Transportation-Network-Inspired Network-on-Chip

Hanjoon Kim, Gwangsun Kim, Seungryoul Maeng, Hwasoo Yeo† , John Kim
Dept. of Computer Science
KAIST
{hanj, gskim, jjk12, maeng}@kaist.ac.kr
†Dept. of Civil & Environmental Engineering
KAIST
hwasoo@kaist.ac.kr

Abstract

A cost-efficient network-on-chip is needed in a scalable many-core systems. Recent multicore processors have leveraged a ring topology and hierarchical ring can increase scalability but presents different challenges, including higher hop count and global ring bottleneck. In this work, we describe a hierarchical ring topology that we refer to as a transportation-network-inspired network-on-chip (tNoC) that leverages principles from transportation network systems. In particular, we propose a novel hybrid *flow control for hierarchical ring topology to scale the topology efficiently. The flow control is hybrid in that the channels are allocated on flit granularity while the buffers are allocated on packet granularity. The hybrid flow control enables a simplified router microarchitecture (to minimize per-hop latency) as router input buffers are minimized and buffers are pushed to the edges, either at the output ports or at the hub routers that interconnect the local rings to the global ring – while still supporting virtual channels to avoid protocol deadlock. We also describe a packet-quota-system (PQS) and a separate* credit network *that provide congestion management, support prioritized arbitration in the network, and provide support for multiflit packets. A detailed evaluation of a 64-core CMP shows that the tNoC improves performance by up to 21% compared with a baseline, buffered hierarchical ring topology while reducing NoC energy by 51%.*

1. Introduction

The network-on-chip (NoC) is necessary to enable scalable manycore processors. The NoC is critical in determining overall system performance since it impacts latency and bandwidth as well as overall cost, including area and energy [34]. Some recent multicore processors, such as Intel Nehalem [36] and Larrabee [42], have adopted a ring topology for NoCs because of its simplicity. To scale the ring networks, hierarchical ring topology can be used but it is not clear how to properly scale the hierarchical ring topology efficiently. The hierarchical ring topology increases the hop count and the global ring that interconnects the local rings can become the bottleneck. In this work, we propose a novel *hybrid* packet-flit flow control for hierarchical ring topology that enables simplified router microarchitecture while providing high performance through the packet quota system (PQS) that we propose. We leverage principles from the transportation network domain and propose a transportation-network-inspired network-on-chip (tNoC).

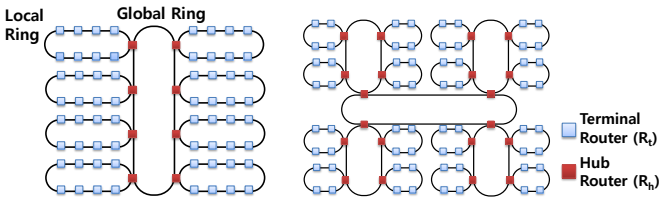
Recent urban transportation design avoids grid-like road structure, such as stop lights and traffic lights, as much as possible since they are inefficient in handling large amounts of traffic [15]. Instead, roundabouts or ring roads are more efficient in handling traffic, and these roads can be scaled using a hierarchical structure, by introducing *hubs*. A hierarchical structure minimizes the complexity at the "endpoints" but it introduces some complexity in the hubs as they allow traffic to be transferred from one region to another region. Properly managing traffic or cars is also crucial as traffic congestion results in delays and other social costs, including pollution and lost man-hours. As a result, a vehicle quota system has been proposed in Singapore to limit the number of vehicles on the roads and to manage congestion [38].

Based on these principles from urban transportation system design, we describe a hierarchical ring topology that we refer to as the transportation-network-inspired network-on-chip (tNoC). We propose a novel hybrid *packet-flit* flow control where channels are allocated on *flit* granularity while the buffers are allocated on *packet* granularity. This flow control simplifies the router microarchitecture such that the router input buffers are minimized and buffers only exist at the ejection ports and at the intermediate *hub* routers that interconnect the local rings to the global ring. The hybrid flow control leverages our proposed packet-quota-system (PQS) such that the router microarchitecture can be simplified with prioritized arbitration while supporting conventional virtual channels [5] . The topology is similar to other hierarchical ring topologies that have been previously proposed [39] consisting of local rings and a global ring but the tNoC differs in the flow control and implementation of the router. We introduce two types of routers – a *terminal* router that has minimal complexity and is connected to the endpoint terminals in the local ring and a *hub* router that interconnects the local rings to the global ring.

In particular, the novel contributions of this work include the following:

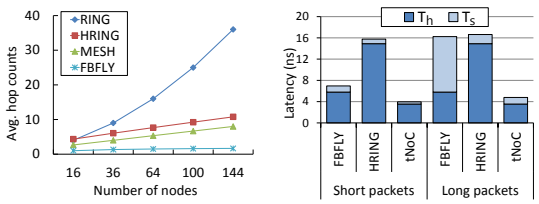
- We propose a novel *hybrid* packet-flit flow control for the hierarchical ring topology, in which channels are allocated on *flit* granularity while the buffers are allocated on *packet* granularity.
- Based on this hybrid flow control, we propose packet quota system (PQS), which enables a lightweight router microarchitecture with support for virtual channels while simplifying the switch arbitration.

978-1-4799-3097-5/14/$31.00



(a) $k = 8, n = 2$ (b) $k = 4, n = 3$

Figure 1: Different hierarchical ring organizations for 64-node network.



(a) (b)

Figure 2: (a) Average hop count and (b) average network latency comparison for alternative topologies.

- We propose a *credit network* in parallel with the data network to support hybrid flow control with PQS.
- Our detailed comparison shows the tNoC improves performance by up to 21% compared with a buffered hierarchical ring while reducing NoC energy by 51%, and compared with the flattened butterfly, improves performance (energy) by up to 7% (20%).

2. Background/Motivation

2.1. Hierarchical Ring Networks

The ring networks have been used in multicore CPUs (including Intel Nehalem [36], Larrabee [42], and IBM Cell [18]) because of their simplicity. The detailed implementation of the ring architecture is not publicly available and some aspect of our work is similar to these ring architectures (e.g., prioritized arbitration [48]); however, our main contribution of this work is in how to create a scalable hierarchical ring network with the proposed novel hybrid flow control. The ring network can be greatly simplified if we assume a single-flit packet – i.e., assume the channels are wide enough to transmit an entire cache line in a single cycle. This approach might be feasible in a small-scale network but is very inefficient since a significant number of packets tend to be short packets [30]. Thus, the wide datapath would be not utilized for most packets. In addition, scaling such wide datapath to larger network size also becomes very inefficient.

A hierarchical ring [40] consists of local rings and a global ring that interconnects the local rings together. Using a similar notation as k-ary n-cube [4], a hierarchical ring can be described with k and n, where k is the number of terminal nodes in the local ring and n is the number of levels. If $n = 1$, the topology is a k-node ring. For $n > 1$, the hierarchical ring topology consists of $n-1$ global rings. In this work, we assume a bidirectional ring architecture with minimal routing. Thus, for each ring (both local and global), it consists of two rings – a clockwise (CW) ring and a counter-clockwise (CCW) ring. We also focus on a 64-node network where we choose $k = \sqrt{N}$ using $n = 2$, where N is the number of nodes but the topology can be scaled by increasing either k or n (Scalability is discussed in Section 4.4). A 64-node hierarchical ring example with $n = 2$ and $n = 3$ is shown in Figure 1.

Scaling a ring topology with a hierarchical ring topology presents two challenges: (1) high hop count and (2) global ring performance bottleneck. In this work, we try to address the challenges of (1) by simplifying the router microarchitecture to reduce zero-load latency while for (2), we propose a hybrid flow control to minimize blocking and improve the throughput of the hierarchical ring.

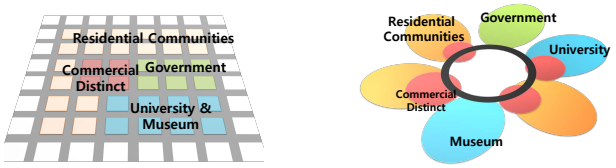
2.1.1. Network Latency The zero-load latency (T_o) [6] of a packet can be summarized as follows.

$$\begin{aligned} T_o &= T_h + T_s \\ &= Ht_r + L/b \end{aligned}$$

where T_h is the header latency, T_s is the serialization latency, H is the hop count, t_r is the per-hop router latency, L is the packet size, and b is the channel bandwidth. Alternative topologies [21, 2, 11] have been proposed to reduce network latency, by reducing the network diameter (H). However, increase in the router radix can increase router complexity (which increases t_r) and can also increase T_s since the network channels are narrower, assuming constant bisection bandwidth across the different topologies. Low-dimensional topologies, such as a ring, often provide higher channel bandwidth (lower T_s) at the cost of higher network diameter (H).

In Figure 2(a), the average hop count (H) for the different topologies are shown for uniform random traffic with minimal routing. The hierarchical ring (HRING) reduces H compared with the RING topology but it is still higher than other alternative topologies. For example, for $N = 64$, HRING reduces the hop count by 52.5% compared with the RING but it is still 43% higher than that of the 2D mesh topology and 5.1× higher than 2D flattened butterfly (FBFLY). Instead of trying to reduce H, we explore reducing t_r through a simplified router microarchitecture. The zero-load latency for a 64-node network is shown in Figure 2(a), based on t_r from Table 6. Comparing FBFLY and HRING, the zero-load latency for long packets is relatively similar but the latency of short packets is significantly higher with HRING because of higher H. However, with the same H as HRING, tNoC is able to reduce latency for both short and long packets by reducing t_r.

2.1.2. Hybrid Flow control Conventional, buffered flow control with virtual channels (VCs) [5] complicates router microarchitecture and makes it hard to reduce t_r due to complex VC allocation stage. Recently proposed bufferless flow control [8] can simplify the router microarchitecture but the high deflection routing across the global ring can reduce performance while increasing the complexity of the router. In addition,



(a) (b)

Figure 3: Block diagram of (a) grid roadway and (b) hierarchical, ring roadway.

supporting VCs to avoid protocol deadlock in bufferless flow control becomes very difficult.

In this work, we propose a *hybrid* flow control where buffers are allocated on packet granularity while channels are allocated on flit granularity. The hybrid flow control enables simplified router microarchitecture with prioritized arbitration for packets in-flight while providing support for multiflit packets and fairness. The proposed architecture is also able to provide support for virtual channels – e.g., prior work on reducing router complexity [20] simplified the router microarchitecture but did not support VCs. One key observation that we make is that ***VCs are not necessarily needed at all buffers if the packet does not hold on to the buffers indefinitely.*** With prioritized arbitration and the hybrid flow control (Section 3), packets in-flight are guaranteed to make progress towards their destinations; thus, avoid adding VCs at all buffers in the network but introduce them only at the ejection ports and hub routers.

2.2. Transportation Network

There are many similarities between roadway traffic and network traffic. In roadway traffic, controlling of high demand traffic can be achieved using traffic signal that allows a group of vehicles to pass for a direction without conflict with traffic for other directions. However, this type of control necessarily stops movement of vehicles and causes delay (Figure 3(a)). Therefore, highways including ring roads without traffic signal can process higher volume of traffic than arterials [15]. Highway is used for high volume traffic and arterials with intersections are used for local accesses.

Connecting arterials to main highway, controlling traffic from arterial to highway is important to keep the highway from congestion. In this approach, highway has higher priority than arterial traffic, because when the highway is congested, it will also block arterials soon. In grid-type arterial network with a number of intersections, the main purpose of control is to prevent ”spill over” of queue to the adjacent intersection. If a queue length increases and blocks the upstream intersection, we can sometimes meet a severe congestion state called ”gridlock” in which no vehicle can move to any direction. To prevent this disastrous situation, we can increase the number of lanes (increase bandwidth), give special lane for turning vehicles, and control the number of incoming vehicles from upstream. Ramp metering (injection control) is widely used to adjust the entering flow rate to highway using traffic signal. For the similar purpose, vehicle quota system has been proposed in Singapore to limit the amount of vehicles on the roads and manage congestion [38, 25].

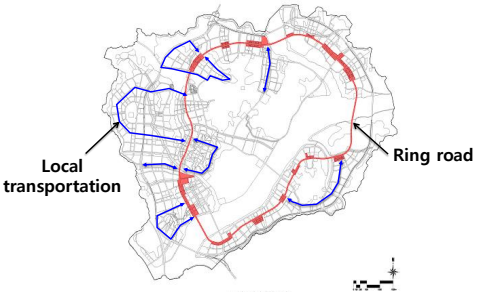


Figure 4: Roadway example of new city that is built hierarchically [43].

Pursuing efficient urban transportation system design, which minimizes total delay, strategies to mix of highway systems and arterials can be set with hierarchical network design. In hierarchical traffic network design, transfer of people or goods follows three steps: (1) aggregating traffic demand from local areas, (2) transferring the aggregated traffic using wide roads or high-speed mass transit, and (3) distribution of the aggregated traffic to destinations using local roads. Hub-and-spoke network is one example widely used for air transportation and freight transportation [10].

The traffic is significantly influenced by overall patterns of urban form. Although there is no clear metropolitan organization that is most efficient, an urban form based on “decentralized concentration” attempted in Denmark and Sweden is one promising approach [41, 33]. This approach groups infrastructures and housings to form different “concentration” of urban location and they are interconnected with other concentration. Similarly Tokyo has also announced “Circular (Ring) Megalopolis Structure” to create a *polycentric* cities connected with ring roads in 2025 [50, 45]. The Sejong city [43] in Korea, which is a new settlement of Korea government complex that is under development, is another example of the hierarchical design of transportation network (Figure 4). An inner ring road is placed to connect major destination centers, functionally grouped, and local transportation using arterials delivers people to the nearby station on the ring road. Outer ring highway is used for traffic from outbound to other cities and inbound to the city. By shaping traffic pattern and volume, the Sejong city is expected to achieve high efficiency with minimum delay for travel.

The traffic found in transportation networks is not necessarily similar to on-chip network traffic, but our goal is to leverage insights from transportation network in the design of a scalable, efficient on-chip network. Thus, we leverage various principles of the transportation network (as summarized in Table 1) and propose transportation-network-inspired network-on-chip (tNoC) in this work – a hierarchical ring topology that leverage the novel *hybrid* flow control.

Table 1: Comparison of the transportation network and the proposed tNoC.

	Transportation network	Proposed tNoC
Topology	hierarchical road structure [33, 50]	hierarchical ring topology
Router micro-architecture	hub-and-spoke network [10]	terminal and hub router with network buffering only at the hub router
Flow control	priority of highway traffic over arterial traffic [15]	prioritized arbitration for in-flight packets
Congestion management	vehicle quota system [38]	packet quota system

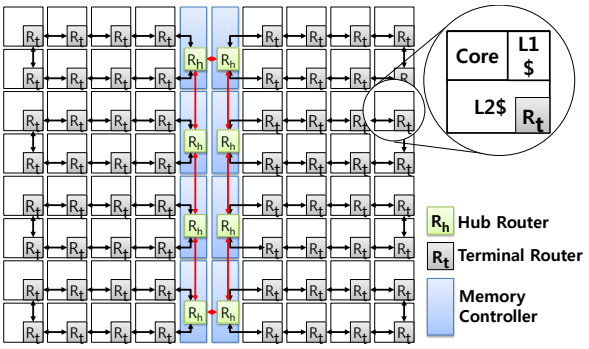


Figure 5: Floorplan diagram of 64 cores connected with hierarchical ring topology

3. Hybrid Flow Control

In this section, we first provide an overview of our proposed tNoC architecture. We then describe the *hybrid* flow control that enables simplified router microarchitecture with prioritized arbitration for a hierarchical ring topology. Leveraging the concept of vehicle quota system from transportation network, we describe the packet-quota system (PQS) for the hybrid flow control and describe how fairness can be provided.

3.1. Transportation-Network-Inspired Network-on-Chip (tNoC) Architecture

The topology of the tNoC is based on a hierarchical ring topology and consists of two types of routers – a *terminal* router (R_t) for the local ring and a *hub* router (R_h) that is used within the global ring. A high-level floorplan diagram of a 64-node network is shown in Figure 5. The ports in a router can be classified as either terminal ports, which are connected to terminal or endpoint nodes, or network ports, which are connected to other routers. A high-level block diagram of the router microarchitecture of the terminal and the hub router are shown in Figure 6. The router data path is simplified to include only a 2-to-1 mux and a pipeline register [20]. In addition, the router microarchitecture supports prioritized arbitration – i.e., packets in-flight have priority over packets that are being injected into the network. However, we introduce intermediate buffers at the hub routers for packets that need to be routed from one local ring to another local ring through the global ring. The intermediate buffers are organized per network (one set of buffers for the CW ring and another set of buffers for the CCW ring). As necessary, multiple virtual channels are provided to avoid protocol deadlock. In addition, we leverage

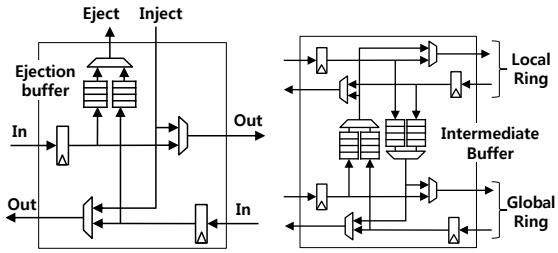


Figure 6: Microarchitecture of a (a) terminal router (R_t) and a (b) hub router (R_h).

Table 2: Flow Control Categorization.

		Channel allocation unit: Packet	Channel allocation unit: Flit
Buffer allocation unit	Packet	Packet-based flow control (store&forward, virtual cut-through)	***hybrid flow control***
	Flit	N/A [1]	Flit-based flow control (wormhole, virtual channel)

the ejection buffers that exist at the output of the ejection ports in the network. As we discuss in the following sections, the ejection buffer needs to be deeper than that of other topologies to support the tNoC and its flow control. However, since there is only two ejection buffers for each router, the total number of buffers required is smaller. The number of buffers is proportional to $O(N^{1/k}+N)$ in the tNoC and not $O(Np)$ as in conventional, input-buffered router networks, where p is the number of router ports, and N is the number of nodes in the network. The ejection buffers are included in the cost evaluation presented in Section 4.

3.2. Flow Control Description

Flow control determines how the network resources (i.e., buffers and channels) are allocated [6]. Packets are often partitioned into one or more *flits* or flow control units. Most flow controls in interconnection networks are either packet-based flow control or flit-based flow control (Table 2). In a packet-based flow control, which is commonly used in off-chip networks, both the channels and the buffers are allocated in units of packets. In comparison, flit-based flow control allocates both resources in units of flits. Examples of the flit-based flow control include wormhole or virtual channel flow controls. On-chip networks have often utilized the flit-based flow control. In this work, we propose a *hybrid* flow control where the buffers are allocated in units of packets while the channels are allocated on flit granularity. An example illustrating the different flow controls is shown in Figure 7(a) with a time diagram of a channel that is used by two packets. With the flit-based flow control, the flits from different packets can be interleaved on the channel while with packet-based flow control, packets are not interleaved. With the hybrid flow control, interleaving can still occur, similar to the flit-based flow control. However, the main difference is in how the buffer resource is allocated.

[1] Packet-channel and flit-buffer flow control is not possible since if a channel is allocated on packet granularity but there is not sufficient buffers, the flow control would not operate properly.

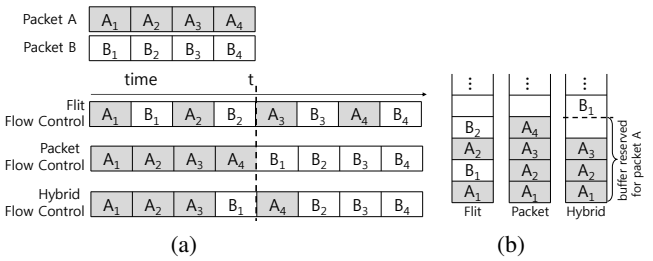


Figure 7: Different flow controls comparisons with (a) time diagram and (b) destination buffer occupancy.

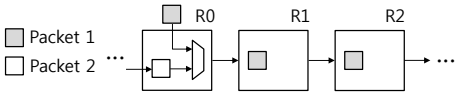


Figure 8: Block diagram of packet 2 interrupting multi-flit packet 1 with prioritized arbitration.

In our tNoC architecture with prioritized arbitration, packet-based flow control cannot be supported since channels cannot be allocated on packet granularity. An example of this problem is shown in Figure 8 – R0 is injecting packet1 into the network and if packet2 arrives, packet2 has priority. Thus, packet1 will be interrupted and the channel cannot be allocated on packet granularity. In comparison, flit-based flow control can be used but it significantly complicates the destination buffer since flits from different packets can be interleaved (Figure 7(b)) and a complex, re-order buffer is needed. To overcome this, the hybrid flow control allows flit-granularity allocation of the channels while the buffers are allocated on packet granularity.

Before any packet is injected into the network, the buffer for the entire packet needs to be allocated; thus, for a packet of L flits, the router needs to obtain L credits before the packet can be sent. Once the buffer for the entire packet has been allocated, the channel resource can be allocated on flit granularity. Because of prioritized arbitration, the injection of a multi-flit packet can be interrupted; thus, the packet will not necessarily be sent continuously. However, when the head flit arrives at the destination, it reserves the next $L-1$ slots in the buffer (Figure 7(b)) such that L flits can be continuously written in case the packet transmission is interrupted.

There are several characteristics of the proposed hybrid flow control in the tNoC that simplify the design of re-order buffers; 1) flits from a single packet will still arrive in-order but not necessarily in consecutive cycles, and 2) there will not be multiple partial packets within a re-order buffer that are sent from the same source to the same destination. Thus, to support re-ordering, each flit needs to carry the source and destination information within the local ring or the global ring (e.g., $2log(k/2)$ bits), and in the ejection buffer, multiple write pointers are needed for each source that is sending a packet to this particular destination. Similar to a FIFO, only a single read pointer is needed since the read is done in order.

3.3. Packet Quota System (PQS)

The purpose of flow control in the tNoC is for packets injected into the ring to proceed to the destination buffer without contention – where destination buffer is either the intermediate buffers at the hub router or the ejection buffers at the packet destination. This can be achieved by the following policies: (1) prioritizing in-flight packets, (2) removing contention at the ejection port, and (3) guaranteeing a buffer at the destination network interface. Although there is no contention in the network, there can be contention for the ejection ports in a bidirectional ring since packet can arrive from both the CW and CCW ring. This problem can be avoided with per-network buffering at the ejection port, as shown earlier in Figure 6(a). However, even with such buffering, backpressure can build (i.e., the terminal node can be stalled and not be able to eject a packet) – which is problematic without any buffers in the router.

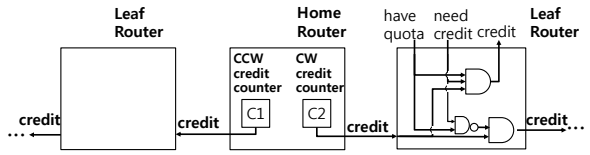


Figure 9: Block diagram of the proposed credit network.

3.3.1. Credit Network In this work, we propose packet quota system (PQS) to provide support for hybrid flow control, while minimizing network congestion. The PQS ensures that when packet arrives at the destination (or the intermediate buffer at the hub router), the packet can be buffered. The PQS is based on conventional the credit-based flow control where a *credit* signals the availability of a buffer entry. We propose adding a narrow *credit network* in parallel to the data network. The credit network is responsible for circulating the credits within each level of the hierarchical ring. The credit network that we propose is different from prior token ring [46] approaches used in LAN networks and other ring networks as tokens represented the ability to access the channel (or the medium). In comparison, the credits that we circulate in the ring represent buffer availability at the destination (either the output ejection port buffer or the intermediate buffer at the hub routers) and are decoupled from the channel usage. As long as there is buffer space available, the destination or the "home" node injects a credit into the credit network.

The source is only allowed to inject a packet after it obtains a credit for its destination (or the intermediate buffer). For a multi-flit packet, multiple credits are needed. Once a packet arrives at the destination, the particular credit can be re-injected into the credit network. By first grabbing credits, it ensures that, when a packet arrives at the destination, it will be ejected (or removed from the network) since there is buffer space available and not cause backpressure. Thus, packets that are injected into the network, in combination with priority arbitration, are guaranteed to make progress towards the destination or the hub router and avoid the need for network input buffers.

A block diagram of a credit network is shown in Figure 9, which illustrates credits flowing from a single "home" router

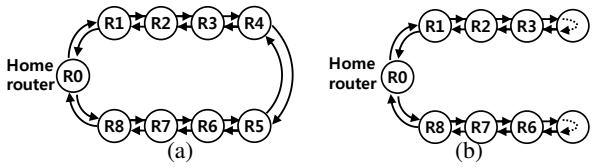


Figure 10: (a) Baseline credit network and (b) folded credit network.

to the other "leaf" routers in the ring. The home router is the router that injects the credits into the credit network while the leaf routers either consume the credit or pass the credit to the neighboring router. For a k-node ring, this simple network needs to be duplicated k times since each terminal (endpoint) or the hub router needs to be a home router in the credit network to distribute its credits. Initially, each home router initializes each credit counter (C_{cr}) to the size of the buffer. If ($C_{cr} > 0$), the home node router injects a credit into the credit network. Unused credits will return back to the home router.

3.3.2. Fairness Fairness is an issue in the credit network in a ring topology since upstream nodes can grab more credits than downstream nodes. An example of the problem is shown in Figure 10(a) where credits are injected from $R0$. The credit network consists of bidirectional rings as credits are injected in both directions. Without any support for fairness, the nodes closest to $R0$ will grab all of the credits. To illustrate such situation, a hotspot traffic where all nodes send traffic to $R0$ is simulated and the results are shown in Figure 11(a). The results show a bimodal distribution of throughput across all of the nodes. The two nodes closest to the hotspot nodes ($R1$ and $R8$) have the highest throughput, while the nodes farthest away ($R4$ and $R5$) are unable to inject any packet into the network and are starved.

To prevent this unfairness, we impose a quota on the number of credits that each node can grab. Although the credits are injected into the credit network in both directions, with minimal routing, only half of the nodes use each type of credit. That is, nodes $R1$, $R2$, $R3$ and $R4$ would only use the CCW data network to send traffic to $R0$ and only use the CW credit network. Thus, we restrict each router to grab only $1/d$ of the credits where d is the number of downstream routers, including itself, that can use the credits. For example, $R1$ is restricted to only grab 1/4 of the credits that pass $R1$, and $R2$ is restricted to grab 1/3 of the credits, while $R3$ is allowed to grab 1/2 of the credits that pass by. The unused credits flow across the ring and return to the source (e.g., $R0$). However, since the other downstream nodes ($R5, R5, R7, R8$) cannot use these credits, we *fold* the credit network and allow the credits to return in the opposite direction, as shown in Figure 10(b). This allows the unused credits to be grabbed by the nodes (e.g., $R1, R2, R3$) without any restriction and enables higher utilization of the credits.

The PQS algorithm implementation is described in Algorithm 1. The PQS and the quota at each router node is implemented using a quota counter (C_q) that describes the amount of quota that the router has, another counter (C_{grab}) that describes

Algorithm 1 PQS Implementation

init: $C_q = 0$

At each router cycle:
if current flit is a head flit **then**
  $C_{grab} = L$ % L: *packet size*
  $cost = k/2 - D + 1$ % D: *distance to the home router*
end if
if $C_q > cost$ **and** $C_{grab} > 0$ **and** credit is available **then**
  grab a credit
  $C_q = C_q - cost$
  $C_{grab} = C_{grab} - 1$
end if
if credit is available **then**
  $C_q = C_q + 1$
end if

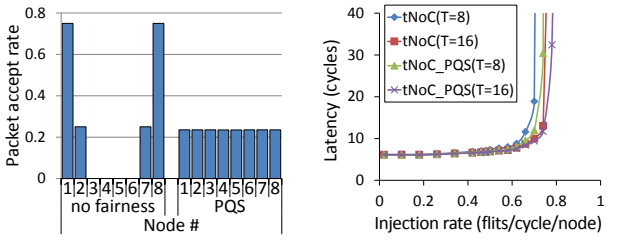


Figure 11: (a) Packet accept rate for node0 hotspot traffic and (b) latency-throughput curve of tNoC with folded credit network in UR traffic.

how many credits are currently needed, and *cost* parameter to provide fairness. The *cost* parameter is equal to d or the number of downstream routers and is essentially the *cost* that a router needs to pay to grab a token to ensure fairness. Routers closer to the home node have a higher *cost* compared with downstream routers. C_{grab} is initialized to the packet size, in terms of the number of flits, and represents the number of credits that needs to be grabbed. For each credit that passes the router node, C_q is incremented by 1, and when quota is available ($C_q > cost$), the credit is consumed.

Using this folded credit network with a quota, the fairness across all of the nodes improves significantly as shown in Figure 11(a). In addition, the impact of the folded credit network and quota is shown using a latency-throughput curve for uniform random traffic in Figure 11(b) [2]. Folding the credit network enables similar throughput with a smaller number of ejection buffers. In the rest of this work, we will assume the tNoC with the PQS, and the folded-credit network in our evaluation.

3.4. Deadlock

A deadlock can occur in tNoC if all nodes have some credits while waiting for more credits to transmit a multi-flit packet. To guarantee deadlock freedom, the destination buffers need to be deep enough to ensure such deadlock does not occur. The worst-case scenario is if all nodes are sending a

[2] The evaluation setup is described in Section 4.1. The evaluation in this section focuses on a single ring while the evaluation in Section 4 presents the hierarchical ring.

Table 3: Simulation Parameters

Common parameter	
Processor	64 in-order/out-of-order cores @ 2GHz
L1 Caches	Split I&D, 32 KB 4-way set associative, 2-cycle access time, 64B line
L2 Caches	Shared L2, 256KB per tile (total 16MB) with various sharing degree [16], 64B line
Cache coherence	Directory-based MOESI with sharing degree of 8 and 64
Memory controllers	8

NoC parameters for different flow controls	
Buffered flow control (BUFF)	speculative 2-stage virtual channel router 3 message class, 4 VCs/class, 8 flits/VC dateline routing for routing deadlock avoidance
tNoC	3 message class, 13 flits/class for interm. and ejection buffer 1 cycle router delay for terminal router 3 cycles for passing through intermediate buffer

Table 4: Workload description

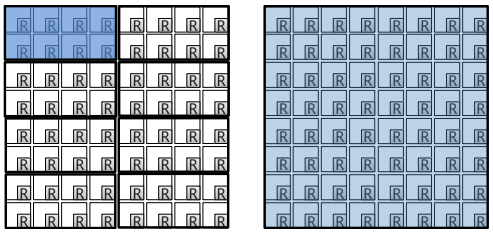
	SpecCPU2006 benchmarks
MIX 1	perlbench bzip2 gcc bwaves
MIX 2	gromacs cactusADM leslie3D namd
MIX 3	gobmk dealII soplex povray
MIX 4	calculix hmmer sjeng GemsFDTD
MIX 5	libquantum h264 tonto lbm
MIX 6	perlbench gamess gromacs gobmk
MIX 7	mcf soplex libquantum GemsFDTD
MIX 8	perlbench povray calculix tonto
MIX 9	mcf lbm milc libquantum
MIX 10	leslie3d soplex sphinx3 GemsFDTD
SPLASH2	barnes, cholesky, fft, fmm, radiosity, radix, raytrace, volrend

packet with a size of *max_packet_size* flits and each node has *max_packet_size* -1 credits. Thus, the amount of buffers needed is $k/2 * (max\_packet\_size - 1) + 1$, in units of flits, to avoid such deadlock. With the wide channels in the hierarchical ring, the maximum packet size, in terms of flits, is relatively small (4 flits in our tNoC evaluation); thus, the number of ejection buffers needed is 13 entries to avoid deadlock. To reduce the cost of ejection buffer, we leverage the network interface buffer that often exists and is often big enough to hold the maximum size packet for depacketization. Note that only two ejection buffers are need per router in tNoC, compared with 5 set of input buffers for a conventional 2D mesh router.

4. Evaluation

4.1. Methodology

In this work, we modified Booksim [17], a cycle accurate network simulator, to model our proposed tNoC network, including the hierarchical topology, both the data and the credit network, the router microarchitecture, and the flow control. Booksim was used for synthetic workload evaluation and integrated into McSimA+ [1] to evaluate a 64-node CMP architecture for multiprogrammed/multithreaded workloads. GEMS [31] was used to model the cache hierarchy. The parameters used in our simulation are summarized in Table 3. In our workload evaluation, we evaluate cache locality with hierarchical cache using the concept of the sharing degree (SD) [16]. Figure 12 illustrates the hierarchical cache with cache sharing degree of 8 and fully shared cache in a 64-node CMP. We used SpecCPU 2006 [14] benchmark suite for multiprogrammed workloads and SPLASH2 [53] benchmarks for multi-threaded workloads. For multithreaded workload results,



(a)sharing degree = 8 (SD8) (b)sharing degree = 64 (SD64)

Figure 12: Various sharing degree of shared L2 cache

Table 5: Comparison of alternative flow controls in the hierarchical ring topology.

	Buffered (BUFF)	Bufferless (HiRD [9])	Hybrid (tNoC)
Contention	buffer backpressure	deflection or drop/retransmit	minimized by PQS
Deadlock	virtual channels	injection guarantee by global coordinator, swap at global router	virtual channels at intermediate buffer
Buffers	per-hop buffering	re-order buffer, intermediate buffers	intermediate and ejection buffers
Router delay (cycles)	2	2 (global router) 1 (local router)	2 (global to local [3]) 1
Fairness	local (round-robin)	N/A	PQS
Additional overhead	N/A	global coordinator, complex coherence protocol support	credit network

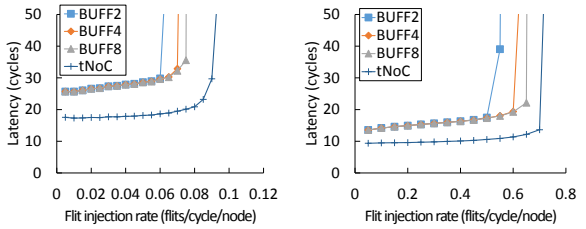
performance results represent the execution time of each workload. The multiprogrammed workload results presented in this paper are weighted speedup results from 10 mixes shown in Table 4. We choose four benchmarks from SpecCPU 2006 benchmarks and a copy of each benchmark is simulated on 16 cores. Verilog RTL was implemented for the alternative router microarchitecture and synthesized for critical path analysis as well as area/power comparison. All of the overhead, including the credit network and the hybrid flow control, are faithfully modeled and their overheads are included in the cost evaluation. McPAT [27] was used for chip-level power and area estimation.

4.2. Comparison to Alternative Flow Controls

We first compare the proposed hybrid flow control of the tNoC with baseline, buffered flow control (BUFF) on the hierarchical ring topology. Recently, bufferless flow control (with intermediate buffers) [9] [4] has been proposed and we provide a qualitative comparison in Table 5. Both the tNoC and the HiRD have intermediate buffering between the local and the global ring but main difference is that the HiRD uses deflection routing when congestion occurs while we leverage the hybrid flow control and PQS and avoid the cost of deflection. In addition, the HiRD requires a global coordinator to prevent deadlock and provide fairness in the network and conventional virtual channel semantics are not supported in the bufferless HiRD architecture. In comparison, no global structure is necessary in the tNoC and the main overhead is the additional

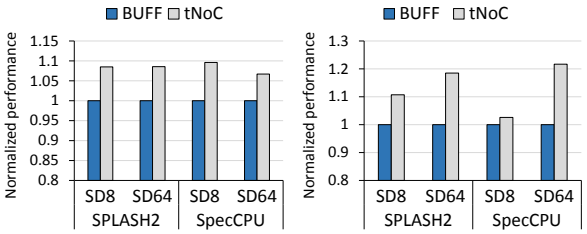
[3] Additional latency of one cycle is only added when a packet switches from a local ring to a global ring, or vice versa.

[4] This work was done concurrently with [9] but all of the details of the HiRD architecture were not clear from the technical report. As a result, providing an accurate quantitative comparison with HiRD was difficult to achieve.



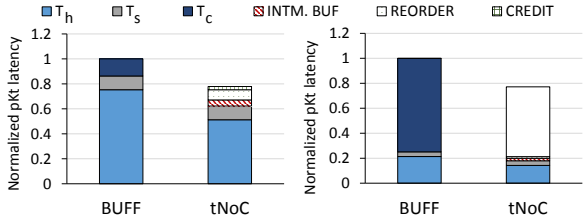
(a) UR (b) LOC

Figure 13: Latency-throughput curve comparison of alternative flow control for (a) UR, and (b) LOC=0.9 traffic.



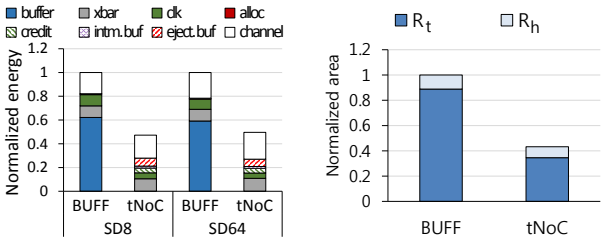
(a) in-order (b) out-of-order

Figure 14: Normalized performance results



(a) in-order (b) out-of-order

Figure 15: Normalized latency breakdown in SD64.



(a) energy (b) router area

Figure 16: Energy breakdown results and router area normalized to the BUFF. R_t: terminal router, R_h: hub router.

credit network.

4.2.1. Synthetic Workload For the synthetic workloads, we evaluate two types of traffic: uniform random (UR) and traffic with locality (LOC). UR traffic represents a fully shared cache organization, while LOC represents traffic with various degrees of locality in the cache organization. For the LOC synthetic traffic pattern, we use SD8 organization and vary the locality, where a locality of 0.9 means 90% of the traffic is sent to the shared local caches, while the remaining 10% of the traffic is sent to remote caches.

Figure 13 shows latency-throughput comparison for the different traffic patterns. Since we are comparing alternative flow controls on the same hierarchical topology, we assume single-flit packets in our initial comparison. Other results with real workloads include both short packets and long (multi-flit) packets. For the BUFF, we evaluate with 2, 4, and 8 virtual channel configurations. The performance of the BUFF increases as VCs are increased but results in higher zero-load latency (because of the higher per-hop latency) and reduced throughput because of the blocking in the network. For UR traffic, the zero-load latency for the tNoC is 32% lower than that of the BUFF. The latency for the tNoC includes the latency to grab the credits but it has minimal impact on overall performance. Since there are sufficient credits at zero-load and no additional latency is required to grab the credits. At high load, near saturation, the network data channel becomes the bottleneck and thus, the additional waiting time for credits has minimal impact on overall performance. In addition, the tNoC improves throughput by 33% and 25% on UR and LOC respectively, compared with the BUFF. Simulations show that increasing the number of VCs increased throughput but at additional cost. With the BUFF, the in-flight packets can be blocked by packet injection by terminal node or packets blocked at hub router and it results in reduction in throughput. By minimizing the network contention, the tNoC is able to provide higher throughput compared with the BUFF.

4.2.2. Real Workload Figure 14(a) shows the performance results of SPLASH2 and SpecCPU normalized to the BUFF. In SD8, the tNoC improves performance by up to 9.6%, compared with the BUFF because of lower per-hop latency. The tNoC also shows better performance in SD64, especially for memory intensive workloads such as fft, raytrace, and radix – improving performance by up to 11% (8% on average) compared to the BUFF. With out-of-order (OoO) cores, the additional network traffic generated from the OoO cores results in high contention for the BUFF (Figure 14(b)). Thus, there is an increase in the performance gap as the tNoC exceeds the performance of the BUFF by up to 21%. [5]

Figure 15 shows the latency breakdown of the different flow controls. For in-order core, both contention delay in the BUFF and latency to grab credits in the tNoC are low. The tNoC reduces the average packet latency by 22% by reducing head latency with lower per-hop latency. For OoO core, the latency to grab credits in the tNoC is lower than contention delay in the BUFF. While for the high traffic load, the BUFF blocks the packets in the router input buffers which can result in blocking of other packets that can proceed, the tNoC only blocks the packet at injection or intermediate router. As a result, the tNoC reduces the packet latency by 23% for OoO cores.

4.2.3. Overhead The costs of the BUFF and the tNoC are compared in Figure 16 in terms of network energy and area. This includes all of the overhead for the tNoC, including the ejection/intermediate buffer the credit network, and the termi-

[5] Because of page limitation, only OoO core results for the alternative flow control comparison is shown. The rest of the results are shown only for in-order cores.

Table 6: Alternative NoC Comparison parameters.

	MESH	CMESH	FBFLY	HNET	HRING	tNoC
Ports	5	8	10	12	3	3
Message class	3	3	3	3	3	3
VCs / class	4	4	4	4	4	1
Buffers depth	8	8	12	8	8	13
Critical path(ns)	0.99	1.10	1.16	1.22	0.86	0.42
Router delay	2	2	2	2	2	1 for R_t
Channel delay	1	1	0.5/tile	1	1	2 for R_h

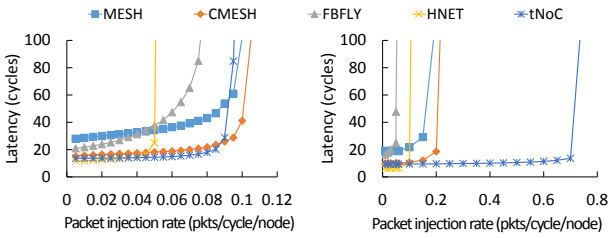
nal and the hub routers. The tNoC reduces the buffer leakage energy by reducing the buffer size and also reduces the buffer read/write energy and pipeline register access energy by removing input port buffers and reducing pipeline stages. Although the tNoC introduces additional energy consumption at credit network and additional storage, the tNoC reduces NoC energy by 52% and 50% compared with the BUFF, in SD8 and SD64 respectively. The tNoC also reduces router area with the reduction in the amount of input buffers.

4.3. Comparison to Other Topologies

We compare the tNoC to alternative topologies, including MESH, MESH with 4-way concentration (CMESH) [2], flattened butterfly with 4-way concentration (FBFLY) [21], and a hierarchical network (HNET) with an 8-way external concentration [24] as an implementation of a hybrid network of bus and mesh [7]. [6] The router designs of the different topologies were based on the Verilog model [47] which implements a two-cycle speculative virtual channel routers. The proposed tNoC was also implemented in Verilog and all overhead was modeled, including both the data network router and the credit network router, as well as the buffers for both terminal routers and hub routers. Synopsys Design Compiler with topographical mode was used to provide a better estimate of the wires, and we used TSMC 45nm technology.

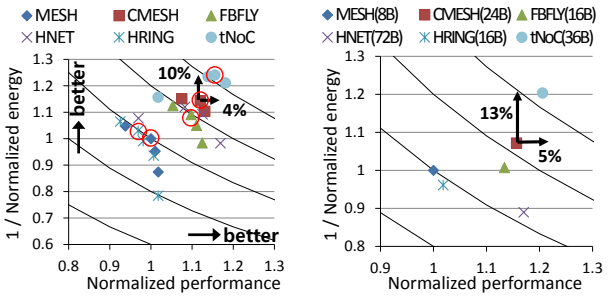
4.3.1. Timing Analysis The synthesis results of the virtual channel routers in the various topologies and the tNoC router are shown in Table 6. Except for the tNoC, all the routers are synthesized as two-stage routers. The tNoC terminal routers are single cycle per hop, while the hub routers are two cycles. The critical paths for the other routers are the virtual channel/switch allocation, and the critical paths increase with higher port count. However, the removal of the arbitration from the critical path in the tNoC significantly reduces the critical path for both terminal and hub routers, by up to 57% compared with the flattened butterfly topology router. In the tNoC router, the control (credit network) and the data network are decoupled, and this helps reduce the critical path. Synthesis results show that the tNoC critical path in the credit network is 0.42ns while the datapath of the tNoC is only 0.33ns. Even though critical path reduction enables higher network frequency for the tNoC, we conservatively assume that the tNoC runs at the same network frequency for the initial comparisons. We also presents results when applying the different network frequency for each topologies, which enables the tNoC to have more network bandwidth with higher router frequency.

[6] We do not model X-share [7] in our HNET implementation. X-share will likely improve performance at additional cost, but can also be applied to other topologies.



(a) UR (b) LOC

Figure 17: Latency-throughput comparison of different topologies with (a)UR and (b)LOC=0.9 traffic patterns.



(a) SPLASH2 (b) SpecCPU

Figure 18: Total performance vs energy scatter plot for SD8.

4.3.2. Synthetic Workload We evaluate various topology under constant bisection bandwidth. With constant bisection bandwidth, packets have different numbers of flits – 4 flits for MESH, 2 flits for CMESH, 8 flits for FBFLY, and 1 flits for HNET, HRING, and tNoC. Figure 17 shows latency throughput curves under UR and LOC traffic. For UR traffic, despite the large hop count, the tNoC provides a comparable latency because of lower router delay with simplified router pipeline stages. FBFLY shows higher zero-load latency because of high serialization latency due to its narrow channel. HNET has the lowest zero-load latency since it has the lowest average hop count (with 8-way concentration). However, throughput of HNET is very limited because of the 8-way *external* concentration [24]. For LOC traffic, the tNoC outperforms other topologies in both latency and throughput. The tNoC and HNET show a lower zero-load latency because both topology have wider channels that minimize serialization latency while header latency is relatively small because of the low average hop count from the LOC traffic. Similarly, the higher per channel bandwidth improves the throughput of the tNoC compared with the alternative topologies.

4.3.3. Real Workload Figure 18 shows the performance and energy results for alternative topologies and different channel widths. The x-axis is the overall performance, while the y-axis is the inverse of the total *chip* energy. Each curve represents the same cost per performance metric; different points on the same line represent a trade-off between performance and energy. An *ideal* network approaches the upper right corner of the plot,

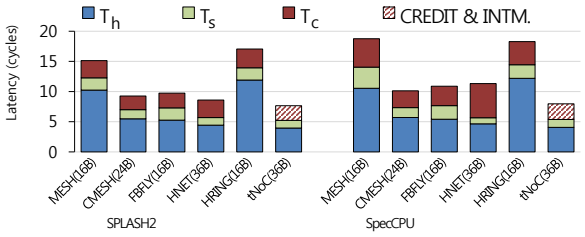


Figure 19: Packet latency breakdown in SD8.

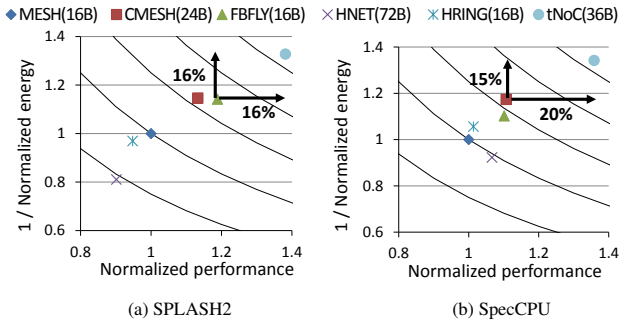
which indicates the network with the lowest overall system cost and highest system performance. In Figure 18(a), we plot the performance of each network as we vary the network channel width from 8B to 72B, as the short packets are 8B and the long packets are 72B to better understand the topology trade-off. Our results show that the optimal network designs for various topologies are different. Thus, for the rest of the comparison, instead of assuming constant bisection bandwidth, we choose the most energy-efficient channel width for each topology, as highlighted as an example with circles in the Figure 18(a).

The various NoCs are compared in Figure 18. The tNoC results in 21% performance improvement over MESH, while reducing energy by 20% for SpecCPU and resulting in 16% (24%) improvement in performance (energy) for SPLASH2. Compared to CMESH, the tNoC achieves 5% (13%) improvement for SpecCPU workloads while the tNoC achieves 7% (20%) improvement over FBFLY. Latency breakdown of the different topologies is shown in Figure 19, where latency is divided into T_h, T_s, as well as contention latency T_c (or queuing latency in the network) components. For the tNoC, additional component is the latency to acquire a credit as well as queuing latency in the intermediate buffer of the hub router. The tNoC shows lower header latency (T_h) by reducing hop delay and lower serialization latency with wider channels. As described earlier, the channel width of other topologies can also be increased but the design points used in the comparisons are based on the most efficient design for each topology. On average, the tNoC reduces latency by 53% compared with the mesh and by 24% compared with the FBFLY.

Figure 20 is the same comparison as Figure 18 but different NoC frequencies are used based on synthesis results for each topology. By leveraging the higher frequency of the tNoC, the efficiency of the tNoC is further improved – energy efficiency and performance by 38% (34%) and 20% (16%), respectively, for multiprogrammed (multithreaded) workloads, compared with the most efficient, alternative topology.

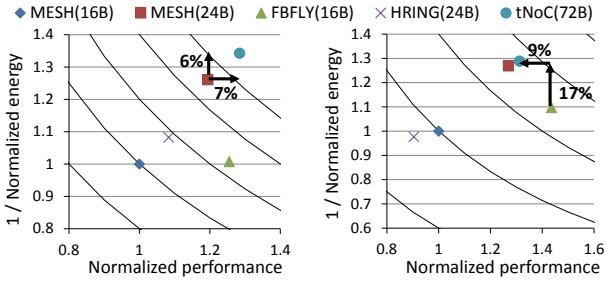
4.4. Scalability

We evaluate the scalability of the tNoC by comparing the performance in a 256-node network with SD8 and SD256 organization (Figure 21). For the tNoC, we still maintain a two-level hierarchical ring ($k = 16, n = 2$) in our comparison. With SD8, the tNoC is able to achieve both improvement in performance and reduction in energy, compared with the most efficient alternative topology (CMESH). For a fully shared or-



(a) SPLASH2 (b) SpecCPU

Figure 20: Energy vs. performance results for SD8 with different NoC frequencies based on synthesis timing results.



(a) SD8 (b) SD256

Figure 21: tNoC Comparison for N=256 system.

ganization (SD256), the most efficient topology is the FBFLY and there is a performance-cost trade-off – the tNoC results in 9% reduction in performance but 17% reduction in energy. Even though the per-hop delay is still lower with tNoC, the much higher hop count from global traffic in $N = 256$ network results in performance degradation. However, complexity of the routers, especially the high-radix routers in FBFLY increases the energy consumed in the routers and thus, the tNoC results in improvement in energy consumption. Thus, the tNoC and the hybrid flow control proposed in this work are scalable, compared with alternatives. In addition, to further scale the network, concentration can also be leveraged as well – i.e., instead of having a single core connected to a terminal router, multiple cores can share a single terminal router.

4.5. Worst-case Traffic Pattern Analysis

The hierarchical nature of the proposed tNoC can result in an adversarial traffic pattern when all traffic is sent from the local ring through the global ring to another local ring. An adversarial traffic pattern in a ring topology is the tornado traffic [6] and the worst-case traffic pattern [7] for a hierarchical ring is where all traffic pass through the global ring (i.e., all traffic generated from one local ring is send to a different local ring) and the global traffic pattern results in tornado traffic. As shown in Figure 22, the throughput of the tNoC suffers compared with other topologies, resulting in a reduction of throughput by 44% compared with CMESH. However, it is very unlikely that such an adversarial traffic pattern will occur

[7] Note that the worst-case traffic pattern is not necessarily the worst-case for other topologies.

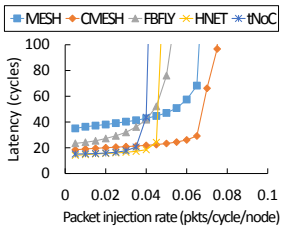


Figure 22: Worst-case traffic pattern for the tNoC comparison.

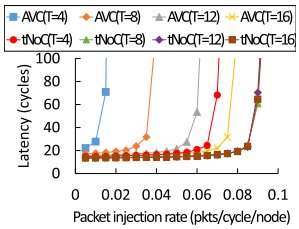


Figure 23: Comparison with atomic VC allocation.

in real workloads. In addition, a recent study on scale-out processors [29] showed that hierarchical and modular memory hierarchy makes optimal use of die area. Recent processors including SPARC T4 [44] and AMD Bulldozer [3] also have hierarchical memory hierarchy. However, if the system needs to support such adversarial traffic pattern as described above, the tNoC can be enhanced to provide more bandwidth by either increasing the channel bandwidth or duplicating the global network.

4.6. Atomic VC Allocation Comparison

Atomic virtual channel allocation (AVC) [6] has some similarity with the proposed hybrid flow control, since with the AVC the head flit of a packet is not buffered behind the tail flit of another packet in the same virtual channel buffer. In effect, buffers are implicitly allocated on packet granularity, even if flit-based flow control is used. However, one challenge with the AVC is the amount of buffers required to provide a high throughput since the buffer depth needs to be equal to the size of long packets and the buffer utilization will be low for short (single-flit) packets. As a result, much more buffering is necessary with the AVC. Figure 23 compares the AVC with the tNoC as the amount of endpoint buffers (T) is varied with single-flit packets to understand the limitation of the AVC. Since we assume large packets are 4 flits, the AVC needs to dedicate 4 entries to each "VC" and thus, for $AVC(T = 4)$, there is significant loss in throughput. To achieve similar throughput as $tNoC(T = 4)$, the AVC needs approximately $3\times$ to $4\times$ the amount of buffers to achieve similar throughput as the tNoC.

5. Related Work

Hierarchical Topologies: The hierarchical ring has been proposed in the multi-processor domain to extend ring topology to scale to a large number of nodes [40]. Our hierarchical ring topology is similar to prior topologies, but it differs in how the routers and the flow control are implemented. Udipi et al. [49] proposed a segmented bus-based hierarchical bus with a bloom filter to effectively track data presence and restrict bus broadcasts to a subset of segments. However, a hierarchical bus differs as the arbitration is done by a centralized arbiter.

Router Microarchitecture: The proposed router microarchitecture is similar to previously proposed router microarchitecture [20, 22] – simplifying the router microarchitectures and minimizing the number of buffers while introducing *intermediate* buffers. However, prior work had no support for VCs and credit-based flow control impacted the router cycle time. In this work, the proposed router microarchitecture provides support for VCs while removing credit-based flow control. Bufferless NoC [8] has been proposed which removes router input buffers. By properly accounting for the cost of buffers and channels, [32] showed minimal benefits of a bufferless NoC compared with a buffered NoC. Another problem with a bufferless NoC is avoiding protocol deadlock since virtual channels are not supported. NoC-out [28] also proposed a simplified router microarchitecture by exploiting the communication characteristics of the scale-out workloads. However, the communication pattern for the workloads that we evaluate is different from the scale-out workloads.

Flow Control: The token-ring has been widely used in local area networks. Recently, the token-ring has been proposed for channel arbitration in the on-chip nanophotonics [51, 35] but tokens are primarily used for channel arbitration to guarantee a slot on the data channel. The folded-credit network proposed in this work is similar to the two-pass token network proposed for on-chip nanophotonics [35]. However, the key difference is that in the first pass in [35], each token was dedicated for a particular node which can increase the amount of time it takes to grab a token whereas in our approach, such dedication is not required but is determined by the predetermined quota. In addition, prior works on nanophotonic assume single-flit packets which simplify the arbitration.

Token flow control (TFC) [23] is similar to our proposed credit-token network as the *guaranteed* token represents buffer space availability, similar to our credit token. However, the main difference is that the tokens are used to generate lookahead signals that enable bypassing of intermediate routers, while our proposed token is leveraged to minimize arbitration in the network. In addition, TFC is implemented on top of a baseline buffered flow control and requires large buffers and complex management of the different tokens while our mechanism is implemented on top of simplified router microarchitecture. Flit-reservation flow control [37] shares similarity as channel and buffer resources are reserved by a control flit. However, we decouple the allocation of channel and buffer and avoid the complexity of a reservation table. Quota-based schemes have been previously proposed to provide quality-of-service (QoS) [26, 12]. However, they rely on complex mechanisms such as global barrier network and large source buffers or managing per-flow states by each router to provide QoS. The purpose of our PQS is not necessarily to support QoS but provide support for the proposed hybrid flow control. It remains to be seen if PQS can be extended to provide QoS support. Source throttling [52, 19] has been proposed to prevent buffer congestion but these have been proposed on top of conventional, buffered flow control or bufferless networks. Grot et al. [13] also proposed a hybrid flow control in their kilo-NoC but their hybrid flow control combined

elastic-buffered flow control with minimal VC flow control.

6. Conclusion

In this work, we proposed a hybrid flow control for a hierarchical ring topology where the channels are allocated on flit granularity while buffers are allocated on packet granularity. The hybrid approach enables the terminal routers to be simplified with minimal buffers while supporting priority-based arbitration that simplifies the router allocation in a hierarchical ring topology. Borrowing the idea of the vehicle quota system from transportation networks, we propose packet quota system (PQS) that minimizes network congestion. Although we minimize buffers, we provide support for conventional virtual channels – which simplifies the network support for protocol deadlock. Detailed evaluations with various workloads show that the tNoC is able to improve performance by up to 21% compared with the baseline hierarchical ring topology while reducing NoC energy by 51%. We also compare with the alternative topologies, and our results show the tNoC is able to improve performance by up to 7% compared with the flattened butterfly topology while reducing chip energy by up to 20%.

Acknowledgements

We would like to the thank the anonymous reviewers for their comments. This work was supported in part by the IT R&D program of MSIP/KEIT (10041313, UX-oriented Mobile SW Platform) and in part by Basic Science Research Program through the NRF of Korea funded by the MSIP (NRF-2011-0015039).

References

[1] J. H. Ahn *et al.*, "McSimA+: a manycore simulator with application-level+ simulation and detailed microarchitecture modeling," in *ISPASS*, 2013.

[2] J. Balfour *et al.*, "Design tradeoffs for tiled cmp on-chip networks," in *ICS*, 2006.

[3] M. Butler *et al.*, "Bulldozer: An approach to multithreaded compute performance," *Micro, IEEE*, vol. 31, no. 2, pp. 6–15, 2011.

[4] W. J. Dally, "Performance analysis of k-ary n-cube interconnection networks," *IEEE Trans. Comput.*, vol. 39, no. 6, pp. 775–785, 1990.

[5] W. J. Dally *et al.*, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Trans. Comput.*, vol. 36, pp. 547–553, May 1987.

[6] W. J. Dally *et al.*, *Principles and Practices of Interconnection Networks*. SF, CA: Morgan Kaufmann, 2004.

[7] R. Das *et al.*, "Design and evaluation of a hierarchical on-chip interconnect for next-generation CMPs," in *HPCA*, 2009.

[8] C. Fallin *et al.*, "CHIPPER: A low-complexity bufferless deflection router," in *HPCA*, 2011.

[9] C. Fallin *et al.*, "HiRD: a low-complexity, energy-efficient hierarchical ring interconnect," Tech. Rep., 2012.

[10] A. Ghobrial, "Analysis of the air network structure: the hubbing phenomenon," Ph.D. dissertation, University of California, Berkeley, 1983.

[11] B. Grot *et al.*, "Express cube topologies for on-chip interconnects," in *HPCA*, 2009.

[12] B. Grot *et al.*, "Preemptive virtual clock: a flexible, efficient, and cost-effective qos scheme for networks-on-chip," in *MICRO*, 2009.

[13] B. Grot *et al.*, "Kilo-NOC: A heterogeneous network-on-chip architecture for scalability and service guarantees," in *ISCA*, 2011.

[14] J. L. Henning, "SPEC CPU2006 benchmark descriptions," *SIGARCH Comput. Archit. News*, vol. 34, no. 4, pp. 1–17, Sep. 2006.

[15] W. Homburger *et al.*, *Fundamentals of traffic engineering 16th edition*. Institute of Transportation Studies, UC Berkeley, 2007.

[16] J. Huh *et al.*, "A NUCA substrate for flexible CMP cache sharing," in *ICS*, 2005.

[17] N. Jiang *et al.*, "A detailed and flexible cycle-accurate Network-on-Chip simulator," in *ISPASS*, 2013.

[18] J. A. Kahle *et al.*, "Introduction to the cell multiprocessor," *IBM J. Res. Dev.*, vol. 49, no. 4/5, pp. 589–604, 2005.

[19] H. Kim *et al.*, "Clumsy flow control for high-throughput bufferless on-chip networks," *Computer Architecture Letters*, vol. 12, no. 2, pp. 47–50, 2013.

[20] J. Kim, "Low-cost router microarchitecture for on-chip networks," in *MICRO*, 2009.

[21] J. Kim *et al.*, "Flattened butterfly topology for on-chip networks," in *MICRO*, 2007.

[22] J. Kim *et al.*, "Router microarchitecture and scalability of ring topology in on-chip networks," in *NoCArc*, 2009.

[23] A. Kumar *et al.*, "Token flow control," in *MICRO*, 2008.

[24] P. Kumar *et al.*, "Exploring concentration and channel slicing in on-chip network router," in *NOCS*, 2009.

[25] S. Lam and T. Toan, "Land transport policy and public transport in singapore," *Transportation*, vol. 33, pp. 171–188, 2006.

[26] J. W. Lee *et al.*, "Globally-synchronized frames for guaranteed quality-of-service in on-chip networks," in *ISCA*, 2008.

[27] S. Li *et al.*, "McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *MICRO*, 2009.

[28] P. Lotfi-Kamran *et al.*, "NOC-Out: Microarchitecting a scale-out processor," in *MICRO*, 2012.

[29] P. Lotfi-Kamran *et al.*, "Scale-out processors," in *ISCA*, 2012.

[30] S. Ma *et al.*, "Whole packet forwarding: Efficient design of fully adaptive routing algorithms for networks-on-chip," in *HPCA*, 2012.

[31] 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.

[32] G. Michelogiannakis *et al.*, "Evaluating bufferless flow control for on-chip networks," in *NOCS*, 2010.

[33] P. Næss, "Urban form and travel behavior: Experience from a nordic context," *Journal of Transport and Land Use*, vol. 5, 2012.

[34] J. D. Owens *et al.*, "Research challenges for on-chip interconnection networks," *IEEE Micro*, vol. 27, no. 5, pp. 96–108, 2007.

[35] Y. Pan *et al.*, "FlexiShare: Channel sharing for an energy-efficient nanophotonic crossbar," in *HPCA*, 2010.

[36] C. Park *et al.*, "A 1.2 TB/s on-chip ring interconnect for 45nm 8-core enterprise Xeon processor," in *ISSCC*, 2010.

[37] L.-S. Peh *et al.*, "Flit-reservation flow control," in *HPCA*, 2000.

[38] S.-Y. Phang *et al.*, "Singapore's experience with car quotas : Issues and policy processes," *Transport Policy*, vol. 3, no. 4, pp. 145–153, October 1996.

[39] G. Ravindran *et al.*, "A performance comparison of hierarchical ring- and mesh- connected multiprocessor networks," in *HPCA*, 1997.

[40] G. Ravindran *et al.*, "On topology and bisection bandwidth of hierarchical-ring networks for shared-memory multiprocessors." in *HIPC*, 1998.

[41] P. Rickaby, "Six settlement patterns compared," *Environment and Planning B: Planning and Design*, vol. 14, pp. 193–223, 1987.

[42] L. Seiler *et al.*, "Larrabee: a many-core x86 architecture for visual computing," in *SIGGRAPH*, 2008.

[43] Sejong city(2011), "http://www.macc.go.kr."

[44] M. Shah *et al.*, "Sparc T4: A dynamically threaded server-on-a-chip," *Micro, IEEE*, vol. 32, no. 2, pp. 8–19, 2012.

[45] A. Sorensen., "Subcentres and Satellite Cities: Tokyo's 20th Century Experience of Planned Polycentrism," *International Planning Studies*, vol. 6, no. 1, pp. 9–32, feb 2001.

[46] W. Stallings, "Local networks," *ACM Comput. Surv.*, vol. 16, pp. 3–41, March 1984.

[47] Stanford NoC projects, "http://nocs.stanford.edu."

[48] T.Fossum, "Keynote : On-die interconnect and other challenges for chip-level multi-processing," in *Hot Interconnects*, Stanford, CA, 2007.

[49] A. Udipi *et al.*, "Towards scalable, energy-efficient, bus-based on-chip networks," in *HPCA*, 2010.

[50] Urban Development in Tokyo, "http://www.toshiseibi.metro.tokyo.jp/pamphlet/pdf/udt2011english.pdf," 2011.

[51] D. Vantrease *et al.*, "Light speed arbitration and flow control for nanophotonic interconnects," in *MICRO*, 2009.

[52] I. Walter *et al.*, "Access regulation to hot-modules in wormhole nocs," in *NOCS*, 2007.

[53] S. C. Woo *et al.*, "The SPLASH-2 programs: characterization and methodological considerations," in *ISCA*, 1995.