Review

From the above discussion, it can be seen that the RMB offers an advantage over the hypercube and fat-tree architectures using the metrics of number of links, cross points, and area for embedding k-permutations. It is also comparable to the mesh using these criteria. From the perspective of ease of use, linear arrays and rings are easier to manage. The RMB uses constant length wires and that offers a major advantage in operating a network at high clock rates. It should be remembered that communication through the RMB is asynchronous, and the uniformity of wire length has primary benefit in the rate at which buses can be compacted. Moreover, the design of each INC and independence from global clock requirements is beneficial.

4 Concluding Remarks

In this article we introduced the reconfigurable multiple bus RMB as a flexible interconnection network for a ring array of processors. The study of multiple bus systems is not new (e.g.. [5]). The use of reconfiguration has eliminated the hardware required for bus arbitration. In addition, the compaction protocols for messages permit full utilization of the multiple bus bandwidth in a straightforward manner. The maximum utilization of the RMB with no prior knowledge of future incoming requests for communication, i.e.., on-line, is achieved through continuous *compaction* of existing virtual-buses. A measure of effectiveness of this approach is its "competitiveness", i.e.., the ratio of its required time for communicating all messages to the time required by an optimal off-line schedule. We plan to pursue research to evaluate the competitiveness of our on-line routing protocol for random communication patterns and for communication patterns emerging from practical applications.

It is worth pointing out that an RMB with k buses should not be considered equivalent of a k bus system. An RMB with k buses can support many more than k virtual buses simultaneously. In the worst case it will support k virtual buses each of length N.

Future research plans also include extending the RMB design to allow multiple send/receive messages per node, comparison with other universal interconnection networks such as the k-ary n cube network, scalability issues and the design of reconfigurable multiple bus systems for 2- and 3-D grid connected computers.

References

[1] K. Hwang, *Advanced Computer Architecture: Parallelism, scalability and programmability*. McGraw-Hill Computer Science Series, 1993.

[2] C. L. Seitz, "The Cosmic Cube," *Communications of the ACM* Vol. 28, pp. 22 – 33, January 1985.

[3] S. B. Choi and A. K. Somani, 'The Generalized Folding Cube Network," *NETWORKS, An International Journal*, Vol. 21, pp. 267 – 294, March 1991.

[4] S. B. Choi and A. K. Somani, "Rearrangeable Hypercube Architecture for Routing Permutations," *JPDC*, Vol. 19, pp. 125–133, 1993.

[5] T. N. Mudge, J. P. Hayes and D. C. Winsor, "Multiple bus architectures," *Computer*, pp. 42 – 48, June 1987.

[6] C. E. Leiserson, "Fat-trees: Universal networks for hardware-efficient supercomputing," *IEEE Transactions on Computers*, Vol. C-34, pp. 892 – 901, 1985.

[7] C. B. Stunkel, D. G. Shea, B. Abali, M. M. Denneau, P. H. Hochschild, D. J. Joseph, B. J. Nathanson, M. Tsao and P. R. Varker, "Architecture and implementation of Vulcan," *Proceedings of 8th International Parallel Processing Symposium* pp. 268 – 274, April 1994.

[8] A. Agarwal, "Limits on interconnection network performance," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 2, No. 4, pp. 398-412, October 1991.

[9] K. Nakano, "A bibliography of published papers on dynamically reconfigurable architectures," *Parallel Processing Letters*, Vol. 5, No. 2, pp. 111 – 124, 1995.

[10] W. J. Dally, "Virtual-Channel Flow Control," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 3, No. 2, pp. 194 – 205, March 1992.

[11] Y. Saad and M. H. Schultz, "Topological properties of hypercubes," *IEEE Transactions of Computers*, Vol. C-37, pp. 867 – 872, 1988.

[12] R. L. Greenberg and C. E. Leiserson, "Randomized routing on fat-trees," *Proc. of 26th Annual Symposium on Foundations of Computer Science*, pp. 241 – 249, October 1985.

lows any arbitrary k messages to pass through the RMB concurrently.

This measure is equivalent to the bisection bandwidth. The bisection bandwidth of the RMB network is equal to $k * B_w$ where B_w is the bandwidth of one link.

Number of Links, Cross points, and Area for the RMB

To support a k-permutation, the RMB architecture requires k buses. Thus the number of links is exactly equal to $N * k$ and all wires are of equal unit length. For the computation of cross points in the RMB, each output can receive data from 3 inputs in each INC. Thus each output has three cross points. There are exactly $N * k$ output ports in all INCs together. Hence the total number of cross points is $3Nk$. The corresponding RMB structure can be laid out using an area of the order of Nk. Thus the RMB structure is efficient from the area point of view as well.

Links, Cross points, and Area for Hypercube

The hypercube and its derivatives have $NlogN$ links, where $N = 2^n$, and link lengths vary in different dimensions in any layout. For example, the EHC which can embed an arbitrary permutation has a degree of $n + 1$ for each node and hence $N * (\log N + 1)$ links. For $k = logN$, this is comparable to the RMB but the hypercube has better permutation embedding capability. To bring a hypercube to support only k permutation capability, we can use a scaled GFC structure with degree d and appropriate number of nodes so that the GFC has d/k links in each dimension and 2^d nodes. This will have a total of $2^d * d$ links and $N/2^d$ should be greater than k. This yields that the total number of links is less than $(N/k)\log(N/k)$. That is much less than that required for the RMB. However, the number of cross points in the EHC structure is $N * (\log N + 1)^2$ and the area to lay it out is $\Omega(N^2)$. Similar is the case for the GFC. This makes the hypercube structure and its variation quite unattractive for VLSI implementations as they are much worse than the RMB.

Number of Links, Cross points, and Area for the Fat-tree

The minimum fat-tree structure required to support a k permutation among N processors has the structure illustrated in Figure 11.

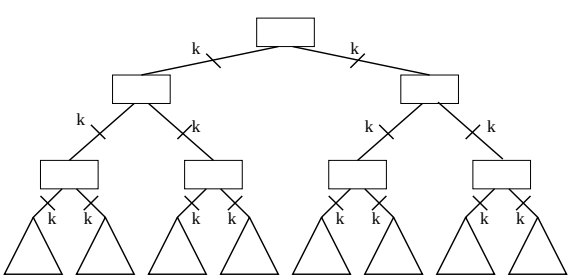

Figure 11: A fat tree supporting k-permutation

Assuming that the system has N nodes, they are divided into N/k leaf nodes. Each leaf node has k PEs and can send up to k messages to the rest of the network. The tree above it has to support only k links at each level from each side.

Each leaf node is internally organized as a complete fat tree. This means at each level in the tree there are k links. Each leaf node has $\log k$ internal levels and k external links. Thus the total number of internal links in all leaf nodes together is given by $(N/k) * k * \log k = N * \log k$. The remaining interconnection network has $(N/k - 2) * k$ links assuming that the root node of network does not have any out-going links. Otherwise, there will be additional k links from the root. Therefore, the total number of links in this structure is $N * \log k + N - 2k$. For $k = N$, the number of links is of the same order of as in hypercube supporting a full permutation of nodes. The RMB has more links than a hypercube or a fat tree to support k-permutation supporting fat tree.

The link lengths in fat tree depend on the layout. Assuming an H-tree layout could be of the order of $O(\sqrt{N})$ or $O(\sqrt{NLogN})$ depending on if I/O is not provided or provided on the boundary. The total wire length turns out to be more than the wire length in the RMB. The wire length affect the propagation time along a wire and the overall average delay in message delivery. In case of synchronous implementation, it also affects the clock rates for communication. Thus the RMB scales well with respect to the fat tree structure.

Computing cross point is problematic in fat trees. Each routing step in tree has k input and k output links from each of the left and right children and parent node. However, only $6k^2$ cross points are required to support the required connections. There are $N/k - 1$ such nodes. So total number of cross points are $(N/k - 1) * 6 * k^2 + N/k * O(k^2) = O(Nk)$ cross points where the constant is more than 6. The number of cross points in the RMB architecture is seen to be favorable comparable with the k-permutation supporting fat tree architecture.

An m node tree laid out as an H-tree requires $O(m)$ area. In the case of a fat tree, there are $2N/k - 1$ switching nodes, including the leaf nodes and internal routing nodes. Each node is $O(k^2)$ in size. So the total area of the k-permutation supporting fat-tree is $2N/k * O(k^2) = O(Nk)$ with a constant of at least twelve (each node is about $6k^2$ and $2N/k$ nodes). Thus, the area for fat-tree is higher than the RMB architecture.

Links, Cross points, and Area for Mesh

The mesh architecture has $2N$ links. Each node has a 4×4 crossbar. Therefore, the total number of cross points is $4 \times 4 * N$. The mesh can be laid out in $O(N)$ area. However, to embed a k-permutation, the k nodes may be in an $O(\sqrt{k} \times \sqrt{k})$ submesh, i.e.. there are not enough input/output links to support k wires. Therefore, each dimension of the mesh has to be expanded by a factor of $\sqrt{k}$. Thus the total area of the mesh becomes $O(Nk)$. The routing to establish an arbitrary permutation is still not well understood. An RMB with the same area and number of links, on the other hand, offers very simple routing.

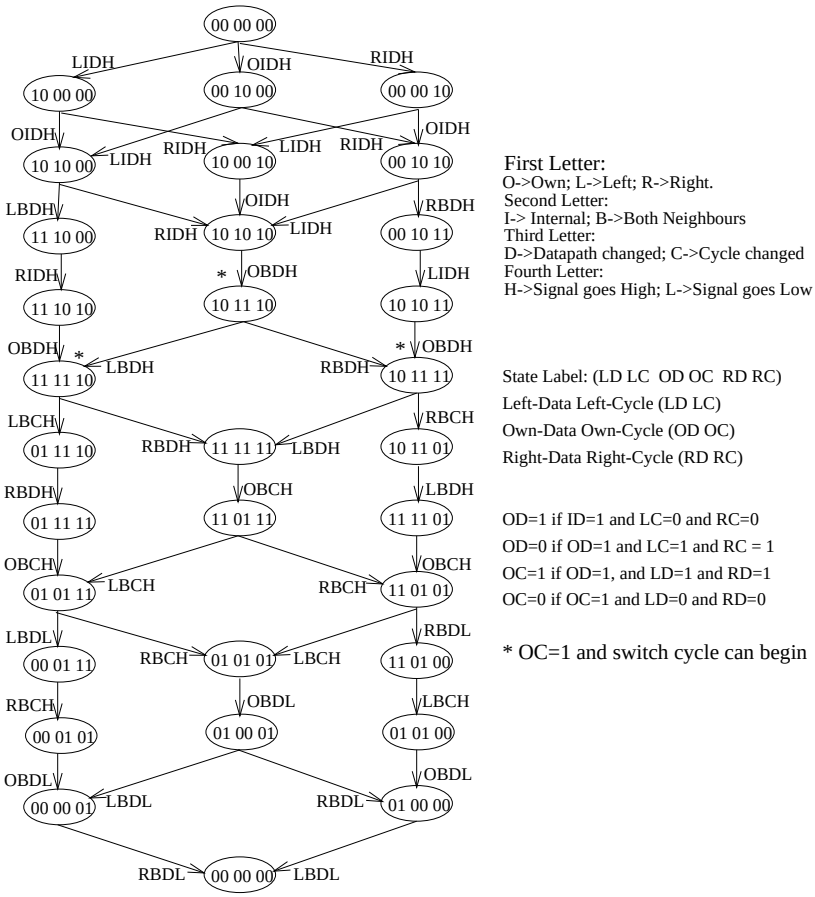

Figure 10: State transitions in odd/even switch

The fat tree, first introduced by Leiserson [6], is a network based on the complete binary tree structure. A set of N processors is located at the leaves of the fat tree. Each edge of the tree structure corresponds to two channels: one from parent to child, the other from child to parent. Each channel consists of a bundle of wires, and the number of wires in a channel is called its capacity. Each internal node of the fat tree contains circuitry that switches messages between incoming channels and outgoing channels. It is assumed that the circuitry can switch in constant time. Thus, the time required for delivering a message in a fat tree with N processors is $O(\log N)$. Routing in the fat-tree is straightforward since a unique path exists between each pair of the N processors.

A channel capacity of 2^i, where i is the distance of the channel from its leaves, is required to deliver simultaneous communication between arbitrary pairing of the N processors. The organization of a fat tree in an H-tree form requires $O(N)$ layout area with variable link lengths between the different levels of the tree. This makes the synchronization of message departures and arrivals demanding, though this may be handled through buffering of messages. An H-tree layout does not directly allow for I/O nodes to be on the boundary. To include I/O nodes on the boundary, the area required in $O(NlogN)$. The fat tree is off-line universal in the sense that it can efficiently simulate the traffic on any other network of the same volume where switch settings can be determined in advance. A randomized routing protocol [12] has demonstrated that it is nearly on-line universal.

The mesh architecture [1, page 371] is another attractive structure. With degree 4 nodes, any arbitrary size structure can be derived. The layout is straightforward and routing remains simple.

3.2 Analysis

There are several metrics of importance. First is the number of links and cross points (wire intersections) which one has to use to realize a network. We assume that the cost of a cross point and the cost of a link are similar in different architectures and costs also depend on the length of the wire. We compare the number of links, number of cross points, and the VLSI layout area for the RMB with the corresponding parameters of the hypercube-like architectures, the fat tree, and the mesh architecture. For comparison purposes, we pick a metric that defines the permutation capability of the network. An RMB with k buses can support any k-permutation where a k-permutation al-

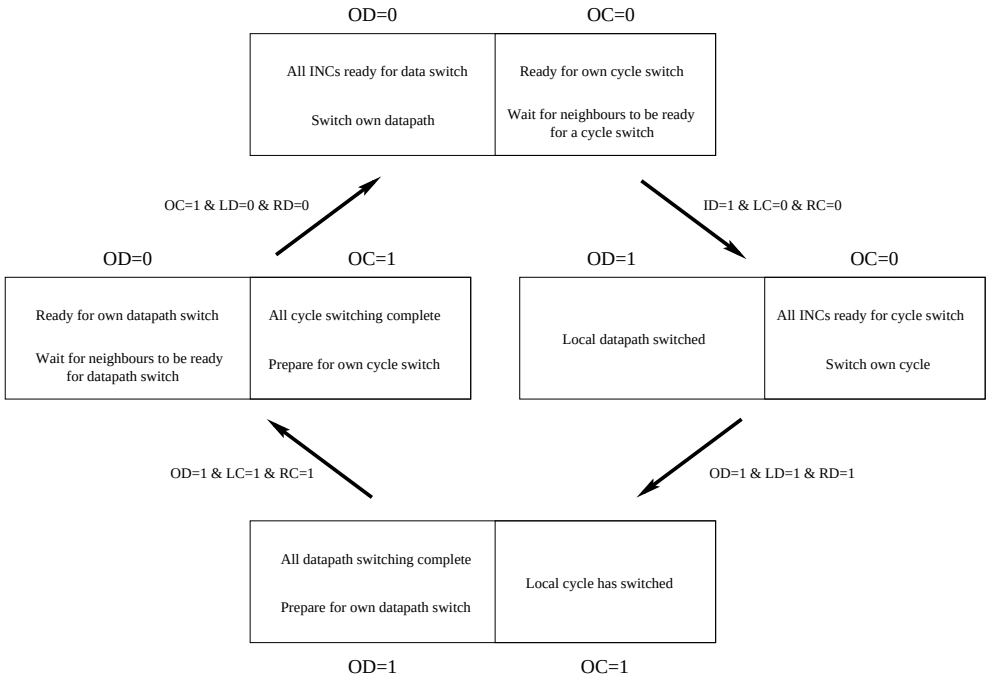

Figure 9: The four switching states of each INC

two nodes remain the same. In the earlier situation, node x is made to wait as it is not allowed to lower and raise its "Data" state until node y has raised its "Cycle" state. So node y will change first and again the number of transitions at the two nodes remain the same. This is true for any pairs of nodes.

The even/odd cycle switch state machine is shown in Figure 10. It is possible to see through the state transitions that once started each INC completes one cycle and then waits for its neighbors to complete their cycle before starting another transitions. Thus the neighboring INCs can alternate between odd/even pairs if started correctly. □

Theorem 1 *The systolic compaction protocol correctly maintains transactions over all existing virtual buses and provides full utilization of the RMB (that is, a request for communication is provided if a bus segment is available between the sending and receiving nodes in the clockwise direction).*

Proof: This follows directly from lemma 1, where the restriction that each node can send/receive a single message enforces one of the four possible transitions illustrated in Figure 7 to occur and no other. □

3 Performance & Advantages

3.1 Previous Networks

Several networks have been proposed and used in multiprocessor design and implementation. We describe the hypercube and some of its variants, the fat-tree and the mesh briefly as we will compare them to the RMB implemented on a ring network.

A hypercube network [2] is a highly concurrent multiprocessor topology. The regularity, symmetry, and strong connectivity of the hypercube make it suitable for various parallel algorithms. An n-cube can be constructed recursively from a lower dimensional cube [11]. Conversely, the n-cube can be decomposed recursively into smaller sub-cubes. Point-to-point routing is straightforward using an e-cube routing [11]. It is not known if routing of a permutation can be embedded in hypercube in contention free mode. For this purpose, several derivatives of hypercube have been proposed. Of particular interest are two new architectural concepts, called the generalized folding cube (GFC) [3] and the enhanced hypercube (EHC) [4] which have been developed to achieve permutation embedding capability. These structures have roughly the same number of slightly more links as the original binary cube. A hypercube with duplicate pairs of links in any one dimension is defined as the Enhanced Hyper Cube (EHC). An n-dimensional EHC has 2^n nodes and each node has $n + 1$ links. The GFC and EHC networks can embed any arbitrary permutation in circuit switching mode. Another serious limitation of a regular hypercube structure is the granularity for expansion in terms of number of processing elements (nodes) when scaling the architecture. To overcome this difficulty, a number of other variation of this architecture has also been developed. It has been shown that the hypercube and its variants are very powerful interconnection structures [2]

an odd or an even node depending on its position.

- An even numbered INC i considers moving a virtual bus on an even physical bus segment l in even cycles. It considers moving a virtual bus on an odd physical bus segment l in odd cycles.
- An odd numbered INC i considers moving a virtual bus on an even physical bus segment l in odd cycles, and moving a virtual bus on an odd physical bus segment l in even cycles.

The pairs of bus segments which are assessed for compaction are shown in Figure 8. As time progresses the nodes alternate between being in odd cycles and being in even cycles, and eventually the process of compaction shown in Figure 5 becomes possible as alternate switches compress the virtual buses downwards.

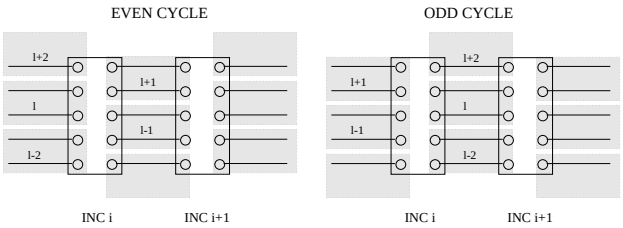

Figure 8: The states of each bus segment in an INC

2.5 Maintaining Even and Odd Cycles

The major action in switching the bus operation is in maintaining even and odd cycles. The assumptions we make here are that individual INCs operate off independent clocks and the timing of communications on the virtual buses is entirely independent of these clocks.

The scheme adopted in the RMB to coordinate these odd-even cycles is discussed below, however, it is necessary first to outline the general process of the cycling and to introduce the key signals which are involved.

In order to effect the local odd-even cycling and the safe movement of virtual buses two key states which are associated with each INC are used. These states indicate whether a virtual bus movement has occurred and whether a change of cycle (from odd to even or vice-versa) has taken place.

Each INC needs to know its own local state information, and the state information of its immediate neighbors. The states and signals are summarized in Table 2 above.

Essentially, an INC will not move any virtual bus until it is ready to do so and both its neighbors have signalled that they are ready also. Further, an INC will not switch from an odd to an even cycle (or vice versa) until it is ready to do so and both its neighbors are also ready.

A cycle cannot switch whilst a virtual bus is being moved, and a virtual bus cannot move whilst a cycle is switching. A simplified diagram of the state transitions within an INC is presented in Figure 9

Mnemonic	Interpretation
States	
OD	Own Datapaths have switched (virtual bus switch)
LD	Left neighbour's Datapaths switched
RD	Right neighbour's Datapaths switched
OC	Own Cycle has changed (odd to even or vice versa)
LC	Left neighbour's Cycle has changed
RC	Right neighbour's Cycle has changed
Signals	
ID	Internal signal to INC indicating all Datapath switches (virtual bus movements have been completed

Table 2: States/signals used in odd-even cycle control

Provided that all INCs are initially reset (i.e. have their OD and OC states set to zero), then either by asserting an ID signal or an OC signal in a single INC, the cycling procedure will propagate through the entire array.

More formally, the process can be embodied in the following five rules:

1. At reset, ensure OD = OC = 0 for all INCs
2. OD = 1 if ID = 1 and LC = 0 and RC = 0
3. OC = 1 if OD = 1 and LC = 0 and RC = 0
4. OD = 0 if OD = 1 and LC = 1 and RC = 1
5. OC = 0 if OC = 1 and LD = 0 and RD = 0

The complete state diagram for the switching sequence can be seen in Figure 10

We now show the correctness of the systolic compaction protocol. **Lemma 1** *While using the even/odd cycle algorithm, all nodes will alternate between the two states even and odd and the number of transitions performed by a pair of neighboring nodes at any time will not differ by more than one.*

> **Proof:** Each pair of neighbors are initialized properly. A node changes state between odd and even only when both of its neighbors are ready to change (LD=RD=1). Suppose node x changes first from even to odd and its neighboring node y is waiting to change from odd to even. So both neighbors are in odd state. Node x will keep its "Data" and "Cycle" flags high since the "Data" flag will only be set to zero after both neighbors acknowledge with a high "Cycle". Now, one of the two things will happen. Either node x will get done first with its internal operation in odd state and is ready to change again (ID=1) or node y changes to even state. In the latter case, the number of transitions by

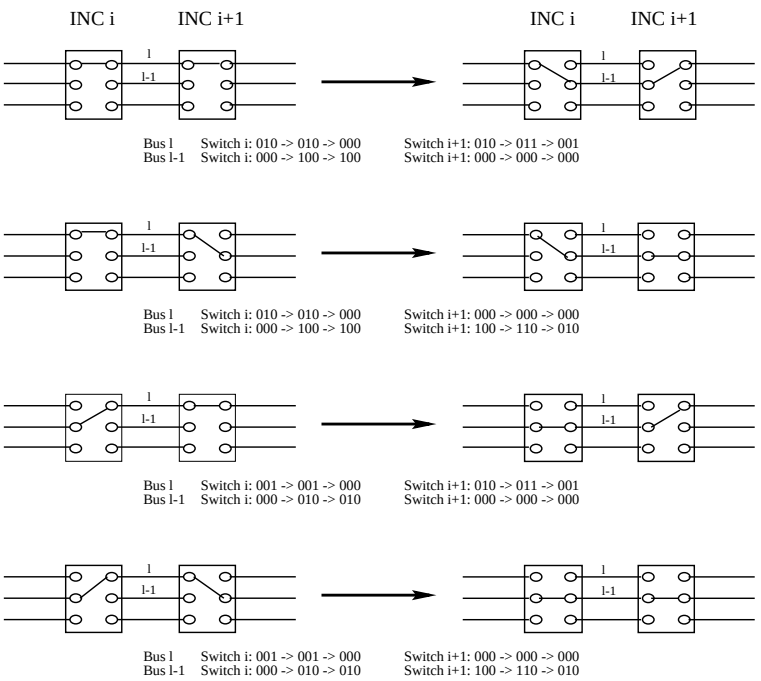

Figure 7: Four conditions for transitions.

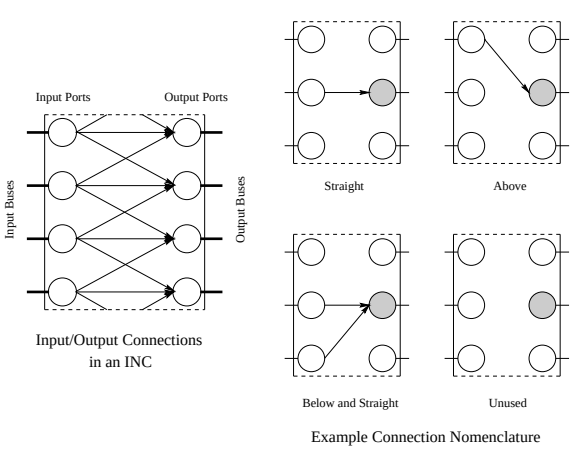

Figure 6: Mapping between I/O ports of an INC

put port. Thus, switching to a lower bus segment is feasible. Notice also that the intermediate state code shown in in the diagrams often indicates the make-before-break interconnection strategy.

- If the operation of a bus segment can be switch to a segment below it then such a bus segment is defined to be in a state called switchable down. A bus segment between a pair of INCs can be in one of the following states: (a) not used; (b) in-use and not switchable down; and (c) in-use and switchable down. In state (c), the operation is switched to the next lower bus segment.

Port state	Interpretation
000	Bus is unused
001	Port receives from below
010	Port receives straight
011	Port receives from below and straight
100	Port receives from above
101	Not allowed
110	Port receives from above and straight
111	Not allowed

Table 1: Interconnections between input and output ports of an INC (viewed from the output port)

Unfortunately, it is not possible to move all used physical bus segments down by one position concurrently, even if there were unused channels available. The reason for this is that there would be races caused between adjacent INCs as to which bus segments would move and in what order. Also, there are situations, due to limited switching at each INC (from input port l to output port $(l-1,\ l,\ l+1)$, when switching a virtual operation down may breaking the virtual bus connection. To circumvent this problem, our architecture causes pair wise agreements to be made between adjacent nodes as decide which bus segments are moved at a time. This process is coordinated with a two phase local synchronization strategy. These phases are termed *odd* and *even* cycles, and the operation is as follows:

- Each node or INC in the array becomes marked as

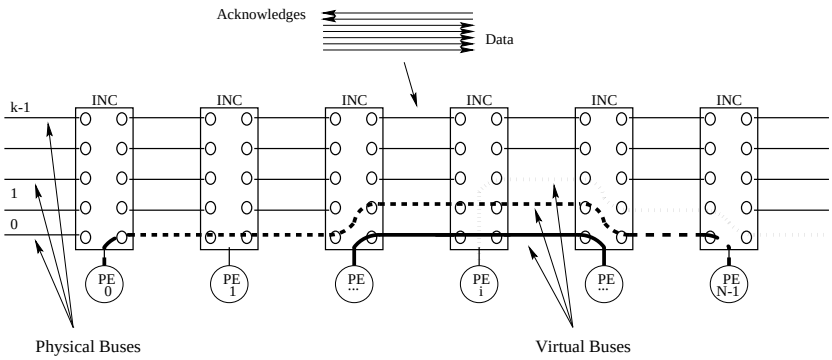

Figure 2: Physical bus segments and virtual buses

electrical propagation of signals. In this strategy our protocols establish an alternative path before disconnecting the old one as shown in Figure 4. This strategy allows for the communication on a virtual bus to progress independently of the process of compaction.

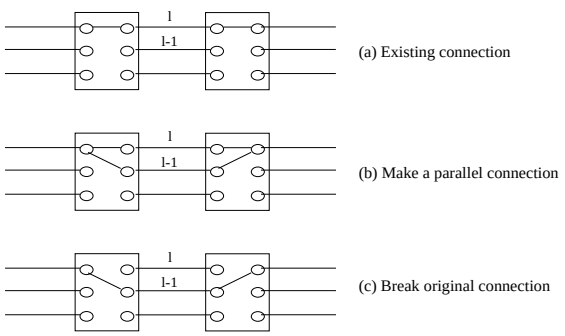

Figure 4: Make-Before-Break connection strategy

The process of moving a virtual bus downwards is performed by a sequence of local downward moves. This compaction process is independent of the message propagation through the INC at the same time. The compaction protocol which operates in each INC is based on local state information, and the states of the INC's immediate neighbors.

The process of moving a virtual bus down in one even and one odd cycle is illustrated in Figure 5.

A detailed description of the compaction protocol the process of maintaining a valid communication path between neighboring INCs is given in the following two sub- sections.

2.4 Compaction Protocol

The protocol operates in each INC based on local state data and information exchanged with the neighboring INCs. If a physical bus segment l is being used and physical bus segment $l-1$ is free then the transaction on l is moved down to $l-1$ in a such a way that the virtual bus is not disconnected at any time. Notice that signals flow in both directions on the buses (i.e. data in a clockwise direction and acknowledgements in a counter-clockwise direction).

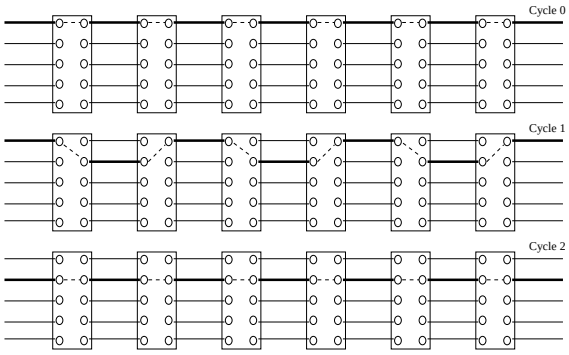

Figure 5: Moving an entire virtual bus in two cycles.

We now outline the compaction algorithm and switching operation in some detail. The algorithm resembles a systolic behavior in the way it makes decision in regular cycles based on local information.

- Each INC maintains a 3 bit status register for the output port of each physical bus segment. The states correspond to the connections illustrated in Figure 6, and are shown in Table 1.

  The status bits for the lth output bus segment directly indicate the corresponding input bus segment ($l-1$ or l, or $l+1$). An output bus segment may receive data from more than one input bus segments provided the information being received from each is the same. Such a situation takes place during the make-before-break process of the downward motion of a virtual bus.

- Since an output port in an INC can only receive from one input data stream at any time, a transaction on bus segment l may only be switched to bus segment $l-1$ if, after the switching, the $l-1$th output port can still be connected to the corresponding input port. Thus, there are only four possible scenarios in which this condition can be satisfied. Figure 7 shows those conditions.

  Note from the figure that in each of the cases, the new output port can receive input from the old in-

not affect the protocols.

- The protocol borrows concepts from wormhole routing. Each message is split into flits. A request with the message destination address is encoded into a header flit which is sent along a virtual bus towards the destination node. An acknowledgement signal is sent back along the same virtual bus in the reverse direction. The INC at the destination node will accept the request if the INC and PE receive ports at that node are both free.

- New channels of communication are introduced only at top bus, bus segment $k-1$ at that node, in the multiple bus system. The request then propagates along that bus. During the lifetime of this communication, the virtual bus operation may be moved down to other buses – $k-2$ ... $k-3$ ..., which is the reason for calling the channel a virtual bus (Figure 2). The goal of moving the virtual bus operation downwards is to make use of only the lowest physical free bus segments. We call this process *bus compaction*.

  The advantage of restricting the initiation of new requests to bus segment $k-1$ is that each INC has to monitor only the top bus segment for header flits. It also avoids any deadlocks while establishing virtual bus connection.

  This restriction has the potential of causing long delays for header flits and being unfair in providing network access to different PEs. These drawbacks are alleviated by allowing the compaction process to start even before any acknowledgement to the header is received. Thus, the top bus is released as soon as possible to accommodate other incoming requests. The request then continues to pass through the network drawing a virtual bus behind it, where that virtual bus can be provided by some physical bus segment other than the top bus segment (Figure 3). A request can only be initiated if the top bus segment at that INC is not being used to serve another request.

- A request consists of *header flit* (HF) followed by *data flits* (DFs). Data flits are only transmitted after an acknowledgement is received for the HF from the destination. This is in order to avoid buffering of DFs at intermediate nodes and is where our protocol differs from traditional wormhole routing.

  A request which is not accepted will have to be tried again at a later time.

- A request is terminated by a special flit called the *final flit* (FF) sent by the initiating PE.

- Each flit, or a group of flits, is acknowledged. Four different kinds of acknowledgement signal are associated with a request:

  – The *header acknowledgement* (Hack) is used by an INC to allow data flits to be transmitted.

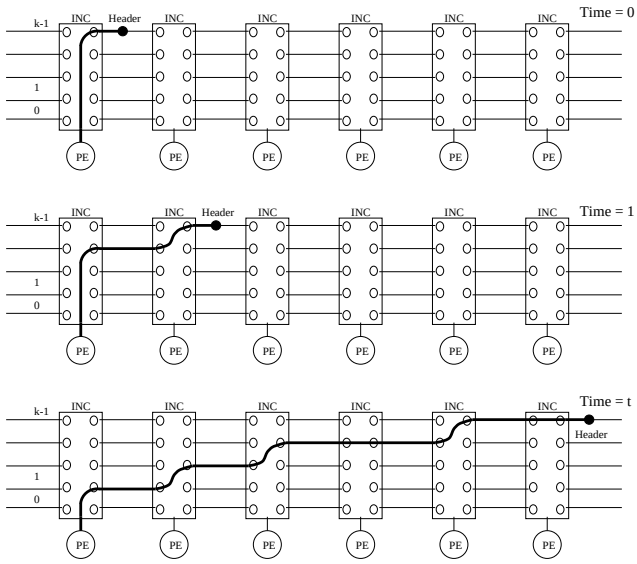

Figure 3: Buses and the compaction process

  – The *data flit acknowledgement* (Dack) is used for continuation of data flit transmissions and may also be used for flow control.

  – The *final flit acknowledgement* (Fack) is used to remove a virtual bus from the RMB. A "Fack" signal is used by all intermediate INCs to free a port being used by that virtual bus connection. Once a virtual bus (the bus segments that it was occupying) is freed up, the compaction process will bring the virtual buses above it down.

  – A *negative acknowledgement* signal (Nack) is used to refuse a request and to release the virtual bus associated with that request.

- The motion of virtual-buses for the purpose of compaction is only downwards. This feature provides an order on the virtual buses in all INCs they pass through.

2.3 Multiple Bus Operation

A node with a message to be sent, attempts to insert the header flit HF at the top output port of its INC. If the port is busy, then the node buffers the HF and waits. Otherwise the HF is permitted to move forward towards its destination along the top bus segments of the RMB.

After an HF has moved forward forming a virtual bus which spans a number of nodes, the corresponding INCs immediately start to move the newly formed virtual bus downwards to their lowest free input and output ports. This process frees the top bus segments for new communication requests to be inserted into the multiple bus system. The process of moving an existing virtual bus at a particular INC requires transferring the connections to new input/output ports. We adopt a make-before-break strategy to ensure proper

efficiency of this topology with respect to established architectures in supporting permutations. For scalability, the ring-based medium-sized system is used as a module. Multiple modules can be used to create larger systems where these modules are interconnected using specific topologies. For example, generalized folding cube [3] architecture recommend usage of richly connected multiple processors node as a building block for larger system configurations. Efficient use of buses in specific topologies is a topic of our future research and is beyond the scope of this paper. Whilst the RMB concept can also be extended to support broadcasting and multicasting, these issues are also not addressed in this paper.

Processors of our chosen ring-based array communicate by sending messages through the RMB using a mechanism based on *wormhole* routing [10]. The messages are decomposed into small fixed length *flits* (*fl*ow control dig*its*). The message as a whole comprises a single header flit which specifies the destination for the message, a number of data flits and a message termination flit. The header flit passes through the interconnection network of the RMB towards its destination and at each switching node sets up a channel for the data flits to follow. The data flits follow the trail of the header flit in a 'pipelined' fashion (however we do not assume that this movement is necessarily clocked as would be the case in systolic arrays). The RMB hardware and protocols ensure that the flits in a message remain contiguous in the channels whilst maximizing the utilization of the bus segments that form the network. The main advantage of the RMB is its organization which makes both implementation and routing straightforward.

The architecture of RMB, the switching nodes and the communication protocols are presented in Section 2 together with studies of the correctness of the strategy and performance characteristics. A comparison between the ring based RMB and the hypercube, fat tree, and a 2-D mesh networks in supporting permutations is presented in Section 3. Finally, we conclude with a discussion of future research directions.

2 Reconfigurable Multiple Bus System

2.1 System Components

The multicomputer system we support with the RMB network has a ring topology with N nodes. Each node consists of a processing element (PE) and an interconnection network controller (INC). The reconfigurable multiple bus system consists of k buses. The ring based nature of the network defines that each INC connects to only two other INCs. Each pair of adjacent INCs are connected using the k bus segments. An RMB with N nodes and k buses is depicted in Figure 1. This simple organization allows for all pair wise communication to be supported with one-way traffic through the network. It should be noted that although, for simplicity, we describe the communication as a one-way ring, for efficiency reasons, one may like to organize the communication as two parallel unidirectional rings.

The PE in each node connects to an INC through a single interface; this interface permits the PE to read

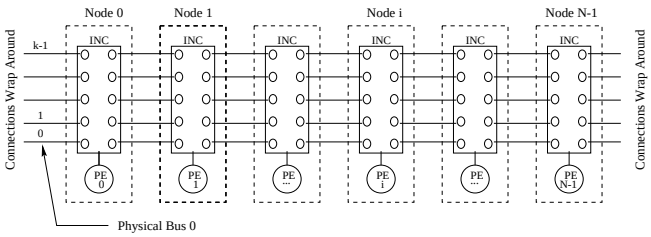

Figure 1: A multiple bus system

from any one input bus in the INC and to write to any one output bus in the INC. It is possible for the interface to be enhanced to permit the PE to talk concurrently with multiple inputs and outputs however for simplicity this is not discussed here. We number the nodes of the multiprocessor 0 to $N-1$. The same number will be used to refer to the PE and the INC connecting a node to the bus system.

2.2 INC Design

Each INC communicates with other INCs through k input and k output ports, one for each bus segment. The number k is a design parameter to be chosen based on the number of nodes in the system, length of a bus segment which is tolerable given its delay and electrical parameters and the target applications of the system. Notice that INC does not have to be a single chip. The lth output port of the ith INC is physically wired to the lth input port of the $i+1$st INC, all indices are calculated modulo N.

An INC routes incoming messages destined for other nodes directly to an output port. For the sake of simplicity and practicality of hardware implementation, our design allows each input port to be connected to one of three output ports only. More specifically, the lth input port in each INC can only be connected to one of the $\{l-1, l, l+1\}$ output ports. Input-to-Output connections are dynamically reconfigured based on local state information within the INC and the statuses of its neighboring INCs. We show later that the INCs are capable of creating virtual buses for communication between pairs of PEs, of maintaining ordered propagation of signals on any virtual bus, and of maximally utilizing the multiple bus system.

A PE requiring to communicate with another requests the INC in its node to establish a connection. The INC will eventually establish a new connection when the target PE is ready to receive and at least one virtual bus is available to support the connection. In this implementation the multiple bus system connects the INCs into a ring where the signals flow in a clockwise direction and acknowledgements travel in a counterclockwise direction on the same virtual bus. The routing protocol used to establish a connection is detailed below. The main features of the routing protocol and their implications are also outlined in the description. Adjacent INCs are assumed to communicate asynchronously for setting up a connection and data transfer. It is possible to design systems where adjacent INCs communicate synchronously or the clocks between the two are skewed and that will

RMB – A Reconfigurable Multiple Bus Network

H. ElGindy
Dept. of Computer Science
The University of Newcastle
Newcastle, New South Wales, 2308
Australia

A. K. Somani
Dept. of E.E. and Dept. of C.S.E.
University of Washington
Seattle, WA 98195-2500
USA

H. Schröder
Department of Computer Studies
University of Technology
Loughborough, Leicestershire
LE11 3TU UK

H. Schmeck
Institut AIFB
Universität Karlsruhe
Karlsruhe,
Germany

A. Spray
Dept. Electrical & Computer Engineering
The University of Newcastle
Newcastle, New South Wales, 2308
Australia

Abstract

The heart of a massively parallel computer is its interconnection network. In this article we present a reconfigurable multiple bus network to support circuit switching as means of communication between processors of a multiprocessor machine. The main contribution of the papers is in demonstrating the simplicity of the routing hardware whilst still providing modularity and full utilization of the multiple bus system. A comparison with major interconnection networks is also presented.

Keywords: Reconfigurable Multiple Bus Network, Interconnection Structure, Permutation Routing, Multiprocessor systems.

1 Introduction

In high-performance computers, real-time and distributed multimedia systems, the interconnection network plays a crucial role. It can even be argued that the network's ability to deliver data within a specified/acceptable time delay is more important than the ability of the communicating processors to manipulate them. Many interconnection networks have been proposed by the research community [1, 2, 3, 4, 5, 6, 7]. Some have been prototyped but few have progressed to become commercial products. Various network architecture and their performance limits have been studied in [8]. This article studies a reconfigurable multiple bus strategy from which to construct high performance computer networks.

Fixed and reconfigurable buses have been used to reduce the large diameter of array connected computers, though perhaps surprisingly, their main use has been in performing certain computations more efficiently. Examples of these computations exist in image processing, sorting, selection, geometric and graph algorithms ... etc. A survey of reconfigurable algorithms and architectures can be found in [9].

In this paper we define the reconfigurable multiple bus (RMB) architecture, and illustrate its use in supporting the communication between the processors of a multicomputer.

The reconfigurable multiple bus architecture relies on the use of an array of parallel bus segments between processing nodes. Each processing node can access the reconfigurable bus system to communicate with another processing node. The bus controller connected to each node coordinate the efficient use of the available buses through reconfiguration. The most important aspect of this architecture is that the reconfiguration takes place entirely independently of any current communication in which the bus segments are involved. Furthermore, the protocols which are explained in this paper - the routing protocol (for moving a message to its destination) and the compaction protocol (for enforcing efficient use of the multiple bus system) - are independent of one another with respect to synchronization.

In this paper, we describe RMB as a ring-based topology to be used to implement a medium size multiprocessor system. We study the effectiveness and