Hierarchical Round Robin Arbiter for High-Speed, Low-Power, and Scalable Networks-on-Chip

N. Sertac Artan, Ming Yang, and H. Jonathan Chao

Department of Electrical and Computer Engineering
Polytechnic Institute of NYU, Brooklyn, NY

Abstract—Recently, the Round Robin Arbiter (RRA), a crucial building block for high-speed switches/routers, receives a new attention with the advent of the Networks-on-Chip (NoC). In this paper, we revisit the RRA design with NoC as the new target application platform. We propose the Hierarchical Round Robin Arbiter (HRRA) a high-speed, low-power, area-efficient, and scalable RRA for NoC applications, which completes the arbitration in one short clock cycle. HRRA shows up to 34.28% increase in clock frequency, whereas it decreases the area and power consumption up to 19.53% and 33.81%, respectively, compared to previous methods.

Index Terms—Round robin arbiter, Hierarchical round robin arbiter, System-on-chip, Network-on-chip, switch allocation

I. INTRODUCTION

The Round Robin Arbiter (RRA) is a building block for many applications. In particular, it is a crucial building block for high-speed network switches/routers [1]–[5]. For instance, when inputs of a network switch are competing for the usage of the switch to send traffic to their respective outputs, Round Robin-based sophisticated arbitration schemes, such as iSLIP [6], Ping-Pong Arbitration [7] or DRRM [8] are used for fair sharing of switch bandwidth between inputs.

Recently, with the advent of the Networks-on-chip (NoC), such fair arbitration schemes receive a new attention for similar problems such as Virtual Channel (VC) Flow Control [9] or NoC Switch Allocation [10]. In traditional networks, thanks to the large distances between switches/routers, the arbitration have multiple clock cycles to complete. For NoC, on the other hand, the arbitration is desired to be completed within one clock cycle to avoid large latencies between the cores on the chip (e.g., between a processing element (PE) and a memory block). To achieve the arbitration in one clock cycle, the overall delay introduced by the arbitration should be low so that it will not impact the overall system clock frequency, which introduces new challenges for the design of the arbiters.

In this paper, we revisit the RRA design with NoC as the new target application platform. We propose the Hierarchical Round Robin Arbiter (HRRA) a high-speed and area-efficient RRA for NoC applications, which completes the arbitration in one short clock cycle. HRRA is unique in the way it divides the arbitration process into simpler consecutive steps, where at each step a novel high-performance and light-weight RRA is used, leading to a better overall arbitration performance. We compare HRRA with RRA in previous work and show up to 34.28% increase in clock frequency, and a decrease of up to 19.53% and 33.81% in the area and power consumption, respectively, compared to previous methods. Note that unlike the recent work on RRA for NoC, which addresses the more sophisticated arbitration schemes based on RRA, such as iterative methods [9] and distributed methods [10], our focus in this paper is on the RRA design.

The rest of the paper is organized as follows, in Section II, we briefly describe a generic RRA and identify issues limiting its scalability. In Section III, we introduce the proposed Hierarchical Round Robin Arbiter (HRRA). The performance results comparing HRRA to the previous work is given in Section IV. Section V concludes the paper.

II. PROBLEM DEFINITION

A. A Generic Round Robin Arbiter (RRA)

Given a set of requests from a set of inputs for a resource, the task of an RRA is to grant requests one input at a time in a fairly manner. A generic RRA consists of two main sub-blocks as shown in Figure 1: (1) Input Selector, and (2) Pointer Updater. The Request input port represents the \( N \) inputs with up to a total of \( N \) requests, and Grant output port represents the granted input with \( E = \log_2(N) \) bits. To achieve fairness among inputs, the RRA keeps an \( E \)-bit pointer (RPT) to the next possible input that would be granted in the next request-grant cycle.

The operation of the RRA is as follows. At the beginning of every clock cycle, each input with a request sets the corresponding bit in the Request to high. Input Selector decides the input that would be granted next, based on the Request and the current value of RPT. If there is a request from the input
pointed by RPT (i.e., Request \( [\text{RPT}] = \text{high} \)), this input will be granted by setting the Grant output value to the index of this input, \( g \). If there is no request from the input pointed by RPT, but there are other requests, the Input Selector grants the first input with a request following the pointer in a circular manner (i.e., if there is any request from inputs \( i > \text{RPT} \), the smallest indexed input, \( g > \text{RPT} \) with a request will be granted. If there are no requests from inputs \( i > \text{RPT} \), but there are requests from inputs, \( i < \text{RPT} \), the smallest indexed input, \( g \neq \text{RPT} \) with a request, will be granted.). At the end of the clock cycle, the Pointer Updater sets the pointer value to the input next to the granted input in a circular manner (i.e., \( \text{RPT} = (g + 1) \mod N \)). If there is no request from any inputs, the \( \text{RPT} \) will not change. An optional no request (NoReq) output port can be added to the generic RRA, which will be high when no input has a request.

B. Limitations on the Scalability of RRA

The most time-consuming operation of the generic RRA is the granting of the requests by the Input Selector, which also dominates the critical path delay. The complexity of the Input Selector are due to two main issues: (1) The issue of changing priority: The priority of the inputs changes as inputs are granted and the RPT value changes. This requires the Input Selector circuit to consider all possible priority settings. (2) The issue of circular priority order: The priority order is circular, which makes the priority processing even harder. This leads to two separate conditions for the grant decisions: Grant decisions for requests at or below the Request \( [\text{RPT}] \) and grant decisions for requests above Request \( [\text{RPT}] \). Any grant produced by the former decision has a higher priority over any grant produced by the latter decision. This two parted decision deepens the critical path.

These two issues are even more pronounced as the RRA size gets larger (i.e., an RRA with more inputs). Although, pipelining the RRA may alleviate these issues, the latency introduced by pipelining is not desired, especially in NoC designs.

III. Hierarchical Round Robin Arbiter (HRRA)

A. Overview of HRRA

In HRRA, we apply a hierarchical approach to round robin arbitration to achieve scalability with increased number of inputs. First, we divide the \( N \) requests into \( k \) sets of \( n \)-input RRAs, which are called as SRRAs here. In each SRRRA, we make the arbitration among the local requests. The arbitration among the local winners are done in subsequent stages of SRRAs. This localized approach ensures that only one SRRRA (described as active SRRRA below) at each stage at any given time needs to deal with the two issues given above. All the remaining SRRAs (described as passive SRRAs below) work on a much simpler single, non-circular priority setting which simplifies their operation. A simple mechanism (described as Pass below) ensures the smooth transition of the active state from one SRRRA to another as the round robin order requires.

In the next subsection, we provide an example HRRA for \( N = 16 \) to describe the basic operation of HRRA. The following subsection generalizes the HRRA design to arbitrary \( N \) values. The final subsection discusses the SRRRA design.

B. An example HRRA Architecture, HRRA16

An example 16-input Hierarchical Round Robin Arbiter (HRRA16) is shown in Figure 2. HRRA16 consists of two RRA stages. The RRA submodules in HRRA are called as sub-RRAs (SRRAs) to distinguish them from generic RRAs and the request inputs of each SRRRA in the HRRA are called as the local request inputs of the respective SRRRA to distinguish them from the Request inputs of the HRRA. The first stage of HRRA16 consists of 4 4-input SRRAs, called as SRRA0_3 to SRRA0_0. The second stage of HRRA16 consists of one 4-input SRRRA, called as SRRA1.

1) Operation of the first stage SRRAs: The local request inputs of SRRA0’s are connected to the Request inputs of the HRRA (4 Request input per SRRA0) as shown in Figure 2. The SPTs\(^1\) of all SRRA0s are set to an initial value pointing to the highest indexed local input for their respective SRRA0.

At the beginning of every clock cycle, each SRRA0, selects one local request in a round robin fashion. The index of the selected local request is shown at the Select outputs of the SRRA0s. If there is no request in an SRRA0, its NoReq output will be set to high. For any SRRA0, if the selected request index is below the SPT0 pointer value, the DownRequest output (DRQ) will be set to high. The DRQ is used to distinguish between requests above the SPT0 and requests at or below the SPT0. If an SRRA0 is granted, the SPT0 value of the granted SRRA0 is set to the index of the local request, next to the selected request, in a circular manner. The SRRA0s are granted by the SRRRA1, whose operation is explained in detail below. In order to realize the round robin operation, the requests to HRRA should be followed in a round robin fashion regardless of the SRRA0 a request is connected to. To achieve

\(^1\)To distinguish from the main pointer of the HRRA (the RPT), the pointers in SRRAs of the HRRA are represented as SPT.
this goal, the Pass mechanism is used based on an extra output, Pass_out in the SRRA0, which shows that the last request in an SRRA0 is just granted. The Pass_out value of an SRRA0 will be one, only if the Select output is equal to the smallest request index, which is zero and this SRRA0 is granted.

2) Operation of the second stage SRRA: The local request inputs of SRRA1 are connected to the DRQ outputs of the SRRA0s. The SPT1 pointer is set to an initial value pointing to the highest indexed SRRA0. At every clock cycle, SRRA1 selects one of the SRRA0s with DRQ output set to high in a round robin manner. The index of the selected SRRA0 is shown at the Select output of the SRRA1. SRRA1 has a 4-bit output called Granted. If an SRRA0 is selected by the SRRA1, the corresponding bit in the Granted output will be set to high. Each of these bits show whether the corresponding SRRA0 is granted or not. The SRRA0, that is granted is called the active SRRA0 and all the remaining SRRA0s are called the passive SRRA0s, whose SPT is always equal to the highest local index (e.g., 4). The update of SPT1 pointer of SRRA1 is slightly different than the update of the SPT0 pointer. SPT1 will be set to the index of the granted SRRA0 (active SRRA0), as long as the Pass_out output of the granted SRRA0 is not set. If this output is set, the SPT1 of SRRA1 is set to the index of the SRRA0, next to the granted, in a circular manner. This SPT1 update allows smooth continuation of the granting operation from one SPT1 to another.

The most significant two bits of the Grant output of the HRRA16 is equal to the Select output of the SRRA1. The least significant two bits of the Grant output is selected as the Select output of SRRA0, granted by the SRRA1, as shown in the Grant Generation inset in Figure 2.

If no DRQ is high, this means no passive SRRA0s has a request. However, the active SRRA0 may still have a request above the pointer. The grant to this request is automatic, since the SPT1 pointer will not move. If NoReq output is high for all SRRA0s, this means there is no request to HRRA16, and the NoReq output of SRRA1 is set to high as shown in the NoReq Generation inset in Figure 2. In this case, the SPTS will not change.

C. Generalized HRRA

In general, HRRA can be extended to multiple stages as the number of request inputs increases. An HRRA with \( N \) inputs consists of \( S \geq 2 \) stages, where each stage \( s \) consists of \( n \cdot 2^{s-1} \) n-input RRAs\(^2\). The \( r \)th SRRA in stage \( s \) is represented with SRRA\(_{s,r}\), where \( (r \geq 0, s \geq 1) \). Only the first stage SRRAs have the actual Request inputs of HRRA as their local inputs. The SRRAs on the subsequent stages receive their local inputs from the DRQ output of SRRAs in the previous stage.

The RPT in HRRA is also divided into \( S \) parts, where the first stage SRRA pointers (SPT\(^3\)) provide the least significant \( \frac{s}{s} \) bits of the RPT (bits \( 0 \ldots \frac{s}{s} - 1 \)), the second stage provides

\[
\frac{2}{s} \cdot 2e - 1, \quad \text{where} \quad e = \log_2(n).
\]

In general, stage \( i \) provides the bits \( \left( \frac{(i-1)e}{s} \ldots \frac{ie}{s} - 1 \right) \) of the SRPT. In each stage at any given time there can be only one active SRRA.

Let us consider two consecutive stages; stage \( i \) and stage \( j \), where \( j = i + 1 \). Each SRRA in stage \( j \) has request inputs which are connected to the DRQ outputs of the \( n \) SRRAs in stage \( i \). If a particular SRRA\(_{j,k}\) in stage \( j \) is connected to the active SRRA in stage \( i \), and if this active RRA’s DRQ is high, the SRRA\(_{j,k}\) will grant the active SRRA. If the DRQ of the active SRRA is low, then the SRRAs in stage \( i \) are granted in a round robin manner by the SRRA\(_{j,k}\). Note that if none of the passive SRRA in stage \( i \) has a request, and the only request(s) are coming from the local inputs of the active SRRA, with indices greater than the active SPT, the highest indexed request among these local requests will be granted.

SRRAs in each stage \( j \) grant among the SRRAs from the previous stage that are connected to itself. However, this grant is only effective if the granting SRRA at stage \( j \) is granted by all subsequent \( p \) stages (\( p = S - j \)). To achieve this goal, each stage \( j \) propagates the result of the grant it received from the subsequent stage to the previous stage by means of an AND tree. Each SRRA in stages \( s > 1 \) ANDs its grant with the grant from subsequent stage, before sending the grant to the previous stage. The request to the HRRA is granted when the first stage SRRA receives the grant.

D. The Sub-RRA (SRRA) Design for HRRA

In this section, we describe the details of an SRRA architecture that can be used in any stage of the HRRA. Along with the conventional input Request and output Select, the generalized RRA introduces two inputs for pointer update (PT_Default and PT_Flag). There are also three additional outputs (DRQ, Granted, and Pass_out). The DRQ and Pass_out outputs are used in all SRRAs, except the SRRAs in the last stage. The Granted output is used in all SRRAs, except the SRRAs in the first stage.

1) Pointer Updater Design: The SPT pointer is reset to the maximum request index value, which is equal to \( n \). The pointer value will be determined based on the pointer flag (PT_Flag) input, when PT_Flag is high, the SPT will be set to the index of the input next to the selected input. When PT_Flag is low, the SPT will be set to the PT_Default value. For the SRRA0s, PT_Default = \( n \), and PT_Flag = Granted. Thus,
when an SRRA0 is granted, the SPT value will be decremented and when an SRRA0 is not granted, the SPT value will be reset to \( n \). For the SRRA\( s \) in the subsequent stages, \( PT_{\text{Default}} = \text{Select} \) and \( PT_{\text{Flag}} = \text{Pass\_in} \). So, for the SRRA\( s \) in stages \( s > 0 \), the SPT keeps pointing to the same SRRA in the previous stage until the PASS\_in is high, which means the active SRRA in the previous stage is changed. In this case, SPT will have the value of Select\( -1 \).

2) Input Selector Design: The Input Selector is implemented as a Lookup Table (LUT) in SRRA. For instance, for \( n = 4 \), the lookup table has a 6 bit input consisting of the Request input and the SPT value. The lookup table output for \( n = 4 \) has 3 bits, consisting of the Select output and the DRQ output. In addition to the LUT, a simple logic is implemented to realize \text{Pass\_out} and \text{NoReq}.

IV. PERFORMANCE EVALUATION

To compare the performance of the HRRA to the performance of PPE, we have implemented different size (8, 16, 32, and 64 input) HRRA\( s \) and PPE\( s \) ([2]) using 65 nm technology. If \( N \) is a power of \( n = 4 \) (e.g., 16 or 64), \( n = 4 \)-input SRRA\( s \) are used for the entire HRRA. If \( N \) is not a power of \( n = 4 \) (e.g., 8 or 32), a \( n = 2 \)-input SRRA is used for the last stage of the HRRA and \( n = 4 \)-input SRRA\( s \) are used for the rest of the HRRA. Table I, shows the post-layout clock frequency, area and energy results for the PPE and HRRA arbiters. The results show that all the HRRA\( s \) except HRRA64 can easily go beyond 1 GHz clock frequency and HRRA64 is very close to 1 GHz clock frequency, whereas neither PPE32 nor PPE64 can reach to 1 GHz speed. HRRA is superior to PPE for the three performance parameters, speed, area, and power as number of inputs increases. The performance improvement of HRRA over PPE can be better seen from Figure 4, which shows the percentage change from PPE to HRRA for the three performance parameters. In particular, the HRRA shows up to 34.28\% increase in clock frequency, and up to 19.53\% and 33.81\% decrease in area, and power consumption, respectively, compared to PPE. Furthermore, it can be easily seen that for all three parameters, the improvement from PPE to HRRA increases as the number of inputs increases, demonstrating the scalability of the HRRA. The critical path of the HRRA is dominated by one LUT from each stage. The critical path in each LUT (for \( n = 4 \)) is shown in Figure 5.

V. CONCLUSION

The recent advances in Networks-on-Chip (NoC) introduced new challenges for switches/routers for NoC. Round robin arbiters, crucial building blocks for switches/routers, are required to complete an arbitration within one short clock cycle to provide low-latency links between processing elements and memory blocks. In this paper, we proposed Hierarchical Round Robin Arbiter (HRRA), a high-speed, low-power and scalable solution for round robin arbitration in NoC. We show that HRRA outperforms the previous work for all three performance parameters (clock frequency, area and power) as the design scales to large number of inputs.

REFERENCES