A Optimal Arbiter Design for NoC

Yun-Lung Lee, Jer Min Jou, Yen-Yu Chen, and Sih-Sian Wu
Department of Electrical Engineering, National Cheng Kung University
Tainan, Taiwan, R.O.C.
Email: Prof. Jer Min Jou (周哲民), jou@j92a21.ee.ncku.edu.tw

Abstract*- A new fast centralized arbiter, which is a modular design and easy for hardware implementation, is proposed. We had derived the state diagram of the arbiter as well as its truth tables and Karnaugh maps, and had designed a set of optimal Boolean functions and the corresponding circuit for the arbiter. This new arbiter is fair for any input combinations and faster than all previous arbiters we knew. Using Synopsys design tools with TSMC 0.18μm technology, the design results have shown that our arbiter has 22.8% improvement of execution time and 39.1% of cost (area) reduction compared with the existing fastest arbiter, SA [7]. Because of this small arbiter's the high-performance, it is extremely useful for the realizations of NoC routers, MPSoC arbitration, and ultra-high-speed switches. This new arbiter is being applied for a patent of the ROC (application No.: 0971xxxxx).*

Keywords: Arbiter, Optimal, Boolean functions.

1. Introduction

Following the rapid technological evolution, the complexity becomes one of the most constraining aspects in the design of embedded systems. Cost and timing issues come along to add to the difficulties in realization of network-on-chip [1], NoC, applications, where many IPs (Intellectual Property) such as processor cores, memories, DSP processors and peripheral devices are placed together, on a single die. These modules communicate, most often, by means of a shared resource, the on-chip network. The increasing complexity of the individual devices, the increasing demand for higher bandwidth on the network lines and an operating frequency hitting new limits with almost every new design, place the communication and/or computation resources arbitration being the performance bottleneck of the NoC system.

Arbiters are a fundamental component in systems containing shared resources, and a centralized arbiter is a tightly integrated design for its input requests. In this study, we propose a new centralized arbiter, which may be used in arbitration of a crossbar switch in NoC routers [2], computer networks [9], or any shared resources. Fig. 1 shows the block diagram of an $n \times n$ switch and a crossbar switch fabric implemented with many transmission gates as switches between input ports and output ports. Each input port contains n virtual output queues (VOQs) to avoid head-of-line blocking. The task of the arbiter is to decide a set of contention-free connection between input and output ports by turning ON/OFF those transmission gates.

Current designs in NoC typically use standard round-robin token passing schemes for bus arbitration [1, 2, 9]. In computer network packet switching, previous researches in round-robin algorithms have reported results on an iterative round-robin algorithm (iSLIP) [3] and a dual round-robin matching (DRRM) algorithm [4]. The iSLIP authors implement an $n \times n$ switch arbiters in hardware which they call a Programmable Priority Encoder (PPE) [6]. Furthermore, Chao et al. describe a design of a round-robin arbiter for a packet switch [5]. Chao et al. refer to their hardware design as a Ping Pong Arbiter (PPA). In general, the goal of a switch arbiter in a packet switch is to provide control signals to the crossbar switch fabric as shown in Fig. 1.

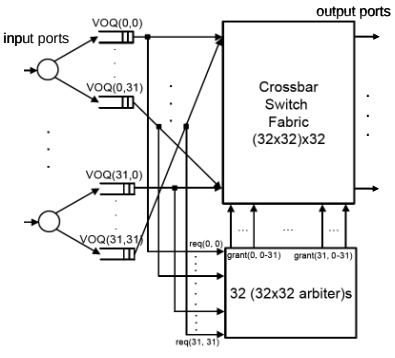


Fig. 1. An $n \times n$ switch: block diagram, crossbar switch fabric and its switch arbiters.

Using the same idea of Ping Pong, another arbiter design, called switch arbiter (SA), was proposed in [7]. An SA is constructed by a tree structure composed of 4×4 SA nodes. An SA node consists of a D flip-flop, four priority encoders, a 4-bit ring counter, five 4-input OR gates, and four 2-input AND gates. As a PPA, the SA is *not* fair for nonuniformly distributed requests. SA is the fastest among known arbiters, but it is more complex in structure. Zhengy and Yang proposed two arbiters, PRRA and IPRRA [8], which are based on a binary search algorithm and have less area but slower speed.

Here, we propose a new arbiter design with fair round-robin arbitration scheme. To balance area and time complexities, we have derived a new arbitration state diagram and the optimal arbitration Boolean functions for the new arbiter, and have proved that the arbitration Boolean functions derived are optimal (simplest). The execution time delay complexity of our arbiter is approximately $O(\log_4 n)$, and the area complexity of our arbiter is $O(n)$ of 2-input logic gates. Practically, our arbiter is the fastest arbiter compared with all previous arbiters we knew, and is area efficient.

This paper is organized as follows: Section 2 describes the design of our new arbiter. Section 3 shows the proof of optimal Boolean logic functions of the new arbiter. Section 4 gives experimental and comparison results. Finally, conclusions will be made in Section 5.

2. Design of the New Arbiter

2.1. Derivation and Formulation of the Function of Our New Arbiter

Based on the round-robin (or fair) scheme, we first design and derive the function of our new arbiter in the following: Given a set of binary inputs (requests) r_i, $0 \le i \le n-1$, under a set of internal states formed with an n-bit binary value t_i (i.e., a set of tokens), $0 \le i \le n-1$, compute a set of binary outputs (grants) g_i, $0 \le i \le n-1$, for each request. During each arbitration (i.e., a state), only one of tokens will be equal to 1 (one-hot coding). Assume t_j is equal to 1 (when all t_i's are equal to 0, let j be 0), then each grant gi can be computed as follows:

$$g_i = \begin{cases} 1, \text{ if exists an } i = \{\min\{(j+b) \bmod n \mid r_{(j+b) \bmod n} = 1, \\ \quad 0 \le b \le n-1\}\} \bmod n \\ 0, \quad \text{otherwise} \end{cases}$$

When all t_i's are euqal to 0, from the formula above we have that r0 always has the highest priority and it forms the *linear* arbitration. When only one of t_i's is equal to 1 (which means that r_i has the only authority of using the resource), it needs the following additional functionality of updating t_i's after the operation specified in (1): if $g_i = 1$ then $t_i \leftarrow 0$ and $t_{(i+1) \bmod n} \leftarrow 1$. Let linear arbitration be a special case of the round-robin arbitration, then the two arbitration schemes can be combined into one scheme, we call it the integrated round-robin arbitration, which is applied in our arbiter.

2.2. The Design of the State Diagram of the New Arbiter

According to our new arbitration function derived above, we shall derive and design the new arbiter's state diagram. Our objective is to design a new arbiter with minimal computing time and less area. We use the combination of all t_i's as the system state of our new arbiter to derive its state diagram. During arbitrating, only one of t_i's will be set to 1, so the initial state is started from that only t0 is equal to 1 and other t_i's are equal to 0, meanwhile all combinations of input r_i's are considered to derive each of its next states. Take $n = 4$ as an example, the initial state $t_0t_1t_2t_3$ is 1000, and when input $r_0r_1r_2r_3$ is 0110, then one of its outputs $g1$ is set to 1 and other outputs g_i's are set to 0, which represent that r_1 obtains the grant and can use the resource. And its next state $t_0t_1t_2t_3$ is 0010, which represents that r_2 will alternately have the authority of using the shared resource next time during the turnaround. Also, at the initial state consider another input: $r_0r_1r_2r_3 = 1101$, then one of its output $g_0 = 1$, and other outputs g_i's are set to 0, which represent that r_0 obtains the grant and can use the resource. Again, its next state $t_0t_1t_2t_3$ is 0100, which represents that r_1 will alternately have the using authority next time. And so forth, the state diagram of the new arbiter for $n = 4$ can be obtained as shown in Fig. 2.

As for n being any value, the state diagram of our new arbiter can be obtained with the same principle above.

Theorem 1: The arbiter's state diagram in Fig. 2 is correct.

Prove: The state value (unsigned binary value) of every state in Fig. 2 uses one-hot coding, and each next state's value depends just on its outputs g_i's, and only one of g_i's will be 1 (granted), assume $g_i = 1$, which only makes $t_{(i+1)}$ mod n to be

1, and other t_i's still 0. Therefore, state coding still in one-hot coding. For example, when at state 0100 and $r_0r_1r_2r_3$ = 0010, then its output g_2 is set to 1 and its next state $t_0t_1t_2t_3$ is 0001. This shows that the only using authority has been rotated fairly from user request r_1 (t_1) to user request r_3 (t_3), which meets the functional definition of our new arbiter defined in Subsection 2.1. Therefore, the state diagram in Fig. 2 is correct, O.E.I.

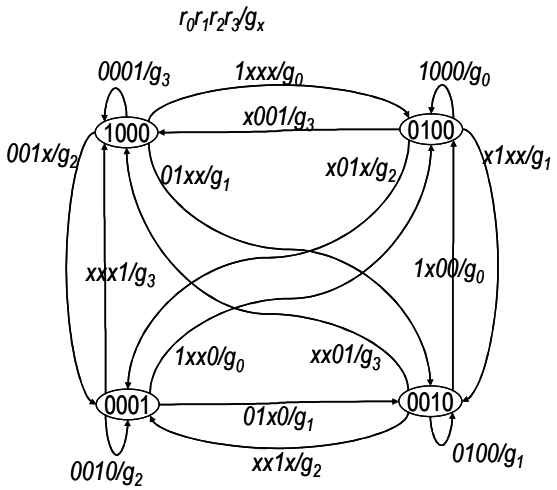


Fig. 2. The state diagram of our new arbiter for n = 4.

2.3. Design of the Optimal n×n Arbiter

According to the derived state diagram of Fig. 2, we shall use the following principles to systematically design the new arbiter: (a) use the systematic and theoretical methods to design the arbiter hardware for increasing its execution speed and for reducing its hardware area, (b) realize the hardware with combinational circuits as far as we can for area reduction and speed improvement, (c) use simple D-FFs to store the state value.

Observing the state diagram of Fig. 2 and using D-FFs to store t_i's, we find that each time whether t_i is set to 1 is determined by $g_{(i-1) \bmod n} = 1$, and the values of g_i's are determined by the values of r_j and t_j, $0 \le j \le n-1$. The arbiter's design is based on the principle of fairness to get the optimal Boolean logic functions. Therefore, the only authority of using the resources in the arbiter will be rotated among n users fairly; every user has the equal opportunity to get the only using authority. But at the same time if more than one of users' t_j's are equal to 1, the grant outputs of our arbiter will be set arbitrarily, in order to achieve the optimal (i.e., simplest) design of the arbiter.

We first derive the optimal 2×2 arbiters output Boolean functions, g_i, where i = 0 or 1. Then, we further generalize them to deduce the optimal Boolean logic functions for an $n \times n$ arbiter.

The 2×2 arbiter's optimal output g_0 set to 1 is determined by when r_0 = 1 under the following two different situations: a). t_0 is 1; this means that only r_0 has the using authority, or b). r_1 is 0; this means that r_1 does not have the request of using the resource. So its optimal Boolean function is $g_0 = t_0r_0 + r_0r_1'$. Additionally, applying the truth table and Karnaugh map approach we also get the same optimal Boolean function as shown in Fig. 3.

Through similar analysis and design, we can obtain the optimal Boolean functions for each output of the new n×n arbiter as follows:

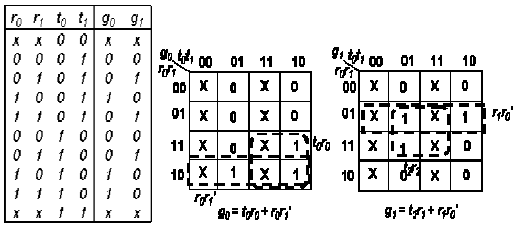


Fig. 3. The derived truth table, Karnaugh map, and optimal Boolean functions of our new 2×2 arbiter.

$$g_0 = t_0r_0 + r_0r_1'\ldots r_{n-1}' + t_2r_2'\ldots r_{n-1}'r_0 + \ldots + t_{n-2}r_{n-2}'r_{n-1}'r_0 + t_{n-1}r_{n-1}'r_0$$

$$g_1 = t_1r_1 + r_1r_2'\ldots r_{n-1}'r_0' + t_3r_3'\ldots r_{n-1}'r_0'r_1 + \ldots + t_{n-1}r_{n-1}'r_0'r_1 + t_0r_0'r_1$$

…

$$g_i = t_ir_i + r_ir_0'r_1'\ldots r_{i-1}'r_{i+1}'r_{i+2}'\ldots r_{n-1}' + t_{i+2}r_ir_{i+2}'\ldots r_{n-1}' r_0'r_1'\ldots r_{i-1}' + \ldots + t_{n-1}r_ir_{n-1}'r_0'r_1'\ldots r_{i-1}' + t_0r_ir_0' r_1'\ldots r_{i-1}' + \ldots + t_{i-1}r_ir_{i-1}'$$

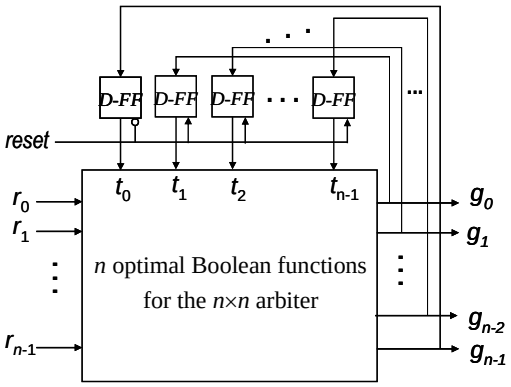
…

$$g_{n-1} = t_{n-1}r_{n-1} + r_{n-1}r_0'\ldots r_{n-2}' + t_1r_1'\ldots r_{n-3}'r_{n-2}'r_{n-1} + \ldots + t_ir_i'r_{i+1}'\ldots r_{n-2}'r_{n-1} + \ldots + t_{n-3}r_{n-3}'r_{n-2}'r_{n-1} + t_{n-2}r_{n-2}'r_{n-1}$$

In the following, we shall prove the Boolean functions above for the $n \times n$ arbiter are optimal (or simplest). Before doing this proof, the corresponding circuit structure of this new n×n arbiter are first derived and designed based on the above optimal Boolean functions.

Using the above-mentioned simplest (optimal) Boolean functions of our new $n \times n$ arbiter and its state diagram in Fig. 2 as well as the revising method for t_i's at each arbitration: If g_i is 1, then t_i is revised as 0, and $t_{(i+1) \bmod n}$ is revised as 1 before arbitrating next time, we designed the circuit structure of our new $n \times n$ arbiter as shown in Fig.

4.



3. Proof of Optimal Boolean Logic Functions of the New Arbiter

3.1. Derivation and Formulation of the Function of Our New Arbiter

In order to prove the above-mentioned Boolean logic functions of the $n \times n$ arbiter is optimal (simplest), some basic definitions are set up as follows, then the definition of an optimal Boolean logic function is given. Finally, we finish the proof with these definitions.

Definition 1: A product term is a set of minterms.

In definition 1, we can see that some minterms make up a product term.

Definition 2: The intersecting of two product terms: If all minterms of a product term are *all totally* included by another product term, we say that the two product terms are intersecting. It can be represented in a mathematical manner: If product term $p_1 \subset$ product term p_2 (or $p_2 \subset p_1$) $\Rightarrow$ $p_1 \cap p_2 \neq \phi$, all other situations we think that this two product terms do not intersect, namely $p_1 \cap p_2 = \phi$ (it means that their intersection is an empty set).

If a product term includes the total minterms of another product term, or inversely, then the two product terms are regarded as intersecting. When they cover each other only partially, then they still are not intersecting.

Definition 3: The union of two product terms: the union of two product terms is the union of all minterms of the two product terms.

It is our object here to prove that the new output Boolean functions of the $n \times n$ arbiter are optimum. The basic condition for an optimal (simplest) Boolean function is that the union of its product terms must includes all its minterms. Secondly, each pair of two product terms in the Boolean function does not intersect each other. And finally, each product term has at least one *unique* minterm. Therefore, we have the optimal Boolean function definition as follows.

Definition 4: (Optimal (Simplest) Boolean logic function definition) Let p_i and p_j be two product terms of a Boolean logic function P, that is, P is the set of all minterms in the Boolean function. Then, 4 conditions that make Boolean function P being optimal (simplest) are listed as follows:

1. $\cup p_i = P$, $0 \le i \le n-1$, it means all minterms of the Boolean function P are included (i.e., it is a complete cover).

2. $p_i \cap p_j = \phi$. Each product in the function does not fully include other product terms (i.e., all are different products).

3. Each of the product can no longer be simplified (i.e., all are the prime implicants).

4. For every pi in P, $\exists$ a minterm $m \in p_i$, and m $\notin p_j$, $i \neq j$, for $0 \le i, j \le n-1$, it means that each product term contains at least an unique minterm (i.e., all are essential implicants)

3.2. Proof of the Optimal Boolean Function of the n×n Arbiter

Theorem 2: The Boolean functions of the new n×n arbiter in Section 2.3 all are optimal (simplest) Boolean functions.

Prove: The derived Boolean functions will satisfy all four conditions of *definition* 4 and are optimum, but due to space limit, here we omit the detailed proof.

4. Experimental Results

In this section, the design results of our arbiters with the Synopsys' synthesis/design tools *Design Vision* (2007.03-sp3) and TSMC 0.18μm standard component library technology are presented. Pre-layout simulation of it after synthesis is done by using the *ModelSim* (*SE PLUS 6.0d*) tool.

Fig. 5 shows the comparison curves of the numbers of the equivalent 2-input NAND gates used in our arbiters and other arbiters in PPE [6], PPA [5], SA [7], PRRA [8], and IPRRA [8], respectively. This figure shows that the area size of our arbiter increases linearly with the number of inputs; and the area results of ours are less than those of PPE and SA designs, but more than those of PRRA and IPRRA.

Fig. 6 shows the comparison curves of the execution time delays of our arbiters and other

arbiters in PPE, PPA, SA, PRRA and IPRRA, respectively. Other arbiters' results in Fig. 5 and 6 are all referred to from [8]. Comparing with other arbiters, our arbiter not only has a definitely fair mechanism, but also its execution time delay grows with the trend of $\log_4 n$, which means that the growing of the execution time delay is less as the number of inputs becomes larger. In the example of case 64×64, the execution time delays of the arbiters in PPE, PPA, SA, PRRA, and IPRRA are larger than that of our arbiter by 3 times , 2.7 times , 1.3 times , 2.7 times and 2.2 times, respectively; Our arbiter is the fastest one among all arbiters that we can find now. In addition, the growth of our arbiter's execution time delays is relatively slow compared to those of other arbiters.

Table 1 shows the compared results of the execution time delays for our arbiter designs and other arbiters of different sizes (n or N) respectively.

Table 2 shows the area compared results in term of the numbers of equivalent two-input NAND gates used for our arbiters and other arbiters of different sizes (n or N) respectively.

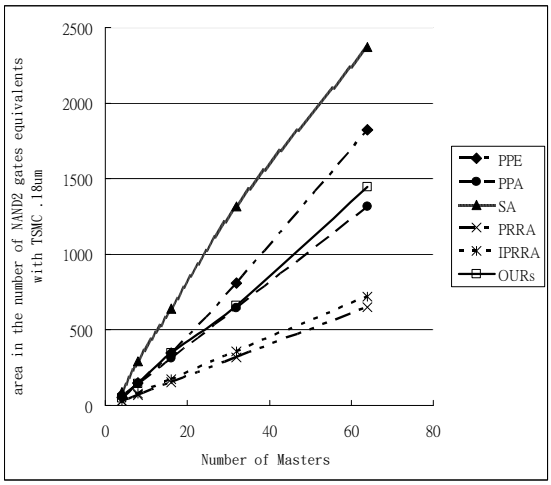


Fig. 5. The growing curves of the numbers of the equivalent 2-input NAND gates used in respective arbiters.

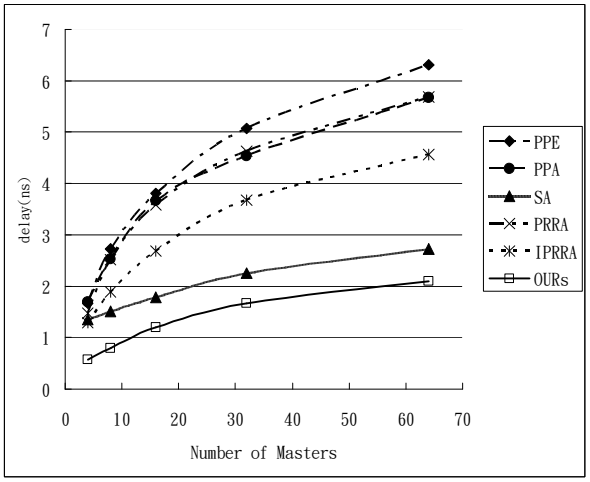


Fig. 6. The growing curves of the execution time delays of respective arbiters.

Table 1. The execution time delay comparisons of our arbiters and other arbiters.

	$N=4$	$N=8$	$N=16$	$N=32$	$N=64$
PPE	1.67ns	2.73 ns	3.8 ns	5.07 ns	6.31 ns
PPA	1.7ns	2.53 ns	3.66 ns	4.54 ns	5.67 ns
SA	1.36 ns	1.51 ns	1.79 ns	2.26 ns	2.72 ns
PRRA	1.47 ns	2.52 ns	3.58 ns	4.63 ns	5.68 ns
IPRRA	1.29 ns	1.89 ns	2.68 ns	3.68 ns	4.56 ns
Ours	0.58 ns	0.80 ns	1.20 ns	1.67 ns	2.10 ns

Table 2. The area compared results of our arbiters and other arbiters in term of the quantity of NAND-2 gates used

	$N=4$	$N=8$	$N=16$	$N=32$	$N=64$
PPE	53	150	349	812	1826
PPA	63	143	313	644	1316
SA	89	292	641	1318	2372
PRRA	31	72	155	320	651
IPRRA	31	82	173	356	723
Ours	48	145	349	661	1445

However, the area of our arbiter is larger than those of PPA, IPRRA and PRRA. Here, the readers will have a major question: Why is the area of our optimal (simplest) Boolean function arbiter larger than that of some other arbiters? After in-depth research and analysis we have found the

most likely reasons are: Because the non-optimal individual Boolean functions of other arbiters may include many of non-optimal product terms which can be *shared* among those functions and then the product terms sharing among the arbiters' outputs makes their circuit areas less. The optimal Boolean functions in our arbiter are designed with respect to the individual function, as a whole, they are not necessarily the simplest design for *all* Boolean functions of the arbiter.

Our individually optimal Boolean functions may have less the same product terms to be shared among the functions, which results in the overall area of our arbiter difficult to small, although, the area of ours is not very large. Our arbiter's individually optimal Boolean functions do not have bad affection for the execution speed of the arbiter, and inversely, they may reduce arbiter's execution delay much. This is reasonable from theory, and it is also seen by the fact in Fig. 6 that our arbiter is fastest comparing with the known arbiters. The execution time delay complexity of our arbiter is approximately $O(\log_4 n)$, which is the same as that of SA [7], and the area complexity of our arbiter is $O(n)$ of logic gates, which is the same as that of IPRRA and PRRA.

Today SoC, NoC, CMP, multi-core designs and tera-bit scale network switches demand the ultra-high speed and the requirement of the small circuit size is less important [1, 2, 9], which just is the same direction as our new fast arbiter.

5. Conclusions

In this paper, we proposed a new fast arbiter design. To balance the gate delay, wire delay, and circuit complexity, a new arbitration state diagram and the optimal arbitration Boolean functions are derived in the design. We proved that the arbitration Boolean functions derived in the design are optimal (simplest), and its arbitration is fair. This fairness is not guaranteed in the design of PPA [5] and SA [7]. The execution time delay complexity of our arbiter is approximately $O(\log_4 n)$, which is the same as that of SA, and the area complexity of our arbiter is $O(n)$ of 2-input logic gates, which is the same as that of SA, IPRRA, and PRRA. Practically, our arbiter is faster than SA, the existing fastest arbiter, and has smaller area. Compared with SA, our arbiter has 22.8% of execution time improvement and 39.1% of area reduction. Since of our small arbiter's the high-performance, it is extremely useful for the realizations of ultra-high-speed switches, MPSoC arbitration, and NoC routers. This new arbiter design is being applied for a patent of ROC (application No.: 0971xxxxx).

References

[1] L. Benini and G. Micheli, "Networks on Chips: A New SoC Paradigm," in Computer Magazine, Vol.35, Issue.1, Jan. 2002, pp.70-78.

[2] S.-H. Hsu, e.t. a.l., "Design of a Reconfigurable NoC Router with Network Interface for AMBA-based IPs" in The 2nd IEEE Asian Solid-State Circuits Confer., 2006.

[3] N. Mckeown, P. Varaiya, and J. Warland, "The iSLIP Scheduling Algorithm for Input-Queued Switch," IEEE Transaction on Networks, 1999, pp. 188-201.

[4] H. J. Chao and J. S. Park, "Centralized Contention Resolution Schemes for a Larger-capacity Optical ATM Switch," Proceedings of IEEE ATM Workshop, 1998, pp. 11-16.

[5] H. J. Chao, C. H. Lam, and X. Guo, "A Fast Arbitration Scheme for Terabit Packet Switches," Proceedings of IEEE Global Telecommunications Conference, 1999, pp.1236-1243.

[6] P. Gupta and N. McKeown, "Designing and Implementing a Fast Crossbar Scheduler," IEEE Micro., vol.19, no.1, pp. 20-29, 1999.

[7] E.S. Shin, V.J. Mooney, and G.F. Riley, "Round-Robin Arbiter Design and Generation," Proc. Int. Symp. Sys. Syn. (ISSS), 2002, pp. 243-248.

[8] S. Q. Zhengy and Mei Yang, "Algorithm-Hardware Codesign of Fast Parallel Round-Robin Arbiters", IEEE Transactions on Parallel and Distributed Systems, Vol.18, Issue.1, Jan 2007, pp.84-95.

[9] W. Stallings, Data and Computer Communications, Fifth Edition, NJ: Prentice Hall, 1997..