A Simple and Fast Parallel Round-Robin Arbiter for High-speed Switch Control and Scheduling

S. Q. Zheng †, Mei Yang †, John Blanton ‡, Prasad Golla ‡, Dominique Verchere ‡
† Department of Computer Science, University of Texas at Dallas, Richardson, TX 75083-0688, USA
‡ Research & Innovation, Alcatel USA, Plano, TX 75075, USA

Abstract— **The design of a fast and fair arbiter is critical to the efficiency of the scheduling algorithm, which is the key to the performance of a high-speed packet switch. In this paper, we propose a parallel round-robin arbiter (PRRA) design based on a binary-tree structure. We show that our design is simpler and faster than existing round-robin arbiter designs.**

I. Introduction

There are four major aspects in the design and implementation of a fast packet switch: (1) a cost-effective switch matrix that provides many non-conflicting paths between inputs and outputs; (2) a scheduling algorithm that chooses input packets to send to outputs; (3) a fast arbitration scheme for resolving output contentions; and (4) a fast mechanism that generates control signals for switching elements to set up non-conflicting paths between inputs and outputs of the switch matrix. For a given switch matrix, the solution of (3) can be used to implement (2) and (4). Hence, the design of a fast arbitration is of critical importance to the design of a high-performance packet switch. In this paper, we focus our discussion on a cell-based crossbar switch for unicast I/O connections. In such a switch, variable length packets are segmented into cells upon arrival, transferred across the switch S, and then reassembled again before they depart.

Consider an $N \times N$ non-blocking switch S with N inputs I_i and N outputs O_i, $0 \leq i \leq N$. To avoid head-of-line blocking [1], virtual output queues (VoQs) are employed at each input. In each cell slot, each input I_i may have multiple connection requests, each being associated with an output, and each output may receive multiple requests. In order for the switch matrix to operate properly, at most one connection request for each output can be granted during any cell slot. Such a setting is called a conflict-free connection. The cell scheduling problem for S can be abstracted as finding a maximum weight matching (which corresponds to an optimal set of conflict-free connections) in a bipartite graph for each cell slot to satisfy a set of performance requirements, such as cell delay, fairness, etc. Since the time complexity of all sequential maximum weight matching (MWM) algorithms is too high to be implemented in hardware, many distributed hardware scheduling algorithms have been proposed ([2], [3], [5], [6], [8], [9]). Instead of finding an MWM, these algorithms assume that all requests have the same weight and approximate a maximum size matching (MSM).

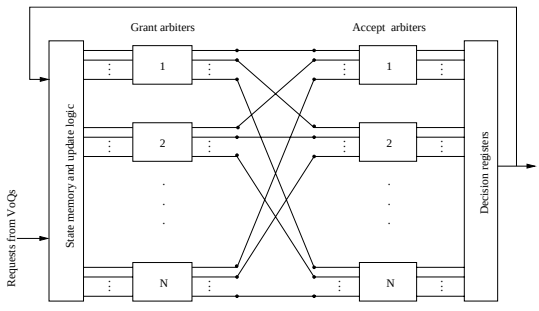

Fig. 1. Hardware scheduler architecture for a crossbar switch.

Figure 1 shows the hardware scheduler architecture on which these MSM approximation algorithms are implemented. In such a scheduler, each input/output is associated with an N-input arbiter. A simple scheme for assuring fairness of an arbiter is the round-robin dynamic priority assignment. In this scheme, all request inputs are arranged as a directed loop. In this loop, the request input that follows the request input being served in the current cell slot is assigned the highest priority in the next cell slot. The priority of other request inputs is determined by their positions in the loop from the request input that is being served. The request input that releases the output will be assigned the lowest priority.

In [7], Gupta and McKeown surveyed previously well-known round-robin arbiter designs, and proposed two new programmable priority encoder (PPE) designs, both having $O(\log N)$ gate delay. Both designs are rather too complicated for the simple round-robin arbitration scheme. In [4], Chao, Lam and Guo proposed a parallel arbiter. This arbiter has a tree structure of $O(\log N)$ levels. Clearly, it has $O(\log N)$ gate delay. By associating a 1-bit memory with each internal node of the tree, this arbiter implements the round-robin selection rule only under the condition that all N requests are present in each cell slot. When there are $N/2+1$ request inputs repeatedly requesting service in a specific pattern, this arbiter grants one request input $N/2$ times more than each of the remaining $N/2$ request input, resulting unfairness.

In this paper, we propose a new parallel round-robin arbiter design (PRRA) based on a binary-tree structure. We formally prove that the proposed PRRA design achieves the round-robin fairness under any input patterns. The latency of our design is $O(\log N)$ gate delay and consumes $O(n)$ gates. Compared to the designs of PPE, our design is much simpler and practically faster.

II. Design of Parallel Round-Robin Arbiter

In this section, we illustrate the parallel round-robin arbiter (PRRA) design taking the example of an output arbiter O_j with N request inputs. Without loss of generality, we assume that $N = 2^n$, $n > 2$. The block diagram of an 8-input PRRA arbiter is shown in Figure 2. The structure of this circuit can be viewed as a $(\log N+1)$-level complete binary tree. The nodes are partitioned into levels. The node in level 0 is called the root node (or, simply, r-node). The nodes in level $\log N$ are called leaf nodes (or, simply, l-nodes). Leaf nodes, as represented by thick rectangles in Figure 2, are connected as a ring. All remaining nodes are called internal nodes. Internal nodes are divided into two types. Type 1 internal nodes (or, simply, i_1-nodes) are those in level $(\log N - 1)$, and type 2 internal node (or, simply, i_2-nodes) are those in levels 1 through level $(\log N - 2)$.

The structure of l-nodes and their connections are shown in Figure 3, in which a dashed rectangle represents an l-node, which consists of an RS flip-flop $Head$. Input $Request_i$ (R_i), which is from I_i, being 1 indicates that I_i is requesting to be connected to O_j. The output H_i of $Head_i$ being 1 indicates that a linear priority scheme of I_i, $I_{(i+1) \bmod N}, \cdots, I_{(i+N-1) \bmod N}$ is currently used, with I_i having the highest priority to be connected to O_j.

At any arbitration time, one and only one of $Head_i$'s can be in the 1-state (assume that initially $Head_0$ is set to 1). Suppose that currently $Head_k$ is in 1-state, i.e. $H_k = 1$. Then, if any I_r is granted connection to O_j, $Head_{(r+1) \bmod N}$ is set to 1 and $Head_k$ is reset to 0; otherwise $Head_k$ remains to in 1-state. Such a rotating assignment is enforced by G_i's.

When I_k requests to connect to O_j, $R_k = 1$. The outputs R_i's and H_i's are used by the arbiter to determine which l-node will be selected. Suppose that currently $H_k = 1$. There are two cases.

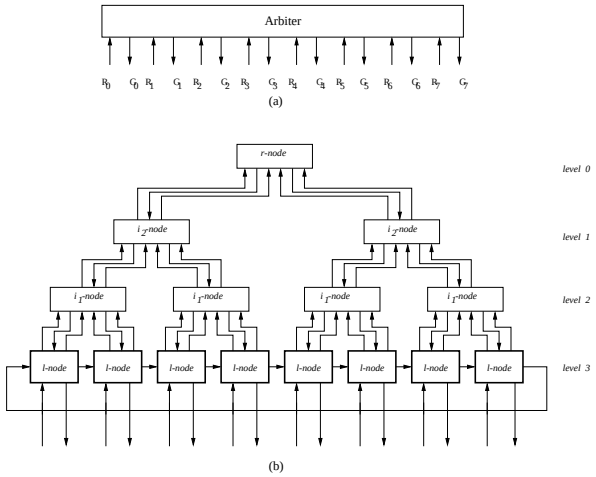

Fig. 2. Structure of an 8-input parallel round-robin arbiter. (a) Block diagram. (b) The tree structure.

S^1	S^0	states
0	0	$H_k = 1$ is not in the subtree rooted at u and the number of $R_i = 1$ in this subtree is 0
0	1	$H_k = 1$ is not in the subtree rooted at u, and the number of $R_i = 1$ in this subtree is > 0.
1	0	$H_k = 1$ is in the subtree rooted at u, and the number of $R_i = 1$ such that $i \geq k$ in this subtree is 0.
1	1	$H_k = 1$ is in the subtree rooted at u, and the number of $R_i = 1$ such that $i \geq k$ in this subtree is > 0.

TABLE I

S^0 AND S^1 USED TO CODE THE STATES OF l-NODES OF A SUBTREE ROOTED AT AN i_1-NODE OR AN i_2-NODE.

Case 1: For some i, $R_i = 1$. In this case, the arbiter returns

$$G_{(k+a) \bmod N} = 1, \text{where}$$
$$a = \min\{b | R_{(k+b) \bmod N} = 1, 0 \leq b \leq N-1\} \quad (1)$$

and

$$G_c = 0 \quad \text{for } c \neq a. \quad (2)$$

The following flip-flops will be affected:

$$Head_k \leftarrow 0, Head_{(k+a+1) \bmod N} \leftarrow 1. \quad (3)$$

$Grant_{(k+a) \bmod N} = 1$ is sent to $I_{(k+a) \bmod N}$.

Case 2: $R_i = 0$ for $0 \leq i \leq N-1$. In this case, the arbiter returns $G_i = 0$ for $0 \leq i \leq N-1$. No l-node is selected because there is no requests. All $Head$'s in l-nodes remain unchanged. In particular, $Head_k$ remains to be in 1-state.

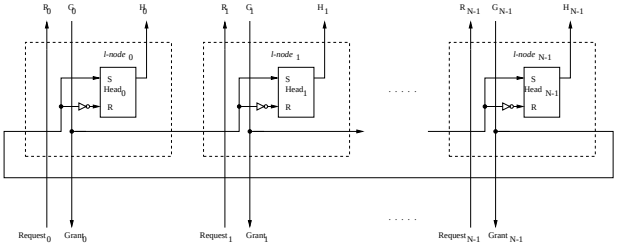

Fig. 3. l-nodes.

Let u be an i_1-node or an i_2-node. We use outputs S^0 and S^1 of u to code the states of l-nodes in the subtree rooted at u as in Table I. The coding is first generated by i_1-nodes for all subtrees of two l-nodes. An i_1-node is a combinational circuit, as shown in the dashed rectangle in Figure 4. It has four inputs from its two child nodes (which are leaf nodes): R_L and H_L from its left child, and R_R and H_R from its right child. It provides two outputs S^0 and S^1 to its parent node. If an i_1-node is the left (respectively, right) child of its parent node, then S^0 and S^1 are S^0_L and S^1_L (respectively, S^0_R and S^1_R) of its parent, respectively. An i_1-node has one input G from its parent node. If this i_1-node is the left (respectively, right) child node of its parent node, this input is the output G_L (respectively, G_R) of its parent node. This i_1-node has two outputs G_L and G_R, which in turn are G inputs of its left and right child node, respectively. The I/O relations of an i_1-node are specified by the following Boolean functions:

$$S^0 = R_R + \overline{H_R} \cdot R_L \quad (4)$$
$$S^1 = H_L + H_R \quad (5)$$
$$G_L = G \cdot R_L \cdot (H_L + H_R \cdot \overline{R_R} + \overline{H_L} \cdot \overline{H_R}) \quad (6)$$
$$G_R = G \cdot R_R \cdot (H_R + H_L \cdot \overline{R_L} + \overline{H_L} \cdot \overline{H_R} \cdot \overline{R_L}) \quad (7)$$

An i_2-node is a combinational circuit, as shown in the dashed rectangle in Figure 5. It has four inputs from its two child nodes (which are either i_1-nodes or i_2-nodes): S^0_L and S^1_L from its left child, and S^0_R and S^1_R from its right child. It provides two outputs S^0 and S^1 to its parent node. If an i_2-node is the left (respectively, right) child of its parent node, then its S^0 and S^1 are S^0_L and S^1_L (respectively, S^0_R and S^1_R) of its parent, respectively. As an i_1-node, an i_2-node has one input G from its parent node. If this i_2-node is the left (respectively, right) child node of its parent node, this input is the output G_L (respectively, G_R) of its parent node. This i_2-node has two outputs G_L and G_R, which in turn are G inputs of its left and right child node, respectively. The I/O relations of an i_2-node are specified by the following Boolean functions:

$$S^0 = S^0_R + S^0_L \cdot \overline{S^1_R} \quad (8)$$
$$S^1 = S^1_L + S^1_R \quad (9)$$
$$\begin{aligned} G_L &= G \cdot (S^1_L \cdot S^0_L + \overline{S^1_R} \cdot \overline{S^0_R} + S^0_L \cdot \overline{S^0_R} \\ &\quad + \overline{S^1_L} \cdot S^0_L \cdot S^1_R \cdot \overline{S^0_R} + \overline{S^1_L} \cdot S^0_L \cdot \overline{S^1_R} \cdot S^0_R) \\ &= G \cdot (S^1_L \cdot S^0_L + \overline{S^1_R} \cdot \overline{S^0_R} + S^0_L \cdot \overline{S^0_R} + S^0_L \cdot \overline{S^1_R}) \quad (10) \end{aligned}$$
$$\begin{aligned} G_R &= S^1_R \cdot S^0_R + \overline{S^1_L} \cdot \overline{S^0_L} + S^1_L \cdot \overline{S^0_L} \cdot \overline{S^1_R} \cdot S^0_R \\ &= G \cdot (S^1_R \cdot S^0_R + \overline{S^1_L} \cdot \overline{S^0_L} + \overline{S^0_L} \cdot S^0_R) \quad (11) \end{aligned}$$

An r-node is a sub-circuit of Figure 5, as shown in the dashed rectangle in Figure 6. It has four inputs from its two child nodes: S^0_L and S^1_L from its left child, and S^0_R and S^1_R from its right child. It provides two outputs G_L and G_R, which in turn are G inputs of its left and right child node, respectively. The I/O relations of an r-node are specified by the following Boolean function:

$$\begin{aligned} G_L &= S^1_L \cdot S^0_L + \overline{S^1_R} \cdot \overline{S^0_R} + \overline{S^1_L} \cdot S^0_L \cdot S^1_R \cdot \overline{S^0_R} \\ &= S^1_L \cdot S^0_L + \overline{S^1_R} \cdot \overline{S^0_R} + S^0_L \cdot \overline{S^0_R} \quad (12) \end{aligned}$$
$$G_R = S^1_R \cdot S^0_R + \overline{S^1_L} \cdot \overline{S^0_L} + \overline{S^0_L} \cdot S^0_R \quad (13)$$

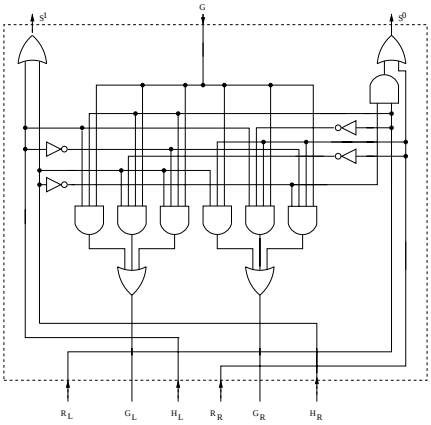

Fig. 4. i_1-node structure.

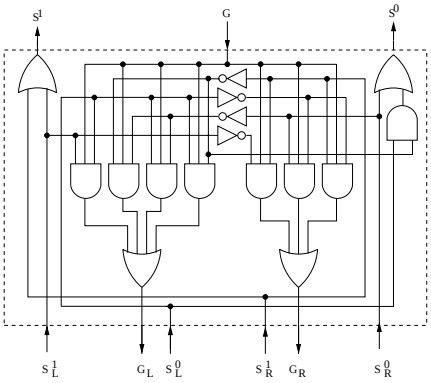

Fig. 5. i_2-node structure.

Now,

III. Correctness

In this section, we show that our design achieves pure round-robin fairness.

Lemma 1: Let u be an i_1-node or an i_2-node of the PRRA. u computes its S^0 and S^1 that satisfy their definitions given in Table I.

Proof: By Equations (4) and (5), it is easy to see that each i_1-node computes its outputs S^0 and S^1 correctly. Assume that the inputs $S_L^1 S_L^0$ and $S_R^1 S_R^0$ of any i_2-node u correctly code the state of u's left and right subtree, respectively. Applying induction on the node levels using Equations (8) and (9), we conclude that the outputs S^0 and S^1 of any i_2-node satisfy their definitions given in Table I. □

Lemma 2: In any level of internal nodes of the PRRA, there is exactly one node whose output $S^1 = 1$.

Proof: Since there is exactly one l-node whose head register can be in 1-state, by the coding of $S^1 S^0$, the claim directly follows from Lemma 1. □

Lemma 3: The following statements hold for the r-node:

(i) The request to be granted, if any, must be found in the left subtree of the r-node if and only if G_L of the r-node is 1.

(ii) The request to be granted, if any, must be found in the right subtree of the r-node if and only if G_R of the r-node is 1.

(iii) One and only one of G_L and G_R of the r-node is 1.

Proof: According to our round-robin priority scheme (Equation (1), (2), (3), the request to be granted, if any, must be found in the left subtree of the r-node if one of the following conditions hold:

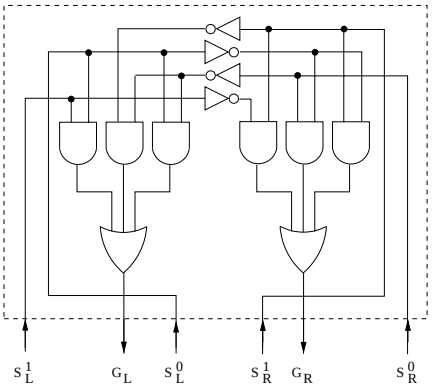

Fig. 6. r-node structure.

(a) $H_k = 1$ is in the left subtree of the r-node, and there is a request $R_i = 1$ in the left subtree of the r-node such that $i \geq k$.

(b) $H_k = 1$ is in the left subtree of the r-node and there is no request $R_i = 1$ in the right subtree of the r-node.

(c) $H_k = 1$ is in the right subtree of the r-node, there is no request $R_i = 1$ such that $i \geq k$ and there is at least one $R_i = 1$ in the left subtree of the r-node.

By Lemma 1, these conditions correspond to $S_L^1 S_L^0 = 11$, $S_R^1 S_R^0 = 00$ and $S_L^1 S_L^0 S_R^1 S_R^0 = 0110$, respectively, of Equation (12). If none of these conditions holds, the request to be granted, if any , must be found in the right subtree of the r-node. This proves (i). The argument for (ii), which corresponds to Equation (13), is almost the same.

By Lemma 2, the following midterms of S_L^1, S_L^0, S_R^1, and S_R^0 cannot be true: $S_L^1 S_L^0 S_R^1 S_R^0 = 0 \times 0\times$ and $S_L^1 S_L^0 S_R^1 S_R^0 = 1 \times 1\times$, where $\times$ denotes "don't-care". The mutual exclusiveness of G_L and G_R can be easily established by the fact that all these midterms are shaded in Equation (12) and (13). □

Lemma 4: In any level of i_2-nodes, there is exactly one node with input $G = 1$. Let u be any i_2-node with input $G = 1$. The following statement holds for u:

(i) The request to be granted, if any, must be found in the left subtree of u if and only if G_L of u is 1.

(ii) The request to be granted, if any, must be found in the right subtree of u if and only if G_R of u is 1.

(iii) One and only one of G_L and G_R of u is 1.

Proof: By Lemma 2, $S_L^1 S_L^0 \neq 00$ for the r-node. By Equations (12) and (13), we know that if $S_L^1 S_L^0 = 00$ then $G_L G_R = 01$, and if $S_R^1 S_R^0 \neq 00$ then $G_L G_R = 10$. According to Equations (10) and (11), we list in Table II all possible combinations of the inputs and outputs of u. By an induction, we know that in any level of i_2-nodes there is exactly one node with input $G = 1$, and one and only one of G_L and G_R of u is 1.

Now let us prove (i). According to our round-robin priority scheme (Equations (1), (2) and (3)), the request to be granted, if any, must be found in the left subtree of u if one of the following conditions hold:

(a) $H_k = 1$ is in the left subtree of u, and there is a request $R_i = 1$ in the left subtree of u such that $i \geq k$.

(b) $H_k = 1$ is in the left subtree of u and there is no request $R_i = 1$ in the right subtree of u.

(c) $H_k = 1$ is in the right subtree of u, there is no request $R_i = 1$ such that $i \geq k$ and there is at least one $R_i = 1$ in the left subtree of u.

(d) $H_k = 1$ is not in the tree rooted at u, and there is at least one request in each of u's left and right subtrees.

Compared with the proof of (i) of Lemma 3, one additional condition is considered, which is (d). We add midterm $S_L^1 S_L^0 S_R^1 S_R^0 = 0101$ to Equation (12), corresponding to this condition, and obtain Equation (10). Clearly, one of conditions (a), (b), (c) and (d) holds

S_L^1	S_L^0	S_R^1	S_R^0	S^1	S^0	G_L	G_R
0	0	0	1	0	1	0	1
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	0
0	0	1	0	1	0	0	1
0	1	1	0	1	0	1	0
1	0	0	0	1	0	1	0
0	0	1	1	1	1	0	1
0	1	1	1	1	1	0	1
1	0	0	1	1	1	0	1
1	1	0	0	1	1	1	0
1	1	0	1	1	1	1	0

TABLE II

POSSIBLE I/O OF AN i_2-NODE, ASSUMING ITS INPUT $G = 1$

for the two i_2-nodes which are child nodes of the r-node. Then, (i) follows from a simple induction on i_2-nodes in level 1 through level ($\log N$-2). The proof of (ii) and (oii) is similar to the proof of (ii) and (iii) of Lemma 3. □

Theorem 1: PRRA operates correctly.

Proof: Consider G_L and G_R of any i_1-node v that takes care of requests R_L and R_R from its two child nodes, which are l-nodes. By Lemma 4, there is exactly one i_1-node with input $G = 1$, and, for this node, $S^1S^0 \neq 00$. The request to be granted, if any, must be found in the subtree rooted at this node. Let v be this i_1-node, and let the request corresponding to R_L and R_R of v be R_{i^*} and R_{i^*+1}, respectively.

If $S^1S^0 = 01$ for v, then $R_{i^*}R_{i^*+1} = 01$, 10 or 11, and $H_{i^*}H_{i^*+1} = 00$. The term $G \cdot R_L \cdot \overline{H_L} \cdot \overline{H_R}$ of Equation (6) sets G_{i^*} to 1 for $R_{i^*}R_{i^*+1} = 10$ or 11, and the term $G \cdot R_R \cdot \overline{H_L} \cdot \overline{H_R} \cdot \overline{R_L}$ of Equation (7) sets G_{i^*+1} to 1 for $R_{i^*}R_{i^*+1} = 01$.

If $S^1S^0 = 10$ for v, then $R_{i^*}R_{i^*+1} = \times 0$ and $H_{i^*}H_{i^*+1} = 01$, or $R_{i^*}R_{i^*+1} = 00$ and $H_{i^*}H_{i^*+1} = 10$. The term $G \cdot R_L \cdot H_R \cdot \overline{R_R}$ of Equation (6) sets G_{i^*} to 1 only when $R_{i^*}R_{i^*+1} = 10$.

If $S^1S^0 = 11$ for v, then $R_{i^*}R_{i^*+1} = 1\times$ and $H_{i^*}H_{i^*+1} = 10$, or $R_{i^*}R_{i^*+1} = \times 1$ and $H_{i^*}H_{i^*+1} = 01$. The term $G \cdot R_L \cdot H_L$ of Equation (6) sets G_{i^*} to 1 when $R_{i^*}R_{i^*+1} = 1\times$ and $H_{i^*}H_{i^*+1} = 10$. The term $G \cdot R_R \cdot H_R$ of Equation (7) sets G_{i^*+1} to 1 when $R_{i^*}R_{i^*+1} = \times 1$ and $H_{i^*}H_{i^*+1} = 01$. The term $G \cdot R_R \cdot H_L \cdot \overline{R_L}$ of Equation (7) sets G_{i^*+1} to 1 when $R_{i^*}R_{i^*+1} = 01$ and $H_{i^*}H_{i^*+1} = 10$.

All possible cases are covered by the proposed PRRA design. Therefore, PRRA correctly grants a request, if any, following the linear priority defined with H_k as the head. It is easy to see that the RS flipflops correctly select the next value of k once G_i's are generated. We conclude that PRRA correctly implements the functions described in Case 1 and Case 2 at the beginning of this section. □

IV. Simulation Results

In terms of time complexity, the proposed PRRA design takes $O(\log N)$ gate level delay to get the grants ready. The area cost of the binary tree structure is $O(N)$ gates.

Simulations of the proposed PRRA design and PPE (PPE_only_smpls in [7]) are conducted on Altera's ACEX1K series CPLD (FPGA). We focus on minimizing the delay from requests R_is to grants G_i. Table III compares the timing and area cost of PRRA and PPE in terms of ns t and number c of logic cells (LCs), with entries t/c. All these designs are optimized under the same operating conditions and the tool is directed to achieve the fastest implementation of each design. It shows that PRRA is faster than PPE for $N >= 16$ and PRRA consumes much fewer logical cells than PPE under all N configurations. Further, we can see that PRRA is more scalable than PPE due to its binary-tree structure.

Design	**N=8**	**N=16**	**N=32**	**N=64**
PPE	13.8/41	17.5/137	30.0/676	39.9/2600
PRRA	15.3/34	17.4/76	19.9/172	25.5/318

TABLE III

Timing and area cost of Design PPE and Design PRRA.

V. Concluding Remarks

In this paper, we proposed a parallel round-robin arbiter and showed that our design is simpler, more scalable than existing round-robin arbiter designs while achieving pure round-robin fairness. Practically the proposed PRRA design is faster and consumes less area than the programmable priority encoder [7]. As a basic building block, PRRA is very useful for implementing well-known iSLIP, DRR and other round-robin matching scheduling algorithms for high packet switches with input buffers. It is also useful for other applications, such as the arbitration of a shared bus, arbitration for permutation networks such as the 3-stage Clos network [11] and multi-stage Benes networks [10].

We can replace N RS flip-flops by an encoder, a decoder and a $\log_2 N$-bit register to make the proposed PRRA programmable. Such a programmable arbiter can operate in two modes. Normally, the arbiter uses round-robin policy to select requests. By providing the register with a specific value, the arbiter can assign any request the highest priority.

References

[1] M. J. Karol, M. G. Hluchyj and S. P. Morgan, "Input vs. Output Queueing on a Space-Division Packet Switch", *IEEE Transaction on Communications*, Vol. 35, No. 12, pp. 1347-1356, 1987.

[2] T. Anderson, S. Owicki, J. Saxe and C. Thacker, "High-Speed Switch Scheduling for Local-Area Networks", *ACM Transactions on Computer Systems*, vol. 1, no. 4, pp. 319-352, 1993.

[3] J. Chao, "Saturn: A Terabit Packet Switch Using Rual Round-Robin", *IEEE Communications Magazine*, vol. 38, no. 12, pp. 78-84, 2000.

[4] H. J. Chao, C. H. Lam, and X. Guo, "A Fast Arbitration Scheme for Terabit Packet Switches", *Proceedings of Globecom 1999*, pp. 1236-1243, 1999.

[5] N. McKeown, "The iSLIP Scheduling Algorithm for Input-Queued Switches", *IEEE/ACM Transactions on Networking*, vol. 7, no. 2, pp. 188-201, 1999.

[6] N. Mckeown, M. Izzard, A. Mekkittikul, W. Ellersick and M. Horowitz, "The Tiny Tera: A Packet Switch Core", *IEEE Micro. Magazine*, Jan.-Feb., pp. 26-33, 1997.

[7] P. Gupta and N. Mckeown, "Designing and Implementing a Fast Crossbar Scheduler", *IEEE Microelectronics*, Vol. 19, No. 1, pp. 20-29, 1999.

[8] D. N. Serpanos and P. I. Antoniadis, "FIRM: A Class of Distributed Scheduling Algorithms for High-Speed ATM Switches with Multiple Input Queues", *Proc. of IEEE Infocom2000*, pp. 548-555, 2000.

[9] Y. Jiang and M. Hamdi, "A Fully Desynchronized Round-Robin Matching Scheduler for a VoQ Packet Switch Architecture", *2001 IEEE workshop on HPSR*, pp. 407-412, June 2001.

[10] V. E. Benes, "On Rearrangeable Three-stage Connecting Networks," *Bell System Technical Journal,* vol. 41, no. 5, pp. 1481-1492, Sep. 1962.

[11] C. Clos, "A study of Nonblocking Switching Networks," *The Bell System Technical Journal*, 32, pp. 406-424, 1953.