Flit Admission in On-chip Wormhole-switched Networks with Virtual Channels

Zhonghai Lu and Axel Jantsch
Laboratory of Electronics and Computer Systems
Royal Institute of Technology, Sweden
{zhonghai,axel}@imit.kth.se

Abstract

Flit-admission solutions for wormhole switches must minimize the complexity of the switches in order to achieve cheap implementations. We propose to couple flit-admission buffers with physical channels so that flits from a flit-admission buffer are dedicated to a physical channel. By the coupling strategy, for input-queuing wormhole lane switches, the complexity of the crossbars can be simplified from $2p \times p$ to $(p+1) \times p$, where p is the number of physical channels; for output-queuing wormhole lane switches, the additional complexity is also minimal. We evaluate the flit-admission solutions derived from the coupling with uniformly distributed random traffic in a 2D mesh network. Experimental results show that these solutions exhibit good performance in terms of latency and throughput.

1 Introduction

Wormhole switching is being proposed for Networks on Chips (NoCs) due to its better performance and smaller buffering requirement [1, 2]. To make efficient use of the Physical Channels (PCs), wormhole switching uses virtual channels (lanes) to gain higher throughput [3]. Several parallel lanes, each of which is a flit buffer queue, share a PC. For on chip wormhole switches, these lane buffers can be customized as dedicated hardware FIFOs instead of register-based or RAM-based FIFOs to reduce the area and thus achieve reasonable buffering cost [2]. To reduce the control complexity of the switches, deterministic routing is favored against adaptive routing. This may also be justified by exploiting the traffic predictability of specific applications [1]. Moreover, regular low-dimension topologies are considered for NoCs to further simplify the control [4, 5].

Figure 1 shows a 2D mesh NoC architecture [1, 4, 5]. Each resource is connected to a switch via a Network Interface (NI). The wormhole switch with bidirectional links has four PCs and several lanes per PC (not shown). Resources feed the network with packets, which are queued in the

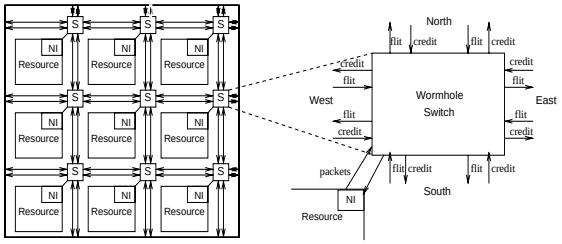

Figure 1. A 2D mesh

packet buffers of the switches. With wormhole switching, a packet is decomposed into a head flit, zero or more middle flit(s), and a tail flit. A single-flit packet is also possible. These flits are stored in flit-uploading buffers called *uploading/admission buffers* before being admitted to the network. There are various ways of organizing the packet buffer and the uploading buffers. In Figure 2.(a), flit-uploading buffers are organized as a FIFO. In Figure 2.(b) and 2.(c), they are arranged as p parallel FIFO queues (p is the number of PCs). Figure 2.(a) and 2.(b) allow at maximum one flit to be admitted to the network at a time while Figure 2.(c) allows up to p flits to be admitted simultaneously.

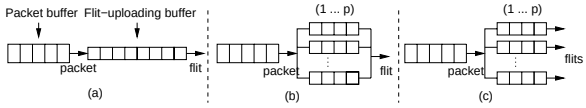

Figure 2. Packet and flit-uploading buffers

In the paper we investigate *flit admission* approaches in wormhole lane switches. We shall see that different solutions largely impact the complexity of the switches. Throughout the paper, we adopt the organization of flit-uploading buffers in Figure 2.(c) since it allows potentially higher performance. In the sequel, Section 2 discusses related work. In Section 3 we present our flit admission solutions for both input-queuing and output-queuing wormhole switches. Section 4 describes experimental results. Finally, we conclude the paper in Section 5.

2 Related Work

Wormhole switching with virtual channels was proposed in [3]. The performance model of a wormhole switch that considers implementation complexity was first noted by Chien [6]. Recently a more efficient canonical wormhole switch architecture for virtual-channel flow control and its performance model was presented in [7].

Admission control is commonly employed for real-time traffic to determine if admitting new real-time traffic can satisfy its timing bounds without jeopardizing the performance guarantees of real-time traffic already in the network. It has been a rich research area in packet-switched computer networks. In cluster computing, based on QoS-capable wormhole switches and network interfaces, an admission control algorithm in conjunction with a congestion control algorithm was designed for the admission of real-time traffic in the networks [8].

Our study on flit admission is different from the admission control for real-time traffic. By effectively sharing physical channels, our flit admission approaches are designed to minimize the complexity of wormhole switches without sacrificing performance.

3 Flit Admission Approaches

3.1 Flit admission in input-queuing switches

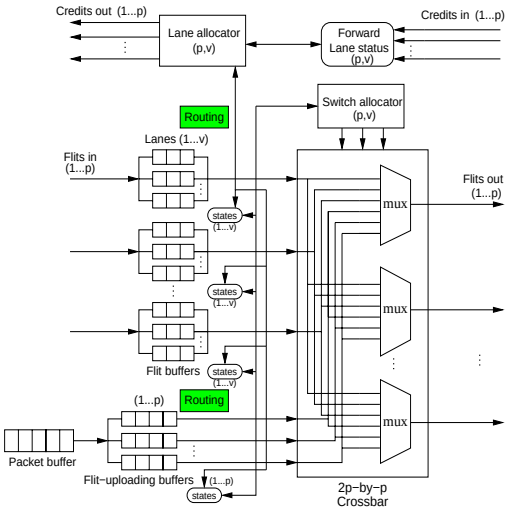

Figure 3. A canonical wormhole lane switch

Figure 3 shows a canonical wormhole switch architecture with virtual channels at inputs [2, 3, 7]. It has p physical channels (PCs) and v lanes per PC. A packet passes the switch through four states: *routing*, *lane allocation*, *flit scheduling*, and *switch arbitration*. In the routing state, the routing logic determines the routing path a packet advances. In the state of lane allocation, the lane allocator *associates* the lane the packet occupies with an available lane on its routing path in the next hop. If the lane allocation succeeds, the packet enters into the scheduling state. If there is a buffer available in the associated lane, the lane enters into the switch arbitration. The first level of arbitration is performed on the lanes sharing the same physical channel. The second level of arbitration is for the crossbar traversal. If the lane wins the two levels of arbitration, the flit situated at the head of the lane is switched out. Otherwise, the lane returns back to the scheduling state. The lane association is released after the tail flit is switched out. Credits are passed between adjacent switches in order to keep track of the status of lanes. Note that a lane is allocated at the packet level, i.e., packet-by-packet. Also, flits from different lanes can not be interleaved in a lane since flits other than head flits do not contain routing information. To guarantee this, a lane-to-lane association must be unique at a time.

In Figure 3, if an uploading buffer is available, a packet is split into flits which are then put into the uploading buffer. An uploading buffer takes the same four states as a lane in order to inject flits into the network. Flits from an uploading buffer can be switched to all the p output PCs. Since the uploading buffers are decoupled from the PCs, the crossbar must be fully connected, resulting in a port size of $2p \times p$.

To alleviate the complexity of the switch, we propose to couple an uploading buffer with a PC in a *one-to-one* manner. In this way, flits from an uploading buffer are dedicated to a PC. Applying the coupling scheme to the switch in Figure 3, an uploading buffer only needs to be connected to one multiplexor instead of p multiplexors. The size of the crossbar is sharply decreased from $2p \times p$ to $(p+1) \times p$, as shown in Figure 4. The number of control signals per multiplexor is reduced from $\lceil log(2p) \rceil$ to $\lceil log(p+1) \rceil$ for any $p > 1$ [1]. Alternatively, an uploading buffer can directly

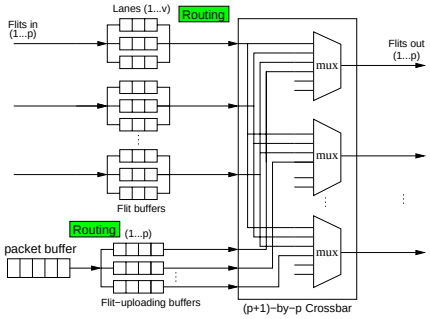

Figure 4. Sharing a (p+1)-by-p crossbar

share an output PC, as depicted in Figure 5.(a). This solu-

[1] $\lceil x \rceil$ is the ceiling function which returns the least integer that is not less than x.

tion can also be regarded as having a crossbar complexity of $(p+1) \times p$ since the combination of a $p \times 1$ multiplexor and a 2×1 multiplexor may be viewed as a $(p+1) \times 1$ multiplexor. The number of control signals per PC is reduced from $\lceil log(2p) \rceil$ to $\lceil logp \rceil + 1$ for any $p > 1$.

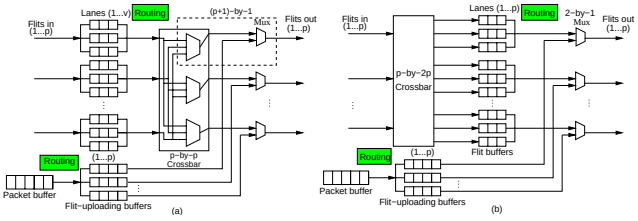

Figure 5. Sharing output physical channels

In order to support the coupling scheme, the routing must be performed before segmenting a packet and storing its flits in an uploading buffer instead. With a routing algorithm, the PC the packet requests can be determined. Hence, the corresponding uploading buffer is identified. One drawback due to the coupling is possible blocking propagation. Specifically, if the head packet in the packet buffer is blocked due to the bounded size of the uploading buffer it aims at, the packets behind the head packet are all unconditionally blocked during the head packet's blocking time.

3.2 Flit admission in output-queuing switches

In addition to sharing the crossbar or output PCs, flit admission may share input PCs of the switch. However, there is a *critical section* problem. Figure 6 illustrates the problem with a simplified graph of two connected wormhole switches, **A** and **B**. Suppose that lane j in switch **B** is available at a certain clock cycle, the uploading buffer sees lane j available and then associates itself to lane j locally. At the same cycle, lane i in switch **A** also detects lane j available and remotely makes an association with lane j. This is possible since both switches maintain a consistent view of the lane status. As a result, two associations with a single lane are established in the same cycle. Consequently, lane j will receive flits from lane i and the uploading buffer, resulting in their flits possibly interleaved in lane j.

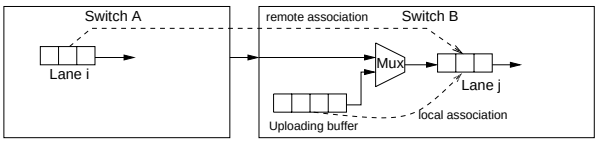

Figure 6. Problem with sharing input PCs

To avoid such a situation and thus achieve a mutual-excluded lane association needs both architectural support and a control protocol. This complicates the switch design and negatively impacts the network performance. Therefore, sharing input PCs is *not* favored as a flit admission solution. This observation illuminates flit admission in output-queuing wormhole switches. If sharing input PCs and crossbars in output-queuing wormhole switches, we encounter exactly the critical section problem, which is costly to resolve. This leads to only one low cost flit-admission path for output-queuing wormhole switches, i.e., sharing output PCs, as drawn in Figure 5.(b), where the multiplexors for admitting flits have a port size of 2×1. If the coupling strategy is not used, the multiplexors must be $(p+1) \times 1$.

4 Experiments

We developed a simulator in SystemC comprising the input-queuing wormhole switch model and other supporting objects. The switch is a single-cycle, flit-level model. We construct a 2D $K \times K$ (K=4) network without end-around connections (Figure 1). The network does dimension-order **X**-**Y** routing, which is deadlock-free and deterministic. The aim of our experiments is two-fold. First, we examine the performance of the two flit-admission solutions derived from the coupling scheme, i.e., sharing simplified crossbars (Figure 4) and sharing output PCs (Figure 5.(a)). The baseline architecture is the one admitting flits via full crossbars (Figure 3). Second, with admitting flits via output PCs, we investigate the impact of multiplexor arbitration.

The simulations were run with uniformly distributed traffic. Resources injected fixed-size packets to random destinations except for themselves at a constant rate. A flit is ejected from the network once it reaches a lane of its destination and the lane passes the routing state. Except otherwise noted, contentions for lanes and channel bandwidth were resolved randomly. Each simulation was run until the network reached steady state, i.e., increasing simulated network cycles did not change the results appreciably. We investigated the average latency of packets and the network throughput. Latency of a packet is calculated from the instant the packet's flits are created to that the last flit of the packet is accepted at the destination, including source queuing time. Throughput λ is defined as the number of flits received per cycle per node.

Number v of lanes per physical channel	3
Size of a lane	2 flits
Size of an uploading buffer	4 flits
One packet	4 flits

Table 1. Simulation parameters

Simulation parameters are listed in Table 1. The size of a lane was chosen to be two, which is the minimal amount of buffer requirement for a lane in order to pipeline flits since sending and receiving credits take two cycles.

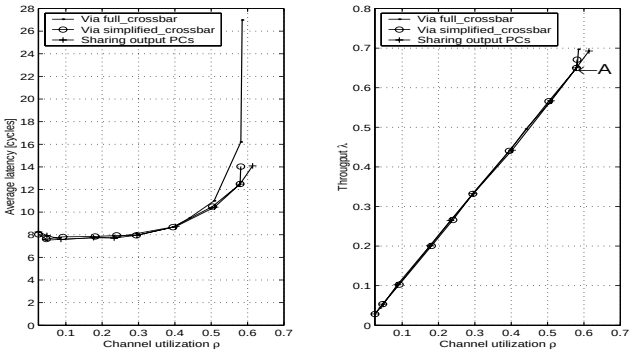

Figure 7. Performance comparison

Figure 7 compares the flit-admission approaches. Admitting flits via simplified crossbars and via output PCs achieve similar performance since they are equivalent for the input-queuing switch. Compared with admitting flits via full crossbars, the three solutions agree well when channel utilization ρ is below 0.5. When ρ is higher than 0.5, the average latency with admitting flits via full crossbars is worse. This suggests that faster uploading of flits results in higher congestion thus higher latency when the network is nearly saturated. It is interesting to note that the three approaches achieve the same channel utilization and throughput (point **A** in the Utilization-Throughput figure) given the same traffic patterns. Above this point, the packet buffers (refer to Figure 3) start to overflow, given a bounded packet buffer size. The Utilization-Throughput figure can serve as a validation of the network operations. The slope of the three lines is $\frac{9}{8}$, because $\lambda = C\rho/(MD_{avg})$, where C, M, D_{avg} are the network capacity, the number of nodes, the average distance traveled by all received flits, respectively. For the 2D mesh, $C = 4K(K-1)$, $M = K^2$, $D_{avg} = \frac{2}{3}K$ for the random traffic. When $K = 4$, $\lambda = \frac{9}{8}\rho$.

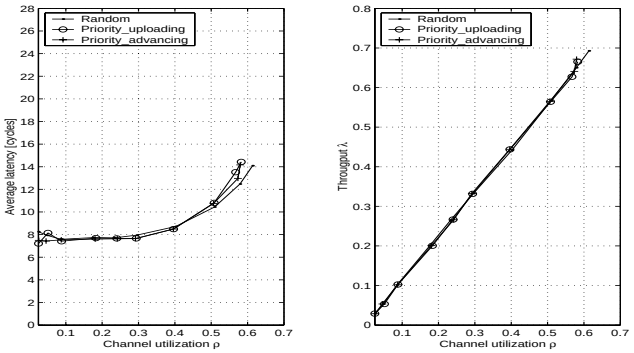

Figure 8. Arbitration impact

Figure 8 shows the performance of admitting flits via output PCs with three different arbitration criteria: random, priority_uploading meaning that flits to be admitted win contentions against network flits, priority_advancing meaning that network flits win contentions against flits to be uploaded. We can see that the three arbitration policies do not make significant difference. This is because, if the arbitration gives priority to uploading flits, uploading flits is faster, but the uploaded flits (network flits) lose arbitration along their routes; if the arbitration favors network flits, they win arbitration along their routes but are difficult to be admitted in the beginning. This sort of balance makes both cases close to the effect of the random arbitration.

5 Conclusions

We have discussed flit admission approaches for wormhole virtual-channel switches. By coupling flit-admission buffers with physical channels, the complexity of the crossbar can be reduced from $2p \times p$ to $(p+1) \times p$ for an input-queuing switch; the additional complexity for admitting flits is also minimal for an output-queuing switch. Simulation results show that these solutions derived from the coupling scheme do not compromise the performance. Although our discussions are equally applicable to macro wormhole-switched networks in parallel computing, the experiments were designed for a NoC that employs a low-dimension topology, deterministic routing, and smaller buffering cost.

Future work will consider flit admission together with packet admission. A higher-level admission control strategy can be devised to track network load so that packets are admitted with reasonable rates. Another direction is to combine flit admission with flit ejection. Practically cost-effective flit ejection models must be taken into account while evaluating the performance of the on-chip network.

References

[1] J. Hu and R. Marculescu. Exploiting the routing flexibility for energy/performance aware mapping of regular NoC architectures. In *DATE*, 2003.

[2] E. Rijpkema, K. Goossens, et al. Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip. In *DATE*, 2003.

[3] W. J. Dally. Virtual-channel flow control. *IEEE Transactions on Parallel and Distributed Systems*, 3(2):194–204, March 1992.

[4] W. J. Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. In *DAC*, 2001.

[5] Mikael Millberg, Erland Nilsson, Rikard Thid, and Axel Jantsch. Guaranteed bandwidth using looped containers in temporally disjoint networks within the Nostrum network on chip. In *DATE*, 2004.

[6] A. A. Chien. A cost and speed model for k-ary n-cube wormhole routers. *IEEE Transactions on Parallel and Distributed Systems*, 9(2):150–162, Feb. 1998.

[7] L. S. Peh and W. J. Dally. A delay model for router microarchitectures. *IEEE Micro*, pages 26–34, Jan.-Feb. 2001.

[8] K. H. Yum et al. Integrated admission and congestion control for QoS support in clusters. In *Proceedings of IEEE International Conference on Cluster Computing*, pages 325–332, Sept. 2002.