A Precise Bandwidth Control Arbitration Algorithm for Hard Real-Time SoC Buses

Bu-Ching Lin, Geeng-Wei Lee, Juinn-Dar Huang and Jing-Yang Jou
Department of Electronics Engineering,
National Chiao Tung University, Hsinchu, Taiwan
{kurt,gwlee,jyjou}@eda.ee.nctu.edu.tw, jdhuang@mail.nctu.edu.tw

Abstract— On an SoC bus, contentions occur while different IP cores request the bus access at the same time. Hence an arbiter is mandatory to deal with the contention issue on a shared bus system. In different applications, IPs may have real-time and/or bandwidth requirements. It is very difficult to design an arbitration algorithm to simultaneously meet these two requirements. In this paper, we propose an innovative arbitration algorithm, RB_lottery, to meet both of the requirements. It can provide not only the hard real-time guarantee but also the precise bandwidth controllability. The experimental results show that RB_lottery outperforms several well-known existing arbitration algorithms.

I. Introduction

In SoC systems, intellectual properties (IPs) need to communicate with each other to accomplish certain functions. For example, CPUs access data from the memory or other I/O devices. Such communication is commonly built by using shared buses which serve as the communication channels between IPs. There are two major components, masters and slaves, in the shared bus architecture. Those IPs who initiate requests to access the shared buses are called masters. These requests could be either read or write transactions. Unlike what masters do, those IPs without bus controllability are called slaves.

Since masters on the same shared bus may initiate requests at the same time, an arbiter is required to decide which master is the current bus owner. The arbiter could be implemented in a centralized or a distributed fashion. Meanwhile, different arbitration algorithms may lead to different bus performance. Therefore, the arbiter should be designed carefully in high performance systems [1–4].

The major difficulty to design an arbiter is to meet different requirements simultaneously. A master has the bandwidth requirement if it requires at least a certain amount of bus bandwidth. In addition, some masters with the real-time requirement demand their requests accomplished within a fixed number of clock cycles. Most arbitration algorithms target at either real-time requirements or bandwidth requirements, but few of them deal with both requirements well. It is a tough challenge to design an arbiter for a real-time system with bandwidth requirements.

To deal with this issue, we propose an innovative arbitration algorithm, RB_lottery. The main motivation is to provide both hard real-time guarantee and precise bandwidth control. RB_lottery is a three-level arbitration algorithm. The first level satisfies all hard real-time requirements. The following two levels cooperate together to provide the precise bandwidth control with only 2% error range. The experimental results also show that RB_lottery outperforms several well-know existing arbitration algorithms.

The remainder of this paper is organized as follows. Several existing arbitration algorithms are briefly reviewed in Section II. Section III presents the details of the proposed arbitration algorithm, RB_lottery. Experimental results are given in Section IV. Finally, we conclude this paper in Section V.

II. Preliminaries

In this section, we briefly introduce several existing arbitration algorithms. The characteristics of each arbitration algorithm are also presented.

II.A. Priority-Based Algorithm

In this scheme, each master is assigned an unique value as its priority, either statically or dynamically. Whenever the contending requests come from different masters, the arbiter grants the bus access to the highest-priority master. Due to the simplicity and low hardware cost, the static priority algorithm is still widely used today. However, the masters with lower priority could be severely starved by higher parity ones.

A methodology for the design of dynamic priority algorithm is proposed in [5]. An addition layer of circuitry, called the Communication Architecture Tuner (CAT), is used to enhance the ability to adapt the system requirements. However, the implementation details of each master IP is required, it is not always possible to get such information if the IP is reused in a black-box fashion.

II.B. TDMA Algorithm

In the time division multiplexed access (TDMA) algorithm, the execution time is divided into several time slots and each slot is statically assigned to a particular master. If the master associated with the current time slot has a pending request, the arbiter grants the transaction immediately and the time wheel is rotated to the next slot. While there is no pending request for the current slot owner, the slot is wasted. In order to alleviate the wasted slots, a second level arbitration algorithm is usually adopted to permit the bus granted to the other requesting masters. Fig. 1 shows an example architecture of the two-level TDMA.

Because every master is allocated a certain amount of time slots, TDMA guarantees not only a minimum bandwidth allocation but also the worse-case response latency. However, it is difficult to design the time slot sequence for an non-periodic system. Another drawback of TDMA is that the time slot can not independently consider real-time requirements and bandwidth allocation at the same time.

II.C. Lottery Algorithm

Another communication architecture, LOTTERYBUS, is proposed in [6]. The algorithm stochastically grants one of the contending masters according to the ticket assignment either statically or dynamically [7–9]. The lottery tickets acted as the weight are accumulated through the lottery manager while bus contentions occur. A master is stochastically selected to get the bus access. In other words, it is a weighted random arbitration mechanism.

However, the ticket assignment influences both the bandwidth allocation and the average response latency. Consider there are a set of bus masters, M_1, M_2, ..., M_n; each of them holds t_1, t_2, ..., t_n tickets, respectively; and all masters have similar traffic behavior. From the probability analysis, the allocated bandwidth of master M_i is $\frac{t_i}{\sum_{j=1}^{n} t_j}$ and the average response latency of master M_i is $\frac{\sum_{j=1}^{n} t_j}{t_i}$. The bandwidth allocation ratio for each master is exactly conformed to the ticket ratio, but the average response latency ratio for each master is proportional to the reciprocal of the ticket ratio.

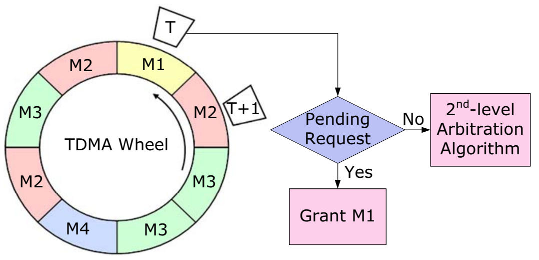

Fig. 1. An example architecture of the two-level TDMA.

From the previous discussions, the ticket assignment determines not only the bandwidth allocation but also the average response latency. The more lottery tickets a master owns, the higher granted probability and the more fraction of bandwidth allocation it gets. In addition, masters with more tickets would also have shorter average response latency. That is, the bandwidth allocation and the average response latency cannot be independently controlled by the ticket assignment. Furthermore, it also assumes that all masters have similar traffic behavior. This assumption may be true for network switch designs [6], however, it is unlikely to be true for many other SoC applications.

II.D. RT_lottery Algorithm

A lottery-based arbitration algorithm, RT_lottery, is proposed in [8]. As shown in Fig. 2, it is a two-level arbitration algorithm including a real-time handler and a Lottery with tuned weight. The real-time handler providing hard real-time guarantees governs all requests with real-time requirements first. Then the Lottery with tuned weight allocates the bandwidth according to the fine tuned tickets to meet the bandwidth requirements.

A set of real-time counters in the real-time handler record the interval from the current request to corresponding deadlines for all masters with real-time requirements. It decreases by one every cycle until the current request is served. If there are still pending requests while the real-time counter reaches zero, it is a real-time violation. The real-time handler also uses a warning line mechanism which considers the worst contending case to provide hard real-time guarantee. If the request is still pending and the real-time counter is less than the warning line, the real-time handler assigns the highest priority to the master and grants the bus access immediately.

Ticket assignment plays an important role in Lottery-based algorithms. It should consider not only the bandwidth requirement but also the traffic behavior at the same time. A simulation-based ticket redistribution mechanism, weight tuning, is used to decide the proper ticket assignment. While the simulation result violates the bandwidth requirements, the master with the most over-allocated bandwidth than required gives part of its tickets to the master with the most under-allocated bandwidth than required.

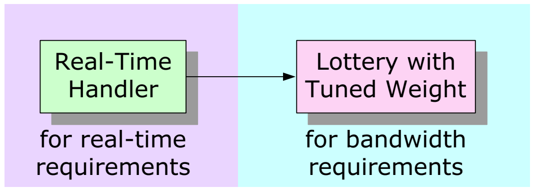
Real-Time Handler
Lottery with Tuned Weight
for real-time requirements
for bandwidth requirements

Fig. 2. The architecture of RT_lottery.

However, the weight tuning mechanism does not always successfully fulfill the bandwidth requirements. For example, there are three masters, M1, M2 and M3, each requires at least 25% of total bandwidth. We define the maximum bandwidth as the greatest possible bandwidth that a master can get. The maximum bandwidth of M1, M2 and M3 are 80%, 30% and 30%, respectively. As shown in Table I, each master has 100 tickets at the beginning in the second column. After several iterations of weight tuning, M1 takes 53% of total bandwidth even when it has only two tickets. M2 and M3 still get not enough bandwidth though they own most of tickets. Hence, a more powerful arbitration scheme is surely demanded.

III. Proposed Algorithm

This section describes our newly proposed arbitration algorithm, real-time bandwidth-regulating lottery (RB_lottery). First, the architecture is introduced in detail. Then we present the implementation to realize the idea from the hardware point-of-view. Finally, the algorithm flow is given.

III.A. The Architecture

In RB_lottery, the *bandwidth regulator* is added into RT_lottery [8]. It monitors the bus communication behavior and provides better controllability over the bandwidth allocation.

As shown in Fig. 3, RB_lottery, based on RT_lottery, is a three-level arbitration algorithm. The real-time handler and the Lottery with tuned weight are the same as RT_lottery. The real-time handler uses the warning line mechanism which considers the worst contending case to provide the hard real-time guarantee. A simulation-based mechanism named *weight tuning* decides the proper ticket assignment for the lottery manager to provide fair bandwidth allocation. The additional circuitry, the bandwidth regulator, is actually a traffic monitor with the controllability over bandwidth allocation according to the dynamic bus communication behaviors. It constantly checks the allocated bandwidth to each master and temporarily blocks the masters that have been allocated the bandwidth they require within a fixed period of time. The detailed implementation is described in the next section.

TABLE I
Failure of Weight Tuning in RT_lottery

	the ticket assignment of M1, M2 and M3				
	100:100:100	60:120:120	34:133:133	16:142:142	2:149:149
M1	60%	58%	56%	55%	53%
M2	19%	20%	21%	22%	23%
M3	19%	20%	21%	22%	23%

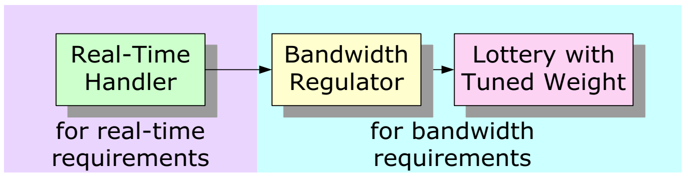
Real-Time Handler
Bandwidth Regulator
Lottery with Tuned Weight
for real-time requirements
for bandwidth requirements

Fig. 3. The architecture of RB_lottery.

We use a simple example shown in Fig. 4 to demonstrate how RB_lottery works. In the example, the four masters M1, M2, M3 and M4 are assigned 1, 2, 3 and 4 tickets, respectively. Assume that only M1, M3 and M4 issue requests at some instance of time. At the first level of real-time handler, no master gets granted because there are no urgent requests. The bandwidth regulator at the second level temporarily blocks requests from the over-served M4 according to the monitored traffic. At the third level, only one of M1 or M3 can be stochastically selected by the lottery manager to access the bus. In this example, M3 is the master that get granted finally.

III.B. The Implementation

In order to record the communication behavior in a fixed period of time, *observation windows* and a set of *bandwidth registers* are used.

- observation window:
  The time is divided into a sequence of fixed size windows for observation. The size of an observation window is configurable. In each observation window, all bus transactions are monitored to obtain the current status of bandwidth allocation. Once a master has already got its required bandwidth, the following requests from this master are temporarily blocked until the next window starts.

- bandwidth register:
  The bandwidth registers record the bandwidth already allocated to masters in the current observation window. When a master gets granted, the corresponding bandwidth register accumulates the

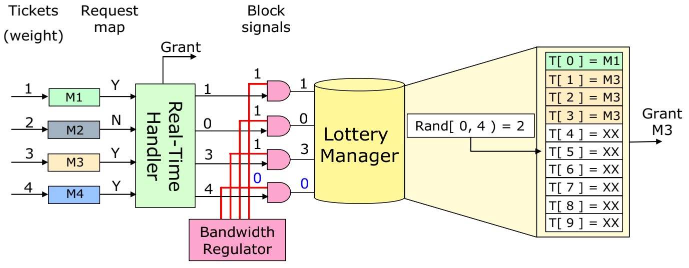

Fig. 4. An example of the proposed arbitration architecture.

amount of the current transaction. For example, if a granted master initiates an 8-beat transaction, then the corresponding bandwidth register is increased by eight when the transaction is completed.

The bandwidth regulator utilizes the information stored in the bandwidth registers to precisely control the overall bandwidth allocation in an observation window.

III.C. The Algorithm Flow

Fig. 5 shows the flow of RB_lottery algorithm. After the system starts, the system is examined in a cycle-by-cycle fashion. The flow is divided into three major blocks. In Block 1, the time is checked to see if it has to proceed to the next window. When the next window starts, all the masters that are previously blocked are released. In Block 2, the real-time handler detects whether there are masters with urgent real-time requests and then grants the most urgent master to avoid real-time violations. If no master is granted by the real-time handler, the lottery manager stochastically grants a request from the contending unblocked masters. In addition, if no unblocked master initiates requests, the lottery manager can still grant one of the pending blocked masters for higher bus utilization. Then the transaction amount of the granted request is accumulated into the bandwidth register belongs to the corresponding master. In Block 3, if the granted master has met its bandwidth requirement, its following requests are temporarily blocked within the current observation window.

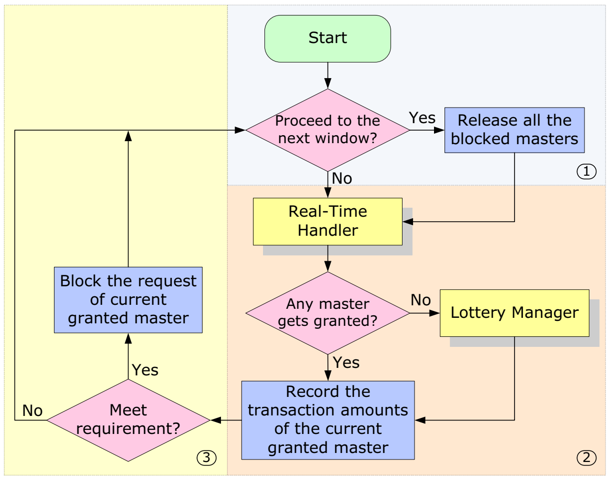
Start
Proceed to the next window?
Yes
Release all the blocked masters
No
①
Real-Time Handler
Any master gets granted?
No
Lottery Manager
Yes
Record the transaction amounts of the current granted master
②
Block the request of current granted master
Yes
No
Meet requirement?
③

Fig. 5. The flow of RB_lottery algorithm.

IV. Experimental Results

IV.A. Experiment Environment

IV.A.1 SystemC model

Our experiment environment is developed under SystemC v2.1 with the transaction level model (TLM) library. Fig. 6 shows an evaluation environment consisting N masters and one slave connected through TLM channels.

In the following experiments, five arbitration algorithms, Static Priority, Lottery, TDMA + Lottery (the first level arbitration is TDMA and the second level arbitration is Lottery), RT_lottery and RB_lottery, are evaluated for comprehensive comparisons.

The high abstraction-level SystemC model is capable of providing a fast simulation speed that can be up to one million cycles per second. Therefore, we can efficiently estimate the system performance and explore the proper system design parameters in this environment.

IV.A.2 Traffic Models

In this work, we use the traffic models developed in [8] to emulate the behaviors of different masters. For clarity, the three models are briefly described as follows:

- D type(D for dependency):
  A D type master has no real-time requirements and issues a request at the time depending on the finish time of the previous request.

- D_R type(D for dependency, R for real-time):
  A D_R type master behaves like a D type master with the real-time requirement. Each request must be completed before its deadline.

- ND_R type(ND for non-dependency, R for real-time):
  An ND_R type master is another kind of master with the real-time requirement. But the issue time of a request from an ND_R type master is independent of the finish time of its previous request.

IV.A.3 Traffic Behavior

In the following experiments, we set up a system with eight masters. As shown in Table II, the second column lists the master type. The third and fourth column give the probabilities of the burst size and the interval time between two successive requests initiated by a master.

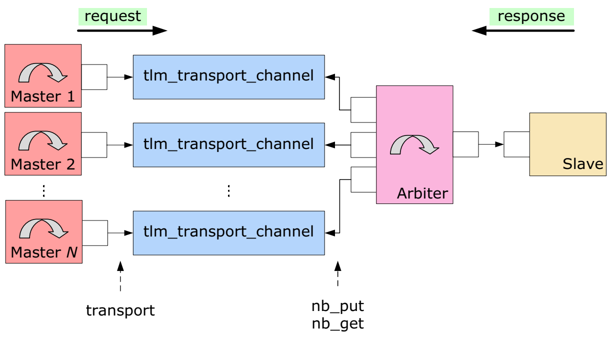

Fig. 6. An evaluation environment using TLM in SystemC.

TABLE II
THE BEHAVIOR OF EACH MASTER IN THE EXPERIMENTS

	type	beat/prob.		interval/prob.				
M1	D	8/50	16/50	6/10	7/20	8/40	9/20	10/10
M2	D	1/50	4/50	10/10	11/20	12/40	13/20	14/10
M3	D	8/50	16/50	6/10	7/20	8/40	9/20	10/10
M4	D	1/50	4/50	10/10	11/20	12/40	13/20	14/10
M5	D_R	8/50	16/50	10/10	11/20	12/40	13/20	14/10
M6	D_R	1/50	4/50	10/10	11/20	12/40	13/20	14/10
M7	ND_R	8/50	16/50	65/10	66/20	67/40	68/20	69/10
M8	ND_R	1/50	4/50	85/10	86/20	87/40	88/20	89/10

▨ heavy-traffic   light-traffic

TABLE III
THE NUMBER OF FAILED PATTERNS OF DIFFERENT ARBITRATION ALGORITHMS

Workload (%)	Static Priority	Lottery	TDMA + Lottery	RT _lottery	RB _lottery
60	100	100	95	0	0
65	100	100	98	0	0
70	100	100	100	0	0
75	100	100	100	10	0
80	100	100	100	18	0
85	100	100	100	37	1
90	100	100	100	55	12
95	100	100	100	74	44

There are four D type masters, two D_R type masters and two ND_R type masters in the system. Besides the bandwidth requirements, M5, M6, M7 and M8 also have the real-time requirements. M1, M3, M5 and M7 are heavy-traffic masters while the others are light-traffic masters. The heavy-traffic masters have larger burst beats and shorter intervals than the light-traffic ones. In other words, a heavy-traffic master generates a heavier load to the shared bus than a light-traffic one does.

IV.B. Experiment 1

In Experiment 1, we compare the performance of different arbitration algorithms, Static Priority, Lottery, TDMA + Lottery(the second level arbitration is Lottery), RT_lottery and RB_lottery. The level of difficulty to meet both real-time and bandwidth requirements generally depends on the bus workload in terms of the percentage of bus bandwidth utilization. As a result, we randomly generate patterns for different bus workloads and compare the results. As shown in Table III, the first column gives the bus workload varying from 60% to 95%. For each bus workload, 100 random patterns of different required bandwidth combinations for the eight masters are generated. And then we simulate the input patterns with different arbitration algorithms. The results in 102400 simulation cycles are recorded and analyzed to see if the arbitration algorithms can meet the real-time and bandwidth requirements simultaneously. If the real-time requirements are not all met or the allocated bandwidth is less than the required bandwidth with 2% error range during simulation, it is a failed pattern.

The parameters of arbitration algorithms are set as follows:

- Static Priority:
  The priority of each master is assigned according to the required bandwidth. The master with higher required bandwidth has a higher priority.
- Lottery:
  The weight of each master is assigned according to the required bandwidth. That is, the required bandwidth ratio is regarded as the weight ratio.
- TDMA + Lottery:
  1st level – TDMA: Masters with real-time requirements are allocated with time slots accordingly.
  2nd level – Lottery: The weight of each master is assigned according to the required bandwidth. The required bandwidth ratio is regarded as the weight ratio.
- RT_lottery:
  The weight of each master is assigned according to their bandwidth requirements and the traffic behaviors initially. To achieve better bandwidth allocation, a weight tuning mechanism is used to redistribute tickets among masters. More details can be found in [8].
- RB_lottery:
  The weight of each master is assigned and tuned as the process of RT_lottery. The size of observation window is set to 256 cycles in the experiment.

As shown in Table III, since Static Priority and Lottery do not consider the real-time requirements, they fail in all 100 random patterns. TDMA + Lottery may survive in the cases of low bus workload. Compared to other existing arbitration algorithms, RT_lottery is remarkably good. However, RB_lottery performs even better, which reduces about 60% of the average number of failed pattern. The first failed pattern in RB_lottery comes up when the bus workload reaches 85%, which is an extremely high traffic load.

The number of failed patterns stepwise increases while the bus workload rises in all arbitration algorithms. However, the proposed algorithm, RB_lottery, still holds more than 50% successful cases even when the bus workload reaches 95%.

IV.C. Experiment 2

The size of observation window is crucial to the performance of the RB_lottery algorithm. In Experiment 2, the difference sizes of observation window are compared. We set the size of observation window from 128 to infinite and observe the performance. Similar to Experiment 1, 100 random patterns with different required bandwidth combinations for each bus workload are generated and 102400 cycles are simulated for each pattern.

As shown in Table IV, the number of failed pattern is stepwise increased in the higher bus workload but stepwise decreased in the larger size of observation window. Compared to the size of 128 and 2048, the number of failed patterns is reduced by 60%. As a result, a larger size of observation window in RB_lottery can provide better results. However, larger observation window leads to the higher hardware cost. Hence, there is a trade-off between arbitration performance and hardware cost.

In summary, if the bus workload is not that high (less than 70%), RT_lottery, which requires less hardware cost, provides fairly good outcomes. However, when the bus workload is extremely high and the precise bandwidth control is required, RB_lottery should be the best choice.

IV.D. Hardware Implementation

Five arbitration algorithms, Static Priority, TDMA, Lottery, RT_lottery and RB_lottery, are implemented for hardware cost comparisons. All arbiters are designed to handle eight masters. In Lottery, RT_lottery and RB_lottery, the random numbers are generated by an 8-bit LFSR and the tickets are statically assigned. In addition, the size of observation window of RB_lottery is set to 256 cycles. All of the arbiters are synthesized using Synopsys Design Compiler with UMC 0.18-μm standard cell library. The NAND2-equivalent gate counts of Static Priority, TDMA, Lottery, RT_lottery and RB_lottery after synthesis are 215, 1543, 4296, 5134 and 5814, respectively. The hardware cost of Static Priority is the least as expected and then TDMA. The lottery-based algorithms require more logic gates because of the extra circuitry for bandwidth control. Thought RT_lottery and RB_lottery require more logic compared to Lottery, they can provide hard real-time guarantee. Meanwhile, RB_lottery can provide even better bandwidth control than RT_lottery with only slight hardware overhead.

TABLE IV
THE NUMBER OF FAILED PATTERNS UNDER DIFFERENT WINDOW SIZES

Workload (%)	The observation windows size of RB_lottery				
	128	256	512	1024	2048
85	4	1	0	0	0
87	11	1	0	0	0
89	25	11	4	2	0
91	37	25	10	7	7
93	42	31	24	20	14
95	57	44	33	32	28

V. CONCLUSIONS

A three-level arbitration algorithm, RB_lottery, is proposed in this paper. It provides not only the hard real-time guarantee but also the better capability of bandwidth control. The bandwidth regulator is utilized to dynamically monitor the bus communication and thus can precisely control the bandwidth allocation. Four existing arbitration algorithms, Static Priority, Lottery, TDMA + Lottery, and RT_lottery, are compared with RB_lottery. The experimental results show that RB_lottery has the best performance among the five algorithms, but only requires an acceptable hardware overhead compared to the other two lottery-based algorithms. Hence, RB_lottery can be a better choice for those SoC buses with both the real-time and bandwidth constraints.

ACKNOWLEDGMENT

This work was supported in part by the National Science council of Taiwan, R.O.C., under Grant NSC 94-2220-E-009-029. The authors also want to thank the anonymous reviewers for their valuable comments.

REFERENCES

[1] H. Chang, L. Cooke, M. Hunt, G. Martin, A. McNelly and L. Todd, *Surviving the SoC Revolution - A Guide to Platform-Based Design,* Kluwer Academic Publishers, 1999.

[2] C. H. Pyoun, C. H. Lin, H. S. Kim and J. W. Chong, "The Efficient Bus Arbitration Scheme in SoC Environment," *International Workshop on System-on-Chip for Real-Time Applications*, 2003, Page(s): 311-315.

[3] M. Yang, S. Q. Zheng, Bhagyavati and S. Kurkovskyt, "Programmable Weighted Arbiters for Constructing Switch Schedulers," *Workshop on High Performance Switching and Routing*, 2004, Page(s): 203-206.

[4] F. Poletti, D. Bertozzi, L. Benini and A. Bogliolo, "Performance Analysis of Arbitration Policies for SoC Communication Architectures," *Journal of Design Automation for Embedded Systems*, 2003, Page(s): 189-210.

[5] K. Lahiri, A. Raghunathan and S. Dey, "Design of High-Performance System-On-Chips Using Communication Architecture Tuners," *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2004, Page(s): 620-636.

[6] K. Lahiri, A. Raghunathan and G. Lakshminarayana, "The LOTTERYBUS On-Chip Communication Architecture," *Transactions on Very Large Scale Integration Systems*, 2006, Page(s): 596-608.

[7] C. A. Waldspurger and W. E. Weih, "Lottery Scheduling: Flexible Proportional-Share Resource Management," *Proceeding of the First Symposium on Operating Systems Design and Implementation*, 1994, Page(s): 1-11.

[8] C.-H. Chen, G.-W. Lee, J.-D. Huang and J.-Y. Jou, "A Real-Time and Bandwidth Guaranteed Arbitration Algorithm for SoC Bus Communication," *Asia South Pacific Design Automation Conference*, 2006, Page(s): 600-605.

[9] Y. Zhang, "Architecture and Performance Comparison of A Statistic-Based Lottery Arbiter for Shared Bus on Chip," *Asia South Pacific Design Automation Conference*, 2004, Page(s): 1313-1316.