Design of a Uni-Directional MultiRing Switch

Xiangjian He[†], *and Hamid Arabnia*[††]

[†]*Department of Computer Systems, University of Technology, Sydney, PO Box 123, Broadway 2007,Australia*
[††]*Department of Computer Science, University of Georgia, Athens, GA 30602, USA*

Summary
MultiRing is a network of 2^n nodes which can be configured into different ring networks of n different configurations. It supports a wide variety of algorithms, such as algorithms for parallel image processing. In this paper, a MultiRing is implemented using a star topology with a MultiRing switch at the centre. We present a hierarchical design of the MultiRing switch. We demonstrate that the construction of the switch is economical in terms of gate count and port numbers. Our design preserves the need that all nodes can communicate simultaneously and independently in a ring configuration. We show that our design is scalable.
Key words:
MultiRing, Network Scalability, Switch Organization, Network Control.

1. Introduction

MultiRing described in [1] is a network containing different rings of processors (or nodes). A MultiRing network consists of 2^n processors physically connected as a star or other topologies. The effectiveness of a MultiRing system is its *reconfigurability*. The MultiRing has the capacity to be configured into R rings of D nodes each, where $R = 2^i$ and $D=2^{n-i}$ for any $i \in \{0, 1, \ldots, n-1\}$. As an illustration, a MultiRing network of $8=2^3$ nodes is displayed in Figure 1. Figure 1 shows all three configurations in the MultiRing, that are 1 ring of 8 nodes, 2 rings of 4 nodes and 4 rings of 2 nodes.

Because providing a direct link between any two nodes on a MultiRing is expensive, the MultiRing network consisting of 2^n processors can be physically interconnected through a MultiRing switch. Figure 2 shows the physical interconnection of a MultiRing of 8 processors. Through the MultiRing switch, nodes can connect with other nodes for a given configuration. Each node in a configured ring has two connections: one to its left node (the first node found in counter clockwise direction) and one to its right node (the first node found in clockwise direction) in the ring. Hence, each node has 2(n-1) possible connections in the 2^n-node MultiRing, of which only two connections are active at one time. With a given configuration, all nodes in the configured ring can send and receive messages simultaneously.

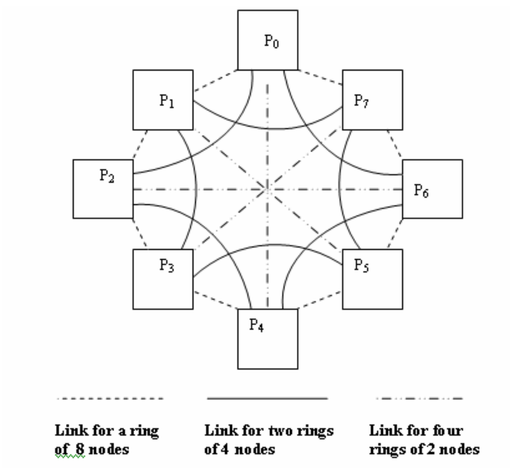


Fig. 1. Configurations of an 8-node MultiRing.

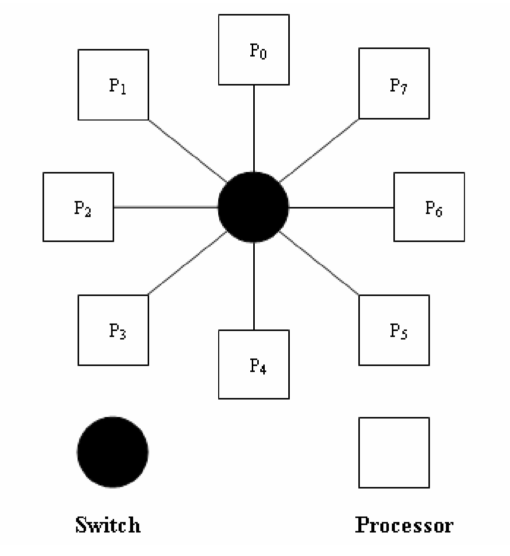


Fig. 2. Physical topology of an 8-node MultiRing.

There are many applications of MultiRing. Many algorithms requesting massive computation can be implemented on a MultiRing. For example, its application

Manuscript revised June 23, 2006.

to the areas of computer vision and image processing is found in [2] and [3] for pipelining message passing and for distributed and parallel processing.

There are many ways for configuring MultiRing depending on the needs. These require different designs for MultiRing switch. We list various ideas for MultiRing configuration as follows.

1). *Automatic switch reconfiguration*

The MultiRing switch automatically cycles through all of the configurations, starting with 2^0 ring of 2^n nodes down to 2^{n-1} rings of 2 nodes. The switch does not open messages. It just provides a path for a message to go through from a node to its next node for a given configuration. The switch will remain at a configuration for a set time to allow messages to go through. As an enhancement, the switch may maintain a cache of recently used configurations and increase the time it spends on the preferred configurations. Switching to a new configuration happens really quickly, so even if only one configuration is needed for an algorithm, the time spent on cycling through the unused configurations is negligible.

With automatic switch reconfiguration, an algorithm designer can think of the network as fully connected and not worry about requiring a certain configuration.

2). *Manual switch reconfiguration*

A MultiRing may be designed with a switch that waits until all nodes agree on a certain configuration before reconfiguring the network. Most of the algorithms implemented on the MultiRing require the nodes to operate in barrier synchronized fashion so that the other nodes that have data to send will eventually require the same configuration. Only after the switch receives *all* expected requests for a configuration, does the switch reconfigure the network.

A manual switch may also be useful when the MultiRing needs to maintain its configuration for a long time such as when implementing a ring of nodes as a pipeline where each node in the ring represents one processing phase. When one node is finished processing data, it will send its results to the next node in the ring. The configuration remains the same until all data has passed through all nodes in the ring.

3). *Smart switch*

When a smart switch is used, each node just sends a message to the switch without waiting for a particular configuration. The switch maintains a composite routing table that specifies the required configurations necessary to establish communication between all nodes in the MultiRing. The switch reads the messages and determines the expected configuration and direction from the composite routing table. The network is reconfigured and the message is sent along the correct path.

In this paper, we will concentrate on the design of MultiRing switch for automatic switch configuration. For message passing on the MultiRing, the source node must know the ring needed and the path for sending the message to the destination node. This require two tables, called *neighbour table* and *routing table* [3] respectively, to be maintained on each node. In addition, data link layer message frame must be formatted for the MultiRing to include the information that is needed by the next node in the configured ring for making the forwarding decision.

In [4], Arabnia and Smith designed a switch for MultiRing network. The design was based on the techniques called *perfect shuffle* [5] and *barrel shifting* [6]. Though many properties such as scalability, hierarchy and ability for parallel processing may also be found in this design, it requests a lot of more hardware and much larger count of Boolean operation gates for implementation than the MultiRing switch to be presented in this paper. Each switch element takes three inputs excluding the control input. The inputs of an element are not absolutely independent from those of other elements. This forces the design to use more hardware and more gates the switch implementation.

In this paper, we present a different approach for the design of MultiRing switch. Each switch element in our approach needs only two inputs. Furthermore, the inputs of any switch element are totally independent from the inputs of the other elements. Therefore, the gate count and hardware for switch implementation greatly decreases, which is important in VLSI design. This new design simplifies the switch implementation.

The context of this paper is arranged as follows. We propose the structure and organization of a MultiRing switch in Section 2. In Section 3, the detailed implementation of the MultiRing switch with link control is presented. This is followed by the discussion on scalability of the MultiRing switch in Section 4. In Section 5, a comparison is made between our new switch design and the one shown in [4]. We conclude in Section 6.

2. MultiRing Switch

A MultiRing switch allows communication directly from source to destination, without going through intermediate nodes. It also allows simultaneous communications between nodes under a MultiRing configuration.

2.1. Popular Switch Organizations

There are two popular switch organizations. A fully connected, or *crossbar*, interconnection allows any node to communicate with any other node in one pass through the interconnection as shown in Figure 3 [7]. An *Omega* interconnection as shown in Figure 4 [7] uses less hardware than crossbar interconnection. In crossbar organization, the outputs of AND gates in each column of switches as shown in Figure 3 are ORed to get a single input to a corresponding node. The implementation of the OR operation may have to use many OR gates when number of nodes (2^n) is big because the number of inputs allowed on a gate is limited by the electronic technology used to build the gate [8]. For a 2^n-node MultiRing, crossbar organization needs 2^{2n} interior switches each implemented using an AND gate with a separate control input. These internal switches are shown as black circles with node shown as squares in Figure 3. Because of the huge amount of internal switches and the corresponding control inputs, this organization is impractical for scalable MultiRing network. However, Omega organization needs only $2^{n-1}n$ switch boxes shown as rectangles in Figure 4. Each switch box has 4 switches shown as black circles in Figure 5 [7].

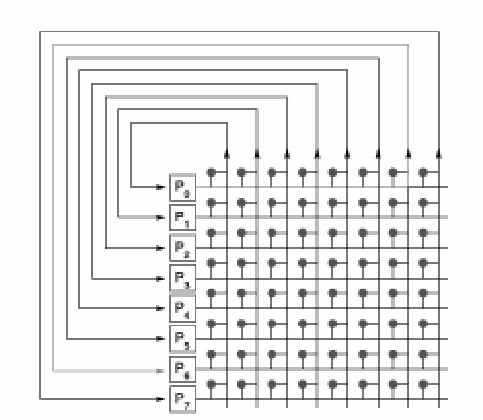


Fig. 3. Crossbar organization of a switch connected with 8 nodes.

2.2. Characteristics of Omega Organization

One of the advantages of Omega organization is the less number of switches required for the interconnection. This is important when a MultiRing has a large number of nodes and when the scalability of an interconnection has to be considered. However, Omega organization has the following two disadvantages.

1) Communication between any two nodes with Omega organization has to go through many passes (or many switch boxes). For example, in Figure 4, P_1 communicates with P_2 by going through 3 passes.
2) Contention between messages is more likely to occur in Omega interconnection. For example, in Figure 4, a message from P_1 to P_2 blocks at a switch box in the middle column while waiting for a message from P_0 to P_1. Of cause, if two nodes are sending messages to the same destination, there will be contention in both Omega and crossbar interconnections.

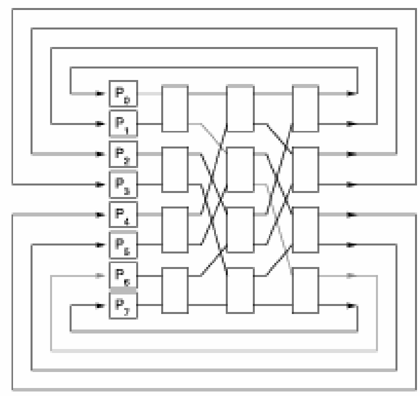


Fig. 4. Omega organization of a switch connected with 8 nodes.

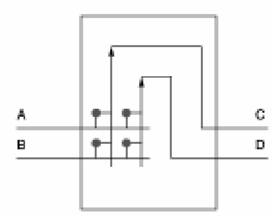


Fig. 5. Omega switch box.

2.3. Switch Organization for MultiRing

In order to take the advantages of Omega organization and to bypass the disadvantages of Omega organization, we present a new switch organization for MultiRing network. The new organization for 8-node MultiRing is displayed in Figure 6. The interior construction of this new switch is similar to the butterfly structure for network connection but not identical to it. The exterior connections to processors are particular for MultiRing network. Let us call this new organization *MultiRing switch organization*.

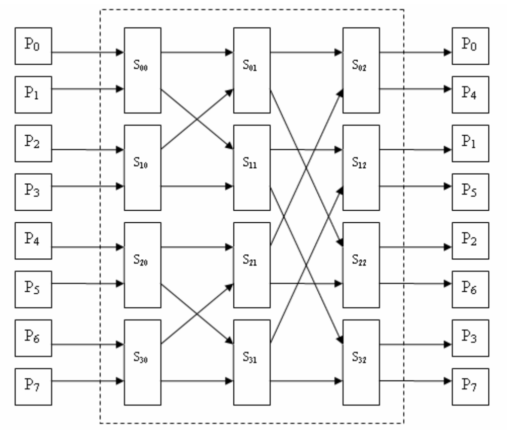


Fig. 6. MultiRing Switch Organization.

In Figure 6, the dashed rectangle represents the MultiRing switch. The solid rectangles represent switch boxes inside the MultiRing switch. Each switch box has four switches with two inputs and two outputs similar to Figure 5. S_{ij} denotes a switch box for each i ∈ {1, 2, 3, 4} and j ∈ {1, 2, 3}.

The MultiRing switch organization uses the same amount of switch boxes inside the switch as Omega organization. It meets the needs of MultiRing network. Unlike Omega organization, MultiRing organization does not have the disadvantages mentioned in Subsection 2.2.

The first disadvantage of Omega organization does not really affect the MultiRing network performance. The simple reason is that more passes does not necessarily mean more Boolean gates that communication between two nodes has to go through. The number of passes (n) is getting exponentially smaller and smaller than the number of nodes as n increases. Hence, for a large MultiRing network, n or the number of passes can be really neglected. The second disadvantage of Omega organization is no longer a problem in the MultiRing switch organization. For each ring configuration on a MultiRing, each node will be using a completely different path to communicate with its next node as shown in Figure 6. So there is no contention on any configured ring. Tables 1, 2 and 3 below show the paths from source nodes to destination nodes corresponding to the configurations of 1 ring of 8 nodes, 2 rings of 4 nodes and 4 rings of 2 nodes respectively. One will find that the paths from two different source nodes to two different destination nodes in the same configuration are not sharing any links between switch boxes. For example, in the configuration of 1 ring of 8 nodes, the path from P_1 to P_2 is going through S_{00}, S_{01} and S_{32}, the path from P_0 to P_1 is going through S_{00}, S_{11} and S_{12}.

Table 1: Communication path for 1 ring of 8 nodes configuration

Source	Destination	Path
P_0	P_1	S_{00}-S_{11}-S_{12}
P_1	P_2	S_{00}-S_{01}-S_{22}
P_2	P_3	S_{10}-S_{11}-S_{32}
P_3	P_4	S_{10}-S_{01}-S_{02}
P_4	P_5	S_{20}-S_{31}-S_{12}
P_5	P_6	S_{20}-S_{21}-S_{22}
P_6	P_7	S_{30}-S_{31}-S_{32}
P_7	P_0	S_{30}-S_{21}-S_{02}

Table 2. Communication path for 2 rings of 4 nodes configuration

Source	Destination	Path
P_0	P_2	S_{00}-S_{01}-S_{22}
P_1	P_3	S_{00}-S_{11}-S_{32}
P_2	P_4	S_{10}-S_{01}-S_{02}
P_3	P_5	S_{10}-S_{11}-S_{12}
P_4	P_6	S_{20}-S_{21}-S_{22}
P_5	P_7	S_{20}-S_{31}-S_{32}
P_6	P_0	S_{30}-S_{21}-S_{02}
P_7	P_1	S_{30}-S_{31}-S_{12}

Table 3. Communication path for 4 ring of 2 nodes configuration

Source	Destination	Path
P_0	P_4	S_{00}-S_{01}-S_{02}
P_1	P_5	S_{00}-S_{11}-S_{12}
P_2	P_6	S_{10}-S_{01}-S_{22}
P_3	P_7	S_{10}-S_{11}-S_{32}
P_4	P_0	S_{20}-S_{21}-S_{02}
P_5	P_1	S_{20}-S_{31}-S_{12}
P_6	P_2	S_{30}-S_{21}-S_{22}
P_7	P_3	S_{30}-S_{31}-S_{32}

3. Control Data Flow in MultiRing Switch

The path for data going through the MultiRing switch is selected according to the control signals.

3.1. Control Data in a Switch Box

Each switch box in the MultiRing switch takes two data inputs and a control input, and produces two data outputs. Let denote two inputs by I_0 and I_1, two outputs by O_0 and O_1, and the control signal by C. The switch box can be implemented using four AND gates, two OR gates and one NOT gate as shown in Figure 7. From this figure, it is easy to see that I_0 is connected to O_0 and I_1 is connected to O_1 when C = 0, and I_0 is connected to O_1 and I_1 is connected to O_0 otherwise.

Figure 8 shows an alternative approach to implementation of the switch box. In Figure 8, two AND gates, two XOR gates, one OR gate and one NOT gate are used. Hence the gate count has been decreased by 1 in this approach. It is reduced from seven in Figure 7 to six in Figure 8.

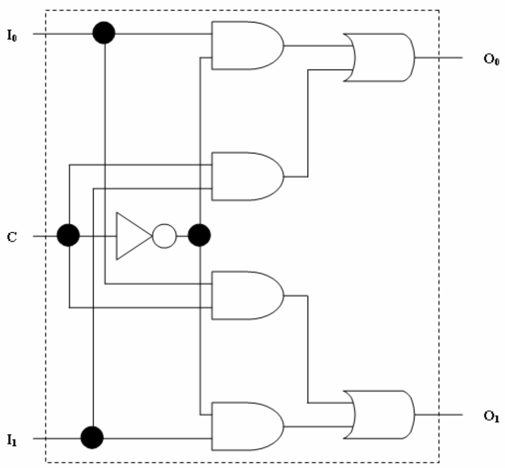


Fig. 7. Switch box implementation.

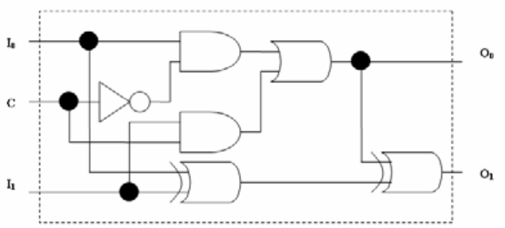


Fig. 8. Alternative implementation of switch box.

3.2. Control Inputs of MultiRing

Given a MultiRing of 2^n nodes, we need n control bits for the n different ring configurations. Let us denote the n control bits by $C_0, C_1, \ldots, C_{n-1}$. Define

$$C_{ij} = C_i \text{ OR } C_{i+1} \text{ OR } C_{i+2} \text{ OR}, \ldots, \text{OR } C_j \text{; i.e.,}$$
$$C_{ij} = C_i + C_{i+1} + C_{i+2} +, \ldots, + C_j$$

for i, j $\in$ {0, 1, 2, …, n-1}and i≤j. C_{ij} defined here are used as control inputs to the switch boxes in a MultiRing switch.
Hence, there are

$$n + (n-1) + \ldots + 1 = n(n+1)/2$$

control inputs for a MultiRing of 2^n nodes. However, there is only a single control input for each switch box. If we denote the switch box in the r^{th} row and the s^{th} column by $S_{(r-1)(s-1)}$ for $r \in \{1, 2, \ldots, 2^{n-1}\}$ and $s \in \{1, 2, \ldots, n\}$ (refer to Figure 6), then assignment of inputs to the switch boxes is shown in the following.

$$\text{Control Input of } S_{rs} = C_{0s} \text{ if } 0 \le r \bmod 2^s < 1 \text{; and}$$
$$\text{Control Input of } S_{rs} = C_{is} \text{ if } 2^{i-1} \le r \bmod 2^s < 2^i \text{ and } i \in \{1, 2, \ldots, n\}$$

As an illustration, let us consider the MultiRing switch for a MultiRing of 8 nodes. The 12 switch boxes as shown in Figure 6 are organized into 4 rows and 3 columns. Using the formula defined above for control inputs of switch boxes, the control input for the switch box S_{rs} (where $r \in \{0, 1, 2, 3\}$ and $s \in \{0, 1, 2\}$) is displayed in Table 4.

Table 4. Control inputs of 8-node MultiRing

Control input for each S_{rs}		**s for columns**		
		0	1	2
r for rows	0	C_{00}	C_{01}	C_{02}
	1	C_{00}	C_{11}	C_{12}
	2	C_{00}	C_{01}	C_{22}
	3	C_{00}	C_{11}	C_{22}

3.3. Control Unit

The control unit on a MultiRing of 2^n nodes creates control signals (Ci, where i $\in$ {0, 1, n-1}), and forms controls inputs (C_{ij}, where i, j $\in$ {0, 1, 2, …, n-1}and i≤j) for switch boxes. As an illustration, Figure 9 shows the organization of the control unit for an 8-node MultiRing.

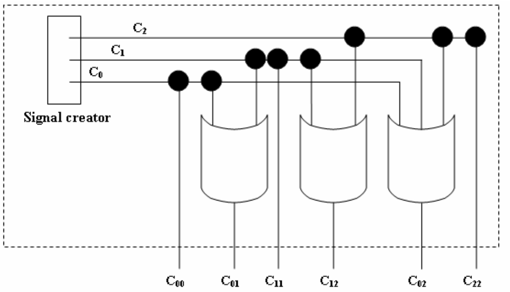


Fig. 9. Control unit for 8-node MultiRing.

It is easy to see that the number of OR gates needed in the control unit of a 2^n-node MultiRing is

Number of control inputs – n.

Hence, the number of gates in the control unit equals to

$$n(n+1)/2 - n = n(n-1)/2.$$

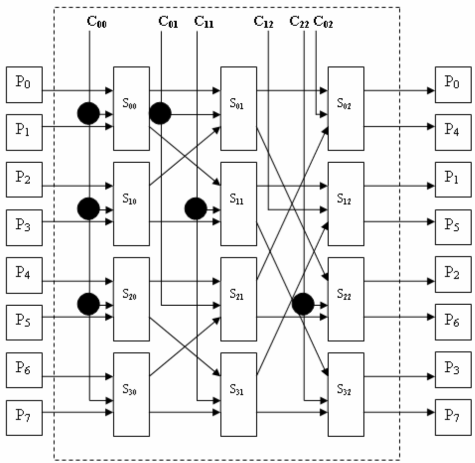


Fig. 10. Data flow in 8-node switch.

3.4. Sample Data Flow in a Switch

Figure 10 shows the data flow with control inputs in an 8-node MultiRing switch. The control inputs are the outputs of the control box displayed in Figure 9.

3.5. Configuration Signal

The signal sequence C_0, C_1, …, C_{n-1} of a 2^n-node MultiRing can be easily set as follows for different ring configurations.

If the configuration is for 2^i rings of 2^{n-i} nodes, where $i \in \{0, 1, 2, \ldots, n-1\}$, then we set $C_i = 1$ and $C_j = 0$ for any $j \neq i$ and $j \in \{0, 1, 2, \ldots, n-1\}$.

As an illustration, let us again consider an 8-node MultiRing. Based on the method for signal setting described above, we have that

1) if the configuration is for 1 ring of 8 nodes, then $C_0 = 1$, $C_1 = C_2 = 0$;
2) if the configuration is for 2 rings of 4 nodes, then $C_1 = 1$, $C_0 = C_2 = 0$;
3) if the configuration is for 4 rings of 2 nodes, then $C_2 = 1$, $C_0 = C_1 = 0$.

Hence, using the control inputs defined in Subsection 3.2, we obtain the inputs of S_{rs} ($r \in \{0, 1, 2, 3\}$ and $s \in \{0, 1, 2\}$) in Table 4 for different ring configurations. The results are shown in Tables 5, 6 and 7 with respect to 1 ring of 8 nodes, 2 rings of 4 nodes and 4 rings of 2 nodes respectively.

Table 5. Control inputs of 1 ring of 8 nodes

Control input for each S_{rs}		**s for columns**		
		0	1	2
r for rows	0	1	1	1
	1	1	0	0
	2	1	1	0
	3	1	0	0

Table 6. Control inputs of 2 rings of 4 nodes

Control input for each S_{rs}		**s for columns**		
		0	1	2
r for rows	0	0	1	1
	1	0	1	1
	2	0	1	0
	3	0	1	0

Table 7. Control inputs of 4 rings of 2 nodes

Control input for each S_{rs}		s for columns		
		0	1	2
r for rows	0	0	0	1
	1	0	0	1
	2	0	0	1
	3	0	0	1

Referring to Figure 10, the paths between adjacent nodes in different ring configurations as shown in Tables 1, 2 and 3 can be easily obtained and found using the results in Tables 5, 6 and 7. For example, looking at the path from P_0 to P_1 in the configuration of 1 ring of 8 nodes, the bit (or data) from P_0 goes to S_{00}. S_{00} switches the bit because its control input is 1. The bit is then sent to S_{11}. S_{11} does not switch the bit because its control input is 0. The bit is then forwarded to S_{12}. S_{12} does not switch the bit either because its control input is 0 too. Hence the bit will arrive at P_1 as expected.

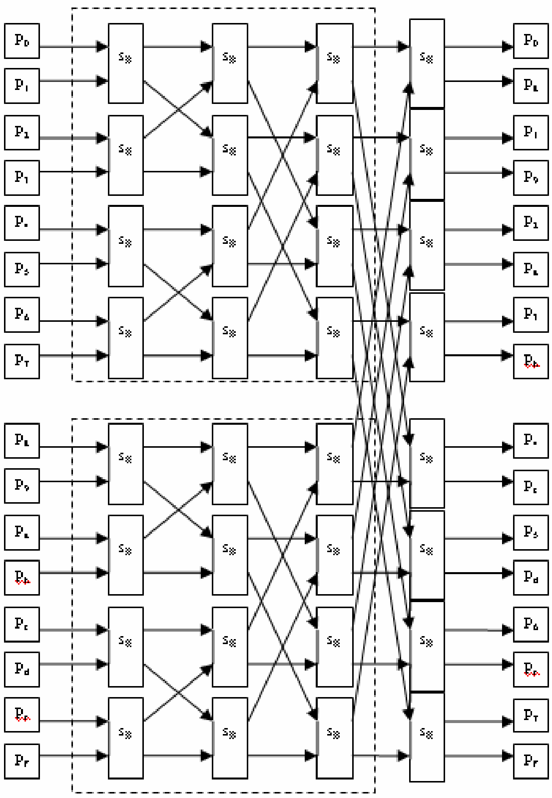


Fig. 11. MultiRing switch organization for 16 nodes.

4. Scalability of MultiRing Switch

MultiRing switch organization proposed in Section 2 is scalable. It can be easily scaled up for a MultiRing containing more nodes. Figure 11 shows the organization for 16 nodes. It is constructed from two 8-node switches and eight switch boxes each having two input ports and two output ports. Inside each dotted rectangle in Figure 11, a MultiRing switch organization for 8 nodes is found.

4.1. 2^n-node MultiRing Switch Organization

Let us now generalize the MultiRing organization. For a 2^n-node MultiRing, the MultiRing switch contains $2^{n-1}n$ switch boxes organized in 2^{n-1} rows and n columns (see, for example, Figure 6 when n=3 and Figure 11 when n=4). It can be constructed using two switches of 2^{n-1}-node MultiRing and 2^{n-1} switch boxes.

The organization of the switch boxes in the left n-1 columns and the top 2^{n-2} rows is the same as that for a 2^{n-1}-node switch. Similarly, the organization of the switch boxes in the left n-1 columns and the bottom 2^{n-2} rows is the same as that for a 2^{n-1}-node switch. Number the input ports (ports on the left hand side of switch boxes) from the top down to the bottom of the switch boxes in the last (right) column as 0, 1, …, 2^n-1. Similarly, number the output ports (ports on the right hand side of switch boxes) from the top to the bottom of the 2nd last column as 0, 1, …, 2^n-1. The output port j in the 2nd last column is connected to input port k at the last column for $j \in \{0, 1, 2, \ldots, 2^n-1\}$ in according to the following.

1) When $0 \le j < 2^{n-1}$,
   if j is even, then k := j;
   else k := j + 2^{n-1} - 1.
2) When $2^{n-1} \le j < 2^n$,
   if j is odd, then k := j;
   else k := j - 2^{n-1} + 1.

Table 8 shows relationship between the values of k and their corresponding values of j for a 16-node switch as shown in Figure 11 using the formula defined above.

Table 8. j-k values for scaling up MultiRing from 8 nodes to 16 nodes

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
k	0	8	2	10	4	12	6	14	1	9	3	11	5	13	7	15

4.2. I/O of MultiRing Switch

Similar to the previous subsection, let us number the input ports of the switch boxes in the first (left) column as 0, 1, …, 2^n-1. We call these ports the *input ports of the MultiRing switch*. Furthermore, let us number the output ports of switch boxes in the last column (right) from the top to the bottom as 0, 1, …, 2^n-1 as well. These output ports are called the *output ports of the MultiRing switch*. If we denote the 2^n nodes on the MultiRing by P_i for i = 0, 1, …, 2^n-1, then we have the following assignment of inputs and outputs on the MultiRing switch.

1) The output from P_i is connected to input port i of the MultiRing switch for each $i \in \{0, 1, \ldots, 2^n-1)$;
2) The input to P_k is connected to the output port j of the MultiRing switch for each $j \in \{0, 1, \ldots, 2^n-1\}$ in according to a perfect shuffle [5] as follows.

If j is odd then k:= 2^{n-1} + (j-1)/2,
else k:= j/2.

Table 9 shows relationship between the node index values of k and their corresponding output values of j of a 16-node MultiRing switch as shown in Figure 11 using the formula for perfect shuffle described above.

Table 9. j-k values for connecting output ports of a 16-node MultiRing switch to nodes

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
k	0	8	1	9	2	10	3	11	4	12	5	13	6	14	7	15

5. Comparison

Arabnia and Smith designed a switch for the scalable MultiRing network in their paper [4]. We compare their design with our new design presented in this paper.

5.1. Switch Boxes

For a 2^n-node MultiRing, both of designs use $2^{n-1}n$ switch boxes.

Each switch box for the old design contains two multiplexers, which need at least four AND gates, one OR gate and two OR gates for implementation. So in total, there are

$$(4+1+2) \times 2^{n-1}n = 7n2^{n-1}$$

gates needed for switch implementation.

On the other hand, our new switch design requires maximum six gates only for switch box implementation as shown in Figure 8. This gives a total of $6n2^{n-1}$ gates for all switch boxes in the MultiRing switch.

Hence, the old design needs $n2^{n-1}$ more gates than the new design for all switch boxes.

In additional to the gate count, each switch box in the old design takes three inputs excluding the control input compared to the two inputs only in the new design. If we call each input interface on a switch box an input port, then the old design again needs $n2^{n-1}$ more input ports than the new design.

As $n2^{n-1}$ is significantly large even when n is not very big, the old design needs a lot more hardware for implementation. This can be a big disadvantage in VLSI chip design.

5.2. Control Units

The old design does not have any gate count in the control unit as a single control bit is used for all switch boxes in the same column.

Our new design needs additional n(n-1)/2 gates in the control unit. But his amount is really small compared to the total number of gates needed in the switch design, which is

$$n(n-1)/2 + 6n2^{n-1}.$$

Hence, the additional gate count is negligible.

5.3. Wiring for Scaling

Without modification, re-wiring is needed in the old design if more nodes are added to the MultiRing. For example, when two existing 8-node MultiRings are to be combined to form a 16-node MultiRing, the nodes previously plugged into the switches must be unplugged and then be re-plugged in order to function properly.

In our new design, to combine two MultiRing into a bigger MultiRing, it is no longer necessary to unplug the connections of nodes to the switches. All we need to do is to plug the ports (output ports) on the back plane of the switches to a set of switch boxes without shutting down the nodes (see Figure 11 for illustration).

Similarly, when separating an existing MultiRing into two smaller MultiRings, the new design does not need the connections to the nodes to be unplugged.

As it may not be practical to plug and unplug the connections when the node count is large, our new design is a more realistic and more efficient.

6. Conclusion

In this paper, we have proposed a new design of MultiRing switch. It is scalable, efficient, realistic, doable and practical.

The switched has been designed for uni-directional MultiRing, which has many applications. One may find an application of uni-directional MultiRing to Machine Vision in [3].

For a bi-directional MultiRing, the switch organization is the same. An acceptable mount of gates must be added to the control unit. We will show the detailed design of switch for bi-directional MultiRing in another paper. One other alternative is to interchange the inputs and outputs in the old switch design, and keeps the same amount of gate count as the old design.

The next step following the switch design is the research work for message passing through the MultiRing switch. Various methods for message passing have been proposed. We will present a technique for efficient message passing in a paper following this one.

References

[1] H. R. Arabnia and M. A. A. Oliver, *A Transputer Network for Fast Operations on Digitised Images*, International Journal of Eurographics Association (Computer Graphics Forum), Vol. 8, No. 1, 1987, pp.3-12.

[2] Hamid Arabnia and Xiangjian He, *Edge Detection Using MultiRing on Spiral Architecture*, Proc. International Conference on Parallel and Distributed Processing Techniques and Applications, Las Vegas, 2004, pp.413-419.

[3] Xiangjian He and Hamid Arabnia, *Parallel Edge Detection Using Uni-Directional MultiRing on Spiral Architecture*, Proc. International Conference on Parallel and Distributed Processing Techniques and Applications, Las Vegas, 2004, pp.420-426.

[4] Hamid Arabnia and Jeffrey Smith, *A Reconfigurable Interconnection Network For Imaging Operations And Its Implementation Using A Multi-Stage Switching Box,* The Proceedings of the seventh annual international high performance computing conference. The 1993 High Performance Computing: New Horizons, Supercomputing Symposium. Calgary, Alberta, Canada, June, 1993, pp. 349-357.

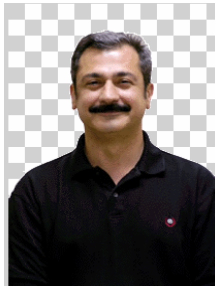
[5] H. S. Stone, *High Performance Computer Architecture*, Addison-Wesley, 1990.

[6] C. Mead and L. Conway, *VLSI Systems Design*, Addison-Wesley, 1980.

[7] John. L. Hennessy and David A. Patterson, *Computer Architecture*, Morgan Kaufmann, 2003.

[8] Sajjan G. Shiva, *Computer Design and Architecture*, Harper Collins, 1991.

Xiangjian He received his PhD degree in Computing Sciences from the University of Technology, Sydney (UTS) in 1999. Dr He is currently an Associate Professor at UTS. He is the Deputy Director of the UTS university-level Computer Vision Research Group, and a Senior Member of IEEE. His research interests are in the areas of Computer Vision, Image Processing, Computer Networks, and Parallel and Distributed Computing. He is the chair of the 3rd IEEE Sponsored International Conference on Information Technology and Applications (ICITA2005), and the chief editor of the conference proceedings published by IEEE CS Society. He has received many research grants including four Australian national (ARC) grants. He has had over 100 research refereed publications.



Hamid R. Arabnia received a Ph.D. degree in Computer Science from the University of Kent (Canterbury, England) in 1987. Dr. Arabnia is currently a Professor of Computer Science at University of Georgia (Georgia, USA), where he has been since late 1987. His research interests include parallel algorithms, reconfigurable machines, interconnection networks, and applications of parallel processing in remote sensing and imaging science. Prof. Arabnia has chaired many national and international conferences and technical sessions in these areas. He is Editor-in-Chief of The Journal of Supercomputing (Springer) and is on the editorial boards of 13 other journals. Prof. Arabnia is the recipient of William F. Rockwell, Jr. Medal for promotion of multi-disciplinary research (Rockwell Medal is International Technology Institute's highest honor). In 2000, Prof. Arabnia was indicted to the World Level of the Hall of Fame for Engineering, Science and Technology (The World Level is the highest possible level for a living person - there are two higher levels which are posthumous.) Prof. Arabnia has published extensively in journals and refereed conference proceedings; he has over 220 publications (including edited and co-authored books). Prof. Arabnia has been the PI/Co-PI of over $4M of grant fundings.