An Optimal Round-Robin Arbiter Design for NoC*

JER-MIN JOU AND YUN-LUNG LEE
Department of Electrical Engineering
National Cheng Kung University
Tainan, 701 Taiwan

A new optimal arbiter is designed. We proposed a set of optimal Boolean functions and the corresponding circuit for it, and showed that the arbitration Boolean functions derived are optimal (simplest). 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 R.O.C. (application No.: 0972020612-0).

Keywords: arbiter, optimization, Boolean function, NoC, network

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. 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.

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

(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). PPA features an $O(\log_2 N)$-level tree structure. Clearly, PPA has $O(\log_2 N)$ in critical path delay. Each node in the tree is 2-input ping-pong arbiter (AR2). The basic function of an AR2 is favoring its two subtrees alternately if both subtrees have requests. However, when the input requests are less than $N$, it is possible that PPA becomes unfair. The performance of scheduling algorithm based on PPA is worse than iSLIP and DRRM which are based on PPE.

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 $4 \times 4$ 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 in some specific cases. 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 simple binary search algorithm and they outperform PPE and PPA both in area and speed. The PRRA design has $O(\log N)$-gate delay and consumes $O(N)$ gates. IPRRA design significantly reduces the timing of all subtrees. Although both of PRRA and IPRRA achieve round-robin fairness, they are not fast and small enough for a SoC design with crossbar switches in it.

Here, we propose a new arbiter design with the 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. NEW DESIGN OF THE ROUND-ROBIN 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) \( r_i, 0 \leq i \leq n - 1 \), under a set of internal states formed with an \( n \)-bit binary value \( t_i \) (i.e., a set of tokens), \( 0 \leq i \leq n - 1 \), compute a set of binary outputs (grants) \( g_i, 0 \leq i \leq n - 1 \), 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 \( g_i \) can be computed as follows:

\[
g_i = \begin{cases} 
1, & \text{if exists an } i = \min \{(j + b) | t_{(j + b) \mod n} = 1, 0 \leq b \leq n - 1\} \mod n \\
0, & \text{otherwise}
\end{cases}.
\]

When all \( t_i \)'s are equal to 0, from the formula above we have that \( r_0 \) 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 \( t_{(i+1) \mod n} \leftarrow 1 \). 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 \( t_0 \) 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 \( g_1 \) 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 \( r_1 \) 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 0010, 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.

Proof: 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 $r_0 r_1 r_2 r_3 = 0010$, then its output $g_2$ is set to 1 and its next state $t_0 t_1 t_2 t_3$ 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 section 2.1. Therefore, the state diagram in Fig. 2 is correct, O.E.I.

2.3 Design of the Optimal $n \times n$ 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 $g_{(i-1)\mod n} = 1$, and the values of $g_i$’s are determined by the values of $r_j$ and $t_j$, $0 \leq j \leq n - 1$. 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 $2 \times 2$ 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 $2 \times 2$ 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 $g_0 = t_0 r_0 + r_0 r_1$. 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 $n \times n$ arbiter as follows:

$g_0 = t_0 r_0 + r_0 r_1' \ldots \ldots \ldots + t_n r_n r_n' r_n \ldots + t_0 r_0$,

$g_1 = t_1 r_1 + r_1 r_2 + \ldots + t_n r_n r_n' r_n \ldots + t_0 r_0 r_0'$,

$g_{i} = t_i r_i + r_i r_{i+1} + \ldots + t_{i-1} r_{i-1} r_{i-1}' r_{i-1} \ldots + t_0 r_0$,

$g_{n-1} = t_n r_n + r_n r_n' + \ldots + t_{n-1} r_{n-1} r_{n-1}' + t_0 r_0$.

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 $n \times n$ 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 $t_{(i+1) \mod n}$ 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 FUNCTIONS OF THE NEW ARBITER

3.1 Definition of an Optimal Boolean Logic Function

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 $p_2 \subset p_1$) $\Rightarrow p_1 \cap p_2 \neq \emptyset$, all other situations we think that this two product terms do not intersect, namely $p_1 \cap p_2 = \emptyset$ (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. $\cup p_i = P$, $0 \leq i \leq n - 1$, 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 $p_i$ in $P$, $\exists$ a minterm $m \in p_i$, and $m \not\equiv p_i$, $i \not\equiv j$, for $0 \leq i, j \leq n - 1$, 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 Functions for the $n \times n$ Arbiter

**Theorem 2** The Boolean functions of the new $n \times n$ arbiter in section 2.3 all are optimal (simplest) Boolean functions.

**Proof:** Without losing general, we shall use one of the arbiter’s Boolean functions, $g_0$, as an example, and prove it satisfying all four conditions of Definition 4 and is an optimal Boolean function. The proof of other arbiter’s Boolean functions are then done with the same manner and optimal.

According to the arbiter’s Boolean function $g_0$, we have $g_0 = t_0r_0 + r_0r_1'...r_{n-1}' + t_2r_2' ...r_{n-1}'r_0 + ... + t_0r_0r_1'r_2'...r_{n-1}' + ... + t_0t_1r_0'r_1'r_2'r_3'...r_{n-1}'r_0 + t_0t_1r_0'r_1'r_2'r_3'...r_{n-1}'r_0$ and let $p_0 = t_0r_0$, $p_1 = r_0r_1'...r_{n-1}'$, $p_2 = t_2r_2'...r_{n-1}'$, ..., $p_i = t_ir_i'...r_{n-1}'$, ..., $p_{n-2} = r_{n-2}r_{n-2}'r_{n-1}'$, and $p_{n-1} = t_{n-1}r_{n-1}'r_0$. We first prove that this Boolean function meets condition 1 of Definition 4 as follows:

Suppose $r_0 = 1$, in accordance with Boolean function $g_0$’s product terms we get the following characteristics: $p_i$ includes all situations of making $g_0$ set to 1 under the condition $t_i$ is 1, namely, $p_0 = r_0d_0 = r_0d_0(r_0 + r_0')(r_1 + r_1')...(r_{n-1} + r_{n-1}')(t_1 + t_1')...(t_{n-1} + t_{n-1}')$.

$p_1$ includes all situations of making $g_0$ set to 1 under the conditions $t_1$ is 1 or all $t_i$’s are 0, namely, $p_1 = r_0r_1'...r_{n-1}' = r_0r_1'r_2'...r_{n-1}'(t_1 + 1) = r_0r_1'r_2'...r_{n-1}'(t_0 + t_1')(t_2 + t_2')(t_3 + t_3')...(t_{n-1} + t_{n-1}') = r_0r_1'r_2'...r_{n-1}'t_1(t_0 + t_1')(t_2 + t_2')(t_3 + t_3')...(t_{n-1} + t_{n-1}') = r_0r_1'r_2'...r_{n-1}'t_1(t_0 + t_1')(t_2 + t_2')(t_3 + t_3')...(t_{n-1} + t_{n-1}')$.

And so forth, we also have that $p_i$ includes all situations of making $g_0$ set to 1 under the condition $t_i$ is 1, namely, $p_i = r_0r_i'...r_{n-1}' = r_0r_i'r_{i+1}'...r_{n-1}'(r_1 + r_1')(r_2 + r_2')(r_3 + r_3')...(r_{i-1} + r_{i-1}')(t_0 + t_i')(t_1 + t_1')(t_2 + t_2')...(t_{i-1} + t_{i-1}')(t_{i+1} + t_{i+1}')...(t_{n-1} + t_{n-1}')$.

Finally, $p_{n-1}$ includes all situations of making $g_0$ set to 1 under the condition $t_{n-1}$ is 1, namely, $p_{n-1} = t_{n-1}r_{n-1}'r_0 = t_{n-1}r_{n-1}'r_0(r_1 + r_1')(r_2 + r_2')(r_3 + r_3')...(r_{n-2} + r_{n-2}')(t_0 + t_i')(t_1 + t_1')(t_2 + t_2')...(t_{n-2} + t_{n-2}')$.

From the analysis above, we know that $p_0 \cup p_1 \cup ... \cup p_{n-1}$ include all minterms to make $g_0$ set to 1 $\Rightarrow \cup p_i = P$, $0 \leq i \leq n - 1$, O.E.I.

Next, we prove that the Boolean function $P$ meets condition 2 of Definition 4 as follows:

We arbitrarily choose from $P$ two product items $p_i = t_ir_i'...r_{n-1}'$ and $p_j = t_jr_j'...r_{n-1}'$, $0 \leq i, j \leq n - 1$. According to the intersecting of two product terms in Definition 2,
we have \( p_i \preceq p_j \) and \( p_j \preceq p_i \) immediately. So, \( p_i \cap p_j = \phi \), \( 0 \leq i, j \leq n - 1, i \neq j \), O.E.I.

Following, we prove that the Boolean function \( P \) meets the condition 3 of Definition 4 as follows: By analyzing of \( g_0 \)'s arbitrary product term \( p_i = r_0f_i' \ldots r_m1' \), we found that its variables are composed of the following three categories: \( r_0 \) the using authority \( t_i \), and respective inverse requests \( r'_i, x \neq 0 \). Therefore, we divided it into three cases to discuss whether this product term can be further simplified. In \( p_i = r_0f_i' \ldots r_m1' \), \( r_0 = 1 \) is essential for making \( g_0 \) set to 1, that is to say if without (simplifying) \( r_0 = 1, g_0 \) is impossible to be set to 1 (without the exclusive resource using request, there will be no grant output). Therefore, \( r_0 \) can not be removed (simplified) in all product terms.

Also if we simplify (remove) the using authority \( t_i \) from the product term, we get a new product term \( \alpha = r_0f_i' \ldots r_m1' \) is simplified (removed) by one of \( r_mi', i \leq m \leq n - 1, \) we get a new product term \( \beta = r_0f_i' \ldots r_m1' \) meets the condition 3 of Definition 4 and are optimal, O.E.I.

Finally, we prove that the Boolean function \( P \) meets the condition 4 of Definition 4 by analyzing each product term in it as follows:

Product term \( p_0 = r_0f_0 = r_0f_d(r_1 + r_1')(r_2 + r_2')(r_3 + r_3') \ldots (t_0 + t_1')(t_2 + t_2') \ldots (t_{m-1} + t_{m-1}'), \) we find it has at least an unique minterm: \( r_0f_2' \ldots r_{m-1}'t_0't_2' \ldots t_{m-1}' \).

Product term \( p_1 = r_0f_1' \ldots r_{m-1} = r_0f_1'(r_2 + r_2')(r_3 + r_3') \ldots (t_0 + t_1')(t_2 + t_2') \ldots (t_{m-1} + t_{m-1}'), \) we find it has at least an unique minterm: \( r_0f_2' \ldots r_{m-1}'t_0't_2' \ldots t_{m-1}' \).

Product term \( p_2 = r_0f_2' \ldots r_{m-1}' = r_0f_2'(r_1 + r_1')(r_3 + r_3') \ldots (t_0 + t_1')(t_2 + t_2') \ldots (t_{m-1} + t_{m-1}'), \) we find it has at least an unique minterm: \( r_0f_2' \ldots r_{m-1}'t_0't_2' \ldots t_{m-1}' \).

... 

Product term \( p_n-1 = t_0t_1t_2 \ldots t_{m-1}'r_0 = t_0t_1t_2 \ldots t_{m-1}'r_0(t_0 + t_0')(t_1 + t_1')(t_2 + t_2') \ldots (t_{m-2} + t_{m-2}'), \) it has at least an unique minterm: \( r_0f_2' \ldots r_{m-1}'t_0't_2' \ldots t_{m-1}' \).

So, we know each product term has at least an unique minterm. And, Boolean functions of our new arbiter all satisfy all conditions of Definition 4 and are optimal, O.E.I.

4. EXPERIMENT RESULTS

In this section, the design results of our arbiters with the Synopsys' synthesis/design tools Design Vision (2007.03-sp3) and TSMC 0.18\( \mu \)m 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 Figs. 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 $64 \times 64$, 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>
<thead>
<tr>
<th></th>
<th>$N = 4$</th>
<th>$N = 8$</th>
<th>$N = 16$</th>
<th>$N = 32$</th>
<th>$N = 64$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPE</td>
<td>1.67 ns</td>
<td>2.73 ns</td>
<td>3.8 ns</td>
<td>5.07 ns</td>
<td>6.31 ns</td>
</tr>
<tr>
<td>PPA</td>
<td>1.7 ns</td>
<td>2.53 ns</td>
<td>3.66 ns</td>
<td>4.54 ns</td>
<td>5.67 ns</td>
</tr>
<tr>
<td>SA</td>
<td>1.36 ns</td>
<td>1.51 ns</td>
<td>1.79 ns</td>
<td>2.26 ns</td>
<td>2.72 ns</td>
</tr>
<tr>
<td>PRRA</td>
<td>1.47 ns</td>
<td>2.52 ns</td>
<td>3.58 ns</td>
<td>4.63 ns</td>
<td>5.68 ns</td>
</tr>
<tr>
<td>IPRRA</td>
<td>1.29 ns</td>
<td>1.89 ns</td>
<td>2.68 ns</td>
<td>3.68 ns</td>
<td>4.56 ns</td>
</tr>
<tr>
<td>Ours</td>
<td>0.58 ns</td>
<td>0.80 ns</td>
<td>1.20 ns</td>
<td>1.67 ns</td>
<td>2.10 ns</td>
</tr>
</tbody>
</table>
Table 2. The area compared results of our arbiters and other arbiters in term of the quantity of NAND-2 gates used.

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

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.

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, and the solution to find simplest design for all Boolean functions of the arbiter is NP-complete [10].

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 tera-bit scale network switches and SoC, NoC, CMP, multi-core designs 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,
AN OPTIMAL ARBITER DESIGN FOR NOC

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. 0972020612-0).

REFERENCES


Jer-Min Jou (周哲民) received the Ph.D. degree in Electrical Engineering and Computer Science from National Cheng Kung University, Tainan, Taiwan, R.O.C., in 1987. In 1989, he was an Associate Professor in the Department of Electrical Engineering, National Cheng Kung University, where he is currently a Professor. His research interests include SoC hardware-software codesign, system design, ASIC design/synthesis, VLSI CAD, and asynchronous circuit design. Dr. Jou was the recipient of a Distinguished Paper Citation at the 1987 IEEE ICCAD Conference, Santa Clara,

Yun-Lung Lee (李昀隆) received the B.S. degree Electronic Engineering from Feng Chia University, Taichung, R.O.C., in 2005, and is currently working toward the Ph.D. degree in Electrical Engineering from National Cheng Kung University. He received the best paper award in International Computer Symposium 2008. His research interests include reconfigurable system and VLSI chip design.