Leveraging Torus Topology with Deadlock Recovery for Cost-Efficient On-Chip Network

Minjeong Shin, John Kim
Department of Computer Science
KAIST
Daejeon, Korea
{shinmj, jjk12}@kaist.ac.kr

***Abstract*—On-chip networks are becoming more important as the number of on-chip components continue to increase. 2D mesh topology is a commonly assumed topology for on-chip networks but in this work, we make the argument that 2D torus can provide a more cost-efficient on-chip network since the on-chip network datapath is reduced by 2× while providing the same bisection bandwidth as a mesh network. Our results show that 2D torus can achieve an improvement of up to 1.9× over a 2D mesh in performance per watt metric. However, routing deadlock can occur in a torus network with the wrap-around channel and requires additional virtual channels for deadlock avoidance. In this work, we propose deadlock recovery with tokens (DRT) in on-chip networks that exploits on-chip networks – exploiting the abundant wires available while minimizing the need for additional buffers. As a result, deadlocks can be exactly detected without having to rely on a timeout mechanism and when needed, recover from the deadlock. We show how DRT results in minimal loss in performance, compared with deadlock avoidance using virtual channels, while reducing the on-chip network complexity.**

I. INTRODUCTION

As more components are integrated into a single chip, the on-chip network that interconnects these components together is becoming more critical. There has been significant amount of research in on-chip networks over the past decade, with one of the earliest work [1] proposing a 2D torus topology for on-chip networks to leverage the abundant amount of on-chip wires. However, most on-chip network research have instead assumed a 2D mesh topology because of its simplicity and some commercial multicore chips have also implemented a 2D mesh topology [2]. Prior work [3], [4] have provided comparisons between a mesh and a torus topology in on-chip networks but have assumed additional buffers (virtual channels) to avoid deadlock.

In this work, we revisit the comparison of 2D mesh vs 2D torus topology in on-chip networks and provide an argument for why 2D torus can provide a more cost-efficient on-chip network. Assuming a constant network bisection bandwidth, 2D torus network results in narrower network channels. Thus, the area of an on-chip network router can be significantly reduced since the router area is quadratically proportional to the channel width [5]. In addition, our results show that 2D torus can result in up to 1.9× improvement in performance per watt metric compared with a 2D mesh. However, the wrap-around channel is problematic as it introduces circular dependencies and routing deadlock. Instead of relying on virtual channels (VCs) to avoid routing deadlock, which increases on-chip network buffers, we propose a deadlock recovery mechanism using tokens to detect deadlock and then recover. We add two additional lightweight networks – a token network and a deadlock-recovery network – to support deadlock recovery and allow deadlocks to be exactly detected. The additional networks introduces more buffers but compared with adding virtual channels, our results show that there is still over 35% reduction in the amount of on-chip buffers.

The deadlock recovery with tokens presented in this paper shares similarity to other deadlock recovery mechanisms such as DISHA [6] as they consist of deadlock detection and recovery phase. However, the main difference is that we exploit the characteristics of on-chip networks to add additional lightweight networks to be able to exactly detect when deadlock occurs – instead of relying on a timer-based heuristic. Our work does include a timer to detect if a deadlock detection token has dropped or not. However, the timer is not used to detect deadlock but only when a token drops and thus, no false deadlocks can be detected. In addition, the timers are relatively small – on the order of k cycles where k is the radix of the torus network while the timers in DISHA need to be significantly larger to avoid false deadlock detection.

II. BACKGROUND / MOTIVATION

A. Performance comparison between Torus and Mesh

In Figure 1(a), we compare the performance of 2D mesh and a 2D torus topology in a 16-node chip multiprocessor (CMP) system. [1] To provide a fair comparison between the two topologies, we assume a constant network bisection bandwidth – thus, assume a mesh topology with 16B channels and a torus with 8B channels. The results show that there is no significant difference in performance between the two topologies. Except for apache, torus actually shows very small (a few percent at most) improvement in performance. However, torus is able to achieve this performance by using network channels that have $1/2$ the channel bandwidth of the mesh topology.

The wrap-around channel in a torus reduces the average hop count, compared with a mesh topology and can reduce

[1]The simulation results are from GEMS [7] full system simulator with benchmarks from SPLASH-2 [8] (fft, lu), PARSEC [9] (blackscholes, canneal), and server workloads (apache, jbb).

978-1-4577-1954-7/11/$26.00 ©2011 IEEE

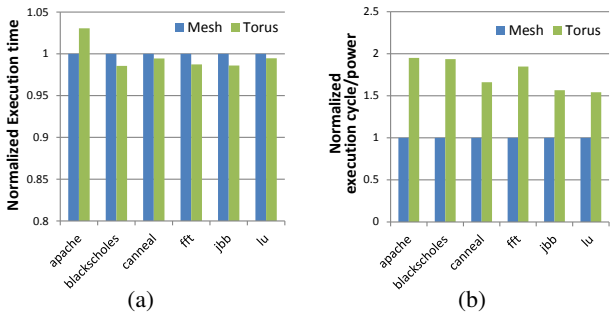

Fig. 1. (a) Performance and (b) performance/watt comparison between a torus and a mesh network. Mesh topology has 16B channel while the torus has 8B channels. Note the zoomed in y-axis for (a).

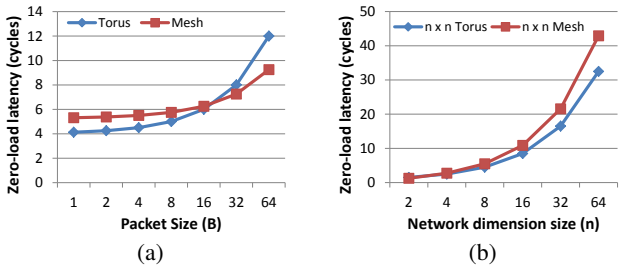

Fig. 2. Zero-load latency comparison of a mesh and a torus (a) as packet size is varied in a 8×8 network and (b) as network size is increased.

the zero-load latency. The zero-load latency of a packet can be expressed as sum of the header latency (T_h) and the serialization latency (T_s),

$$\begin{aligned} T &= T_h + T_s \\ &= H \times t_r + L/b \end{aligned}$$

where H is the hop count, b is the channel bandwidth, t_r is the router latency, and L is the packet size. The hop count H is dependent on the network size (N). Since the average hop count of mesh is $Nk/3$ and the hop count of torus is $Nk/4$ [10], the header latency of the mesh increases faster than torus. In the Figure 2, we compare the zero-load latency between a torus with 8B channel and a mesh with 16B channel, assuming a router latency of 1 cycle, varying the packet size and network size. When the packet size is small (under 16B), torus always has lower zero-load latency even with half of the bandwidth. If the router latency increases, this gap also increases. Thus, although the serialization latency increases with narrower channels, the reduced hop count of the torus topology results in an overall latency which is very comparable to the mesh topology.

However, with narrower channels, the cost of a torus router is much lower than a mesh router. Router area is proportional to $O(p^2w^2)$ where p is the port count and w is the channel width. Since p is identical for both topologies while w is reduced by a factor of 2, the overall area can be reduced by approximately a factor of 4. In addition, router power estimates using Orion2.0 [5] show that the torus topology router provides approximately 2× improvement in power. In Figure 1(b), we plot the *efficiency* of the two topologies by plotting performance/Watt and it shows the torus topology achieving up to 2× improvement in efficiency.

B. Routing Deadlock in Torus network

Even with minimal dimension ordered routing, circular dependency caused by the wrap around channel in a torus can cause routing deadlock. This routing deadlock in a torus network can be handled either through deadlock *avoidance* or deadlock *recovery*. In this section, we describe different deadlock handling mechanisms that are commonly used and motivate deadlock recovery in on-chip networks.

1) Related Work: Deadlock avoidance is commonly done through the use of virtual channels (VCs) [11] to remove the circular dependency. By partitioning the resources (i.e., VCs) appropriately, the circular dependency is removed and routing deadlock does not occur. To avoid the use of virtual channels, bubble router [12] has been proposed. Circular dependencies occur when all resources involved are completely utilized. Bubble router avoids this deadlock by ensuring that there is at least one free buffer within the circular dependency. The bubble router requires virtual cut-through (VCT) flow control and restricts injection such that packets are only injected or allowed to change dimensions when the amount of buffer size available is at least twice the size of the maximum packet size. This restriction ensures that there is always a "bubble" in the circular dependency and prevents a circular deadlock. Bubble router was originally proposed for off-chip networks where buffers are not constrained. However, for an on-chip network where buffers are more restricted, the bubble router requires large amount of buffers and can be inappropriate for on-chip network constraints.

Although deadlock avoidance guarantees deadlock will not occur, it requires additional resources to guarantee deadlock does not occur. If deadlock does not occur frequently, deadlock *recovery* can lead to more efficient mechanism to handle deadlock. Disha [6] is an adaptive deadlock recovery scheme using a timer based detection algorithm and central deadlock buffers for deadlock recovery. At each router, if a packet does not make progress and the timer reaches its threshold, deadlock is assumed to have occurred. A central deadlock buffer is used for deadlock recovery as only one packet is processed for deadlock recovery. However, Disha was developed for large-scale networks and the constraints for on-chip networks are very different. For example, Disha requires a more, complex dynamically accessed queue while simple FIFO queues are often used in on-chip networks. In this work, we propose a deadlock recovery mechanism which can exactly detect when deadlock occurs through the use of a deadlock token. In addition, we use simple FIFO queues while relying on a separate lightweight networks for deadlock recovery.

2) Frequency of deadlock occurrence: The design decision between deadlock avoidance and deadlock recovery is dependent on the frequency of deadlocks. If deadlocks occur very rarely, deadlock recovery is likely more appropriate while deadlock avoidance can be more appropriate if deadlocks occur more frequently. In Figure 3, we plot the saturation throughput for a 8×8 torus network for different traffic patterns using deadlock avoidance with virtual channels (2VCs)

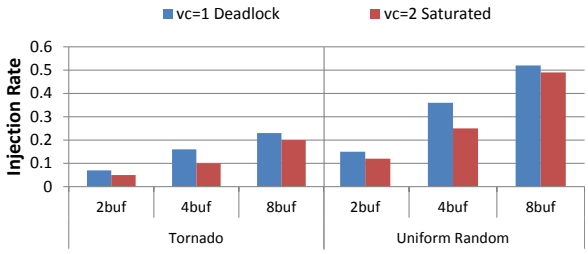

Fig. 3. Network injection rate when the network is saturated ($vc = 2$ data) and the lowest injection rate when first routing deadlock is detected ($vc = 1$ data).

and different buffer size. For the same network, we remove the virtual channel and measure the smallest offered load when deadlock is detected. [2] Compared to deadlock avoidance with VCs, the smallest load when a deadlock is detected is at a higher load than deadlock avoidance network. Thus, with the same resource, using deadlock recovery can not only provide higher performance (i.e., higher throughput) but the likelihood of deadlock occurring is low, especially below network saturation. In addition, deadlock is only likely to occur when traffic exceeds saturation and prior work have shown that traffic in chip multiprocessor (CMPs) is relatively small [13]. As a result, in this work, we avoid using VCs to avoid routing deadlock in a 2D torus topology by using a lightweight deadlock recovery network.

III. Deadlock Recovery in Torus

An overview of the proposed token-based deadlock recovery mechanism is described in Table I. Deadlock recovery consists of deadlock *detection* and deadlock *recovery*. Both parts uses a separate network to reduce the impact of deadlock recovery on the design of the baseline, data network.

A. Deadlock Detection

Two different tokens are used in the deadlock detection process – a *priority* token and a *detection* token. A single priority token exists within each dimension of the torus – e.g., for a 2D torus with radix-k, there are $2k$ priority tokens – one for each ring in the network. The priority token circulates within each ring and the router that has the priority token (defined as the *home* router) can inject a *detection* token into the network. [3] The home router generates a deadlock detection token when a deadlock *might* have occurred – i.e., buffer is full and a packet can not be injected. The detection token is injected in to the separate, token network (Section III-D). If the token is routed around the network and returns to the home router without being dropped, a circular routing deadlock has been detected and a deadlock recovery is needed.

The routing and dropping of the detection token is based on the state of the router and the packets at the head of the queue. If the flit at the head of the queue is destined for an output which is full (i.e., there is no remaining buffer space in the downstream router), we assume a routing deadlock is still a possibility and detection token continues to be routed; otherwise, the detection token is dropped. For example, in Figure 4(a), assume port2 of R_1 is full and a deadlock detection token is generated. The token looks at the head of the queue in port2 and determines which output port that packet uses. If this packet's next destination is R_2, the token is also routed to R_2. Once the token arrives in R_2, it looks at the head of the queue in port1 (R_2), and determines which output it needs to be routed. Thus, the token follows the routing of the head-of-the-queue flits.

Once a deadlock detection token is generated and routed in the network, it can be dropped if the next routing direction is the local ejection port or if the buffer of the target router is not full. In addition, since we are assuming dimension-ordered routing (DOR), the circular dependency cannot occur between the dimensions – i.e., circular dependency cannot be formed between both the X and the Y dimension. Thus, if the routing direction of a flit changes dimensions, the token is also dropped. These conditions signify that a circular dependency can *not* occur; thus, the detection token is dropped to signify that there is currently no deadlock. If a token is dropped, it can signify no deadlock but it can also mean that deadlock has not formed yet. Thus, for each home router, if a detection token has not returned within k cycles where k is the radix of the dimension, the home router assumes a deadlock has not occurred and re-injects the priority token into the network.

B. Deadlock Recovery

If a deadlock detection token circles around and arrives at the home node, a circular dependency exists and a routing deadlock is detected. Once a deadlock has been detected, deadlock is recovered by removing a packet involved in the deadlock and injecting it into a separate, deadlock *recovery* network. For deadlock recovery, a *target* packet is selected and removed from the circular dependency. A target packet is defined as as packet that is involved in the deadlock and is at the head of the queue. We add the constraint that the flit that is at the head of the queue is a head flit to maintain a simple FIFO buffer structure.

Since the routing needs to be done at packet granularity, if the target packet at the home router is not a head flit but a body or tail flit, we drop the detection token and pass the priority token to allow another node to generate a deadlock detection token. If the home router has a head flit in the head of the queue, we redirect the packet in to the separate, deadlock-recovery network. Since a packet involved in the deadlock is moved to another network, cyclical dependency is removed and recovery from deadlock is completed.

However, what if all the other nodes are in the same situation as head of the queues involved in the deadlock are non-head flits? In the following section, we show that there is at least one head flit in the head of the queue when a circular routing deadlock occurs in torus. Thus, the deadlock recovery described above can be implemented without requiring any

[2]We hold the amount of buffers constant in our comparison and thus, double the buffer *per* VC.

[3]Multiple detection tokens can be implemented within each ring without a priority token. However, this significantly complicates the recovery process since multiple packets need to be removed from the network and injected into the recovery network.

TABLE I
DEADLOCK RECOVERY WITH TOKEN ARCHITECTURE OVERVIEW

Priority token	Detection token	Deadlock Recovery Process
Initialization One router within each ring of the 2D torus are initialized to have the priority token for each ring in the 2D torus network. Thus, total of $2k$ tokens exist in the network. *Routing* The priority token is consumed by the current router if deadlock is suspected. Otherwise, it is forwarded to the next router if no deadlock is suspected. In addition, the priority token is re-injected into the network if the deadlock detection token has not returned within k cycles	*Generation* 1) If the router has the priority token *and* 2) the buffer of the target router is full, a deadlock detection token is injected into the token network. The router that injects a detection token is identified as the *home* router. *Routing* The deadlock detection token received at the current router is dropped 1) if the dimension of the *front* flit changes where a *front* flit is defined as the flit at the head-of-the-queue that is suspected to be involved in the deadlock, 2) if the front flit is destined to the local ejection port at the current router *or* 3) if the buffer of the target router is not full. Otherwise, the token is continuously routed in the token network.	*Target Packet* If the deadlock detection token returns to the *home* router, deadlock is detected. A target packet that can be removed from the circular deadlock is identified. Target packet is defined as a packet involved in the circular deadlock and the head flit of the packet is at the head of the router input buffer. *Recovery* The target packet is removed and injected into the recovery network, and routed to its destination.

complex buffer management (such as a DAMQs) or restricting buffer usage by allow only once packet to use a buffer at a given time.

C. Characterization of Routing Deadlock in Torus Network

In this section, we show how when a circular routing deadlock occurs in a 2D torus network, there is at least one buffer involved in the deadlock which is occupied by a head flit of a packet. We first show how this will occur in a 1D torus (or a ring) topology and then, extend it to a 2D torus network.

1) Routing Deadlock in 1D Torus:

Claim. If a routing deadlock occurs in 1D torus, at least one flit in the head of the queue is a head flit.

Proof. Without losing generality, assume a unidirectional ring network as shown in Figure 4. Assume port 2 is the injection queue from the local, terminal node and port 1 is connected to a neighbor node. If all packets in the network are single-flit packets, all flits are head flits and proof is trivial. In the rest of this proof, we assume multi-flit packets. When routing deadlock occurs in this 1D ring, the following two observations can be made.

- All of the buffers involved in the circular dependency are completely full.
- Flits of different packets are never interleaved since unit of routing is a packet and body/tail flit must follow the head flit.

Based upon these observation, there are two different deadlock scenarios as shown in Figure 4.

(a) *One ore more injection port is involved in deadlock:* Circular dependency between the buffers can occur with one or more injection ports (port 2) involved in the deadlock as shown in Figure 4(a). When such routing deadlock occurs, the router where the injection port is involved in the deadlock will have another queue that is involved in the deadlock. In this example, this refers to the queue for port 1 in router R_1 (i.e., $R_1\_q_1$) .
Based on the second observation above, since packets can not be interleaved and $R_1\_q_2$ is injecting a packet for $R_2\_q_1$, the head of the queue $R_1\_q_1$ *must* be a head flit. If the head of queue $R_1\_q_1$ is not an head flit, it either means the flits of two different packets are interleaved in $R_2\_q_1$ (which cannot occur) or the destination of the packet is R_1 and the packet is currently being ejected out the network – in which case, there would be no circular deadlock.

(b) *No injection port is involved in deadlock:* As shown in Figure 4(b), assume that no injection port buffer is involved in the deadlock. If such deadlock occurs, no movement of packets can be made. Assume the last packet injected into the network to cause the deadlock was P – inject from R_1 into $R_2\_q_1$. Since deadlock occurred *after* the injection of this packet and since packet interleaving cannot occur, for similar reason as above, the head of the queue ($R_1\_q_1$) must also be an head flit.

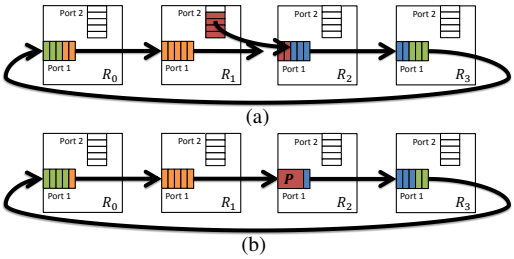

Fig. 4. Routing deadlock scenarios in a 1D torus. Different color represents different packets.

Thus, based on these descriptions, we show that when a circular routing deadlock occurs, there exists at least one queue where the head of the queue is a head flit of a packet. ■

2) Routing Deadlock in 2D Torus: In this work, we assume dimension-ordered routing (DOR) in a 2D torus network. With this DOR such as XY routing, dependencies cannot occur from packets in the Y dimension to the X dimension. Thus, all the circular dependencies occur within each dimension – either the X dimension rings or within the Y-dimension rings. Thus, the proof described in Section III-C1 can be easily generalized to 2D torus with DOR routing.

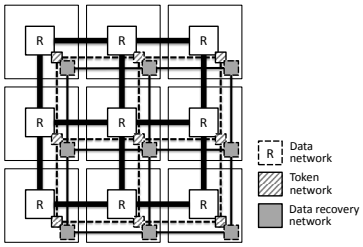

Fig. 5. Network architecture with three parallel networks including the data network, token network, and deadlock recovery network.

D. Deadlock Token Architecture

Figure 5 shows the structure of the deadlock-recovery token network consisting of three networks – the data network, token network, and deadlock recovery network. The token network is a simple, two-bit network and is connected as a ring network (or a 1D torus network) and not a 2D torus network since the token will only traverse within a given dimension. The token network is further simplified as a bufferless network since there is only one token inflight and there is no contention in the network. A narrow data recovery network is also added in parallel. Once a packet has been removed from the data network, it is injected into the recovery network and routed to its destination. Since the packet's destination can be in another dimension, the data recovery network needs to be a 2D torus topology. Routing deadlock also needs to be prevented in the data recovery network but we limit the number of data packet that is injected into data recovery network to one within each ring – thus, a simplified ring microarchitecture [14] can be extended to 2D torus data recovery network. The two networks (token network and the recovery network) add additional area but we show in Section IV-C that there is an overall reduction in area since virtual channel for routing deadlock is removed.

IV. Evaluation

We evaluate the performance of our deadlock recovery scheme with tokens (DRT) and compare with a deadlock avoidance scheme using virtual channels (VCDOR) and bubble router (BDOR) for 8-ary 2-cube torus network. We used a cycle-accurate network simulator [10] and modified it to implement the bubble router and our proposed deadlock recovery scheme. Both open-loop simulation and closed-loop simulations are used to thoroughly evaluate the different schemes.

A. Latency-throughput curve

We compare the latency-throughput of the alternative deadlock mechanisms across different synthetic traffic patterns in Figure 6, 7. All three alternatives use dimension-ordered routing (DOR) and we assume a single-cycle router delay. DRT and VCDOR use wormhole flow control while BDOR requires virtual cut-through flow control. VCDOR requires 2 VCs to avoid routing deadlock while the other two only require a single VC. To maximize the buffer utilization with 2VCs, we do not use the simple dateline approach of buffer partition but adopt a routing algorithm that balances the usage of buffers while still avoiding deadlock [10]. The total amount of buffer

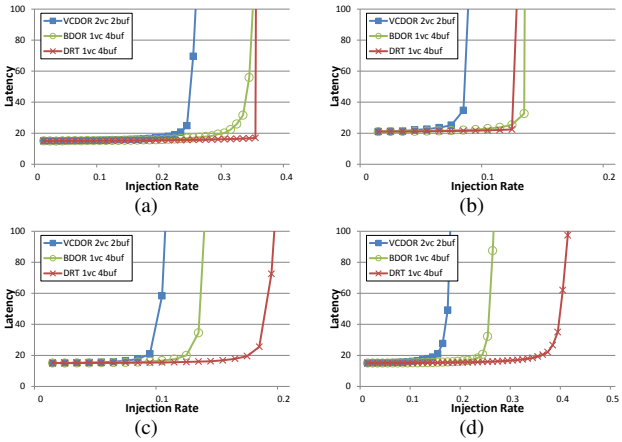

Fig. 6. Comparison for (a) uniform random, (b) tornado, (c) bit reversal, and (d) bit complement traffic pattern in the 8x8 torus network with 4 buffer and 1-flit packets.

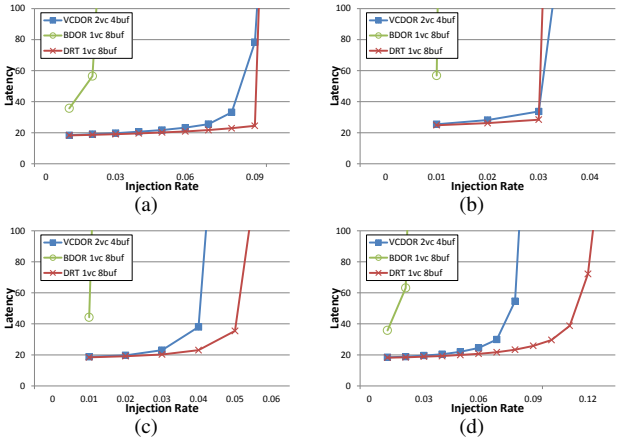

Fig. 7. Comparison for (a) uniform random, (b) tornado, (c) bit reversal, and (d) bit complement traffic pattern in the 8x8 torus network with 8 buffer and 4-flit packets. Injection rate is measured in packets per cycle for each node.

is assumed to be 8 entries *per* physical port – thus, with VCDOR, the buffers are partitioned among the two VCs. We evaluate with single-flit packets to provide a fair comparison since using a large packet size would result in unfavorable results for BDOR since it requires VCT flow control. We also show results with 4-flit packets to evaluate the impact of large packets.

Figure 6 shows the results with single-flit packets when the buffer is limited to 4 entries per router port while Figure 7 shows the result when the buffers are assumed to be 8 entries with 4-flit packets. In our simulations, no deadlock was detected with DRT for the network loads that we evaluated. For single-flit packets, BDOR and DRT outperforms VCDOR as they have deeper buffers. However, since BDOR requires twice the size of the maximum packet buffer entries before sending packets, it has poor performance with 4-flit packets. DRT is able to outperform VCDOR with 4-flit packets since it has deeper buffers as virtual channels are not needed.

B. Batch Traffic

We also compare the alternative schemes using the batch model as described in [15]. Unlike the latency-throughput curves which is an *open*-loop simulation, batch model is a

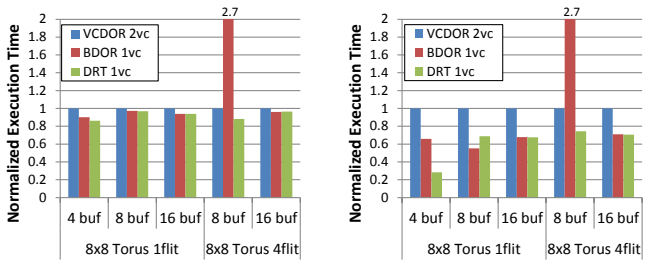

Fig. 8. Closed loop simulation results for (a) uniform random and (b) bit complement traffic pattern.

closed-loop simulation methodology that captures dependencies between messages by modeling request-reply traffic and measures the overall execution time. We use a bath size of 1000 and assume each node has a maximum outstanding request of 16. Figure 8 shows the comparison of the normalized execution time to VCDOR as the buffer size and the packet size is varied (lower value is better). With this approach, we are able to stress the network with large number of packets and evaluate the impact of deadlock recovery. For large buffers (compared with packet size), results of BDOR and DRT are very comparable but for large packets, the BDOR results in significant performance degradation. Compared to VCDOR, DRT reduces the execution time by approximately 8% and 38% on uniform random and bit complement traffic, respectively.

C. Area comparison with Deadlock Avoidance

For applications that we evaluated earlier in Section II, routing deadlock did not occur when evaluated with DRT and the performance of VCDOR is nearly identical to DRT. Thus, we compare the cost of the two approaches by comparing the amount of buffers required. For VCDOR, additional VC buffer is required in each router (bw) where b is buffer depth and w is datapath width. For DRT, the additional buffers required are from the token network (w_{tok}) and the recovery network ($b_{rec}w_{rec}$). Using $w = 128$ and $b = 4$, DRT results in over 70% reduction in additional buffers required to prevent routing deadlock and reduces on-chip network area by approximately 40%. A detailed router area comparison is shown in Table II based on the synthesis results using TSMC 45nm technology. The baseline data network is reduced by approximately 43% since the amount of buffers is reduced by half with the removal of the additional VCs. The token network adds very negligible area while the data recovery adds overhead. However, the total router area using deadlock recovery is reduce by approximately 24% compared with deadlock avoidance with VCs. We assume a baseline data network with 128 bits datapath while the data recovery network was 16 bits. If the width of the data recovery network is further reduced, the overhead of deadlock recovery network can be further reduced.

V. Conclusion

In this work, we showed how the 2D torus topology can provide a more cost-efficient topology compared with a 2D mesh network by relying on reduced hop count and reduced channel width. However, the torus topology can create routing deadlock and we proposed token-based deadlock recovery for on-chip networks that exploits the characteristics of on-chip network – abundant wires but reduced buffers. Routing deadlocks can be exactly detected with a circulating token and a separate deadlock recovery network is used to recover from deadlocks when it occurs. This work focused on routing deadlock in a torus network but deadlock can still occur because of high-level protocol. We plan to expand this work and explore how similar recovery techniques can be applied to high-level protocol deadlocks.

TABLE II
Router area comparison between Deadlock Avoidance and Deadlock Recovery.

	VCDOR	DRT
data network	54186.73	31768.75
token network	N/A	1824.32
deadlock recovery network	N/A	8274.22
Total (um^2)	54186.73	41867.31

Acknowledgment

This research was supported by the MKE (The Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program supervised by the NIPA(National IT Industry Promotion Agency)

References

[1] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proc. of the 38th conference on Design Automation (DAC)*, 2001, pp. 684–689.
[2] D. Wentzlaff, et al, "On-chip interconnection architecture of the tile processor," *IEEE Micro*, vol. 27, pp. 15–31, September 2007.
[3] M. Mirza-Aghatabar, et al.,"An empirical investigation of mesh and torus noc topologies under different routing algorithms and traffic models," *Euromicro Symp on Digital Systems Design,* , vol. 0, pp. 19–26, 2007.
[4] J. Balfour and W. J. Dally, "Design tradeoffs for tiled cmp on-chip networks," in *ICS*, Cairns, Queensland, Australia, 2006, pp. 187–198.
[5] A. Kahng, B. Li, L.-S. Peh, and K. Samadi, "Orion 2.0: A fast and accurate noc power and area model for early-stage design space exploration," in *Proceedings of Design Automation and Test in Europe (DATE)*, 2009.
[6] K. Anjan and T. Pinkston, "An efficient, fully adaptive deadlock recovery scheme: Disha," in *Proc. of the International Symposium on Computer Architecture (ISCA)*, Jun. 1995, pp. 201 – 210.
[7] M. M. K. Martin, et al., "Multifacet's general execution-driven multiprocessor simulator (gems) toolset," *SIGARCH Computer Architecture News*, vol. 33, no. 4, pp. 92–99, 2005.
[8] S. C. Woo, et al., "The SPLASH-2 programs: Characterization and methodological considerations," in *ISCA*, 1995, pp. 24–36.
[9] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The parsec benchmark suite: characterization and architectural implications," in *PACT*, 2008, pp. 72–81.
[10] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA: Morgan Kaufmann, 2004.
[11] W. Dally and C. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *Computers, IEEE Transactions on*, vol. C-36, no. 5, pp. 547 –553, May 1987.
[12] V. Puente, et al., "Adaptive bubble router: a design to improve performance in torus networks," in *Proc. of International Conference on Parallel Processing*, 1999, pp. 58 –67.
[13] S. Cho and L. Jin, "Managing Distributed, Shared L2 Caches through OS-Level Page Allocation," in *MICRO*, Orlando, FL, 2006, pp. 455–468.
[14] J. Kim and H. Kim, "Router microarchitecture and scalability of ring topology in on-chip networks," in *Proc. of the 2nd Intl Workshop on Network on Chip Architectures (NoCArc)*, New York, NY, 2009.
[15] H. Kim, et al., "On-chip network evaluation framework," in *Proc. ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, New Orleans, LA, 2010.