ELectronics EXpress

LETTER

IEICE Electronics Express, Vol.14, No.3, 1–6

BARR: Congestion aware scheduling algorithm for Network-on-Chip router

Nan Su[1,3], Kun Wang[2a)], Xiaoshan Yu[1], Huaxi Gu[1], Yantao Guo[3], and Jiayi Chen[1]

[1] *State Key Laboratory of Integrated Service Networks, Xidian University, Xian, China*

[2] *School of Computer Science and Technology, Xidian University, Xian, China*

[3] *Science and Technology on Information Transmission and Dissemination in Communication Networks Laboratory, Shijiazhuang 050081, China*

a) *kwang@mail.xidian.edu.cn*

Abstract: Scheduling algorithm is crucial to the performance of the Network-on-chip router. Different from traditional scheduling algorithms that concentrate on local fairness, we propose a congestion-aware scheduling algorithm based on input buffer of downstream router. The scheduling algorithm keeps a match dynamically between input and output by detecting the flits number to be transferred in the same packet. It can reduce network congestion especially under heavy traffic loads. Compared to RRM and iSLIP algorithm, the new scheduling algorithm can increase the saturation throughput by 8.2% and reduce the average communication latency by 7.8% under non-uniform traffic.

Keywords: Network-on-chip, router, scheduling algorithm, round-robin

Classification: Integrated circuits

References

[1] M. Tang, *et al.*: "Local congestion avoidance in Network-on-Chip," IEEE Trans. Parallel Distrib. Syst. **27** (2016) 2062 (DOI: 10.1109/TPDS.2015.2474375).

[2] P. H. Pham, *et al.*: "ProMINoC: An efficient Network-on-Chip design for flexible data permutation," IEICE Electron. Express **7** (2010) 861 (DOI: 10.1587/elex.7.861).

[3] N. Choudhary, *et al.*: "Genetic algorithm based topology generation for application specific Network-on-Chip," ISCAS (2010) 3156 (DOI: 10.1109/ISCAS.2010.5537952).

[4] Y. Zhang, *et al.*: "Model of Network-on-Chip routers and performance analysis," IEICE Electron. Express **8** (2011) 986 (DOI: 10.1587/elex.8.986).

[5] X. Hao, *et al.*: "Performance evaluation of scheduling algorithms in network on chip," ICSPCC (2011) 1 (DOI: 10.1109/ICSPCC.2011.6061769).

[6] Y. Chang, *et al.*: "A study of NoC topologies and switching arbitration mechanisms," HPCC (2012) 1643 (DOI: 10.1109/HPCC.2012.241).

[7] E. Fischer and G. P. Fettweis: "An accurate and scalable analytic model for round-robin arbitration in network-on-chip," NoCS (2013) 1 (DOI: 10.1109/NoCS.2013.6558403).

DOI: 10.1587/elex.14.20161247
Received December 18, 2016
Accepted December 20, 2016
Publicized January 20, 2017
Copyedited February 10, 2017

[8] H. Park and K. Choi: "Adaptively weighted round-robin arbitration for equality of service in a many-core network-on-chip," IET Comput. Digit. Tech. **10** (2016) 37 (DOI: 10.1049/iet-cdt.2015.0049).
[9] C. Chan, *et al.*: "A priority based output arbiter for NoC router ISCAS," ISCAS (2011) 1928 (DOI: 10.1109/ISCAS.2011.5937966).
[10] F. Guderian, *et al.*: "Fair rate packet arbitration in network-on-chip," SOCC (2011) 278 (DOI: 10.1109/SOCC.2011.6085085).

1 Introduction

In this age with rapid development of semiconductor technology, more and more transistors can be integrated on a single chip, and the System-on-Chip (SoC) has to face the addition transistors overload. Recently, Network-on-Chip (NoC) is typically employed to substitute the traditional bus-based interconnects [1, 2], which can guarantee the high bandwidth and parallel communications among the IP cores on chips [3]. In the Network-on-Chip architecture, router is considered to be key component. And input virtual-channel router is popularly adopted in NoC because of low latency and high throughput [4]. The performance of routers is extremely sensitive to scheduling algorithm. It is used to configure the crossbar switch, deciding the order in which the packets will be transmitted to and resolving contention in switch allocation. Therefore, it is a key consideration in the design of NoC router.

Traditional Round-Robin Mechanism (RRM) makes arbitration decisions independently at each port [5]. The iSLIP is popular due to higher throughput and fairness in routers for NoC based on round-robin strategy [6]. Therefore, lots of researchers have paid more attention to round-robin method combined with NoC tightly. Fischer et al. introduces an analytic service time model. It can reflect the behavior of round-robin arbiters and predict network throughput and latencies [7]. Park et al. proposes a method for equality of service, which achieves the global fairness by providing the service to each node with fewer resource requirements [8]. A priority based output arbitration method to reduce the congestion of the NoC is proposed in [9]. Two new arbitration mechanisms to obtain fair link bandwidth are introduced in [10], which can achieve almost absolute fairness of link bandwidth.

The previous work mainly considers local information on local router. However, it provides local fairness at each router and can not provide any information across neighbor router. As a result, the packets are often transmitted to the virtual channel (VC) at downstream router which has little available buffer due to local fairness, increasing the probability of congestion. Taking buffer at downstream router into account, we propose a buffer-based adaptive round-robin scheduling algorithm (BARR). The algorithm matches input to output based on buffer and dynamically keeps the match in the same VC, which prevents VC blocking and reduces congestion.

DOI: 10.1587/elex.14.20161247
Received December 18, 2016
Accepted December 20, 2016
Publicized January 20, 2017
Copyedited February 10, 2017

2 Buffer-based adaptive round-robin scheduling algorithm

The BARR is divided into two phases, including matching input to output based on buffer length and adaptively scheduling multiple flits. In order to take downstream routers buffer status into consideration, main referred metrics adopted in the scheduling algorithm will be given:

Definition 1: The free buffer length is denoted as B_{ij}, which represents free buffer length at the requested VC buffer at downstream router, where i and j are the input port and input VC.

Definition 2: The flits in the same packet buffered in current input VC at local router are defined as F_{ij}. T_{im} is shown in Eq. (1). T_{im} represents the flits number transferred to the output m continuously. Here, F_{ij} and B_{ij} mean symbols requesting output m from input port i and input VC j.

$$T_{im} = \min\{F_{ij}, B_{ij}\} \tag{1}$$

2.1 Phase1 - match based on buffer

Let the number of input ports and the number of output ports be N. And let i and m be the input and output. We can describe phase1 as follows:

Request: Each unmatched non-empty input port i send a request to its computed output port m.

Grant: If an output receives more than one request, the output grants a request closest to the grant pointer in round-robin schedule. Once a grant is accepted in the accept process, the grant pointer moves to next position after granted input. When not accepted, it will point to the queue which owns the biggest free buffer length at corresponding downstream router, i.e. B_{ij}.

Accept: Each output accepts the grant that is the closest to the accept pointer in round-robin schedule. If the grant is accepted, the input accept pointer moves to next position after the accepted output.

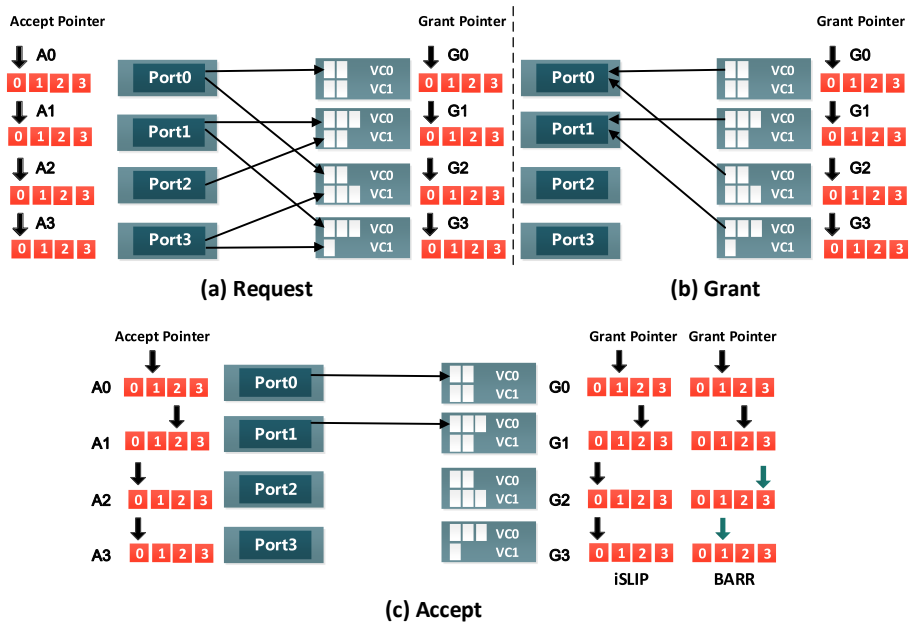

Fig. 1. An example of matching based on buffer.

DOI: 10.1587/elex.14.20161247
Received December 18, 2016
Accepted December 20, 2016
Publicized January 20, 2017
Copyedited February 10, 2017

Fig. 1 shows an example of proposed algorithm. The scheduling algorithm is based on input virtual-channel router, which only has input buffer. In order to simplify the process in Fig. 1, the buffer drawn in output means input buffer at corresponding downstream router. Each input consists of VN virtual-channels (VN equals 2 in the example).

In request phase, each unmatched non-empty input i send a request to its output m. The VC 0 in input 0 requests VC 0 at downstream router corresponding to output 0, and VC 1 in input 0 requests VC 0 at downstream router corresponding to output 2. Requests from other inputs are similar. In grant phase, a grant is given to the request that is equal or closest to the grant pointer. For output 1, it receives requests from input 1 and input 2. Compared with input 2, input 1 is closer to the grant pointer which points input 1. Therefore, the output 1 grants input 1. The difference between BARR and previous algorithms is the way of pointer updating when a grant is not accepted. In accept phase, the grant from output 2 is not accepted by input 0, so all the grant pointers are updated to the port which has the biggest B_{ij}. For output 2, B_{01} equals 2 (virtual-channel 0 in input of downstream router corresponding to output 2) and B_{30} equals 3. Therefore, the grant pointer named g2 is updated to input 3 whose corresponding input of downstream router owns bigger free buffer length. For output 3, its grant pointer updates similarly. All grant pointers are updated to the queue which has the biggest free input buffer at downstream router, which can guarantee to select a more free port and further reduce network congestion especially under heavy traffic loads without adding complexity.

2.2 Phase2 - adaptive multiple-flit scheduling

Once an input matches with an output successfully in phase1, flits belonging to the same packet will be transferred to that output port continuously, which leads to prevention of virtual-channel blocking. The remaining inputs only match with free outputs, then the scheduling process of building a new match is similar to that of phase1 for a smaller switch size. To avoid unfairness under extremely unbalanced traffic pattern, multiple-flit scheduling will set the flits number transferred to the output continuously, which equals T_{im} defined previously. When one flit is transferred to the output port m from input port i, T_{im} will be deducted by one. Once T_{im} equals zero, the process of looking for a new match is similar to that of phase1. T_{im} will be calculated again based on B_{ij} and flits in the same packet buffed in current input VCs.

The number T_{im} is dynamically changed according to free buffer length at downstream router and flits in the same packet that buffered in current input virtual-channel. The whole process of scheduling considers on buffer information and further guarantees the flexibility of scheduling process.

The details of BARR scheduling algorithm are illustrated in Fig. 2. The proposed BARR gives priority to the flit whose input buffer at the downstream node is more free in phase1. And adaptive multiple-flit scheduling in phase 2 will combine the advantages of flit-by-flit and packet-by-packet round-robin, which can release occupied VC quickly and keep agility.

DOI: 10.1587/elex.14.20161247
Received December 18, 2016
Accepted December 20, 2016
Publicized January 20, 2017
Copyedited February 10, 2017

```
INIT: RST (i, j, m), GRT (m, i, j), APT (m, i, j), B_ij, T_im, G_m, V_m, A_i
/*RST (i, j, m) is the request from vc j in input i to output m. GRT (m, i, j) is the grant from output m to vc j
in input i.APT (m, i, j) is the match from input i to output m.
The B_ij is free buffer length at requested vc, and T_im equals flits number transferred continuously. The G_m
represents Grant pointer, which points to position of granted input in Grant step, and V_m records selected vc.
Accept Pointer A_i points to the position of accepted output in Accept step. */
1:  PHASE1:
2:  If T_im = 0 then /*start to match*/
3:     If there are requests from virtual channel j in input i to output m  /*Step1*/
4:        RST (i, j, m) ← TRUE /* initial value is false */
5:     For m ← 0 to N-1 do /*Step2  N is the number of output ports*/
6:        For i ← 0 to N-1 do  /* N is the number of input ports */
7:           For j←0 to VN-1 do  /* VN is the number of virtual channels */
8:              If RST (i, j, m) =TRUE then
9:                 Calculate  B_ij based on requested input /*free buffer length*/
10:                REQ_FLAG_m ←TRUE /*initial value is false, and it will be true if there exits any request */
11:             If G_m = i  then  /*Grant pointer G_m points to current input.*/
12:                GRT (m, i, j) ← TRUE
13:                GR_FLAG_m←TRUE /*initial value is false, and it is true when request is granted*/
14:       If REQ_FLAG_m = TRUE && GR_FLAG_m !=TRUE then /*no request for output m is granted*/
15:          grant the request closest to the grant pointer. /*there is no request that equals the G_m*/
16:    For i ← 0 to N-1 do /*Step3  i is the input port */
17:       For j ← 0 to VN-1 do /* j is the input VC */
18:          For m ← 0 to N-1 do /* m is the output port */
19:             If GRT (m, i, j) = TRUE then
20:                AC_FLAG_i ← TRUE /*initial value is false, and it will be true if there exits any grant */
21:                If A_i = m then  /*Accept pointer A_i points to the requested output */
22:                   APT_FLAG_i ← TRUE /*initial value is false, and it is true when grant is accepted*/
23:                   V_m ← j, calculate t based on accepted input /*V_m  points to current virtual channe*/
24:                   APT (m, i, j) ← TRUE  /* input i matches the output m */
25:                   G_m points to next position after granted input /*update the G_m pointer*/
26:                   A_i points to next position after accepted output /*update A_i pointer*/
27:       If  AC_FLAG_i = TRUE &&  APT_FLAG_i != TRUE /*no grant for input i is accepted*/
28:          accept the grant closest to the grant pointer.  /*there is no grant that equals the A_i */
29:          update G_m and A_i to next position, record V_m
30:       For output with unaccepted grant do
31:          G_m and V_m points to the input port and vc which has the biggest free buffer length, and select
one randomly when buffer length is same  /* the biggest B_ij */
32: PHASE2:
33: For i ← 0 to N-1 do
34:    If input i get matched then
35:       If T_im = 0 then /* all flits that need to be transferred continuously have been sent, and m is the
computed output port*/
36:          Calculate T_im based on accepted input
37:          Implement phase1  /*match input to output again*/
38:       Else T_im ←T_im-1  /* the flits number transferred continuously is deducted by one */
```

Fig. 2. Pseudo code of BARR.

3 Simulation and results

To evaluate performance, we simulate the scheduling algorithms in NoC router based on 8×8 networks. The nodes that generate packets subject to Poisson distribution and virtual-channel buffer size of each input is 8 flits. The simulation is based on X-Y routing algorithm. The whole simulation time is set to be 72000 cycles, and the warm up time is 7000 cycles. We have compared BARR with iSLIP and RRM under different traffic patterns.

Firstly, the scheduling algorithms are compared under uniform traffic pattern. As Fig. 3(a)(b) shows, the three algorithms show almost identical latency and throughput. However the BARR keeps good better performance under uniform traffic pattern and performance gap starts to broaden slowly when traffic loads increase. Secondly, because network traffic is usually non-uniform, we simulate the three scheduling algorithms under hotspot traffic. We choose four nodes (coordinates:(1,2),(2,1),(1,1),(2,2)) as hotspots, and 5% of the traffic is sent to each of these four hotspots. As shown in Fig. 3(c)(d), It is obvious that BARR outperforms the other scheduling algorithms, which achieves the lowest latency and highest throughput. The results show that BARR increases the throughput by 8.2% and reduces End-To-End (ETE) delay by 7.8% compared with iSLIP. The BARR performs better under uniform and hotspot traffic. The reason is that BARR gives

DOI: 10.1587/elex.14.20161247
Received December 18, 2016
Accepted December 20, 2016
Publicized January 20, 2017
Copyedited February 10, 2017

priority to the flit whose buffer length is bigger in phase1. It can reduce network congestion, especially under heavy traffic loads. In addition, the free VC buffers are utilized more efficiently. Because the free buffer that has been reserved by head flit is not used by forwarding body or tail flit in traditional algorithm. With the BARR mechanism, this condition can be alleviated in phase2 because flits belonging to the same packet can be transferred to that output continuously.

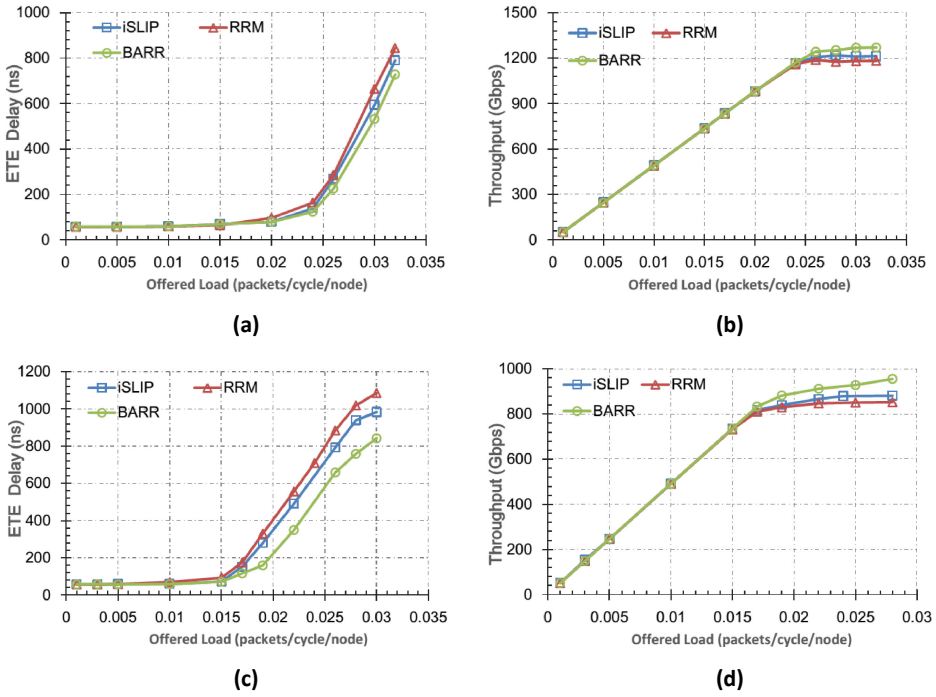

Fig. 3. (a) Delay under uniform traffic; (b) Throughput under uniform traffic; (c) Delay under hotspot traffic; (d) Throughput under hotspot traffic.

4 Conclusion

In this letter, we present a new scheduling algorithm. Taking buffer length of downstream router into account, the BARR algorithm reduces network congestion especially under heavy traffic loads. Compared with traditional scheduling algorithm, the flexibility of scheduling process is guaranteed by keeping the match dynamically. The simulation results show that BARR increases the throughput by 8.2% and reduces ETE delay by 7.8%.

Acknowledgments

This work was supported by the National Science Foundation of China Grant No. 61472300, the Fundamental Research Funds or the Central Universities Grant No. JB150318, the 111 Project Grant No. B08038, and the Research Funds of the Science and Technology on Information Transmission and Dissemination in Communication Networks Laboratory.

DOI: 10.1587/elex.14.20161247
Received December 18, 2016
Accepted December 20, 2016
Publicized January 20, 2017
Copyedited February 10, 2017