Layered Switching for Networks on Chip

Zhonghai Lu
zhonghai@kth.se

Ming Liu
mingliu@kth.se

Axel Jantsch
axel@kth.se

Royal Institute of Technology, Sweden

ABSTRACT

We present and evaluate a novel switching mechanism called *layered switching*. Conceptually, the layered switching implements wormhole on top of virtual cut-through switching. To show the feasibility of layered switching, as well as to confirm its advantages, we conducted an RTL implementation study based on a canonical wormhole architecture. Synthesis results show that our strategy suggests negligible degradation in hardware speed (1%) and area overhead (7%). Simulation results demonstrate that it achieves higher throughput than wormhole alone while significantly reducing the buffer space required at network nodes when compared with virtual cut-through.

Categories and Subject Descriptors:
B.4 [Hardware]: Input/Output Data Communications;

General Terms: Design, Theory, Performance

Keywords:
Network-on-Chip, System-on-Chip, Switching Technique

1. INTRODUCTION

The communication platform is becoming increasingly crucial for complex System-on-Chip (SoC) integration in the nanometer regime [12]. Systems such as mobile platforms, personal handheld sets and multimedia terminals require the on-chip interconnects to handle a huge amount of traffic, driving the traditional bus-based architectures towards network-based architectures. As the Network is fabricated on Chip (NoC), both high performance and small area are basic requirements [4, 8, 9].

The switching scheme of an interconnection network determines how packets flow through each node and how to handle packet blocking. It is the primary factor in dominating the network's performance [3]. Store-and-forward switching (**SAF**), wormhole switching (**WH**) [2] and virtual cut-through switching (**VCT**) [6] are three commonly used schemes. Whereas an SAF switch must receive an entire packet before forwarding it downstream, a WH or VCT switch starts to transmit portion of a packet, i.e., flit[1], once the downstream buffer is available to hold the flit. In this way, WH and VCT make flit transmission pipelined, resulting in lower latency than SAF if the network is not saturated. WH and VCT are both *cut-through* switching schemes. They mainly differ in how they handle packet blocking. With WH, buffers and links are allocated at the flit-level and the switch buffering capacity is a multiple of a flit. If a packet is blocked, flits of the packet are *stalled* in place. With VCT, a switch, at which a packet is blocked, must receive and *store* all flits of the blocked packet. This enforces that buffers and links are allocated at the packet-level and the buffering capacity in switches must be a multiple of a packet. VCT utilizes the network's bandwidth more efficiently, achieving higher throughput than WH but requiring higher buffering capacity. From the performance perspective, implementing VCT for on-chip networks is preferable. However, switch buffers take a large portion of area and consume significant power. In the Æthereal WH switch [4], the flit buffers occupy about 30% of area even after being optimized as hardware FIFOs. It is estimated in [15] that the access of flit buffers consumes about 30% of total node power in a 4×4 torus network. From the cost and power perspective, implementing WH for on-chip networks is more desirable.

We present a *layered* switching strategy which combines the salient features of both WH and VCT. It performs better than WH, and consumes less storage than VCT. The idea is to introduce a data abstraction, *group*, between *flit* and *packet*. In order to minimize communication overhead and hardware implementation cost, we realize a *logical* group by which flits of a packet are virtually partitioned into groups of a uniform size. This abstraction enables us to vertically lay WH on top of VCT. As virtual channels are allocated *packet-by-packet*, and buffers and links are allocated *group-by-group*, we say that WH is performed on packets and groups; Due to the fact that buffers and links are allocated *group-by-group* and link pipeline is conducted *flit-by-flit*, we say that VCT is performed on groups and flits. The synergy of WH and VCT results in a layered WH-VCT switching.

In the sequel, Section 2 briefs related work. In Section 3, we describe the operation of a canonical wormhole switch. Then we detail the layered switching strategy, compare it with WH and VCT, present a hardware implementation study, and show synthesis results in Section 4. Simulation results are reported in Section 5. Finally, we conclude the paper in Section 6.

2. RELATED WORK

Wormhole switching dates back to the Torus Routing Chip [2], a fast switch in 1986 technology. Dally proposed Virtual Channel (VC) flow control in [1] to break the dependency between physical channel (PC) allocation from VC allocation. Since VCs decouple buffer resources from transmission resources, this scheme allows

[1] Virtual cut-through does not divide packets into flits. We use the division to allow a consistent comparison in the paper.

DAC 2007, June 4-8, 2007, San Diego, California, USA

active flits to pass blocked flits using link bandwidth that would otherwise be left idle. Network throughput is thus improved.

Based on VC flow control, much work has been done to further enhance the network performance by optimizing wormhole switch architectures. Typical optimizations are centered on the control path. In [14], routing and arbitration latency may be eliminated by statically scheduling buffer and link resources. This leads to the increasing size of routing memory, thus very high cost. In addition, this static method can not handle dynamic traffic. Flit-reservation flow control [10] allows to pre-schedule resources but scheduling decisions are determined at run-time. Link and buffer usage is scheduled by sending control flits ahead of data flits over independent channels. Scheduling decisions are stored in reservation tables in association with each input and output PC. This scheme improves performance but adds significant complexity to the switch architecture. Recently, techniques to hide routing and arbitration latency are proposed in [9]. This is realized by pre-determining the results of routing and arbitration decisions one cycle before they are requested. All the above works implement purely WH and greatly complicate the switch designs.

In contrast, our proposal is a novel switching strategy. Like *hybrid switching* [13], our mechanism bridges the performance gap between WH and VCT. However, our approach is fundamentally different from it. The hybrid switching combines WH and VCT, performing network flow control on either flits like WH or packets like VCT. It differentiates blocked packets by *selectively* performing either WH or VCT blocking policy on them. With WH, the blocked packets are stalled in place. With VCT, the blocked packets are buffered locally in the node where the packets lose arbitration. Our switching scheme performs network flow control on groups. It does not differentiate blocked packets. The *same two-level blocking policy* applies to all blocked packets. Upon packet blocking, both WH and VCT blocking polices are employed, but on different data abstraction. The WH blocking policy is performed on groups, i.e., all groups are stalled in place. The VCT blocking policy is performed on each individual group, i.e., flits of an entire individual group is to be received and buffered locally at the node where the group is stalled. Besides, since the hybrid switching differentiates packets, it requires additional informative bits in a packet. This is not necessary with our switching scheme. In [5], Hu *et al*. proposed a *hybrid routing* technique that judiciously switches between deterministic and adaptive routing according to the network congestion conditions. Our work focuses on the switching strategy, thus is orthogonal to theirs. If needed, both works could be combined.

3. WORMHOLE SWITCH ARCHITECTURE

Figure 1 illustrates a canonical input-queuing wormhole switch architecture [11]. The Æthereal NoC [4] adopts this architecture after carefully analyzing the performance and cost trade-offs between different queuing strategies. The switch has p physical channels (PCs or links) and v lanes (VCs) per PC. It conducts credit-based link-level flow control to coordinate packet delivery between adjacent switches to avoid buffer overflow and flit loss.

As sketched in Figure 2, for each packet, a switch passes through the following steps: *routing* S_1, *lane allocation* S_2, *flit scheduling* S_3, *switch arbitration* S_4, *switch traversal* S_5 and *lane release* S_6. In the routing step, the routing logic determines the routing path over which the packet advances. Routing is performed only when the head flit of a packet becomes the earliest-come flit in the lane. After routing, the output PC is determined. In the step of lane allocation, the lane allocator tries to associate the lane the packet occupies with an available lane in the next hop, i.e., to make a *unique* lane-to-lane association. Note that it is not necessarily required that

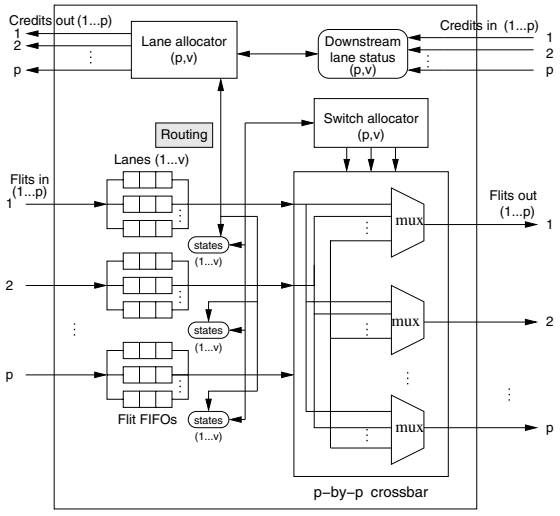

Figure 1: A canonical wormhole switch

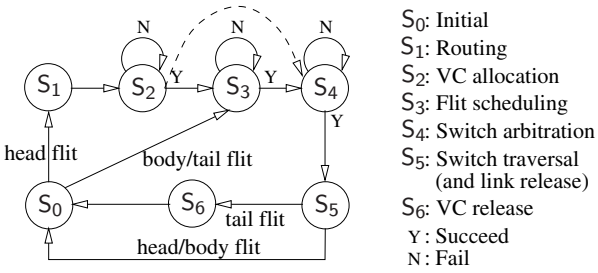

Figure 2: Sequence of steps based on flits

there is an empty buffer in the requested lane in order for the lane to be associated or allocated. But, if the buffer availability (at least one buffer) is a pre-condition to complete VC allocation, the flit-scheduling step is not needed any more for the head flit of a packet. In such a case, a head flit directly transits from step S_2 to S_4, as indicated by the dashed line in Figure 2. A lane-to-lane association fails when all requested lanes are already associated to other lanes in directly connected switches, or the lane loses arbitration in case multiple lanes in the switch request the same downstream lane. If the lane-to-lane association succeeds, the packet enters the flit scheduling step. If there is a buffer available in the associated lane, the lane enters the switch arbitration (S_4), which can be done with a two-level arbitration scheme. The first level of arbitration is performed on the lanes sharing the same input port to the crossbar. The second level of arbitration is for the crossbar traversal to output links. If the lane wins the two levels of arbitration, the earliest-come flit in the lane enters the step of switch traversal (S_5). The flit is switched out to the next hop and the granted link is released. Otherwise, the lane stays in the arbitration step. Once the tail flit is switched out, the lane-to-lane association is released (S_6), thus the allocated lane is available to be reused by other packets. Credits are communicated between adjacent switches in order to keep track of the status of downstream lanes, such as if a lane is free, and a count of available buffers in the lane.

4. THE LAYERED WH-VCT SWITCHING

4.1 Operation

Wormhole switching allocates lanes at the packet level, i.e., packet-by-packet, but packet buffering and link allocation are conducted at

the flit level. This mismatch between lane allocation and buffer/link usage causes three major drawbacks. First, flits other than a head flit must again enter the flit scheduling step, and contend for switch arbitration and traversal. This incurs extra control cycles. Second, since flits of a packet may be distributed in nodes along the packet's routing path, they occupy allocated VCs and independently contend for the link bandwidth along the path. This increases contention for outgoing links, limiting the network's maximum throughput [13]. Third, the spread-out flits may occupy only a portion of allocated lane buffers. The buffers between the flits can not be used by any other packets, making the buffer utilization less efficient.

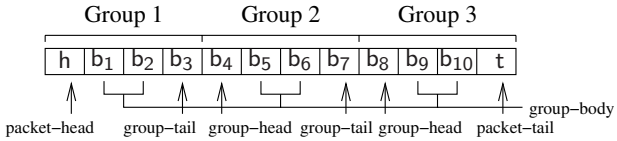

Figure 3: Partitioning a packet into 3 groups

We are motivated to shorten the control cycles and make efficient use of flit buffers and links. To this end, we introduce a data abstraction, *group*, between flit and packet, and perform network flow control on groups instead of flits. We partition flits of a packet into groups according to the depth d of a VC. Groups thus have the same size. Specifically, we partition a packet of m flits into $\lceil m/d \rceil$[2] groups, and each group contains d flits. If m/d is not an integer, we have to pad the last group with extra flits[3]. As shown in Figure 3, a packet of 12 flits is virtually partitioned into three groups if d is four, and one group contains four flits. As groups are of the uniform size, we can use a simple counter to implement the grouping (See Section 4.3). No additional bits are needed to explicitly distinguish the group-head, group-body and group-tail flits. Therefore a group is a *logical* group.

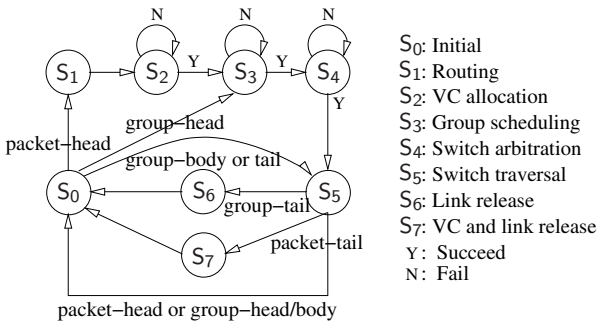

Figure 4: Sequence of steps with grouping

The sequence of steps with the grouping scheme is depicted in Figure 4. After a packet-head flit performs routing, VC allocation, scheduling and switch arbitration, the rest of flits in the first group directly transit to the switch traversal step S_5. For a group-head flit, the lane transits from the initial step S_0 to the scheduling step S_3. Since it is the scheduling step for one group, we call this step *group scheduling*. After completing this step, the group-head flit enters step S_4 (switch arbitration). By winning the switch arbitration, the group-head flit is allocated with the requested link and then goes to step S_5 (switch traversal). This link allocation will be inherited by the body and tail (either group-tail or packet-tail) flits of the group, which will directly go from step S_0 to S_5. This shrinks the flit-scheduling and switch arbitration cycles for group-body, group-tail and packet-tail flits, thus speeding up the flit traversal. After the group-tail flit is switched out, the granted link is released (S_6). When the packet-tail flit is switched out, both the granted link and the allocated VC are released (S_7). To complete the group-scheduling step S_3, *an entire group of free flit buffers* must be detected and then allocated. This is a strong request on the number of available buffers. However, it is in fact sufficient to detect *one buffer* available in the requested lane. The availability of one buffer means either there are buffers for an entire group or the group is in the process of being transmitted to the next switch since links are allocated on a group basis. While the previous group is switched out, the new group can be switched in, occupying the lane buffers in a pipelined fashion. Thus the condition is satisfied *on-the-fly* with the *one-buffer-availability*.

The resulting switching mechanism implements a layered WH and VCT switching. It performs WH on packets and groups since VCs are allocated to packets, and buffers and links to groups. Meanwhile, it performs VCT on groups and flits since buffers and links are allocated to groups and the link pipeline is still conducted on flits. The layered switching requires a switch to hold entire group(s) in case of packet blocking. The switch buffering capacity is a multiple of a group. This seems implying a higher buffering consumption than WH. However, this is not necessary. If we use the VC depth d or a factor of d of a wormhole switch as the group size, the layered switching does not incur extra buffers in switches.

4.2 Discussion

Switching (Flow control level)	VC allocation	Packet buffering	Link allocation	Link pipeline
WH (Flit)	Packet	Flit	Flit	Flit
WH-VCT (Group)	Packet	*Group*	*Group*	Flit
VCT (Packet)	Packet	Packet	Packet	Flit

Table 1: Flow control granularity

The granularity of network flow control can be used to differentiate one switching technique from another [3]. WH with or without VCs performs flow control on the *flit level*; VCT does this on the *packet level*. The layered switching in fact realizes a *group-level* flow control. The group size g in flits is in the range $[1, m_{max}]$, where m_{max} is the maximum number of flits of packets. Table 1 compares the three pipeline switching methods. All of them allocate VCs to packets and pipeline flits on links. The unit for packet buffering and link allocation reflects their flow control granularity.

The group-level flow control (WH-VCT) can be positioned between the flit-level (WH) and the packet-level flow control (VCT). If $g = 1$, one group is one flit, and the group-level flow control becomes the flit-level one; if $g = m_{max}$, one group is one packet, and the group-level flow control resembles the packet-level one. Note that a group is not simply a larger flit. If this were the case, a switch must *always* receive an entire group before forwarding it downstream (in the SAF fashion). The link pipeline would be based on groups. In our case, a switch sends portion of a group, i.e., flit, downstream as soon as enough buffering is available to hold the flit (in the VCT fashion). The link pipeline is still based on flits.

4.3 An implementation study

To investigate the hardware implementation overhead of our proposal, we implemented a wormhole switch and a layered WH-VCT switch at RTL by following the canonical switch model. Figure 5

[2] $\lceil x \rceil$ is the ceiling function returning the least integer that is not less than x.

[3] We expect that, in NoC designs, it is possible to select m and d in a way to minimize or avoid this overhead entirely.

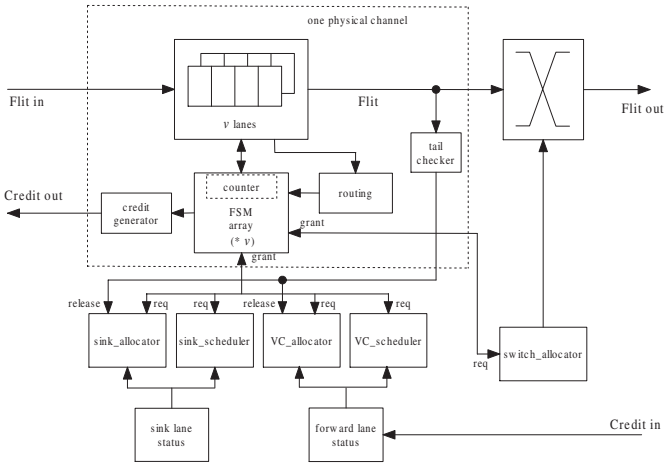

Figure 5: The implementation structure

sketches their implementation structure. For simplicity, only a single input and a single output are shown. The sink modules (sink allocator/scheduler/status) in Figure 5 are used for flit ejection. They implement a cost-effective p-sink model that ejects flits using p shared sink queues instead of $p \cdot v$ sink queues as required by an ideal flit ejection model [7].

Both designs share most of the modules in the structure. The main differences lie in the FSMs and switch allocator. There are v FSMs per PC, one for each lane. An FSM manages the states of a lane, as described previously. With the grouping scheme, the FSMs are added with a *counter*. Together with the flit type information in a flit, this counter is used to distinguish different flit types. Wormhole switching encapsulates a packet into a *head flit*, *body flit(s)* and a *tail tail*. Single-flit packets are also possible but irrelevant to our discussion since it is meaningless to perform grouping on them. In a flit, there is a *flit type field* to indicate its flit type. With the grouping scheme, there are still physically three flit types but logically five flit types, i.e., the *packet-head*, *group-head*, *group-body*, *group-tail* and *packet-tail* flits (See Figure 3). If a group has d flits, the counter has a value range from 0 to $d-1$. Starting with 0, the counter value is incremented whenever a flit is switched out. Once a group-tail or packet-tail flit is sent, it is reset to 0. The counter value 0 represents either a packet-head or group-head flit. The counter value $d-1$ represents either a packet-tail or group-tail flit. Since group-head and group-tail flits are physically body flits, this helps to distinguish a group-head from a packet-head flit, and a group-tail from a packet-tail flit. The counter value(s) between 0 and $d-1$ are group-body flits. The switch allocator also uses the counter values to distinguish flits of different types. This simple counter enables to virtually partition a packet into groups of flits. The implementations are not single-cycle models. Instead they are cycle-true. With WH, it takes six cycles for a head flit and four cycles for a body or tail flit to pass through the switch. With the WH-VCT, it takes six cycles for a packet-head flit, four cycles for a group-head flit, and one cycle for other types of flits to pass through the switch. This implies that the WH-VCT can also improve the zero-load latency. Despite the high latency for flit traversal of a switch, flits are pipelined on links, thus the network bandwidth can be efficiently utilized, achieving high throughput.

We synthesized the designs for performance using 180*nm* technology. Both designs implement the dimension-order XY routing. We set the flit width $W_{flit} = 32$, the number of lanes per PC $v = 4$, and the lane depth $d = 2$ flits. Similarly to [5], we used registers to implement the flit FIFOs in order to achieve better performance and power efficiency. The switch speed is constrained by the control path. The WH switch can be clocked up to 396 MHz. As a data cycle comprises two control cycles, the data operating frequency is 198 MHz. The WH switch has an estimated gate count of 41K gates. The control clock of the WH-VCT switch can operate at 392 MHz, resulting in a data clock of 196 MHz. The WH-VCT switch consumes 44K gates. About 20% of the areas is counted for buffers. The frequency degradation is 1% and area overhead is 7%. Note that this area overead is less significant when compared with a VCT switch. For a VCT switch, the lane depth d must be a multiple of a packet. Assume that $d = 2$ packets and a small packet size $m = 4$ flits, then the number of lane buffers is quadruple, resulting in a gate count of about 64K for the VCT switch.

5. SIMULATION RESULTS

5.1 Experimental setup

To investigate the performance of the proposed technique, we construct 4×4 mesh networks using the two switch designs. The baseline network uses WH, the other the layered WH-VCT switching. The XY routing guarantees deadlock-free on the mesh [3].

The simulations were run with uniformly distributed traffic. Fixed-size packets were injected synchronously to random destinations except for themselves at a constant rate. Except otherwise noted, contention for lanes and channel bandwidth were resolved using round-robin. Each node injected 1500 packets into the network. Simulation statistics were collected after all the packets were received. For each simulation, warm-up and cool-down cycles were not included in the results, i.e., the results show the performance of the network at a steady state. We investigated the packet latency and network throughput. Latency of a packet is calculated from the instant the packet is injected into the packet source queue to that the packet is ejected from the network. This packet latency has two components. One is the queuing time in the source queue. The other is the network delivery time counting from the instant the packet enters the network to that the packet is ejected from the network. Throughput is defined as the number of flits received per cycle per node in normalization with the network capacity, i.e., the ideal throughput, which is 1 flit/cycle/node for a mesh network under the uniform traffic.

	Test 1		**Test 2**		**Test 3**		**Test 4**		**Test 5**	
	W1	**L1**	**W2**	**L2**	**W3**	**L3**	**W4**	**L4**	**W5**	**L5**
v	4									
m	8		8		8		8		16	
d	2		4		8		4		4	
g	-	2	-	4	-	8	-	4	-	4
n	-	4	-	2	-	1	-	2	-	4
A	RR		RR		RR		FR		RR	

Table 2: Switch parameters in all 5 tests (v: the number of VCs per PC; m: packet size in flits; d: the VC depth; g: group size in flits; n: the number of groups; A: switch arbitration.)

We conducted five tests. The switch parameters are listed in Table 2, where **Wx** and **Lx** mean **Test x** for **W**ormhole and the **L**ayered switching, respectively. We set the group size g to the VC depth d, i.e., one VC can hold exactly one group of flits. With the layered switching, Test 1, 2 and 3 result in different numbers of groups due to a different VC depth. They partition a packet of 8 flits into four, two and one group(s), respectively. Test 3 with the layered switching (L3) mimics virtual cut-throughput since its VC

depth equals to the packet size. Test 1, 2 and 3 use round-robin (RR) for switch arbitration. To study the effect of the arbitration mechanism, Test 4 uses fixed-priority (FR) for switch arbitration. Test 5 has a longer packet size of 16 flits.

5.2 Basic comparison

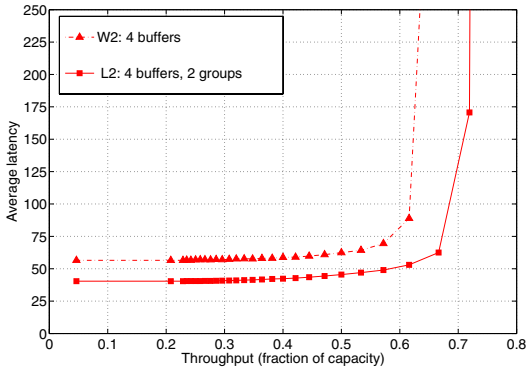

Figure 6: Basic performance comparison

Figure 6 draws the average latency in data cycles as a function of the measured throughput for Test 2. With the same amount of storage, the layered switching (L2) saturates the network at a higher throughput than WH (W2) due to more efficient use of VC buffers and link bandwidth. The graph also reflects a lower latency due to shrinking flit-scheduling and switch arbitration cycles for group-body and tail flits, as well as reducing VC and link contention. Specifically, the layered switching (L2) reduces the minimum latency by 28% from 57 to 41 cycles. The network saturates at 72% of capacity, 12% higher than 64% of its WH counterpart (W2). As we shall see in Figure 8, this throughput is even higher than the throughput (68%) in Test 3 with WH (W3: 8 buffers per VC), which doubles the buffer capacity of Test 2.

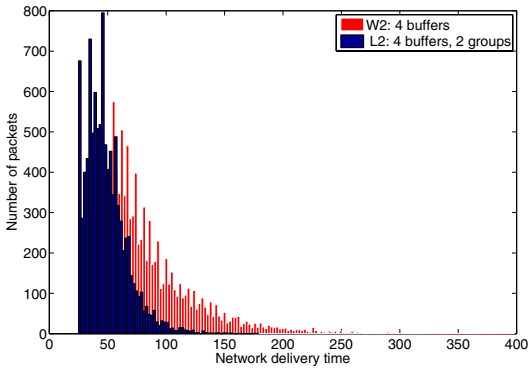

Figure 7: Histogram of network delivery time

In Figure 7, we depict a distribution graph by the number of packets with their experienced network delivery time. The packet injection rate is 1 packet every 13 cycles (0.615 flit/cycle/node). At this point, both networks are highly utilized but not saturated. We can observe that the layered switching (L2) moves the envelop of the WH (W2) towards the origin along the X-axis, implying that more packets enjoy lower network delivery time. In addition, the number of packets experiencing high latency is reduced. Particularly, the layered switching shrinks the observed maximum delivery time from 396 to 179 cycles, reducing it by 55%.

5.3 The effect of group size

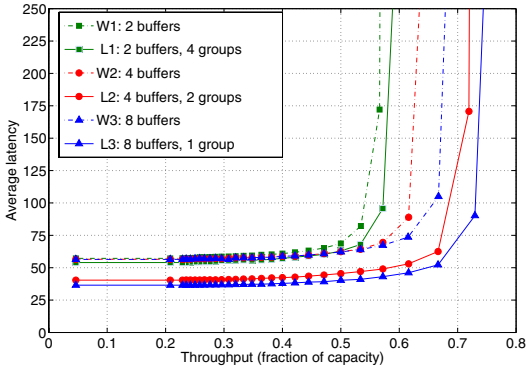

Figure 8: Performance with group sizes

Test 1 and 3 are designed to observe the minimum and maximum improvement with the layered switching. Figure 8 shows the results. As the number of partitioned groups decreases, the size of a group is closer to a packet and the benefit is generally increased. With Test 1, 2, 3, the minimum latency improvements are 6%, 28% and 35%, respectively; the throughput improvements are 5%, 12% and 10%, respectively. The minimum improvement in both latency and throughput occurs in Test 1, where the depth of a VC is two, which is minimal to counter for the round-trip credit latency (2 cycles) so as to enable fully pipelining flits [3]. The best latency improvement happens in Test 3 where one packet is partitioned into a single group. The best throughput improvement occurs in Test 2, not in Test 3. This is due to the fact that Test 3 with WH (W3) is already approaching the bound of saturation throughput. Although the layered switching (L3) still improves this throughput, the acceleration is slowed down. As a larger group size implies higher buffer consumption, Test 2 addresses the tradeoff between high performance and small buffer requirement.

5.4 The impact of arbitration mechanism

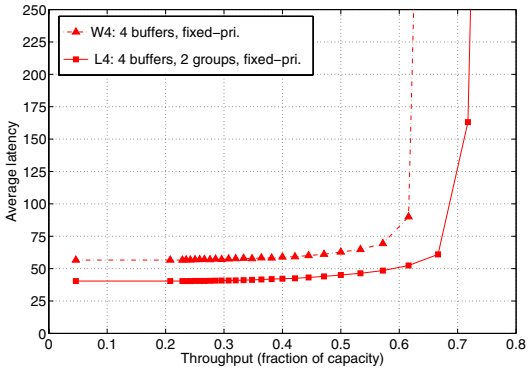

Figure 9: Performance with prioritized arbitration

An important property for arbitration is fairness. The round-robin arbitration for switch traversal is fair. We implemented also a priority-based arbitration policy. This unfair policy sets a fixed priority for each VC of an input port. Flits situated in a VC with a higher priority win switch arbitration to use the contended link. Simulation results on latency are shown in Figure 9. With the unfair arbitration, the layered switching (L4) improves performance

similarly to the round-robin policy (W4). The latency and throughput are improved by 28% and 15%, respectively. We also looked at the distribution graph of delivery time with the fixed-priority arbitration. Similarly to the fair arbitration, the group-level switching largely moves the graph envelop towards the origin along the X-axis. This means that the layered switching improves the network performance without bias towards switch arbitration policies.

5.5 The effect of packet size

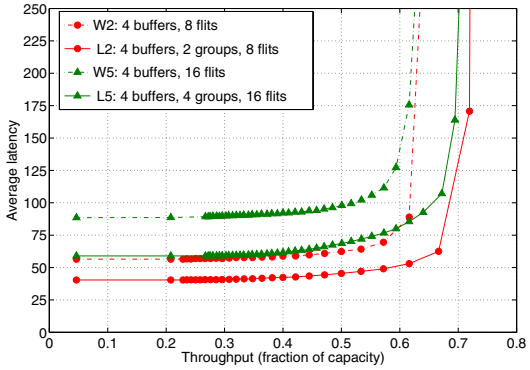

Figure 10: Performance with longer packets

The previous experiments use packets of eight flits. Test 5 uses longer packets of 16 flits. The results are depicted in Figure 10 together with Test 2 that uses the same amount of buffer storage. The minimum-latency and throughput are improved by 34% and 11%, respectively. Enlarging the packet size without increasing flit buffers per VC leads to a larger number of groups per packet. In this case, the group size is increased from two to four. The incremental latency improvement from 28% in Test 2 to 34% in Test 5 comes mainly from the increasing number of flits enjoying fast switching due to the deduction of flit-scheduling and switch arbitration cycles. For the latency, with the layered switching, longer packets benefit more from the grouping scheme. But the improvement in throughput is not linear due to the network's approaching the throughput bound.

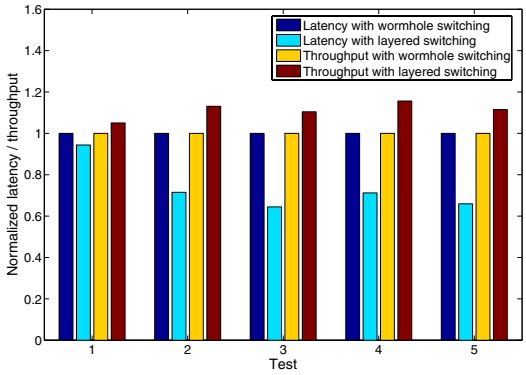

Figure 11: Normalized latency and throughput

We summarize the results normalized with their corresponding wormhole counterparts for the five tests in Figure 11. The maximum/minimum improvement in latency and throughput is 35% (Test 3) /6% (Test 1) and 15% (Test 4) /5% (Test 1), respectively.

6. CONCLUSION

We have presented the concept of layered WH-VCT switching. By virtually partitioning packets into groups of flits, network flow control can be performed on groups but pipelining is still conducted on flits. It reduces the packet scheduling and switch arbitration cycles, and makes efficient use of links and buffers. Therefore the envelope of the latency-throughput graph is lowered and extended, when compared with WH. For instance, with 4 VCs per input and 4 flit buffers per VC, the layered switching scheme partitioning one packet into two groups reduces the average packet latency by 28%, and improves throughput by 12.5% from 64% to 72% of capacity, which is even higher than the throughput (68%) of WH with doubled buffer capacity, i.e., 8 flit buffers per VC. In comparison with VCT, the layered switching requires significantly less storage since it buffers groups not packets upon packet blocking. The grouping is logical and implemented by a simple counter in a switch, thus the communication and implementation overhead is minimal. We believe that such an efficient and cost-effective switching technique is beneficial to the design of cost-constrained on-chip networks.

Our future work is to investigate the potential of power saving of the layering switching. Another interesting track is to dynamically change the group size under different traffic conditions for further performance enhancement.

7. REFERENCES

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

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

[3] W. J. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufman Publishers, 2004.

[4] K. Goossens, J. Dielissen, and A. Rădulescu. The Æthereal network on chip: Concepts, architectures, and implementations. *IEEE Design and Test of Computers*, 22(5):21–31, Sept-Oct 2005.

[5] J. Hu and R. Marculescu. DyAD: Smart routing for networks-on-chip. In *Proc. of the 41st Design Automation Conference*, June 2004.

[6] P. Kermani and L. Kleinrock. Virtual cut-through: A new computer communication switching technique. *Computer Networks*, 3:267–286, January 1979.

[7] Z. Lu and A. Jantsch. Flit ejection in on-chip wormhole-switched networks with virtual channels. In *Proceedings of the IEEE Norchip Conference*, pages 273–276, November 2004.

[8] F. G. Moraes, N. Calazans, A. Mello, L. Möller, and L. Ost. HERMES: an infrastructure for low area overhead packet-switching networks on chip. *Integration, the VLSI Journal*, 38(1):69–93, 2004.

[9] R. Mullins, A. West, and S. Moore. Low-latency virtual-channel routers for on-chip networks. In *Proceedings of the 31st Annual Symposium on Computer Architecture*, 2004.

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

[11] L. S. Peh and W. J. Dally. A delay model for router microarchitectures. *IEEE Micro*, 21(1):26–34, Jan.-Feb. 2001.

[12] M. Sgroi, M. Sheets, A. Mihal, K. Keutzer, S. Malik, J. Rabaey, and A. Sangiovanni-Vencentelli. Addressing the system-on-a-chip interconnect woes through communication-based design. In *Proceedings of the 38th Design Automation Conference*, 2001.

[13] K. G. Shin and S. W. Daniel. Analysis and implementation of hybrid switching. *IEEE Transactions on Computers*, 45(6):684–692, 1996.

[14] M. B. Taylor et al. The Raw microprocessor: A computational fabric for software circuits and general purpose programs. *IEEE Micro*, 22(2):25–35, 2002.

[15] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik. Orion: A power-performance simulator for interconnection networks. In *Proceedings of the 35th International Symposium on Microarchitecture (MICRO)*, November 2002.