



## Efficient iSLIP Arbiter with iSLIP Scheduling Algorithm for NoC

Authors

**Deepali Thakur Mahobiya**

M-TECH Scholar

Department of Electronics & Communication Engineering  
RKDF Institute of Science & Technology, Bhopal M.P India

Email:- [thakur.deepali1@gmail.com](mailto:thakur.deepali1@gmail.com)

### Abstract

*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 computation resources arbitration being the performance bottleneck of the NoC system. The arbitration is desired to be completed within one clock cycle to avoid large latencies between the cores on the chip. To achieve the arbitration in one clock cycle, the total 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 design iSLIP arbiter using iSLIP scheduling algorithm with mesh router for NoC. An iterative, round-robin algorithm, iSLIP can achieve 100% throughput for uniform traffic. It is well known that if simple first in first out (FIFO) input queues are used to hold packets then, even under benign conditions, head-of-line (HOL) blocking limits the achievable bandwidth to approximately 58.6% of the maximum. HOL blocking can be overcome by the use of virtual output queuing, which is described in this paper. It is designed using VHDL and using ModelSimSE 6.3f.*

**Keywords:** iSLIP, iterative, scheduling algorithm, mesh router, arbiter.

### 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, 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. Arbiters are 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. As fabrication technology continues to improve, smaller feature

sizes allow increasingly more integration of system components onto a single die.

In recent years, there has also been a significant amount of research work on scheduling algorithms for crossbar switches that use virtual output queuing. Scheduling algorithms have been developed by a number of researchers [5],[6],[7],[8],[11], for research prototypes [2],[4], and commercial products [3]. What makes a good crossbar scheduling algorithm for the backplane of a router?

We desire algorithms with the following properties:

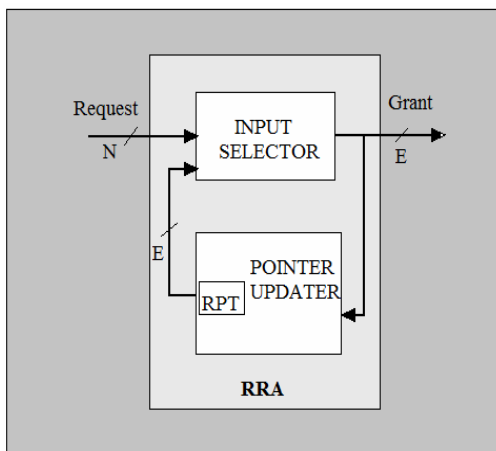
1. *High Throughput* — An algorithm that keeps the backlog low in the VOQs (virtual output queuing). Ideally, the algorithm will sustain an offered load up to 100% on each input and output.
2. *Starvation Free* — The algorithm should not allow any VOQ to be unserved indefinitely.

3. *Fast* — To achieve the highest bandwidth switch, it is important that the scheduling algorithm does not become the performance bottleneck. The algorithm should therefore select a crossbar configuration as quickly as possible.
4. *Simple to implement* — If the algorithm is to be fast in practice, it must be implemented in special-purpose hardware; preferably within a single chip.

Satisfying the above criterions here we present *i*SLIP arbiter using *i*SLIP scheduling algorithm with on-chip mesh router. The rest of the paper is organized as follows, in Section 2, we briefly describe a generic RRA(Round RobinArbiter) and identify issues limiting its scalability. In Section 3, we introduce the proposed *i*SLIP Arbiter and its interconnections. Section 4 describes the *i*SLIP scheduling algorithm. Section 5 shows simulation results. Section 6 concludes the paper.

## 2.Problem Formulation

### 2.1 Generic Round Robin Arbiters (RRA)



**Figure 1:** Generic Round Robin Arbiter architecture.

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 [ $RPT$ ] = 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  $i > RPT$ , the smallest indexed input,  $g > RPT$  with a request will be granted. If there are no requests from inputs  $i > RPT$ , but there are requests from inputs,  $i < RPT$ , the smallest indexed input,  $g \neq RPT$  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.*,  $RPT = (g + 1) \bmod N$ ). If there is no request from any inputs, the  $RPT$  will not change. An optional no request (*No Req*) output port can be added to the generic RRA, which will be high when no input has a request.

### 2.2 Limitations on the Scalabilities 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 [ $RPT$ ] and grant decisions for requests above Request [ $RPT$ ]. 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). Also RRA has its throughput near about 63%.

### 3. The iSLIP Arbiter

In this study, we propose a new centralized arbiter, which may be used in arbitration of a crossbar switch in NoC routers. The iSLIP arbiter is similar to Round-Robin arbiter, but with one change that here we keep a track of the requesting node and the grant given to the nodes previously, so that no node needs to wait for long time to get a grant. Maximum waiting period would be 1 clock cycle. iSLIP arbiter would keep a variable to check if this particular node has been assigned the grant signal. Each node gets the grant signal for one clock cycle. Here mesh router will send the data effectively from source node  $N_S$  to destination node  $N_D$  routing it through first in x-direction then in y-direction using iSLIP algorithm. The input packet format is shown in Figure 2 for source nodes:

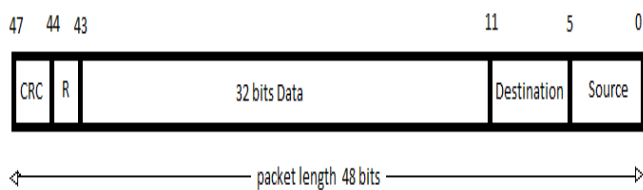


Figure 2. Input Packet Format

Field	Size	Description
CRC	3 bits	check bits
R	1 bit	request bit
Data	32 bits	length of data
Destination	6 bits	destination address
Source	6 bits	source address

As the requesting node enables its request bit (i.e.  $R=1$ ) the arbiter will match the source address to its destination and give the grant to requesting node. iSLIP arbiter will keep the record of the nodes to whom grant was previously given and therefore it will arbitrate to only requesting nodes to ensure that no node would starve for getting the grant. Table 1 shows the grant giving process of requesting nodes .

Table 1. Grant giving process

	1	2	3	4 ..... →	Requesting Nodes
CLK ↓	0	0	1	1 →	Request Bit
1st	0	0	1	0 →	Request of node 3 processed
2nd	0	0	0	1 →	Request of node 4 processed
3rd	0	0	1	0 →	Request of node 3 processed
4th	0	0	0	1 →	Request of node 4 processed
5th	0	0	1	0 →	Request of node 3 processed

As soon as the grant received then the data will route through mesh using the iSLIP algorithm. Figure. 3 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 Figure. 3.

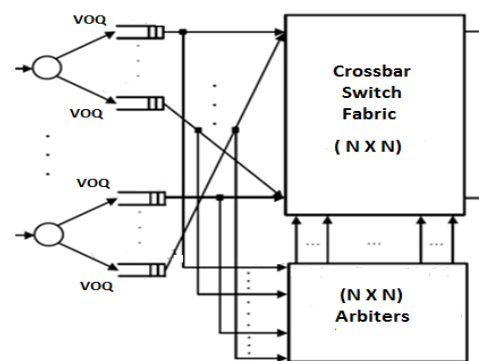


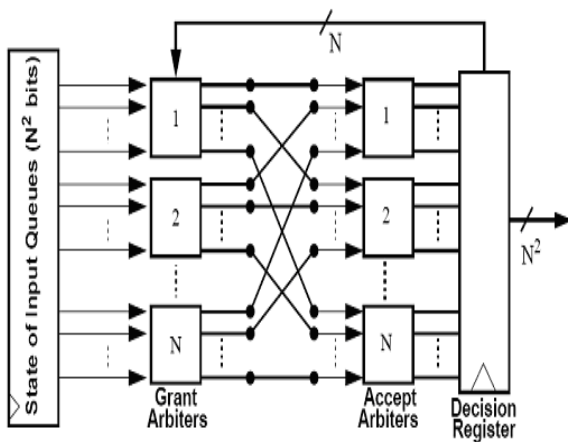
Figure 3: An  $n \times n$  switch: block diagram, crossbar switch fabric and its switch arbiters.

#### Algorithm to configure the crossbar :

1. Centralized scheduler serving the  $N^2$  VOQs – Content of the VOQs examined at beginning of each cell time.
2. Based on iSLIP arbitration – arbiters schedule cells at each output and at each input.

- Input makes request to every output for which it has a queued cell.
- Output grants request that appears next in schedule.
- Input accepts grant that appears next in schedule.
- When arbitration is completed, matching between input and output has been found.

Figure. 4. shows an example baseline implementation of the *i*SLIP algorithm. This implementation requires an  $N \times N$  bit memory to contain the states of the input queues ( $N$  inputs  $\times$   $N$  output queues per input),  $N$  Grant Arbiters,  $N$  Accept Arbiters, and decision register to send control information to the datapath. The Grant and Accept Arbiters each consist of programmable priority encoders and a state pointer to record which input should have the highest priority on the next arbitration cycle. The  $N$ -bit feedback signal from the Decision registers to the Grant arbiters is an enable vector, which enables arbitration only for unmatched ports on each successive iterations.



**Figure 4.**  $2N$  arbiters ( $N$  input and  $N$  output) interconnected to construct an *i*SLIP scheduler for an  $N \times N$  switch Decision register configures the crossbar.

The arbiter is responsible for configuring the input blocks and the crossbar muxes to maximize utilization of the crossbar for each transmit cycle. Each data block indicates which destinations it has pending packets for, and the scheduler arbitrates to match inputs to destinations using the *i*SLIP scheduling algorithm. The crossbar interconnect uses a simple mux-based crossbar to enable a

completely synthesized design. Once the scheduler configures the mux selects and data block destinations, the input\_blocks simply pump the data through the interconnect to the output\_blocks over multiple cycles. Once the output block receives the full packet, it transmits the packet to the connected output device. *i*SLIP is good for high speed as its input buffers are separated. For each input and output there is a Separate scheduler & each works independently.

## 4. *i*SLIP Scheduling Algorithm

### 4.1 *i*SLIP Algorithm

The *i*SLIP algorithm is designed to meet our goals. *i*SLIP is an iterative algorithm — during each time slot, multiple iterations are performed to select a crossbar configuration, matching inputs to outputs. The *i*SLIP algorithm uses *i*SLIP arbitration to schedule each active input and output in turn. The main characteristic of *i*SLIP is its simplicity; it is readily implemented in hardware and can operate at high speed. A prototype version of *i*SLIP is currently being implemented that makes a new scheduling decision every  $40\text{ns}$ <sup>[10]</sup>. *i*SLIP attempts to quickly converge on a conflict-free match in multiple iterations, where each iteration consists of three steps. All inputs and outputs are initially unmatched and only those inputs and outputs not matched at the end of one iteration are eligible for matching in the next. The algorithm for *i*SLIP, taken from <sup>[1]</sup>.

The steps of each iteration are:

**Step 1. Request.** Each input sends a request to every output for which it has a queued cell.

**Step 2. Grant.** If an output receives any requests, it chooses the one that appears next in a fixed, schedule starting from the highest priority element. The output notifies each input whether or not its request was granted.

**Step 3. Accept.** If an input receives a grant, it accepts the one that appears next in a fixed, schedule starting from the highest priority element. By considering only unmatched inputs and outputs, each iteration matches inputs and outputs that were not matched during earlier iterations.

### 4.2 Performance

Despite its simplicity, the performance of *i*SLIP is surprisingly good. A detailed study of its performance and implementation can be found in [9], and more briefly in [10]. In brief, its main properties are:

**Property 1. High Throughput** — For uniform, and uncorrelated arrivals, the algorithm enables 100% of the switch capacity to be used.

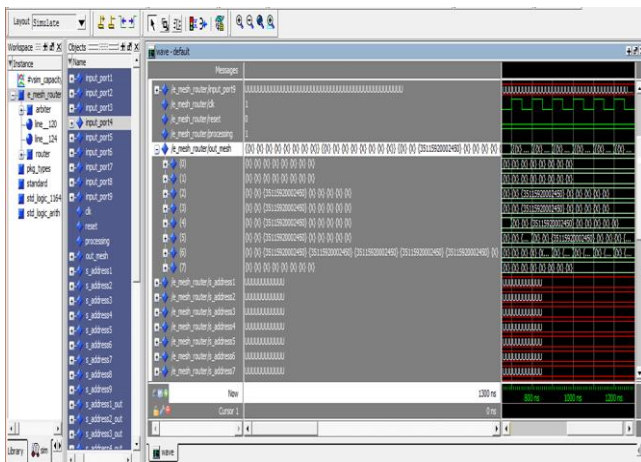
**Property 2. Starvation Free** — No connection is starved. Because pointers are not updated after the first iteration, an output will continue to grant to the highest priority requesting input until it is successful. Furthermore *i*SLIP is in some sense fair: with one iteration and under heavy load, all queues with a common output have identical throughput.

**Property 3. Fast** — The algorithm is guaranteed to complete in at most  $N$  iterations. However, in practice the algorithm almost always completes in fewer than iterations. i.e. for a switch with 16 ports, four iterations will suffice.

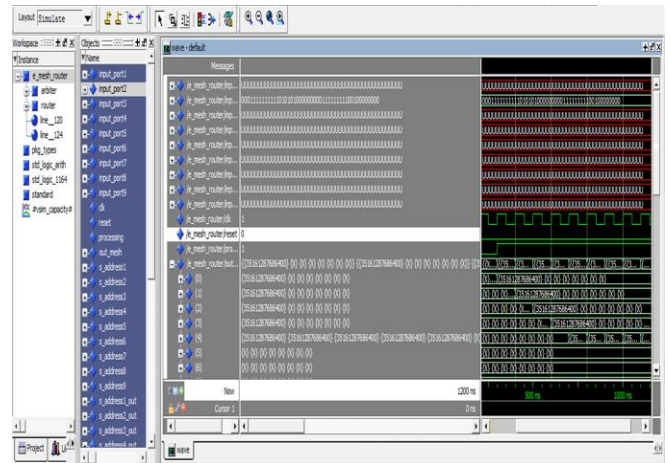
**Property 4. Simple to implement** — An *i*SLIP scheduler consists of  $2N$  programmable priority encoders. A scheduler for a 16-port switch is readily implemented on a single chip.

### 5. Simulation results

1] Data from source (2,3) is routed to the destination node (6,7), as input port 4 requested to send the data.



2] Data from source (0,0) is routed to the destination node (4,4), as input port 2 requested to send the data.



### 6. Conclusion

The recent advances in Networks-on-Chip (NoC) introduced new challenges for switches/routers for NoC. *i*SLIP 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 *i*SLIP Arbiter & *i*SLIP scheduling algorithm for on-chip mesh router, a high-speed, high throughput, starvation free and easy to implement solution for arbitration in NoC. The arbiter trap the source and destination address from the output of buffer and generate the control signal so that input data from source side sending to the output port.

### References

1. McKeown, N. W. 1995 *Scheduling Algorithms for Input-Queued Cell Switches*. Doctoral Thesis. UMI Order Number: UMI Order No. GAX96-02658., University of California at Berkeley.
2. C. Partridge; P. Carvey et al. “A Fifty-Gigabit per Second IP Router,” Submitted to IEEE/ACM Transactions on Networking, 1996.
3. T. Anderson et al. “High Speed Switch Scheduling for Local Area Networks,” ACM Trans. on Computer Systems, Nov 1993, pp. 319-352.
4. N. McKeown et al. “The Tiny Tera: A small high-bandwidth packet switch core,” IEEE Micro, Jan-Feb 1997.

5. LaMaire, R.O.; Serpanos, D.N.; "Two-dimensional round-robin schedulers for packet switches with multiple input queues," *IEEE/ACM Transactions on Networking*, Oct. 1994, vol.2, (no.5):471-82.
6. Lund, C.; Phillips, S.; Reingold, N.; "Fair prioritized scheduling in an input-buffered switch," *Proceedings of the IFIP-IEEE Conf. on Broadband Communications '96*, Montreal, April 1996. p. 358-69.
7. Hui, J.; Arthurs, E. "A broadband packet switch for integrated transport," *IEEE J. Selected Areas Communications*, 5, 8, Oct 1987, pp 1264-1273.
8. Ali, M.; Nguyen, H. "A neural network implementation of an input access scheme in a high-speed packet switch," *Proc. of GLOBECOM 1989*, pp.1192-1196.
9. N. McKeown, "Scheduling Cells in an input-queued switch," PhD Thesis, University of California at Berkeley, May 1995.
10. N. McKeown, "iSLIP: A Scheduling Algorithm for Input-Queued Switches," Submitted to *IEEE Transactions on Networking*.
11. Maher Abdelrasoul, Mohammed Ragab, Victor Goulart "Evaluation The Scalability of Round Robin Arbiters for NoC Routers onFPGA" of 2013 *IEEE 7th International Symposium on Embedded Multicore/Manycore System-on-Chip*.
12. D. Cavendish, M. Lajolo, and H. Liu. On the evaluation of fairness For input queue switches. In *IEEE International Conference on Communications*, 2002. ICC 2002, volume 2, 2002.
13. G. Varghese. *Network Algorithmics: an interdiscilinary aproach to designing fast networked devices*. Morgan Kaufmann, 2005 T. D. Richardson, C. Nicooulos, D. . and V. Narayanan, Y. Xie, C. Das, and V. Degalahal, "A hybrid soc interconnect with dynamic tdma-based transaction-less buses and on-chi networks," in roc. Of *International Conference on VLSI Design*, 2006.
14. S. Bourduas and Z. Zilic, "A hybrid ring/mesh interconnect for networkon chip using hierarchical rings for global routing," in *International Symosium on Network-on-Chi*, 2007
15. H. M. Harmanani and R. Farah, "A method for efficient maping and reliable routing for noc architectures with minimum bandwidth and area," in *IEEE Northeast Workshop on Circuits and Systems and TAISA Conference*, June 2008.
16. W. Lee and G. E. Sobelman, "Mesh-star hybrid noc architecture with cdma switch," in *IEEE roc. of International Symosium onCircuits and Systems*, 2009.
17. Kang, K., K. Park, L. Sha, Q. Wang. Design of a crossbar VOQ real-time switch with clock-driven Scheduling . *Real-Time Systems*, January2013, Vol. 49, Issue 1, pp. 117-135.
18. Tejaswini Rajesh Deotare, Prashant R.Rothe; "I-SLIP algorithm for low latency on hybrid noc architecture" *IJRET: International Journal of Research in Engineering and Technology* eISSN: 2319-1163 | pISSN: 2321-7308 Volume: 03 Issue: 07 | Jul-2014, Available @ <http://www.ijret.org> 509

#### Author Profile



**Deepali Thakur Mahobiya** received the B.E degree in Electronics & commnication Engineering from Sagar Institute of Science & Technology, Bhopal (M.P) INDIA in 2010 and currently pursuing M.E in Digital Communication from RKDFIST, Bhopal (M.P), INDIA.