# Efficient Packet Transmission Priority Control Method for Network-on-Chip

Yusuke Sekihara, Takashi Aoki and Akira Onozawa
NTT Microsystem Integration Laboratories
3-1 Morinosato-Wakamiya, Atsugi, Kanagawa, 243-0198 Japan
e-mail : {sekihara.yuusuke,takashi.aoki,akira.onozawa}@lab.ntt.co.jp

**Abstract – To meet the ever-increasing need for high-performance computing, the performance of a single processor has been improved almost to its limit and parallelization has thus become inevitable. NoC architecture based on packet switching is becoming popular for large-scale parallelism. In this paper, we propose a new packet transmission control method in the NoC architecture that can improve the efficiency of the buffers. The simulation results prove that the proposed method can improve average latency about 10-20% when congestion occurs.**

## I. Introduction

Recent systems-on-a-chip (SoC) have to process a huge amount of data during a very short clock period. SoC performance has been improved by increasing the clock frequency. However, there is a limit to how much the clock frequency can be increased, because of gate delay, clock skew, power consumption, and so on. To solve this problem, performance improvement by parallelization is becoming inevitable. Since, large-scale parallelization is difficult to achieve with the conventional SoC architecture, the network-on-chip (NoC) architecture has been proposed as a way to realize it. NoC architecture is used for packet transmission instead of for traditional bus communication as on-chip communication. Fig.1 shows a schematic diagram of the NoC architecture. Processing elements (PEs) are arranged two-dimensionally and connected through on-chip routers that enable packet transfer among PEs.

This architecture improves scalability, which allows massive parallelism [1]. However, packet transmission has some problems, such as congestion phenomena, that reduce transmission performance and increase the latency of packets.

In this paper, we propose a new packet transmission control method that solves the problem of packet communication and improves average latency by improving the efficiency of the buffers. Section II explains packet transmission and the problem with the NoC architecture. Section III describes the simulation of congestion phenomena. Section IV describes the proposed method for improving transfer performance of the NoC. Section V explains the simulation architecture used for the experiment, and section VI describes the results for the proposed method evaluated by the simulation. This is followed by a conclusion in section VII.

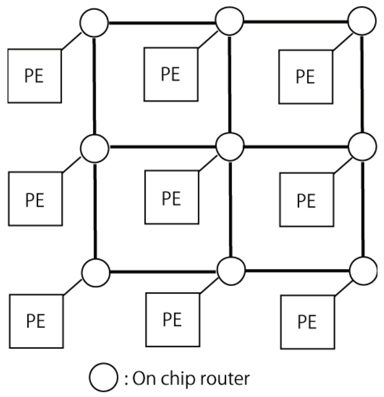


Fig. 1 NoC architecture

## II. Architecture of NoC

The NoC architecture is used for packet communication as on-chip communication. General packet transmission has three types of modes: store and forward, virtual cut through [2], and wormhole routing [3]. Store and forward and virtual cut through require a buffer longer than the length of each packet in the router, which is not very efficient for a NoC. Therefore, in general, wormhole routing is employed in the on-chip router. In wormhole routing, the packet is divided into fixed-length data, called a flit. One packet is divided into three kinds of flits: a header flit, which means the start of packet with routing information; a payload flit, which is a body of data; and a tail flit, which indicates the end of packet. The flits are transmitted across multiple router buffers as shown in lower half of Fig. 2. Therefore, when transmission is blocked, transmission performance deteriorates significantly. In this paper, transmission blockage is an event in which a router cannot transmit flits to the next router for some reasons. We explain below how this blockage occurs.

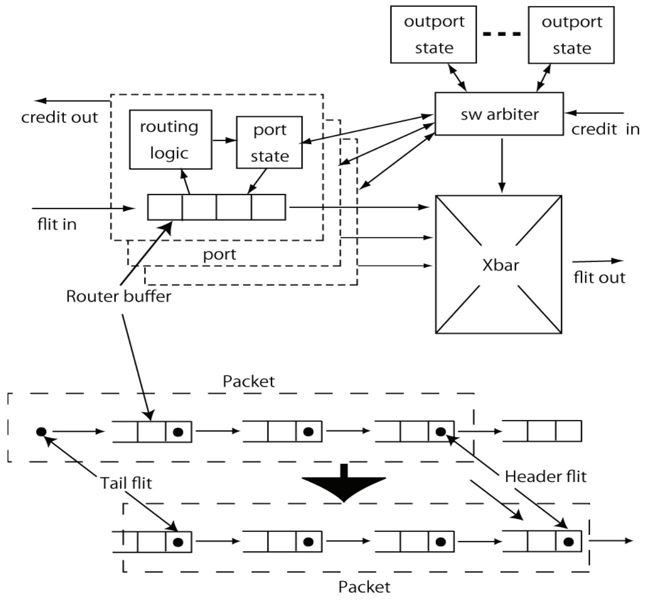
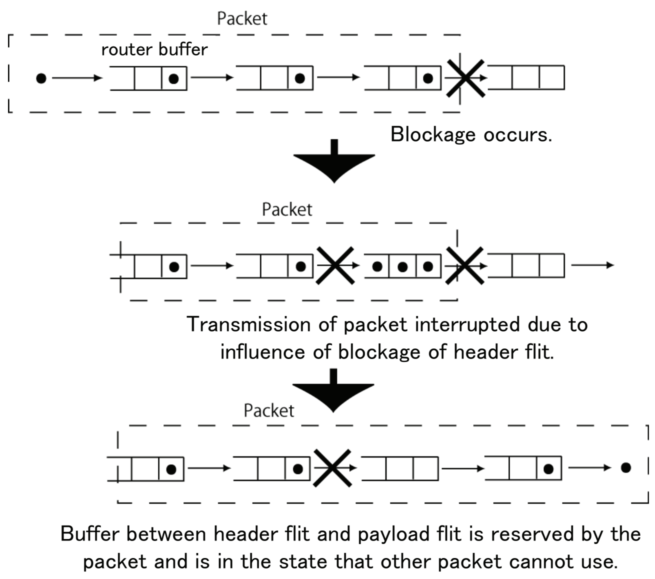


Fig. 2 Router schematic diagram and wormhole routing


Packet
router buffer
Blockage occurs.
Packet
Transmission of packet interrupted due to influence of blockage of header flit.
Packet
Buffer between header flit and payload flit is reserved by the packet and is in the state that other packet cannot use.
Fig. 3 Transmission blockage and blank buffer

The upper half of Fig. 2 shows a schematic diagram of a typical wormhole router [4]. When a header flit reaches a router buffer, the packet to which the header belongs reserves the input controller and buffer; that is, the input port is locked so that the other packets cannot use the buffer. The transmitting direction of flits is determined by routing logic based on the transmission information the header flit has. After transmission has started, the router's crossbar is occupied until packet transmission finishes or destination buffer overflow occurs. Generally, the arbiter schedules permission for the use of a port in a crossbar with the round-robin method. Round-robin is a scheduling method for assigning an equal-length time slice to each of the input ports. The flit in the buffer must wait until it is given the time slice (permission) from the arbiter, which causes a jam in the subsequent flits. Furthermore, since header flit transmission in wormhole routing doesn't wait for subsequent flits to arrive at the router buffer, that buffer is often left blank for a while. A blank buffer is defined as an event in which the router's buffer is empty, but the buffer is reserved by that packet and cannot accept other packets. This blank buffer reduces the transmission efficiency of the router greatly, as shown in Fig. 3.

As mentioned above, the round-robin scheduling method gives equal opportunities to all input ports of a router. However, with this method, time is taken to wait for the time slice from the arbiter and to fill the blank buffer. The resultant blockage, or the lower utilization of the router buffer, causes deterioration of transmission performance.

## III. EVALUATION OF CONVENTIONAL METHOD

Prior to the evaluation of the proposed method, we evaluated the existing scheduling method, i.e., round-robin scheduling, by MATLAB simulation. We constructed a 4 x 4 2D-mesh model in MATLAB Simulink and applied a set of input patterns of uniform traffic, which was created randomly from each core and transferred in an XY routing scheme. Note that this type of NoC architecture, also shown in Fig.1 for a 3x3 arrangement, is used widely [5]. It is assumed that each router has a FIFO buffer of three flits per port. The packets that arrive at each core are kept for ten cycles of data processing and then discarded.

### *A. Delay models in the simulation*

We assumed three types of delay in this simulation: delay due to the transfer in the crossbar (transfer delay), wake-up delay due to power gating (wake-up delay), and delay due to packet blockage caused by congestion (blockage delay). The former two are considered to be constant. Transfer delay is caused by the selection of the destination and arbitration. Wake-up delay is the time spent when the corresponding core wakes up from the power-off state and is due to the power gating operation. Since the power-gating technique is widely used in commercially available multi-core large-scale integration, we think it is necessary to employ the idea in our simulation. The wake-up delay was set to six cycles; that is, it takes six cycles for a core to wake up when a header flit arrives. Table I shows the main parameters of the simulation model. These parameters and architectures are widely used as a general model for NoC [5].

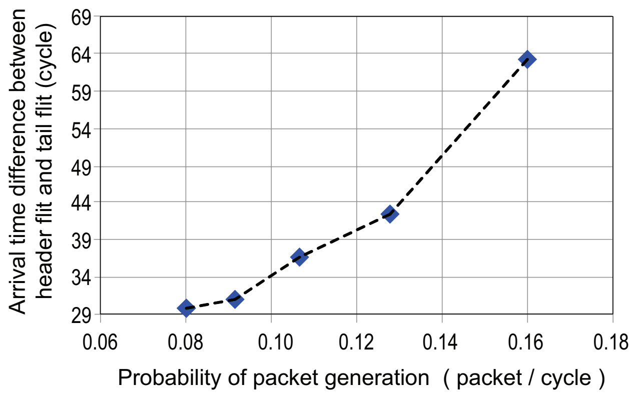


Fig. 4 Evaluation of round-robin scheduling

### B. *Evaluation of round-robin scheduling*

We evaluated the wormhole routing method with the round-robin scheduling using the simulation model above. We thought that the interval between header flit arrival time and its corresponding tail flit arrival time at the destination would become larger when blank buffers are increased due to the blockage. To evaluate the efficiency of this scheduling method, we measured this time interval as the arrival time difference $T^D$. This means the average packet duration in the cycle count at the destination node, which is defined as follows:

$$T^D = \left\langle t_d^T - t_d^H \right\rangle,$$

where

$t_d^H$ : Arrival time of the header flit at its destination,

$t_d^T$ : Arrival time of the tail flit at its destination.

The results are shown in Fig. 4, where the horizontal axis is the packet generation rate at each core and the vertical axis is the average interval between the arrival times of the header flit and its corresponding tail flit. When the probability of packet generation exceeds 0.16 (right most part of the figure), it is hard to measure the interval increase in the arrival time due to simulation cost. From this experiment, we can state that a packet occupies many blank buffers between flits due to traffic increase, and we believe that this is one of major reasons that the packet latency increase and the transmission performance degrades.

TABLE I
Experimental NoC Architecture

| | |
|---|---|
| Topology | 2D-mesh |
| Size | 4 × 4 |
| Router buffer | Three flits |
| Routing algorithm | X-Y routing |
| Transfer delay | One cycle per flit |
| Wake-up delay | Six cycles per flit |

In the next section, we describe our new scheduling method that can decrease the average latency and thus enhance the performance of a NoC.

## IV. Proposed Method

Our priority control method suppresses the generation of blank buffers to improve transmission performance. Previous works [6] and [7] have proposed examples of efficient uses of router buffers. However, those methods require adding additional buffers, which make the router hardware larger. Our method achieves efficiency without additional buffers by modifying the scheduling algorithm as follows:

*Algorithm 1 : Priority Control Scheduling*

```
while(1){
    if( current port p transmission has completed ) {
        // reset priority flag
        list_f [p]=0;
    }
    if( current port p is blocked ) {
        // set priority flag
        list_f [p]=1;
    }
    if(( current port p transmission has completed )||
            ( current port p is blocked ) {
      // round-robin for higher priority ports
      round-robin ( for all f in list_f [f]= =1 ) {
          if ( destination_buffer_of( f ) is not full ){
              give permission to port f ;
              p = f;
              break;}
      }
      // round-robin for lower priority (all) ports
      round-robin ( for all g ) {
          give permission to port g ;
          p = g;
          break;
      }
    }
    // else keeps permission to current port
}
```

If more than two flags are set to multiple ports, round-robin scheduling is applied to select one of them. This algorithm can be executed by the conventional round-robin scheduler, and the additional small control logic can be implemented as several bits of registers and a negligible number of gates compared to the size of the entire NoC for priority flags and decision circuitry. Fig. 5 shows an example of time slot allocation on the crossbar with the proposed method.

In this example, the arbiter gives permission to use the crossbar to port C at t1 with the round-robin scheduling method. Packet transmission to port C is blocked by buffer overflow of the destination buffer and blocked at t2. Then, the arbiter searches for what is

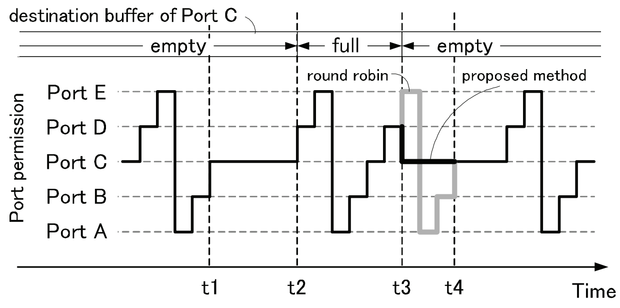


Fig. 5 Example of time scheduling

to be transmitted to the next router. Suppose that the destination buffer overflow is resolved at t3. In the round-robin scheduling method, port C obtains permission to use the crossbar at t4 after the arbiter finishes checking all other ports. That is, port C postpones the transmission till t4 even if it is possible at t3, which may generate blank buffers between t3 and t4. On the other hand, the priority control method gives preferential permission to use the crossbar to port C at t3. In this way, it can prevent generating and expanding a blank buffer and as a result can improve NoC transmission performance.

## V. Experiment

We evaluated the proposed priority control method by simulation. We made a simulation model using MATLAB Simulink with the same set of parameter values as the ones for the round-robin scheduling simulation model. The transmission performance of the proposed method is compared with that of the round-robin scheduling method.

We used two packet traffic patterns in this experiment. One was a uniform traffic pattern, which means that packets are generated under uniform distribution from all nodes. This packet traffic pattern condition is referred as

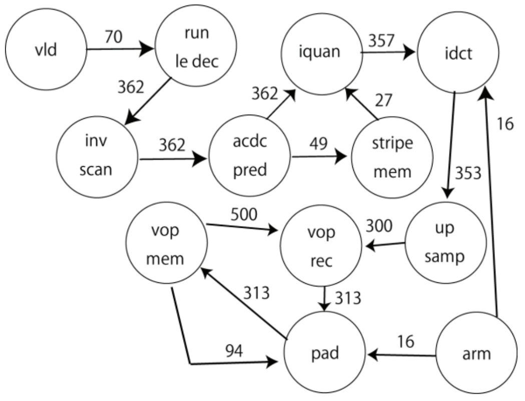


Fig.6 VOPD traffic task graph

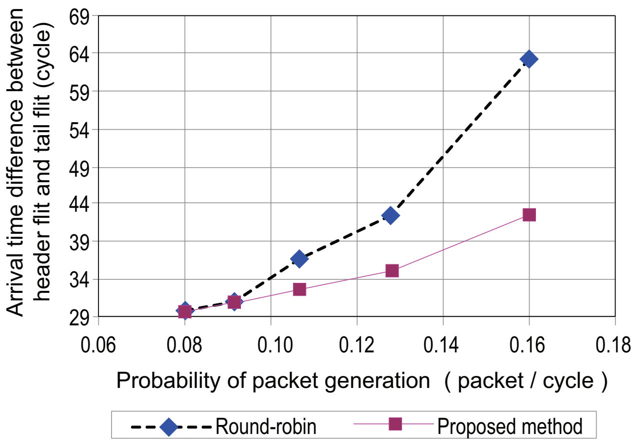


Fig. 7 Arrival time comparison between proposed method and conventional method

"UD". The destination of each packet is determined at random in this case. The other is a Video Object Plane Decoder (VOPD) [8] application traffic pattern. Fig. 6 shows the task graph of the VOPD application, where the nodes represent the tasks and the arcs represent the communication between tasks and the relative bandwidth requirements. We have allocated each task and bandwidth value on 2D mesh topology using the general minimum-path mapping algorithm [8]. The VOPD traffic pattern is generated under the condition where the packet rates between the source nodes and destination nodes is proportional to the value of bandwidth.

## VI. Simulation Results and Discussion

Fig. 7 shows the average arrival time difference between header and tail flits with the UD traffic pattern for the round-robin method and the priority control method.

The results show that our priority control method reduces $T^D$, the average packet duration at the destination node. $T^D$ becomes larger as the probability of packet generation increases, but the priority control scheduling method can suppress the increase compared to the round-robin method. This result suggests that the number of blank buffers between source and destination nodes is reduced by the priority control method. The reduction of the blank buffers shortens packet latency. Average packet latency is defined as

$$T^L = \left\langle t_d^T - t_s^H \right\rangle,$$

where

$t_d^T$ : Arrival time of the tail flit at its destination.

$t_s^H$ : Departure time of the header flit from source node.

Fig. 8 shows the average packet latency $T^L$ for the priority control method and conventional round-robin scheduling method with the UD traffic pattern. The error

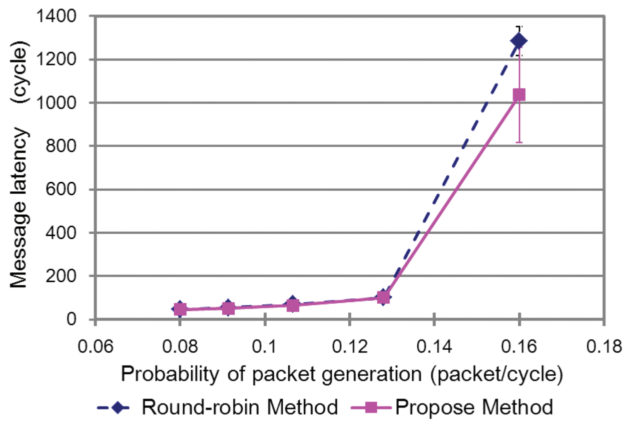


Fig. 8 Latency comparison under UD packet condition

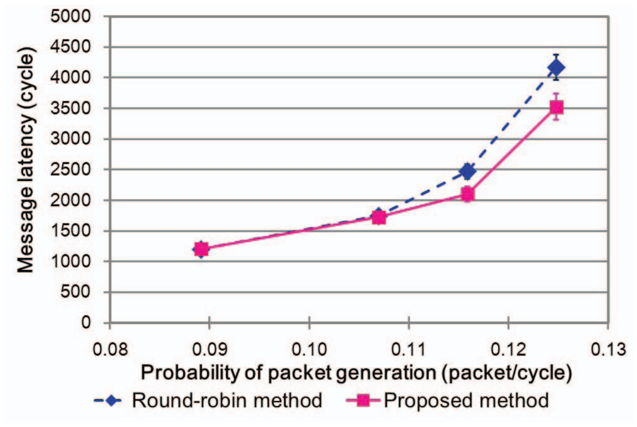


Fig. 9 Latency comparison under VOPD packet condition

bars in the graph indicate the standard deviation for ten measurements.

The results show that, as the probability of packet generation increases, our method can improve the average latency compared to the round-robin method about 20% when the generation rate is relatively high (>0.13), while the average latency for the two methods is comparable when the rate is relatively low (<0.13). However, our method has a larger distribution in latency. In our method, some packets are preferentially transferred when blockage occurs, which could increase the latency distribution. Both methods were simulated under the same UD condition. In this condition, the destination of each packet and the interval of packet generation are determined at random uniformly, which is not very realistic. In order to evaluate the latency and its distribution with the proposed method in a more realistic condition, we simulated our method using the VOPD traffic pattern.

Fig. 9 shows the $T^L$ in the proposed priority control method and conventional round-robin scheduling method with the VOPD traffic pattern. These results show that our method improves the average packet latency about 16% when the generation rate is higher (>0.12) and that the distribution of average latency with our method is almost equal to that of the round-robin method. This could mean that the sensitivity of the distribution of average latency is suppressed in this more realistic application.

From these experimental results, we can state that our proposed method has potential to reduce the average transmission latency by around 16-20% by resolving the congestion in packet traffic. It is also suggested that it increases the distribution of latency compared to the conventional round-robin method in the case of UD traffic, but it may be able to suppress it in realistic applications.

## VII. Conclusion

We proposed an efficient priority control port scheduling method for wormhole routing that can improve the transmission performance of a NoC. Evaluations of the method by simulation show that the average latency can be improved about 16-20% when congestion occurs. In a rather unrealistic case (UD), the distribution of latency was larger than that of for the conventional round-robin method. Constructing a scheduling method that can also reduce the distribution of latency in most applications remains as future work.

## References

[1] Arteris. “A comparison of Network-on-Chip and Buses” www.arteris.com, 2005.

[2] P. Kermani and L. Kleinrock. “Virutal cut-through: A new computer communication switching techniques.” *Computer Network*, vol. 3, No. 4, pp. 267-286, 1979.

[3] L. M. Ni and P. K. McKinley. “A survey of wormhole routing techniques in direct network.” *Computer, IEEE Computer Society,* Vol. 26, No. 2, pp. 62-76, 1993.

[4] L. S. Peh and W. J. Dally. “A delay model for router microarchitectures.” *IEEE Micro,* Vol. 21, No.1, PP. 26-34, 2001.

[5] T. BJerregaad and S. Mahadevan. “A survey of research and practices of Network-on-Chip.” *ACM Computing Surveys,* Vol. 38, pp. 1-51, 2004.

[6] (in Japanese) Y. Miura and T. Abe and S. Horiguchi. “Virtual-channel flow control of wormhole routing.” *The 57th National Convention of IPSJ,* Vol. 1, pp. 48-49, 1998.

[7] S. T. Wu and C. H. Chao. “Dynamic channel flow control of Network-on-Chip systems for high buffer efficiency.” *Signal Processing System, IEEE workshop on*, pp.493-498, 2007.

[8] D. Bertozzi et al. “NoC synthesis flow for customized domain specific multiprocessor systems-on-chip” IEEE Transactions on Parallel and Distributed systems, Vol. 16, No. 2, pp. 113-129, 2005.