Flit Ejection 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

An ideal flit-ejection model is typically assumed in the literature for wormhole switches with virtual channels. With such a model, flits are ejected from the network immediately upon reaching their destinations. This achieves optimal performance but is very costly. The required number of sink queues of a switch for absorbing flits is $p \cdot v$, where p is the number of physical channels (PCs) of the switch; v the number of lanes per PC. To achieve cheap silicon implementations, flit-ejection solutions must be cost-effective. We present a novel flit-ejection model and a variant of it where the required number of sink queues of a switch is p, i.e., independent of v. We evaluate the flit-ejection models with uniformly distributed random traffic in a 2D mesh network. Experimental results show that they exhibit good performance in latency and throughput.

1 Introduction

Chip design is increasingly becoming communication-bound other than computation-bound with the steady technology scaling [1]. Network-on-Chip (NoC) [5, 6, 8, 11] addresses the design challenge by proposing networks to replace buses as a scalable global communication platform. In a NoC, heterogeneous resources such as processor cores, DSPs, FPGAs/ASICs, and memories are interconnected by switches, which route packets to enable communication between resources.

Network flow control governs how a packet is forwarded in a network, concerning shared resource allocation and contention handling. Wormhole switching [4] is a network flow control scheme that allocates buffers and physical channels (PCs) to flits instead of packets. A packet is decomposed into one or more flits. A flit, the smallest unit on which flow control is performed, can advance once buffering in the next hop is available to hold the flit. This results in that the flits of a packet are delivered in a pipeline fashion. For the same amount of storage, it achieves lower latency and greater throughput. However, wormhole switching uses channels inefficiently because a PC is held for the duration of a packet. If a packet is blocked, all PCs held by this packet are left idle. To mitigate this problem, wormhole switching adopts *virtual channels* (*lanes*) to make efficient use of the PCs [3]. Several parallel lanes, each of which is a flit buffer queue, share a PC. Therefore, if a packet is blocked, other packets can still traverse the PC via other lanes, leading to higher throughput. Because of these advantages, namely, better performance, smaller buffering requirement and greater throughput, wormhole switching with lanes is being advocated for on-chip networks [6, 11].

The ejection of flits in a wormhole-switched network concerns when and how the flits reaching destinations are ejected from the network and stored in flit sink queues (sinks) before being composed back into packets. An ideal ejection model has been assumed for wormhole lane switches. With such an ideal model, flits are ejected into sink queues instantly once they reach destinations (after routing). Also, ejecting flits does not interfere with advancing flits. Such a model is optimal for performance, but it is too costly to be suitable for silicon implementations. Given the number of PCs of a switch is p, the number of lanes per PC is v, the required number of sink queues of the switch is proportional to p and v in order to realize the ideal model.

In this paper, we present a novel *flit-ejection* or *sink* model and a variant of it for wormhole lane switches. Our models sharply reduce the required number of sink queues from $p \cdot v$ to p without compromising much performance. In the sequel, Section 2 discusses related work. In Section 3 we describe a canonical wormhole lane switch and the ideal flit-ejection model. We present our flit-ejection models in Section 3, followed by experimental results in Section 4. Finally, we conclude the paper in Section 5.

2 Related Work

The performance model of a wormhole switch that considers implementation complexity was first noted by Chien [2]. A more efficient canonical wormhole lane switch archi-

tecture and its performance model was presented in [10]. In general, the design complexity of a wormhole lane switch is the function of p and v. To gain further performance, flit-reservation flow control [9] was proposed which utilizes control flits to reserve bandwidth and buffers before transferring data flits. All of these works assume an ideal flit-ejection model while evaluating the network performance.

To our knowledge, no prior work discussing flit-ejection models other than an ideal model was reported. Our motivation is to reduce the switch complexity to achieve cost-effective designs on silicon. In line with this idea, Goossens et al. proposed to customize the lane buffers as dedicated hardware FIFOs instead of register-based or RAM-based FIFOs to reduce the area and thus achieve reasonable buffering cost [11]. Recently, cost-effective flit admission approaches for virtual-channel wormhole switches are discussed in [7]. 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 [6], which NoCs target. Moreover, regular low-dimension topologies are considered for NoCs to further simplify the control [5, 8].

3 The Ideal Sink Model

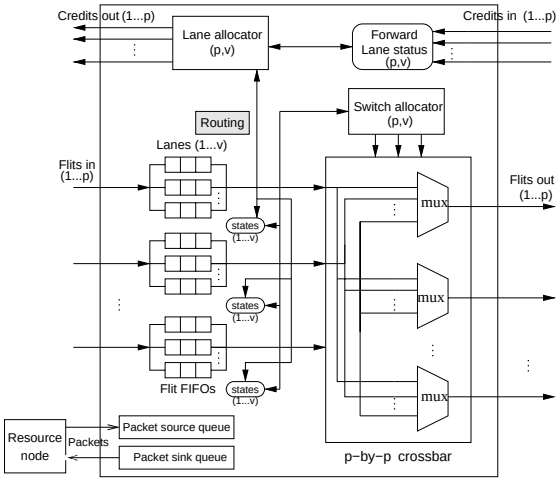

Figure 1. A canonical wormhole lane switch

Figure 1 shows a canonical wormhole switch architecture with virtual channels at inputs [3, 10, 11], connecting to a resource node. 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 PC. 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 while the PC bandwidth is assigned at the flit level, i.e. flit-by-flit. In addition, flits from different lanes can not be interleaved in a lane since flits other than head flits do not contain routing and sequencing information. To guarantee this, a lane-to-lane association must be unique at a time.

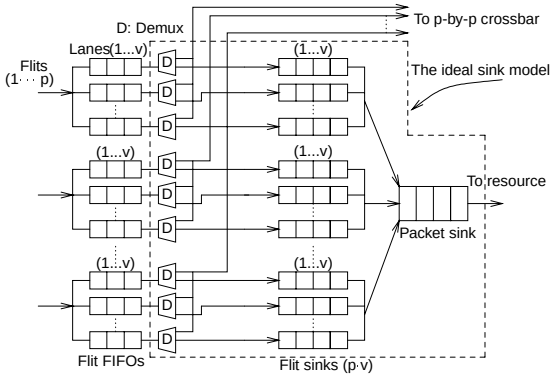

Figure 2. The ideal sink model

In a traditional wormhole switch design, an ideal sink model is assumed, as shown in Figure 2. The lane state is extended with a *reception* state. If the routing determines that the head flit of a packet reaches its destination, the lane enters the reception state immediately. Since flits from different packets can not interleave in a sink queue, there must be $p \cdot v$ sink queues, each of them corresponding to a lane, in order to realize an immediate transition to the reception state. The length of a sink is the maximum number of flits of a packet. After the lane transitting to the reception state, the head flit bypasses the crossbar and enters into its sink. The subsequent flits of the packet are ejected into the sink immediately upon arriving at the switch. When the tail flit is ejected, the lane is freed. This model is beneficial in both time and space. A non-head flit reaching its destination neither waits to be ejected nor occupies a flit buffer. Moreover, it does not interfere with flits buffered in other lanes from advancing to next hops. Upon receiving all the flits of a packet, the packet is composed and delivered into the packet sink. If the packet sink is not empty, the switch outputs one packet per cycle from the sink in a FIFO manner.

4 Proposed Sink Models

4.1 A p-sink model

Our objective is to simplify the ideal sink model with small performance penalty. We observe that the maximum number of flits entering a switch per cycle is p. This means that at maximum p flits may need to be ejected from a switch per cycle. This number is independent of v. By this observation, we can use p sink queues instead of $p \cdot v$ sink queues. The length of a sink is still the maximum number of flits of a packet. Besides, in order to have a more structured design, we could connect the p sink queues to the crossbar, as illustrated in the dashed box of Figure 3.

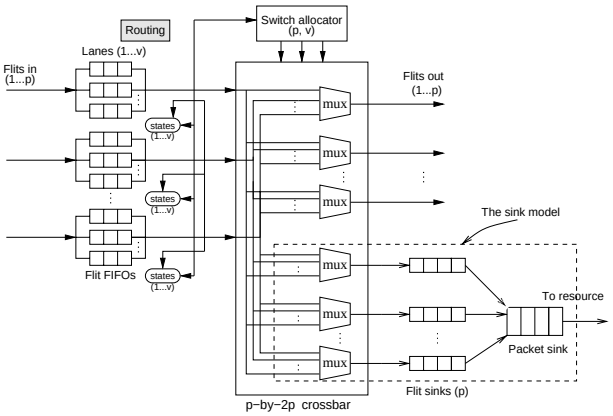

Figure 3. A p-sink flit ejection model

To enable ejecting flits by the p-sink model, we now extend the lane state to a *arriving* and a *reception* state. If a head flit reaches its destination, the lane the flit occupies transits from the routing to the arriving state. Then it will try to associate with an empty sink, i.e., to establish a *lane-to-sink* association. If the association is successful, the lane enters the *reception* state. Subsequently the other flits of the packet follow this association exactly like flits advancing in the network. Upon the tail flit entering into the sink, the association is torn down. If the association fails, the head flit is blocked in place holding the lane buffer. To speed up flit ejection, the contentions for the crossbar input channels and crossbar are arbitrated on priority. A lane in a reception state has a higher priority than a lane in a state for forwarding flits. The drawback due to this sink model is the increase of blocking time when flits reach their destinations. First, the lane-to-sink association may fail since all sink queues might be in use. Second, only one lane per PC can win arbitration to a crossbar input channel. In case of more than one lane of a PC are in an ejection state, only one can use the channel.

To implement this model, the crossbar must double its capacity from p-by-p (p $p \times 1$ multiplexers) to p-by-2p ($2p$ $p \times 1$ multiplexers). The number of control ports of the crossbar is doubled proportionally.

4.2 A coupling scheme

To further simplify the switch, we could modify the p-sink model by using a coupling scheme in which the flits from a PC are dedicated to a sink. In other words, lane(i, j), where i ($i \in [1, p]$) is the PC identifier and j ($j \in [1, v]$) the lane identifier, is dedicated to sink(i). In this way, the p $p \times 1$ multiplexers for sinking flits can be replaced with p 1×2 demultiplexers, as shown in Figure 4.

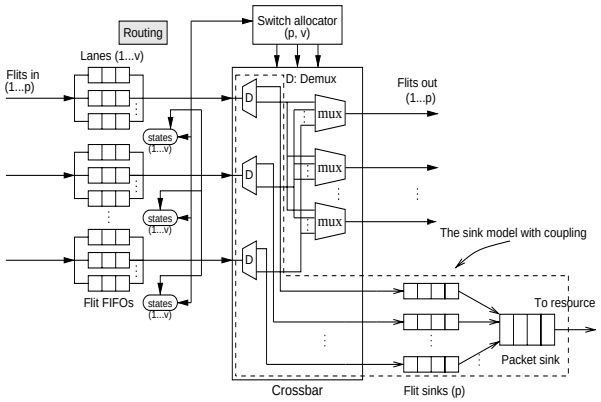

Figure 4. The coupled p-sink model

Due to this coupling, lane(i, j) is not allowed to use a sink(k), where $k \neq i$, even if it is empty. This potentially increases the blocking time of flits and the sink queues might be under-utilized.

	$p \times 1$ Mux	1×2 Demux	Sink queue
Ideal model	-	$p \cdot v$	$p \cdot v$
Decoupled model	p	-	p
Coupled model	-	p	p

Table 1. Cost of the sink models

Table 1 summarizes the number of each component to implement the sink models.

5 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. The simulator is programmable as to network size, packet injection rate, sink model, etc. We construct a 2D 4×4 mesh network with bidirectional channels. The network does dimension-order **X-Y** routing, which is deadlock-free and deterministic. The purpose of our experiments is to examine the performance (latency and throughput) of the p

sink model in Figure 3 and its coupled counterpart in Figure 4. The baseline is the ideal sink model in Figure 2.

The simulations were run with uniformly distributed traffic. Resources injected 4-flit packets to random destinations except for themselves at a constant rate. 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 investigate 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 packet is output to its destination resource, including source queuing time and packet queuing time at the destination. Throughput λ is the number of packets received per cycle per node.

Number v of lanes per physical channel	3
Length of a lane	2 flits
Length of a sink queue	4 flits

Table 2. Simulation parameters

Simulation parameters are listed in Table 2. To minimize buffering cost, the lane length was two, which is the minimal depth requirement of a lane in order to pipeline flits since sending and receiving credits take two cycles.

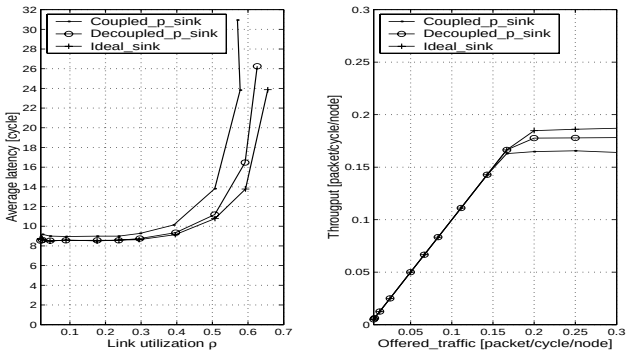

Figure 5. Performance comparison

Figure 5 compares the performance of the proposed sink models with that of the ideal model. The left figure gives the average latency in the function of link utilization (network load). Compared with the ideal model, the decoupled p-sink model achieves equivalent performance below link utilization $\rho = 0.5$. When ρ is higher than 0.5, the latency with the ideal model is better. This is because slower ejection of flits results in higher congestion thus higher latency when the network is nearly saturated. As can be expected, the performance of the coupled p sink model is worse than that of its decoupled counterpart. However, the penalty is not significant. Specifically, when the link utilization is below 0.4, the latency difference is about 0.5 cycle. The right figure reports the throughput versus the offered traffic, which is measured in terms of the number of packets injected per cycle per node. The saturation throughput for the coupled, decoupled, and ideal model is 0.165, 0.178, and 0.186, respectively. Normalized with the ideal model, the relative throughput is 0.96, 0.89, and 1, respectively.

6 Conclusions

Cost-effective flit-ejection models for wormhole switches are desired for chip implementations. We have presented a novel sink model that achieves approximate performance with the ideal model when the network is reasonably loaded (below 0.5 capacity). By coupling a physical channel with a sink queue, the switch complexity is further reduced with small performance penalty. 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 combine flit ejection together with flit admission in order to achieve practically cost-effective flit admission/ejection solutions. Such a combination is essential for evaluating the performance of an on-chip network.

References

[1] V. Agarwal, M. S. Hrishikesh, S. W. Keckler, and D. Burger. Clock rate versus IPC: the end of the road for conventional microarchitectures. In *Proceedings of the 27th annual international symposium on Computer architecture*, pages 248 – 259, 2000.

[2] 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.

[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 C. L. Seitz. The torus routing chip. *Journal of Distributed Computing*, 1(3):187–196, 1986.

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

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

[7] Z. Lu and A. Jantsch. Flit admission in on-chip wormhole-switched networks with virtual channels. In *Proceedings of International Symposium on System-on-Chip*, 2004.

[8] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch. Guaranteed bandwidth using looped containers in temporally disjoint networks within the Nostrum network on chip. In *DATE*, 2004.

[9] L. S. Peh and W. J. Dally. Flit-reservation flow control. In *Proceedings of High Performance Computer Architecture*, pages 73–84, Jan. 2000.

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

[11] 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.