Microelectronics Journal 63 (2017) 81–93

Contents lists available at ScienceDirect

Microelectronics Journal

journal homepage: www.elsevier.com/locate/mejo

Ping-lock round robin arbiter

Alireza Monemi[a,*], Chia Yee Ooi[b], Maurizio Palesi[c], Muhammad N. Marsono[a]

[a] Department of Electrical Engineering, Universiti Teknologi Malaysia, 81310 Johor Bahru, Malaysia
[b] Malaysia-Japan International Institute of Technology (MJIIT), Universiti Teknologi Malaysia, 54100 Kuala Lumpur, Malaysia
[c] Kore University of Enna, Enna 94100, Italy

ARTICLE INFO

Keywords:
Digital circuits
Delay optimization
Round robin arbiter
Fair arbitration

ABSTRACT

Arbiter is the core element in shared-resources systems such as in network-on-chip (NoC), conventional interconnection buses and computer network switch schedulers. Arbiters are located in the critical path delay (CPD) of these systems, that necessitates fast and fair arbitration. This paper proposes two gate-level arbiter architectures. The first arbiter is an improved ping-pong arbiter (IPPA) that is optimized to offer lower execution delay compared to existing round robin arbiters (RRAs). One of the main disadvantages of ping-pong arbiter (PPA) is that fair arbitration is limited to the uniformly-distributed active requests pattern. To solve this problem, we propose a new gate-level RRA, called ping-lock arbiter (PLA). PLA, which is an improved IPPA offers fair arbitration under any distribution of active requests and has the advantage of low execution delay. The FPGA and ASIC implementations of PLA show up to 18% and 12% improvement in average delay, respectively, when compared to existing RRAs in literature.

1. Introduction

Arbiters are used in electronic systems for sharing resources and solving contention when multiple sources request access to a single shared resource that can accept, at most, one request per clock cycle. A round-robin arbiter (RRA) allows fair resource-sharing using a flip-flop based priority pointer p to indicate the priority order of all input requesters. When an RRA receives multiple simultaneous active requests, it grants the one having the highest priority among all requests. In order to provide fair arbitration after granting a request, an RRA updates its priority pointer in such way that the latest granted requester gets the lowest priority order among all requesters.

RRAs are widely used in electronic systems due to its fairness. RRA based switch schedulers in computer network applications provide fair and high utilization of crossbar switches [1–3]. RRAs are also one of the basic elements of multi-core system-on-chips (MCSoCs) that can be found in multi-port memory modules [4], conventional bus-based interconnections [5,6], and in virtual channel (VC) and switch allocators of modern network-on-chip (NoC) routers [7,8]. As arbiters are usually located in the critical path, designing of a faster RRA is important to improve the overall performance of such systems [9].

In this article, we propose the optimization of conventional RRA architecture to minimize arbitration execution delay with reasonable area overhead. The assumption throughout this paper is that arbitration latency is one clock cycle and both requests and grant signals are represented in one-hot format. All arbiters in this paper are described in gate level.[1]

As our goal is to design a fast RRA, we first define two important delay paths in an RRA at the gate level:

1. Arbiter maximum delay (*MD*) is the longest delay between input requests, output grants, and internal priority registers. In other words, it is the maximum among *input-to-output*, *input-to-register*, and *register-to-output* delays.
2. Granting delay (*GD*) is the maximum required delay for an arbiter to assert a grant after receiving a request. In other words, it is the maximum *input-to-output* arbiter delay, where $GD \leq MD$.

Depending on which application the arbiter is used, both *GD* and *MD* values are important metrics to determine the overall arbiter performance. Normally, the output ports of arbiters are connected directly (not registered) to other combinational logics such as multiplexers as shown in Fig. 1. This situation is more common when the arbiter's width is small. In this case, the execution delay of components after the arbiter can be greater than the arbiter priority pointer updating delay. Hence, in the case when arbiter is located in the critical path of a system, reducing the arbiter's *GD* can result in a shorter CPD, rather than reducing the update delay. However, as both *GD* and

* Corresponding author.
E-mail addresses: monemi@fkegraduate.utm.my (A. Monemi), ooichiayee@fke.utm.my (C.Y. Ooi), maurizio.palesi@unikore.it (M. Palesi), nadzir@fke.utm.my (M.N. Marsono).

[1] In this paper, the subscripts following an arbiter name (e.g $IPPA_n$) represents the arbiter size in bits, whereas, when it follows a signal, it represents the n-th bit of the signal.

http://dx.doi.org/10.1016/j.mejo.2017.03.004
Received 30 August 2016; Received in revised form 1 January 2017; Accepted 13 March 2017

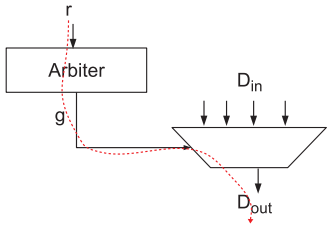

Fig. 1. Bus arbiter.

MD are important parameters, we provide analysis on the effect of both of these delay metrics in this paper.

In this paper, seven most competitive RRAs in literature [2,10–15] are reviewed. In order to compare arbiters' speed, we also developed all arbiters based on their original proposed RTL in Verilog HDL. The RTL codes are described in only using 2-input logic gates (AND, OR, NAND, NOR), NOT gate and flip-flops. Then, we prototyped all arbiters on FPGA by setting the synthesizer to be optimized for delay.

As we will discuss in Section 3, ping-pong arbiter (PPA) shows a better performance than what is expected from its original schematic as proposed in [10]. PPA even performs better than other arbiters when it is mapped to an FPGA device. By analyzing the technology mapped version of PPA by FPGA synthesizer tools, we propose a new delay-optimized schematic version of PPA which is called improved ping-pong arbiter (IPPA). However, the main problem of PPA is that it can only provide fair arbitration under uniformly distributed requests. Unfair arbitration limits the usage of PPA as a general purpose arbiter. To overcome this problem, we propose a new round robin-type arbiter by adding priority lock logic to the IPPA that results in a fair arbitration. As the lock logic delays are only added to the priority updating delay path, our proposed arbiter can offer similar *GT* delay to IPPA while provides a fair arbitration to all requesters.

The rest of this paper is organized as follow: Section 2 reviews the existing works on RRA design while provides extensive analysis on RRA architectures. The related works' delay evaluation and motivation for optimizing arbitration delay are discussed in Section 3. Section 4 proposes an improved version of ping-pong arbiter (IPPA) which is optimized for minimum execution delay. Section 5 shows how ping-pong typed arbiters cannot provide fair arbitration under non-uniform load distribution or when the arbiter's width is not a power-of-two. Section 6 describes our proposed PLA, which is a new RRA based on IPPA architecture. PLA offers an IPPA's minimum delay while provides fair arbitration of an RRA. Section 7 provides fair arbitration observation experiment as well as FPGA and ASIC results for all arbiters. We conclude this paper in Section 8.

2. Related works

This section reviews existing works on RRA design [2,10–17]. This includes description and detailed micro-architecture of each arbiter.

2.1. Baseline RRA

A generic RRA [16] can be built from arbiter cells as shown in Fig. 2. While this structure results in a low area cost RRA architecture, the two wrap-around wires that connect the last cell's c_o and g_o outputs to first cell's c_i and g_i inputs result in combinational loops. This baseline

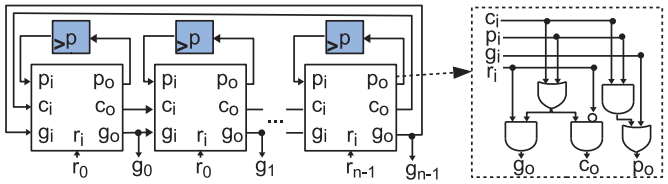

Fig. 2. Baseline RRA.

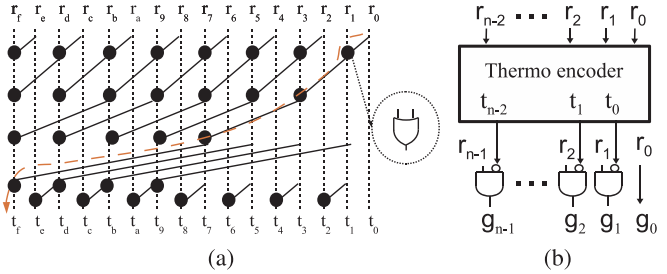

Fig. 3. FPE (a) A thermometer encoder using Han-Carlson parallel prefix adder [21]. (b) Generating FPE using thermometer encoder.

RRA has been improved in several other related works [18–20], although the combinational loop still exists in all aforementioned works. As combinational loops prohibit static timing analysis by commercial synthesis tools and may cause pitfalls in testing, we did not focus on these types of RRAs.

2.2. Programmable priority encoder (PPE) arbiter

Programmable priority encoder (PPE) arbiter [2] is a round robin arbiter that is composed of two fixed priority encoders (FPEs). An FPE is an arbiter which always serves its input requesters by a fixed priority order, where the least and the most significant requests have the highest and the lowest priority order, respectively. FPE grants an active request only if there is no other active request(s) between it and the least significant request.

Fig. 3 illustrates the structure of an FPE that is thermometer-coded input requests masked by ANDing with the input requests. The thermometer encoder can be implemented efficiently using parallel prefix networks (PPNs). Ugurdag et al. [14] compared the performance of RRA using four different PPNs, namely as Kogge-Stone [22], Ladner-Fischer [23], Brent-Kung [24] and Han-Carlson (HC) [21]. The comparison in [14] showed that HC [21] outperforms the rest for most arbiter sizes. As a result, we select HC for generating thermometer encoder as illustrated in Fig. 3(a).

PPE arbiter [2] is generated by using two FPEs working in parallel as shown in Fig. 4. The PPE's input requests are fed-in directly to the first FPE, while the second FPE receives a subset of input requests masked with priority pointer value *p* (which is the thermometer-coded of the highest priority request). Hence, the second FPE searches from the highest priority requester to the most significant requester. If any request is granted by the second FPE, the final grant is the output of second FPE. Otherwise it is selected from the first FPE.

2.3. Ping-pong arbiter (PPA)

Ping-pong arbiter (PPA) was proposed by Chao et al. [10]. PPA is made by connecting narrow arbiters (2-bit arbiters) in a binary tree format to generate wider arbiters. Fig. 5(a) shows the schematic of a 2-bit PPA (PPA_2) as proposed in [10]. The grant signal of a PPA_2 is valid only if it also receives the next level PPA_2 grant. Hence, the grant signals of arbiters in each layer are masked with the grant signal of their

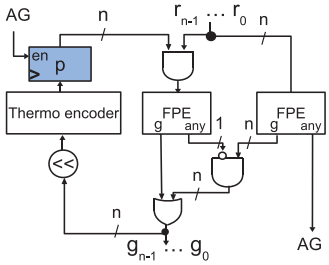

Fig. 4. PPE arbiter functional block diagram.