Enabling Quality-of-Service in Nanophotonic Network-on-Chip

Jin Ouyang* and Yuan Xie
Computer Science and Engineering, the Pennsylvania State University
*jouyang@cse.psu.edu

***Abstract*—With the recent development in silicon photonics, researchers have developed optical network-on-chip (NoC) architectures that achieve both low latency and low power, which are beneficial for future large scale chip-multiprocessors (CMPs). However, none of the existing optical NoC architectures has quality-of-service (QoS) support, which is a desired feature of an efficient interconnection network. QoS support provides contending flows with differentiated bandwidths according to their priorities (or weights), which is crucial to account for application-specific communication patterns and provides bandwidth guarantees for real-time applications.**

In this paper, we propose a quality-of-service framework for optical network-on-chip based on *frame-based arbitration*. We show that the proposed approach achieves excellent differentiated bandwidth allocation with only simple hardware additions and low performance overheads. To the best of our knowledge, this is the first work that provides QoS support for optical network-on-chip.

I. Introduction

The diminishing return from instruction-level parallelism (ILP) drives the shift to many-core processors that exploit task-level parallelism (TLP). However, with the increasing number of cores, the burgeoning on-chip bandwidth requirement is becoming difficult to satisfy. To address this problem, researchers have developed various on-chip interconnection networks (or network-on-chip, NoC). Nevertheless, existing analyses show that interconnects account for 30%–50% of total chip power [1], [2], which set researchers on the quest for power-efficient on-chip interconnects.

A. Optical Network-on-Chip

The emerging nanophotonic technology enables on-chip optical interconnects that are faster and less power-consuming than electrical wires [3]. Therefore it has been leveraged to build various on-chip networks. Kirman *et al.* [4] propose to use optical components to build on-chip buses, which however has limited scalability when the network size increases. A major branch of optical network researches are focused on *direct network topologies*, such as meshes and tori [5]–[7]. These researches migrate the topologies widely used in electrical networks to optical networks. A common feature of these networks is that the optical network is overlaid over an electrical network with the same topology. The optical network uses circuit-switching to avoid intermediate buffering. Circuit set-up is done by sending set-up packets in the packet-switching electrical network. Another major branch of researches are focused on *token-ring based networks* such as *Corona* [8] and its extension [9], *Firefly* [10] and *MPNoC* [11]. All these networks implement all-optical arbitration and flow control mechanisms to exploit the full strength of nanophotonic technology.

While direct network topologies win popularity in electrical networks, optical direct networks have several severe problems. First, the circuit set-up latency is long since set-up packets are delivered in the electrical network. To amortize this overhead and achieve good utilization, the length of data packets needs to be over 2KB [5]. However, in CMPs, the majority of traffics are coherency data which are typically 10s of bytes long but require very short delivery latency. Second, the hop-by-hop electrical network consumes significant power, which offsets the power efficiency of the optical network. Third, direct networks inevitably introduce large number of waveguide crossings which severely affects the integrity of optical signal [12]. In contrast, *token-ring based optical networks* do not have overheads of a second electrical network, and there are few or no waveguide crossings even for a large scale network. Therefore token-ring based optical networks are likely to outperform direct optical networks in future many-core CMPs. However, token-ring based networks suffer from severe fairness issues since aggressive sources can easily starve other sources on the same ring.

B. Quality-of-Service

Another important aspect of on-chip networks is allocating bandwidths to contending flows with different bandwidth requirements. Quality-of-service (QoS) encapsulates mechanisms that service contending flows according to their respective importance and requirements. This is important to account for application-specific communication patterns and improve system throughput. It is also critical to provide bandwidth guarantees to real-time applications. A number of researchers have proposed QoS-enabled NoC architectures for electrical network [13]–[16]. Frame-based arbitration is used in Lee *et al.*'s [15] and Grot *et al.*'s [16] work to achieve differentiated bandwidth allocation with a simple mechanism that incurrs low overheads. However, according to our knowledge, there is no existing work to provide QoS support in nanophotonic NoCs.

In this paper, we propose a QoS-enabled optical network-on-chip that uses frame-based arbitration to provide differentiated bandwidth allocation. Due to the simplicity of proposed architectural innovations, the QoS-enabled optical NoC architecture incurs low hardware and performance overheads compared to a baseline optical NoC, while achieving excellent fairness in bandwidth allocation. Due to its low overheads, we believe the proposed QoS-enabled architecture is suitable to be implemented in future nanophotonic network-on-chips.

II. Nanophotonic Interconnect Components

A nanophotonic interconnect consists of a laser source (typically located off-chip), waveguides carrying light injected by the laser source, and micro-rings to modulate and detect optical signals. A conceptual view of a nanophotonic link is shown in Figure 1. With dense-wavelength-division-multiplexing (DWDM), up to 128 wavelengths can be generated and carried by the waveguides [3], [11], which increases the bandwidth density to over 320Gb/s/um. Micro-rings can be electrically tuned into resonance (the "on" state) and remove light from waveguides; or out of resonance (the "off" state) and let light pass by unaffected. This mechanism is leveraged to modulate light into on-off signals. Doping Ge in a micro-ring turns it into a optical detector. When the doped micro-ring is turned on, it removes light from the waveguide and converts optical signals to electrical ones. Detecting is destructive which means if a detector is turned on then

978-1-4244-7514-8/11/$26.00 ©2011 IEEE

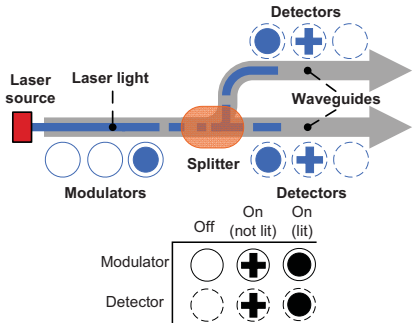


Fig. 1: A conceptual nanophotonic link, which consists of a laser source, waveguides, and micro-rings as modulators or detectors.

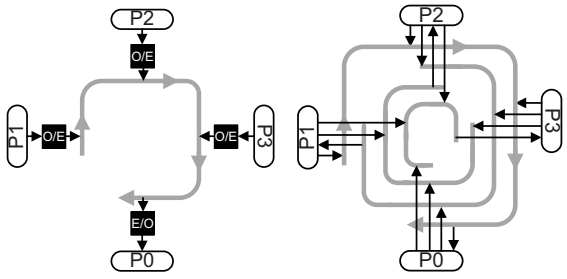


Fig. 2: (a) A single MWSR ring. Black boxes refer to O/E and E/O converters. P0-P4 are processors that send and receive signals. (b) A connected 4-node network with 4 MWSR rings. For clarity, O/E and E/O converters are omitted.

downstream detectors will not be able to detect light. A splitter is used to direct a fraction of light power to another waveguide without affecting modulated signals. It is needed to implement broadcast in nanophotonic links. The pitches of nanophotonic components are small, on the order of 5–10um.

III. Optical NoC Architecture

In this section, we first describe the baseline optical NoC architecture without QoS support, followed by our proposed QoS enhancements to the baseline architecture.

A. Baseline Architecture

Our baseline architecture is derived from Corona [8], [9], which we consider to be more promising than other alternatives for CMPs, as discussed in Section I.

MWSR Token Rings. The on-chip network in Corona consists of multiple token rings, each of which is a Multiple Write, Single Read (MWSR) ring . On a MWSR ring, there is a single destination, and multiple sources that send data to the destination. Light flows unidirectionally in the ring, passing each source and finally terminated by the destination. The sources and the destination modulate and senses the light with micro-rings in the same way as described in Section II. Figure 2a shows a single MWSR ring with three sources (P1, P2, P3) and one destination (P0). For a connected n-node network, n MWSR rings are needed. Figure 2b shows an example of a connected 4-node network. In Corona's terminology, the destination node terminating a MWSR ring is called the *home node* of that ring. For example, P0 is the home node of the MWSR ring in Figure 2a.

Arbitration and Flow Control. Arbitration is needed to avoid data collision on the MWSR rings. A token-based arbitration mechanism called *token slot* is proposed in [9] that achieves all-optical arbitration and up to 100% bandwidth utilization. For a single MWSR ring, the home node emits a one-bit token at every clock cycle. The requesters with data to send try to seize the tokens. Capturing a token grants the requester with the right to send one phit of data (1 phit=1 flit in Corona). As long as the delay between capturing a token and sending the data is a constant, the data sent from different requesters will not collide. To do flow control, the home node can simply stop emitting tokens when there are no sufficient buffers considering the round-trip latency on the ring.

Overall Architecture. Corona assumes a CMP with 256 cores aggregated into 64 clusters. Each cluster contains 4 cores and one optical router. 64 MWSR rings form a wide optical waveguide bundle that visits every cluster. The approximate floorplan is shown in Figure 3. Corona uses DWDM with 64 wavelengths per waveguide. With 64 wavelengths, a single waveguide is used to carry arbitration tokens of all MWSR rings. The length of the rings is estimated to

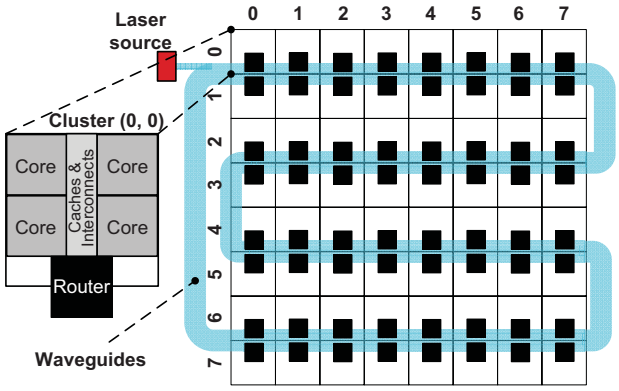
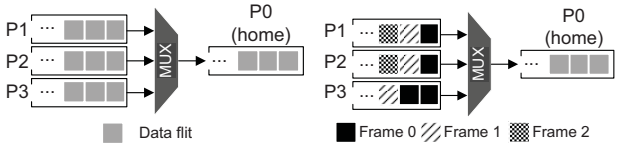


Fig. 3: Floorplan of the 256-core CMP interconnected by optical token-rings (Corona [8]).

be 160mm [8], [9], [11], leading to an 8-cycle round-trip latency with a 5GHz clock. In addition, Corona uses virtual output queues (VOQs) [9], which means each source queue is decomposed into multiple virtual queues. Each virtual queue is dedicated to buffering flits for a different destination. VOQs prevents flits destined for different nodes from blocking each other and improves performance. It also allows us to provide QoS support with frame-based arbitration as discussed in the next subsection.

B. Optical NoC with QoS Support

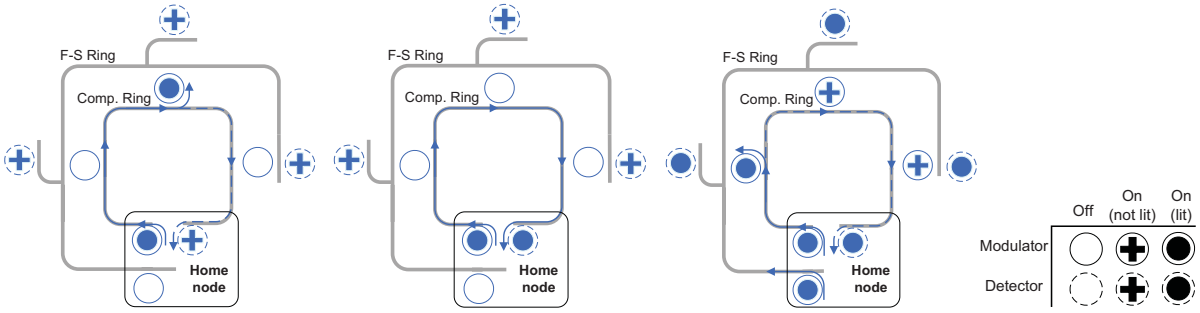
With token slot arbitration, Corona suffers from a severe fairness issue: since the tokens flow unidirectionally, upstream requesters have absolutely higher priority than downstream requesters in seizing tokens; in the worst case, one requester can starve all other requesters on the same MWSR ring (*e.g.,* P1 may starve P2 and P3 in Figure 2a). In the original work, the authors proposed *fair token slot* to address this problem. While this approach tries to provide equal bandwidths to contending requesters, it does not provide bandwidth differentiation and is ignorant of weights of different requesters. Hence, fair token



(a) Abstract model for a 4-node MWSR ring

(b) Frame-based arbitration for a 4-node MWSR ring

Fig. 4: Bandwidth allocation models.



(a) One of the sources still has flits of the head frame and diverts light from the completion ring

(b) The source finishes sending head frame flits and turns off its micro-ring. The home node detects light from the completion ring

(c) The home node then broadcast one-bit frame-switching signal to trigger frame-switching.

Fig. 5: All-optical frame-switching mechanism. The inner ring is the completion ring while the outer is the frame-switching ring.

slot can hardly be considered as providing QoS support.

In order to develop a QoS mechanism for optical NoC, we first derive an abstract model that is focused on bandwidth allocation in token-based MWSR rings. Figure 4a shows this model for the single 4-node MWSR ring in Figure 2a. Each source is represented solely by a source queue, and the destination is represented by a destination queue. To study bandwidth allocation, we ignore all processing and propagation latencies for the moment and view the role of the shared MWSR ring as a multiplexer which picks a flit at each cycle from one source queue and pushes it to the destination queue. For example, when all sources are backlogged, with the default token slot mechanism the multiplexer always picks flits from the requester with the highest priority (P1 in this example). On the other hand, with fair token slot it picks flits from source queues in a round-robin order, achieving equal bandwidth allocation. Note that since VOQs are used, each MWSR ring works independently. This allows us to study the model of a single ring and the conclusion about bandwidth allocation can be readily applied to multiple rings.

To enable QoS in optical NoC, we exploit frame-based arbitration [15], [16] to provide differentiated bandwidth allocation according to the weights of requesters. We first describe the principles of frame-based arbitration, followed by its application to optical token-rings.

1) Principles of Frame-Based Arbitration: In frame-based arbitration, a *frame* is a batch of flits that is delivered in entirety. The *frame size* (F) is defined as the maximum number of flits a frame can contain. A share R_{Pi} from a frame is allocated to each source Pi ($\sum R_{Pi} \leq F$). Multiple frames are allowed to exist simultaneously in a network. For clarity, we associate a frame number (F_N) with each frame, starting from frame 0.

When the network is initially powered up, all queues are empty and no frame contains any flits. When a source, say Pi, pushes its first flit into the source queue, it marks it as belonging to frame 0. Using the same terminology with [15], we call this action *injecting* a flit into frame 0. Further incoming flits of Pi are also injected into frame 0 until the total number of flits injected into frame 0 reaches R_{Pi}. After that, Pi updates its injection frame to be frame 1 and fills it with further incoming flits, until the total number of injected flits reaches R_{Pi} again. This process is repeated forever when the network is operating. That is, each source node injects its share into frames with increasing frame number. Figure 4b shows an example for the 4-node MWSR ring, where flits filled with different patterns belong to different frames. In this example, R_{P1}=1, R_{P2}=1, R_{P3}=2, and F=4.

Frame-based arbitration requires flits belonging to a same frame to be delivered together. In addition, frames are delivered according to their numbers. For example, in Figure 4b, flits belonging to the first frame (black boxes) are delivered before any other flits. In general, flits belonging to frame $(i + 1)$ cannot be delivered until frame i is drained. This method essentially introduces a strict ordering between frames, but not among flits within a same frame. We call the frame currently being drained (the current oldest frame) the *head* frame, and flits of the head frame "ready" flits, while all other flits "unready" flits.

2) Implement Frame-Based Arbitration in Optical NoC: In practice, implementing frame-based arbitration only requires each node to track the status of one frame—the head frame. A source node needs to be throttled if its flits belonging to the head frame have all been delivered, but there are flits belonging to the head frame left in some other source queues (that is, the head frame is not drained). Only when the current head frame is drained, source nodes can generate a new head frame by admitting flits to the new head frame in compliance with R_{Pi}s. We call this action *frame-switching*, which is the key to implementing frame-based arbitration.

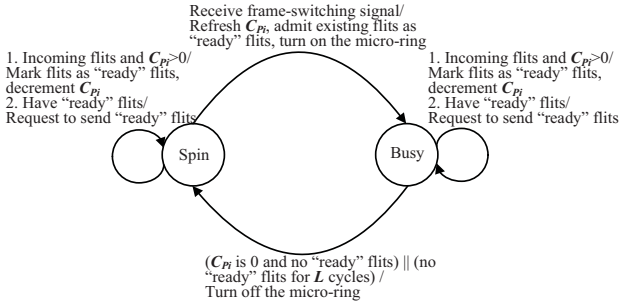
We leverage optical interconnect to provide an efficient and all-optical frame-switching mechanism, which is suitable to be used with our baseline architecture. We propose to introduce two additional rings:

- The **Completion Ring** is used to gather local status of each source node on a MWSR ring. On the completion ring, each source node has a micro-ring which is tuned into resonance when it still has flits of the head frame. The home node injects a continuous light into the ring that passes each source node. Therefore the completion ring essentially implements a "NOR" function. When there is at least one flit of the head frame remaining in the network, the source node owning that flit will remove light from the completion ring (Figure 5a). The home node has a detector at the end of the ring, and only detects light when the current head frame is drained (Figure 5b).
- The **Frame-Switching Ring** is used to broadcast the global status to each source node on the MWSR ring. When the home node detects light on the completion ring, it will send one-bit *frame-switching* signal on the frame-switching ring, which reaches the detectors of all source nodes on this ring (Figure 5c). Receiving a *frame-switching* signal triggers the source nodes to perform frame-switching the operation and tune their micro-rings on the completion ring into resonance again.

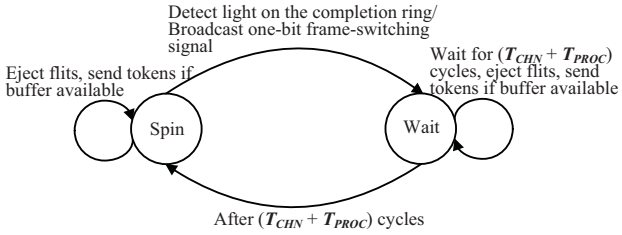
Note that it is possible to use a single broadcast ring to realize the functions of both the completion ring and the frame-switching ring. However, this means this single broadcast ring needs to pass each node twice and its length is doubled. According to our analysis, to account for exponential signal attenuation, this long broadcast

TABLE I: Summary of notations

Var. Name	Range	Description
F	$1 .. \infty$	frame size in number of flits
R_{Pi}	$1 .. F$	node Pi's share in a frame
C_{Pi}	$1 .. R_{Pi}$	node Pi's remaining share in the current head frame
T_{CHN}	$1 .. \infty$	round-trip latency (8 cycles for Corona)
T_{PROC}	$1 .. \infty$	node's processing delay of frame -switching
W	$0 .. (T_{CHN} + T_{PROC})$	wait cycle counter for the home node
L	$1 .. \infty$	idle cycle threshold to trigger early frame-switching
Q_{Pi}	$0 .. L$	idle cycle counter for node Pi



(a) Source node behavior



(b) Home node behavior

Fig. 6: State machines for source and home nodes.

ring requires a laser power of 32 Watts, while the total laser power needed by both completion and frame-switching rings is only 0.85 Watts. Therefore we choose the double-ring architecture with a small hardware overhead.

Node Behavior. The source node and the home node are both controlled by state machines, which are described by Figure 6 and Procedure 1 and 2. The notations used in the figure and pseudo-codes are summarized in Table I. Note that the 3 loops in Procedure 1 runs independently and concurrently.

The source node has two states: *spin* and *busy* (Figure 6a). Initially each source node is put into the busy state, during which the micro-ring on the completion ring is turned on. Upon the generation of a new flit, it is marked as a "ready" flit if there are still available share in the current head frame ($C_{Pi} > 0$). The source node goes to the spin state when its share is used up ($C_{Pi} = 0$) and all "ready" flits have been delivered. In the spin state, the source node is throttled and turns off its micro-ring on the completion ring. The transition from the spin state to the busy state is triggered by the frame-switching signal. During this transition, the source node refreshes its share ($C_{Pi} \leftarrow R_{Pi}$), admits existing flits to the head frame (line 31–34 of Procedure 1), and turns on its micro-ring on the completion ring.

The home node also has two states: *spin* and *wait*. It is initialized to be in the spin state. Upon detecting light on the completion ring, it emits the frame-switching signal and goes to the wait state. In the wait state, the home node waits for ($T_{CHN} + T_{PROC}$) cycles—the time for all source nodes to receive frame-switching signal and complete frame-switching. This is necessary to prevent the home node from sending multiple frame-switching signals during the on-going frame-switching process.

Procedure 1 Behavior of source node Pi.

```
1: Initialize: C_Pi ← R_Pi, Q_Pi ← 0
2: Initialize: Initial state ← busy
3: Initialize: Turn on the micro-ring on the completion ring
4: loop // Flit generation loop
5:    AT EACH FLIT GENERATION EVENT:
6:    Push the flit into the source queue
7:    if C_Pi > 0 then
8:       Mark the flit as "ready"
9:       C_Pi ← C_Pi − 1
10:   end if
11: end loop
12: loop // Flit delivery loop
13:   if there exists a "ready" flit then
14:      Request to send the "ready" flit
15:      Q_Pi ← 0
16:   else
17:      Q_Pi ← Q_Pi + 1
18:   end if
19: end loop
20: loop // Node state manipulation loop
21:   WHILE IN THE busy STATE:
22:   if (C_Pi = 0 and no "ready" flit exists) or Q_Pi = L then
23:      Go to the spin state
24:      Turn off the micro-ring on the completion ring
25:   end if
26:   WHILE IN THE spin STATE:
27:   if detect the frame-switching signal then
28:      Go to the busy state
29:      Turn on the micro-ring on the completion ring
30:      C_Pi ← R_Pi
31:      repeat
32:         Mark the first existing "unready" flit as "ready"
33:         C_Pi ← C_Pi − 1
34:      until C_Pi = 0 or all existing flits are "ready"
35:   end if
36: end loop
```

Procedure 2 Behavior of the home node.

```
1: Initialize: Initial state ← spin
2: loop // Node state manipulation loop
3:    WHILE IN THE spin STATE:
4:    if detect light on the completion ring then
5:       Go to the wait state
6:       Emit frame-switching signal
7:       W ← T_CHN + T_PROC
8:    end if
9:    WHILE IN THE wait STATE:
10:   W ← W − 1
11:   if W = 0 then
12:      Go to the spin state
13:   end if
14: end loop
```

Early Frame-Switching. While providing bandwidth differentiation, frame-based arbitration has the risk to under-utilize the network capacity. Consider the extreme case that one source node does not generate flits at all. This "silent" node blocks frame-switching and other nodes are throttled even if they have data to send. In this extreme case network utilization is reduced to 0. Similarly, network utilization is low when low-injection-rate nodes prevent high-injection-rate nodes from exploiting unused bandwidth.

To address this problem we propose to do early frame-switching:

frame-switching is triggered even when some nodes have not used up their shares, based on the prediction that those nodes are unlikely to generate flits for a prolonged period. In this work we use the length of current idle period (Q_{Pi}) as a predictor. When the current idle period is longer than some threshold (L), a source node also goes from the busy state to the spin state. The threshold L can be determined statically or adaptively according to the network status. Currently we statically set $L = 2$, an empirical value found in experiment to provide good utilization and bandwidth allocation. Adaptive methods to determine L is the subject of our future work. Note that with this modification, a source node may go to the spin state even if its share is not used up; therefore it may still admit and send "ready" flits even in the spin state.

Multiple MWSR Rings. The discussion up to this point assumes a single MWSR ring. Since with VOQs flits destined for different nodes will not affect each other, the above mechanisms can be straightforwardly extended to multiple MWSR rings. Each MWSR ring simply implements the aforementioned frame-based arbitration independently. With 64 wavelengths per waveguide, two waveguides can implement all completion and frame-switching rings in a 64-node network. A source node Pi can have different R_{Pi}s on different rings. If we denote node Pi's share on ring Hi as R_{Pi}^{Hi}, it is still required that $\sum R_{Pi}^{Hi} \leq F$, $\forall Pi$ on ring Hi.

Bandwidth Allocation. With frame-based arbitration, the bandwidth allocated to a source node Pi on ring Hi is R_{Pi}^{Hi}/F of the maximum bandwidth of ring Hi.

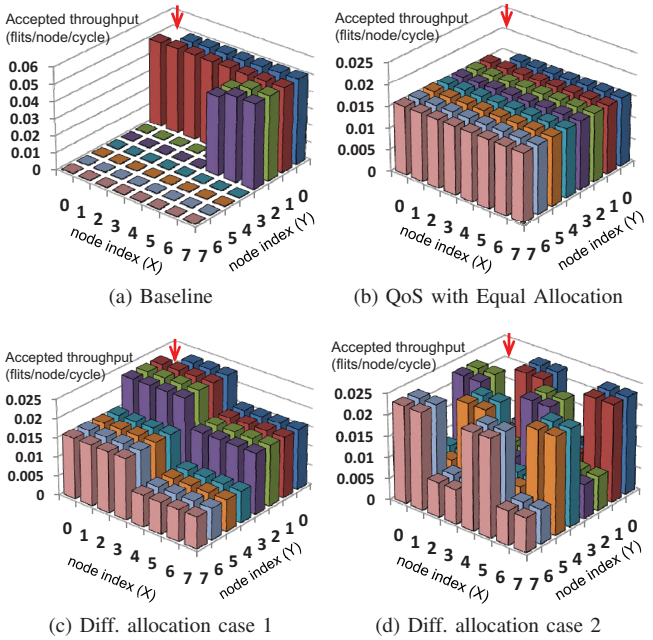
IV. Experiment

We evaluate and compare the original Corona and our enhanced QoS-enabled network using an cycle-accurate NoC simulator. Each simulation is run until results are stabilized. We model a 64-node token-ring based network as shown in Figure 3. The baseline configuration is exactly the same as Corona, while for the QoS-enabled network enhancements discussed in Section III-B are added. The default frame size (F) is set to 128 flits. Synthetic traffic patterns are used to exercise both networks. In addition, considering the features of the baseline architecture, the synthetic traffics are divided into two classes:

- For *uniform* and *hotspot* traffics, multiple sources may send data to one destination. In the token-ring based network, these sources will contend for bandwidth of a single or multiple MWSR rings.
- The other class of traffics are those based-on permutation patterns: *transpose*, *bit-reversal*, *perfect-shuffle*, *complement*, etc. In all these traffics, data are sent only between pairs of nodes; and a given destination only receives data from one source. For the baseline architecture, this means on each MWSR ring, there is only one active source, and no bandwidth contention exists.

A. Fairness and Performance

We use the hotspot traffic to evaluate the fairnesses of the baseline and the QoS-enabled architectures. In this experiment, we pick node $(0, 0)$ as the hotspot, and each other node sends data to this node at a rate of 0.05 flits/cycle. The resulted aggregate offered load exceeds the network capacity. The result for the baseline architecture is shown in Figure 7a. As can be seen, in this case the upstream nodes exhaust all available bandwidth, while downstream nodes only drips traffics. Figure 7b shows the result for the QoS-enabled network with equal allocation, and Figure 7c and 7d show the results with differentiated allocation. In Figure 7c the 64-node network is divided into 4 quadrants and differentiated services are provided to different quadrants. In Figure 7d, the network is partitioned in to 2×2 node groups and bandwidth is allocated in a checkerboard pattern. We see that in all cases, the accepted throughput of each source is compliant with allocation.



(a) Baseline (b) QoS with Equal Allocation

(c) Diff. allocation case 1 (d) Diff. allocation case 2

Fig. 7: Accepted throughput for (a) the baseline architecture and for (b–d) the QoS-enabled architecture with equal and differentiated allocations. The red arrow indicates the hotspot node.

We next examine the performance with synthetic traffics. For uniform and hotspot, we assign $\lfloor F/64 \rfloor$ of a frame to each source. For the second class of traffics, we allocate the whole frame to each source since there is no contention. The results are shown in Figure 8, where offered load and throughput are normalized to network capacity. Due to space limit, we only show results for transpose from traffics in the second class, since their performances share identical traits. As can be seen, the flit latency of QoS-enabled network is almost identical to the baseline architecture. However, beyond the saturation point the flit latency of QoS-enabled network rises more rapidly, especially in hotspot and transpose. The maximum accepted throughputs of QoS-enabled network are 17% and 7% lower than the baseline for uniform and hotspot respectively. This is due to the overheads of frame-switching latency and idle cycles of source nodes. On the other hand, the throughput overhead of transpose is negligible. This is because the share of each source node is a whole frame, and those extra latencies are amortized by the large share (128 flits).

A large frame size can potentially improve throughputs by amortizing overheads of frame-switching. This is reflected by Figure 9a, where the throughputs of the QoS-enabled network improve with increasing frame size. The improvement saturates beyond the frame size of 512 flits. With a frame size of 512 flits, the throughput reductions of uniform and hotspot are only 10% and 2% respectively.

B. Energy and Hardware Overheads

Energy consumption of an optical network consists of both static and dynamic components. The static component includes external laser power and ring heating power. The dynamic power is expended by ring modulation and electrical back-end components including pre-driver, analog receiver, sampling circuits, and amplifier. We use data from [11] and [3] for 22nm node to calculate the overall energy consumption. The results for uniform traffic with different offered loads are plotted in Figure 9b. First, we observe that for both architectures the static energy dominates overall energy when the offered load is low. With offered load increasing, the contribution of static energy is amortized by the increased data rate. On the other hand, the

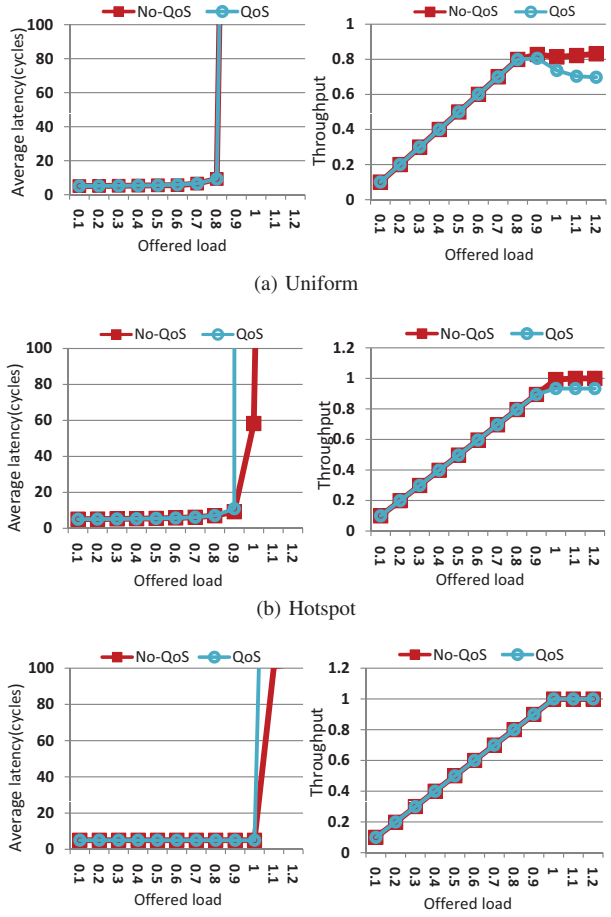


Fig. 8: Average flit latency (left), achieved throughput (right) for (a) uniform , (b) hotspot, and (c) transpose.

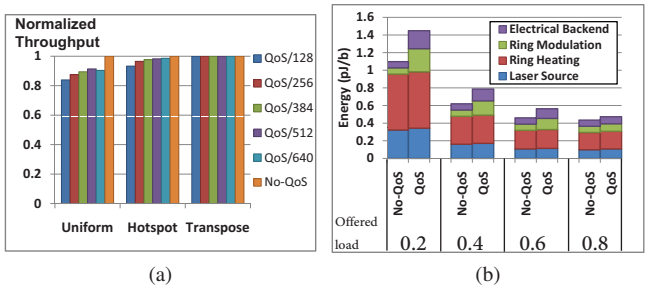


Fig. 9: (a) Maximum throughputs of the QoS-enabled network normalized to the baseline, with different frame sizes (128–640 flits). (b) Energy decomposition with different offered loads.

dynamic energy is almost constant for the baseline architecture. Second, we observe that QoS-enabled network incurs little static energy overhead since only few additional optical components are used. On the other hand, the dynamic energy overhead is significant, due to the activities associated with frame-switching. As expected, the dynamic energy overhead is more prominent with low offered loads when source nodes become idle more frequently; and it is much lower with high offered loads. Overall, the total energy overhead ranges from 32% at 0.2 load rate to 8% at 0.8 load rate. It is possible to adaptively adjust L to control the frame-switching rate, which can reduce the overhead at low loads. This is the subject of our future work.

The optical component budget is shown in Table II, where the overheads introduced by the QoS enhancements are in bold. Frame-based arbitration only introduces 0.8% overheads for both waveguides and micro-rings. Due to the small sizes of optical components, the resulted area overhead is also likely to be small.

TABLE II: Summary of notations

Photonic Subsystem	Waveguides	Micro-rings
Data MWSR rings	256	1024K
Arbitration ring	1	4K
Comp. ring	**1**	**4K**
Frame-switch. ring	**1**	**4K**
Total	259	1036K
QoS overhead	0.8%	0.8%

V. Conclusion

Emerging nanophotonic technology has the potential to boost performance and reduce power of future many-core CMPs. In this work we propose a nanophotonic network-on-chip architecture with quality-of-service support, which to our best knowledge is the first work on this topic. Our frame-based QoS enhancements achieve excellent bandwidth allocation, while only introducing simple extra hardware and small performance overheads. Based on this initial work, we are currently working on adaptively adjusting the idle cycle threshold (L) to further reduce energy overheads.

Acknowledgement

We are thankful to NSF 0905365, 0903432, 0702617, 0643902 and SRC grants for supporting this work, and also to Guangyu Sun for his valuable comments on the paper.

References

[1] L. Cheng *et al.*, "Interconnect-aware coherence protocols for chip multiprocessors," *SIGARCH Comput. Archit. News*, vol. 34, no. 2, pp. 339–351, 2006.
[2] N. Magen *et al.*, "Interconnect-power dissipation in a microprocessor," in *Proc. of SLIP '04*, 2004, pp. 7–13.
[3] C. Batten *et al.*, "Building manycore processor-to-DRAM networks with monolithic silicon photonics," in *Proc. of HOTI '08*, 2008, pp. 21–30.
[4] N. Kirman *et al.*, "Leveraging optical technology in future bus-based chip multiprocessors," in *Proc. of MICRO 39*, 2006, pp. 492–503.
[5] A. Shacham *et al.*, "On the design of a photonic network-on-chip," in *Proc. of NOCS '07*, 2007, pp. 53–64.
[6] M. Petracca *et al.*, "Design exploration of optical interconnection networks for chip multiprocessors," in *Proc. of HOTI '08*, 2008, pp. 31–40.
[7] H. Gu *et al.*, "Odor: a microresonator-based high-performance low-cost router for optical networks-on-chip," in *Proc. of CODES+ISSS '08*, 2008, pp. 203–208.
[8] D. Vantrease *et al.*, "Corona: System implications of emerging nanophotonic technology," *SIGARCH Comput. Archit. News*, vol. 36, no. 3, pp. 153–164, 2008.
[9] D. Vantrease *et al.*, "Light speed arbitration and flow control for nanophotonic interconnects," in *Proc. of MICRO 42*, 2009, pp. 304–315.
[10] Y. Pan *et al.*, "Firefly: illuminating future network-on-chip with nanophotonics," *SIGARCH Comput. Archit. News*, vol. 37, no. 3, pp. 429–440, 2009.
[11] X. Zhang *et al.*, "A multilayer nanophotonic interconnection network for on-chip many-core communications," in *Proc. of DAC '10*, 2010, pp. 156–161.
[12] Y. Xie *et al.*, "Crosstalk noise and bit error rate analysis for optical network-on-chip," in *Proc. of DAC '10*, 2010, pp. 657–660.
[13] K. Goossens *et al.*, "Æthereal network on chip: concepts, architectures, and implementations," *IEEE Trans. Design and Test*, vol. 22, no. 5, pp. 414–421, 2005.
[14] T. Bjerregaard *et al.*, "A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip," in *Proc. of DATE '05*, 2005, pp. 1226–1231.
[15] J. W. Lee *et al.*, "Globally-synchronized frames for guaranteed quality-of-service in on-chip networks," in *Proc. of ISCA 35*, 2008, pp. 89–100.
[16] B. Grot *et al.*, "Preemptive virtual clock: a flexible, efficient, and cost-effective QoS scheme for networks-on-chip," in *Proc. of MICRO 42*, 2009, pp. 268–279.