at any petit cycle(s) within a bus cycle. Another difference between the PR-Mesh and the APPBS is that the APPBS cannot end a bus in the middle of the mesh, each bus must extend to the outer processors in the mesh. The APPBS can either use the coincident pulse technique or the control functions $send(m)$ and $wait(n)$ to send a message. These functions define the number of petit cycles processor m has to wait before sending a message and processor n must wait before reading a message.

The ability of different switches to change their settings during different petit cycles could result in many different model configurations within a single bus cycle. Note that (i) the path any given message traverses is linear, despite all the switch changes, and (ii) a message may follow a different path than the one that initially precedes (or succeeds) it in the pipeline. If we do not allow an increase of processors on an N^2-processor PR-Mesh, then simulating an APPBS appears to require one step to simulate each petit cycle, leading to $O(N^2)$ steps to simulate each step of an N^2-processor APPBS. By allowing the number of processors to increase by a polynomial factor, the PR-Mesh can simulate each step of an APPBS in a constant number of steps.

In the other direction, the obstacles to simulating a PR-Mesh by an APPBS are that the APPBS does not have delay loops and is not able to segment its buses. To overcome these problems, we simulate an LR-Mesh by an APPBS, rather than a PR-Mesh by an APPBS. This, along with the result of Corollary 2, implies that the APPBS can simulate any step of a PR-Mesh in constant steps using polynomial processors.

Theorem 3 *PR-Mesh* $\left(\log^j N, \text{poly}\,(N)\right)$
$=$ *APPBS* $\left(\log^j N, \text{poly}\,(N)\right)$.

Proof: Let $\mathcal{S}$ denote an $N \times N$ APPBS and let s_i denote a processor of $\mathcal{S}$ numbered in row major order. We construct an $O(N) \times O(N) \times O(N^2)$ PR-Mesh $\mathcal{P}$ that simulates each step of $\mathcal{S}$ in $O(1)$ steps. Let layer ψ of $\mathcal{P}$ represent the APPBS configuration at petit cycle ψ, where $0 \leq \psi < N^2$. $\mathcal{P}$ creates a vertical bus representing the path each message would follow over the APPBS, such that the message passes switches in layer ψ corresponding to the APPBS switches it would pass in petit cycle ψ.

Within each layer of the PR-Mesh, we use a 4×4 block of processors to simulate each processor of the APPBS, as in Section 3.1. Let $block_i$ simulate s_i. Each block in layer ψ sets its configuration to simulate the corresponding APPBS processor during petit cycle ψ. Blocks connect within the same layer to the preceding block on the bus and then route the bus up to the next layer.

By using the control functions $send(m)$ and $wait(n)$, each processor holds information on the petit cycles in which it is to read and write. The writing processor for each bus (the block in layer 0) broadcasts the value it holds for $send(m)$. Each block can then determine if it is to receive the message by considering the data read, the value it holds for $wait(n)$, and its layer. Next, each block in layer 0 broadcasts its message. Therefore, *APPBS* $\left(\log^j N, \text{poly}\,(N)\right) \subseteq$ *PR-Mesh* $\left(\log^j N, \text{poly}\,(N)\right)$.

Now let $\mathcal{L}$ denote an $N \times N$ LR-Mesh and let l_i denote a processor of $\mathcal{L}$ numbered in row major order. We construct an $O(N) \times O(N)$ APPBS $\mathcal{S}$ that simulates each step of $\mathcal{L}$ in a constant number of steps.

We use a 3×3 block of processors in $\mathcal{S}$ to simulate each processor l_i of $\mathcal{L}$, as shown in Figure 5(a). The center processor, sc_i, sets its switches corresponding to the port configuration of l_i, and the remaining processors simulate the instances of buses that are segmented in $\mathcal{L}$. All of these processors set their switches to straight. If a bus ends at one of the ports of l_i, then

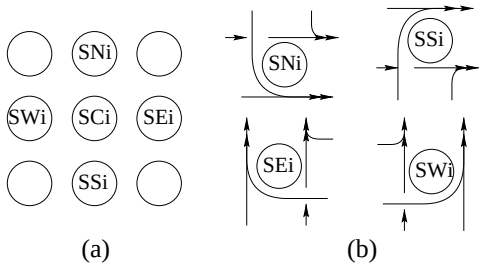

Figure 5. Configuration of APPBS processors to simulate an LR-Mesh: a) 3×3 block of APPBS processors for each LR-Mesh processor; b) configuration of port processors for a bus ending at a port of l_i.

the corresponding "port processor" sets its switches as shown in Figure 5(b). This forms alleyways to shunt messages if a bus is supposed to end. All processors on the alleyway disregard messages sent along alleyways, except for the port processor at which the bus was to end. To simulate a communication step, first set all switches as described above and send the messages along the buses. Next, any port processor that handled a bus termination sends the message to sc_i.

Combining this result with the fact that an LR-Mesh of $O(N^3) \times O(N^3)$ size can simulate each step of an $N \times N$ PR-Mesh in $O(1)$ steps [12], we have $\textit{PR-Mesh}\left(\log^j N, \text{poly}\,(N)\right) \subseteq \textit{APPBS}\left(\log^j N, \text{poly}\,(N)\right)$.

Therefore, $\textit{APPBS}\left(\log^j N, \text{poly}\,(N)\right) = \textit{PR-Mesh}\left(\log^j N, \text{poly}\,(N)\right)$. ■

3.3. Complexity of the AROB

The *Linear Array with Reconfigurable Optical Buses* (LAROB) and AROB [8], are similar to the LARPBS and PR-Mesh, respectively, with some extra hardware features. They are able to segment buses into separate subarrays as are the LARPBS and PR-Mesh.

Each processor of the AROB is equipped with an internal timing circuit that can count an arbitrary number of unit delays between receiving a signal and sending out a new one within the bus cycle; each processor can also add an arbitrary number of unit delays to shift the select pulse with respect to the reference pulse. There is a relative delay counter and an optical rotate-shift register at each processor enabling it to perform a bit polling operation within one step. This is the ability to select the k^{th} bit of each of N messages and determine the number of these bits that are set to 1. Pavel and Akl also presented an extended version of the LAROB. The extended model allows on-line switch settings during a bus cycle and the transmission of up to N messages with arbitrary word size.

These features suggest that the AROB does not have the same complexity as the PR-Mesh. By allowing the number of processors to increase polynomially, however, we establish the same complexity despite these obstacles.

Theorem 4 $\textit{PR-Mesh}\left(\log^j N, \text{poly}\,(N)\right) = \textit{AROB}\left(\log^j N, \text{poly}\,(N)\right)$.

Proof: An $N \times N$ AROB can simulate each step of an $N \times N$ PR - Mesh in a constant number of steps, as it has the same capabilities. Therefore, $\text{PR-Mesh}\left(\log^j N, \text{poly}\,(N)\right) \subseteq \text{AROB}\left(\log^j N, \text{poly}\,(N)\right)$.

Let $\mathcal{B}$ denote an $N \times N$ AROB and let b_i denote a processor of $\mathcal{B}$ in row major order. We construct an $O(N) \times O(N) \times O(N^2)$ PR-Mesh $\mathcal{P}$ that simulates each step of $\mathcal{B}$ in $O(1)$ steps. We will individually present simulations of the features not possessed by the PR-Mesh.

The first feature we simulate is the bit polling operation. We use a similar approach as in the APPBS simulation (Section 3.2) and consider $2N^2$ layers of a PR-Mesh to simulate an AROB. Again, we use a 4×4 block of processors, as shown in Figure 3, to simulate each processor of the AROB on each layer. Each block sets its configuration to form buses up through the layers of the PR-Mesh. In contrast to previous simulations, the base layer here is layer N^2, the center layer.

Consider one of the original buses of the AROB, where the head of the bus is processor b_i. All processors on the bus now determine their distances from the head of the bus by computing prefix sums [8] on the upper N^2 layers of the bus. Call this distance d_k for processor b_k. Do this for all buses of the AROB.

The bus corresponding to b_k begins in layer $N^2 - d_k$. Each block on the center layer broadcasts its message. If processor b_k was to perform a bit-polling operation on the i^{th} bit, then each block p_k on each layer that received a message extracts the i^{th} bit from the message it read and uses this value in the next step. Next, all blocks connect in vertical buses and perform prefix sums [7] to get the bit polling result within a constant number of steps. The sum obtained by p_k represents the number of i^{th} pulses that are 1.

The next feature we consider is the ability to set an arbitrary number of delays. We will use the following lemma to show that the PR-Mesh can simulate setting an arbitrary number of delays in $O(1)$ steps with a polynomial increase in the number of processors.

Lemma 5 *An N^2-processor LARPBS can simulate in $O(1)$ steps any step of an N-processor LAROB that allows an arbitrary number of delays.*

Proof: Let processor p_{Ni} of the LARPBS simulate processor b_i of the LAROB, so that each p_{Ni} has a segment of N processors corresponding to it. Processor p_{Ni} sends a message to each of the N processors in its segment with the value of its delay. For a delay of x_i corresponding to p_{Ni}, each of the first x_i processors of segment p_{Ni} sets its value to 1. Perform prefix sums over all N^2 processors. Processor p_{Ni} then adjusts its prefix sum by x_i. Based on the adjusted prefix sum value, p_{Ni} adjusts its select frame. Processor p_{Ni} sends this information to p_i. Now p_i simulates b_i and sends the messages in a normal state of operation, such that all conditional delay loops are set to straight. Only the first N processors are active in this last step. ■

Lemma 6 *An $O(N) \times O(N) \times O(N^2)$ PR-Mesh can simulate in $O(1)$ steps any step of an $N \times N$ AROB that allows an arbitrary number of delays.*

Proof: We first present this for an $O(N) \times O(N) \times O(N^4)$ PR-Mesh, and then reduce it down to the desired size. Configure all processors to form the buses of the AROB in the bottom layer of the PR-Mesh. Perform prefix sums on each bus so each processor can get its ranking within its bus. The head of each bus sends its ID along the bus to provide a bus ID to all processors on that bus. Let the ID be j. Due to the third dimension, each of the processors on the bottom layer has an N^4-processor LARPBS associated with it. For the bus with ID j, map the i^{th} processor on bus j to processor $p_{N^2 i}$ of the N^4-processor LARPBS beginning at processor j on the bottom layer. From Lemma 5, each processor can determine the number of delays that will affect it, and can adjust its select frame accordingly. Adjust all select frames, then all processors along the bottom layer can send their messages through the bottom layer.

To reduce the PR-Mesh to N^2 layers, first rank processors along each bus as before. Next the tail of each bus sends the count to the head of its bus, so the head holds the total number of processors on its bus. To get the bus IDs, perform a prefix sum of the bus lengths using the heads of buses. By connecting the three-dimensional mesh in a snake-like pattern, the entire mesh is just a one-dimensional LARPBS. Now, place each bus in contiguous segments of the mesh, with the starting location depending on the bus ID. This problem then reduces to the one presented in Lemma 5. Therefore, we can simulate any step of the AROB using arbitrary delays on a PR-Mesh in $O(1)$ steps. ■

The next feature considered is the on-line switching ability of the AROB. This simulation follows the simulation of this feature of the APPBS by the PR-Mesh in Section 3.2. Combining these results, this proves that $AROB\left(\log^j N, \mathrm{poly}\,(N)\right) \subseteq PR\text{-}Mesh\left(\log^j N, \mathrm{poly}\,(N)\right)$, thus establishing that the two models have the same complexity. ■

References

[1] Y. Ben-Asher, K. J. Lange, D. Peleg, and A. Schuster, "The Complexity of Reconfiguring Network Models," *Information and Computation*, vol. 121, (1995), pp. 41–58.

[2] H. ElGindy, "An Improved Sorting Algorithm for Linear Arrays with Optical Buses," Manuscript, 1998.

[3] Z. Guo, "Optically Interconnected Processor Arrays with Switching Capability," *J. Parallel Distrib. Comput.*, vol. 23, (1994), pp. 314–329.

[4] Z. Guo, R. Melhem, R. Hall, D. Chiarulli, and S. Levitan, "Array Processors with Pipelined Optical Busses," *J. Parallel Distrib. Comput.*, vol. 12, (1991), pp. 269–282.

[5] M. Middendorf and H. ElGindy, "Matrix Multiplication on Processor Arrays with Optical Buses," to appear in *Informatica*.

[6] Y. Pan, "Order Statistics on a Linear Array with a Reconfigurable Bus," *Future Generation Computer Systems*, vol. 11, (1995), pp. 321–328.

[7] Y. Pan and K. Li, "Linear Array with a Reconfigurable Pipelined Bus System: Concepts and Applications," *Informations Sciences – An International Journal*, vol. 106, (1998), pp. 237–258.

[8] S. Pavel and S. G. Akl, "On the Power of Arrays with Optical Pipelined Buses," *Proc. Int'l. Conf. Par. Distr. Proc. Techniques and Appl.*, (1996), pp. 1443–1454.

[9] C. Qiao, "On Designing Communication-Intensive Algorithms for a Spanning Optical Bus Based Array," *Parallel Proc. Letters*, vol. 5, (1995), pp. 499–511.

[10] C. Qiao and R. Melhem, "Time-Division Optical Communications in Multiprocessor Arrays," *IEEE Trans. Comput.*, vol. 42, (1993), pp. 577–590.

[11] J. L. Trahan, A. G. Bourgeois, Y. Pan, and R. Vaidyanathan, "Optimally Scaling Permutation Routing on Reconfigurable Arrays with Optically Pipelined Buses," *Proc. 13th Int'l. Par. Process. Symp. & 10th Symp. Par. Distr. Process.*, (1999), pp. 233–237.

[12] J. L. Trahan, A. G. Bourgeois, and R. Vaidyanathan, "Tighter and Broader Complexity Results for Reconfigurable Models," *Parallel Proc. Letters*, vol. 8, (1998), pp. 271–282.

[13] R. Vaidyanathan and J. L. Trahan, "Optimal Simulation of Multidimensional Reconfigurable Meshes by Two-Dimensional Reconfigurable Meshes," *Info. Proc. Letters*, vol. 47, (1993), pp. 267–273.

[14] S. Q. Zheng and Y. Li, "Pipelined Asynchronous Time-Division Multiplexing Optical Bus," *Optical Engineering*, vol. 36, (1997), pp. 3392–3400.