High Radix On-Chip Networks at Incremental Reconfiguration Cost

Ritesh Parikh, Animesh Jain and Valeria Bertacco

Department of Computer Science and Engineering, University of Michigan
{parikh, anijain, valeria}@umich.edu

ABSTRACT

Networks-on-chip (NoCs) have become increasingly widespread due to the extensive integration of components in modern chip multi-processor (CMP) and SoC designs. A fundamental tradeoff in NoC design is the radix of its constituent routers. While high radix routers enable a richly connected and low diameter network, low radix routers provide simple and low power designs.

In this work, we present HiROIC, a solution that provides high-radix like performance at a cost similar to a low-radix network. HiROIC leverages the irregularity in runtime communication patterns to provide short low-latency paths between frequently communicating nodes, while infrequently communicating pairs take longer paths. HiROIC deploys more links at a router than ports, binding ports to links at runtime depending on the traffic demands. Our experiments on a 64-node CMP, running multi-programmed workloads, show that HiROIC reduces average network latency by 19% over an area- and power- comparable mesh NoC.

1. INTRODUCTION

As a result of increasing integration of components into CMP and SoC architectures, NoCs have become the dominant choice for on-chip interconnects, due to the highly concurrent communication paths and better scalability they provide. Moreover, to keep up with the communication demands of the cores/IPs on-chip, NoCs are increasingly incorporating bulky and power-hungry resources, required to meet target latency and bandwidth goals.

One such design decision is the radix of the routers in a topology, that is, the number of I/O ports that a router provides to connect links to adjacent router. High-radix routers (>5 ports) provide low-diameter topologies, enabling packets to traverse fewer routers to reach their destination. However, the router components, such as crossbar and allocators, grow quadratically in area with the radix of the router. In addition, high-radix routers lead to increased signal propagation latencies and slower operating frequencies. Low-radix routers (*e.g.*, mesh), on the other hand, consume less power and can be clocked at a substantially higher rate [15]. Unfortunately, low-radix topologies often lead to large network diameters and large hop counts. Using low-radix routers particularly hurts performance when applications do not have sufficient memory-level parallelism (MLP) to hide the higher latency.

With HiROIC (for **Hi**gh **R**adix **O**n-chip Networks at **I**ncremental re-**C**onfiguration Cost), we want to provide the best of both classes of topologies: low and high radix. HiROIC provides an effective network diameter at par with high-radix topologies, while only utilizing resources comparable to low-radix routers. HiROIC exploits the non-uniformity of communication patterns to provide short, low latency paths only between heavily communicating nodes, while it forces the low traffic source-destination pairs to use longer paths. Thus, on average, HiROIC provides a small hop count for packets traversing the network. With the increasing integration of application-specific components, the location and quantity of heavily used routing paths is likely to be highly unbalanced both across and within applications. We therefore envision great potential for the deployment of HiROIC in upcoming CMP and SoC designs.

HiROIC uses routing and topology reconfiguration to optimize for high-volume source-destination pairs. At the heart of HiROIC is the concept of port-link decoupling: network links are connected to routers' ports only at runtime, and the binding is modified dynamically based on the changes in traffic patterns imposed by the application. Our NoC design includes low-radix routers, but abundant links, as in a high-radix topology, so to *potentially* provide short paths between any source-destination pair. In HiROIC, computation is partitioned into epochs of execution, with port-link binding fixed during each epoch. At the end of an epoch, the mapping is re-evaluated based on the observed traffic patterns, and modified if there is space for improvements. While HiROIC's wiring overhead is greater than conventional topologies (*e.g.*, meshes), we observe that wires never constitute a timing bottleneck in conventional router pipelines [9]. In addition, unused wires during an epoch can be power-gated after the port-to-link binding is complete. Note that, in most NoCs, each router has one *local* port (sometimes more) connecting to the processing node(s). Since this connection is essential to provide connectivity, we always dedicate one router port for binding to the *local* node link. Thus, to simplify the discussion, in the remaining of this paper we exclude the port connected to the local link when reporting the radix of a router.

In summary, our contributions are:

- A novel router architecture to mimic high-radix router's behavior, while consuming resources comparable to a low-radix router.
- A reconfiguration algorithm that predicts an application's upcoming communication needs and periodically adapts the network topology to provide short paths between heavily communicating source-destination pairs.

In our evaluation with non-uniform multiprogrammed workloads from the SPEC CPU 2006 suite, HiROIC's 64-node layout reduces average network latency by 21%, compared to a baseline mesh.

2. RELATED WORK

Much of the research targeting performance improvements in NoC designs has focused on: i) reducing the number of pipeline stages within the router [12, 15], and ii) increasing the clock frequency of the router's operation [2]. Our work leverages an orthogonal approach to improve performance – decreasing the average packet's hop count. Previous works that leveraged application-driven configuration for the NoC targeted the design phase of the NoC, with no ability to reconfigure at runtime. These solutions would characterize all applications that were expected to run on the system and then, based on the analysis, optimize the design's: i) topology [17], ii) routing [13], iii) buffer sizing [8], etc. In contrast, HiROIC adapts dynamically to changing application patterns and reconfigures the topology at runtime.

Runtime reconfiguration solutions have also been proposed to optimize either power [11] or performance [6, 5]. All these techniques provide valuable benefits and can be deployed concurrently with HiROIC, which targets dynamic topology reconfiguration to attain additional gains. There is also a body of work related to runtime topology reconfiguration, whose goal, however, has been reliability. In these solutions, for instance Ariadne [1] and uDIREC [14], the topology changes upon a runtime fault, and the authors propose reconfiguration techniques to update the routing function so to route around the fault. In comparison, HiROIC's key contribution is a dynamic topology reconfiguration solution for performance enhancement.

3. METHODOLOGY

In conventional networks, router ports have fixed one-to-one mapping with links. In contrast, we propose to provide more links than those available in low-radix topologies, eliminating the traditional fixed connections. At runtime, router ports are bound to a subset of the available links, based on the application's communication demands. The internal micro-architecture of the router is not modified, with the exception of the necessary updates to the routing tables, based on the selected configuration. The left side of Figure 1 shows the schematic of a four-port HiROIC-enabled router with the opportunity to bind to eight links. The *glue logic* in the hashed area comprises multiplexers to complete the bindings. The right side of the figure presents two examples of port-to-link bindings for the router.

HiROIC's Execution Flow. In HiROIC, the NoC's execution is partitioned into epochs. During each epoch, our traffic-statistics collection framework monitors the density of communication between all source-destination pairs. Our goal is to identify the pairs that transfer the majority of the traffic so as to minimize their hop count. In the rest of this

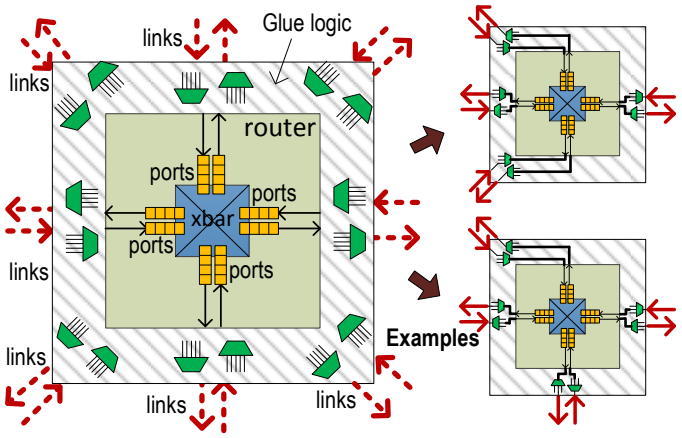

Figure 1: **Port-link decoupling.** The router in the figure can connect up to four router ports by selecting among eight links. Depending on the application's traffic demands, it can adopt different port-binding configurations, by simply assigning select signals of the multiplexers. The right part of the figure shows two such possible bindings.

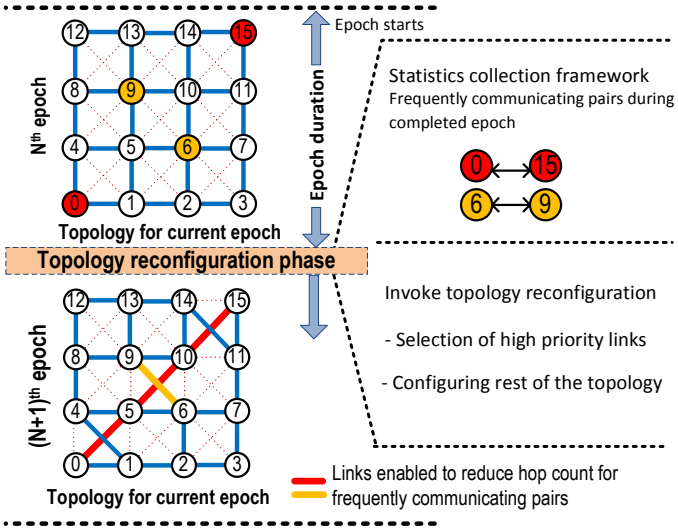

Figure 2: **HiROIC execution flow.** Application's execution is partitioned into epochs, during each epoch, HiROIC monitors the NoC's traffic patterns. At the end of the epoch, we determine, based on the patterns observed, whether a topology reconfiguration is appropriate. If so, the new configuration aims at minimizing the distance between frequently communicating pairs (FCPs).

paper, we will refer to any such pair as the *Frequently Communicating Pair (FCP)*. At the end of each epoch, we analyze the composition of the FCP set and determine whether a topology reconfiguration should occur to improve on the current port-link binding. Figure 2 illustrate a high-level overview of HiROIC's execution flow.

Note that our approach strives to predict application's demands based on the traffic observations during the current epoch. This is a valid approximation, as long as our epochs are short compared to the frequency of major phase changes in the application's behavior. We observe that, in practice, applications' phases are at-least hundreds of thousands of cycles long [4]; thus, in our evaluation we set the epoch length to 10,000 cycles, so to be able to quickly respond to significant traffic pattern changes.

3.1 Topology Reconfiguration

To optimize the topology for high-volume communication patterns, we perform the following steps: i) we collect traffic statistics over execution intervals (epochs) to predict future traffic behavior, ii) we trigger topology reconfigurations when we observe pattern changes, and iii) we set port-link bindings at each router based on the new topology planned. The hardware implementation of our scheme is discussed in-depth in [7], and is not discussed in this document.

Statistics Collection Framework. Knowledge of traffic statistics is necessary to predict application's future communication demands. The insight is that if few nodes are generating high traffic volume in one epoch, they will continue to do so in the upcoming epochs as well. This history-based prediction on the communication pattern helps in optimizing topology for the next epoch. Our framework collects i) the number of packets send between each source and destination, and ii) each router's maximum buffer occupancy [3] averaged over the epoch duration. The maximum buffer occupancy metric is used as an indicator of congestion in NoC. It is used to ensure that the reconfigured topologies do not lead to congestion.

The **Decision Engine** determines when to trigger the topology reconfiguration. At the end of each epoch, it analyses the data received from the statistics collection framework and determines the pairs that are generating the most traffic in the NoC. These pairs are referred as the **Frequently Communicating pairs (FCP)** in the rest of the paper. Topology reconfiguration is triggered when the following conditions are met: i) FCPs share more than a threshold percentage (T_{th}) of the total traffic, ii) and the new FCPs are significantly different from the FCPs of previous epoch. The first condition ensures that there is sufficient traffic irregularity in the network for HiROIC to be effective. The second condition ensures that a topology reconfiguration is triggered only when the current topology is not well suited for the predicted communication patterns. Notice that setting T_{th} to a high value will prevent HiROIC from adapting to communication demands quickly, whereas setting it to a low value will trigger frequent reconfigurations, causing traffic suspension and application slowdown. Therefore, a suitable value of T_{th} is essential for the proper functioning of HiROIC.

As mentioned before, HiROIC utilizes the maximum buffer occupancy metric [3] to detect congestion in the network. Congestion could lead to longer packet wait times, and hence application slowdowns. Our analysis shows that a baseline 2D mesh network is better at balancing traffic than the irregular topologies realized at runtime by HiROIC. Therefore, upon detection of a congestion situation, indicated by a high average value for the maximum buffer occupancy metric, the NoC reverts back to the 2D mesh topology.

3.1.1 *Reconfiguration Algorithm*

After the FCP set is calculated and the decision engine triggers the reconfiguration process, the reconfiguration algorithm determines the link-port bindings that would minimize the hop count between the FCPs. HiROIC's topology reconfiguration falls under the category of constraint satisfaction problems (CSP), where the constraint is the availability of ports and their valid bindings with links. The topology reconfiguration algorithm involves the following steps:

i. Port-link bindings for FCPs. After determining the FCP set at the end of each epoch, the next step is to enable the NoC's links that facilitate low-latency communication for those critical pairs. These links are selected by traversing the network, router by router, and building low-latency paths connecting the FCPs. First, the FCPs are grouped by their destination routers, and one group is considered in a greedy fashion before moving on to the next group. For each FCP entry within a group, HiROIC attempts to enable all the links between the source and the destination router. This is achieved by activating the appropriate select signals on the multiplexers used to bind ports to links.

It is not always possible to enable the shortest paths between all FCPs. This is because HiROIC's architecture is limited by: i) the number of available router ports, and ii) the flexibility provided by the glue logic binding links and ports. Therefore, a check is performed after enabling each link to determine that these constraints are not violated. If any constraint is violated for any port-link along the path of an FCP, then all the links bound for that path are released.

ii. Port-link bindings for non-FCPs. It is often the case that we can enable all the links required to provide shortest paths for the FCPs, and still have several disconnected ports in a number of routers. Therefore, the second phase in a topology reconfiguration binds the free router ports to available links. This again is achieved by traversing the network, router by router. Each router chooses among the locally-available port-link bindings in a greedy fashion. This process is repeated until the current router has successfully bound all its ports, or when all port-link binding options are exhausted. While this does not lead to an optimal mapping, it provides an acceptable solution for the latency of infrequently communicating source-destination pairs.

iii. Routing in the new topology After the generation of the new topology is complete, all routing paths must be updated. To this end, we leverage Ariadne's route-reconfiguration algorithm [1]: Ariadne was proposed for reconfiguration around faulty components and, due to the

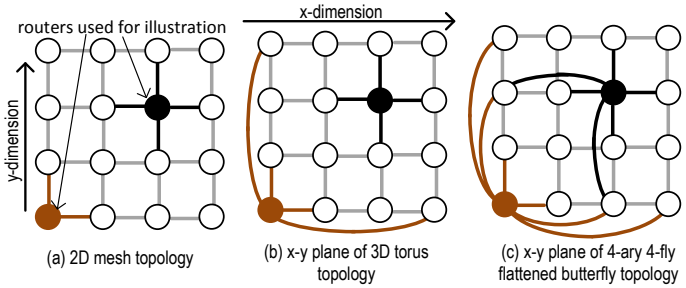

Figure 3: **Organization of links and routers in proposed physical topologies.** We consider two topologies for links: a 3D torus and a flattened butterfly. For simplicity of illustration, the figure shows the x- and y- dimension connections only for the bold colored routers. 3D torus routers have two connections in each dimension, while a 4-ary 4-fly flattened butterfly has routers with three connections in each dimension.

increasing reliability concerns with shrinking transistor sizes, we assume a functionality similar to Ariadne to be already present on-chip. Ariadne leverages the up*/down* algorithm [16] for routing in irregular networks, while proposing a quick and lightweight distributed implementation to update routes upon each topology change. Ariadne is reported to reconfigure a 64-node network in only ∼4K cycles, and it is therefore an ideal fit for HiROIC, if reconfiguration is triggered once every few epochs.

4. PHYSICAL TOPOLOGIES

We refer to a particular arrangement of physical links and ports, independently of any binding, as a *physical topology*. In physical topologies, links are available in accordance to a high radix topology (*e.g.*, 3D torus), while ports are those of a low radix router (*e.g.*, 2D mesh routers). In our evaluation, we consider two such topologies: both use routers with only four ports as in a mesh. We argue that, due to the similar router structure, both topologies have power and area characteristics similar to a 2D mesh network. Thus, all performance analysis is against a 2D mesh topology. A traditional 2D mesh has a one-to-one binding between ports and links, as depicted in Figure 3(a). In contrast, we implement HiROIC with the following physical topologies:

Adaptive 3D Torus. The average hop count between the nodes of a 3D torus is substantially lower than that of a 2D mesh. This is due to the higher (six) radix of its constituent routers, as shown in Figure 3(b). We propose a HiROIC-enabled adaptive 3D torus physical topology that organizes links as in a 3D torus, while maintaining radix-four routers.

Adaptive Flattened Butterfly. A flattened butterfly further reduces the average hop count compared to a 3D tori. For our 64-node NoC, a 4-ary 4-fly flattened butterfly arranges the routers in three dimensions, with direct links between routers on the same (x, y), (y, z) or (z, x) dimensions, as shown in Figure 3(c). Our second physical topology is an adaptive flattened butterfly (radix=9) topology with radix-four routers.

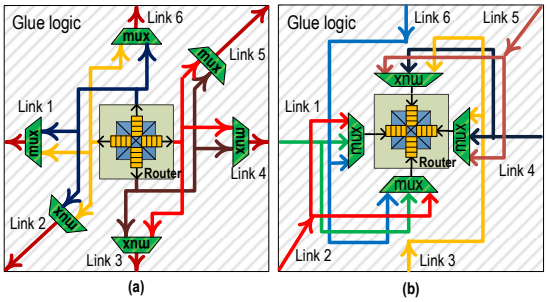

Figure 4: **Glue logic for an adaptive 3D torus router.** At the end of a topology reconfiguration, the glue logic select signals implement the new port-link bindings.

As discussed before, the glue logic incorporates multiplexers to decouple the traditional port-link binding. The size and the number of multiplexers affect how many links a particular port in the router can choose from. Full flexibility in link-port bindings allows a port to connect to any link. However, full flexibility results in large area overhead. We therefore decided to limit the number of links a port can choose from. For example, in an adaptive 3D torus topology, each output port can connect to one of three output links (one in each dimension) as shown in 4(a). With these restrictions, each output link can connect to one of two ports, and therefore six 2:1 multiplexers are sufficient for the output glue logic. For the input glue logic, each input port can bind to one of three links, resulting in four 3:1 multiplexers, as shown in Figure 4(b).

(a) Processor @2GHz

Cores	2-wide fetch/commit 64-entry ROB
coherence	4-hop MESI, 64B block
L1 cache	Private: 32KB/node ways:4 latency:2
L2 cache	Shared: 256KB/node ways:16 latency:6
Memory	Distributed: 1GB/bank banks:4 latency:160

(b) Network @2GHz

Topology	8x8 mesh, 128 bit links
Pipeline	3-stage VC flow ctrl
VCs	4 VCs/port, 8 flits/VC
Routing	up*/down*, XY
Routing-Update	Ariadne [1]: new up*/down* routes
Workload	multi-programmed: SPEC CPU 2006
Simulation	10M cycles

Table 1: **Experimental CMP: configuration of processor and network.**

5. EXPERIMENTAL RESULTS

We evaluated HiROIC on a cycle-accurate trace-driven multi-core simulator [3]. Table 1 shows the characteristics of the processors and the NoC we evaluated. We ran all experiments considering a 64 core system as a baseline with 3-stage pipelined routers in the network. We implemented the HiROIC scheme over the adaptive 3D torus and adaptive flattened butterfly topologies. All our comparisons are against a baseline 2D mesh topology. Finally, we used the up*/down* routing algorithm [16] to navigate the packets within the NoC.

5.1 Synthetic Traffic

Our first set of experiments injects the NoC with synthetic normal random traffic. HiROIC is not supposed to trigger topology reconfiguration for normal random traffic because all pairs have similar amount of traffic between them, resulting in absence of frequently communicating pairs. We observe that our algorithm indeed did not invoke any reconfigurations for normal random traffic, leading to our adaptive topologies behaving exactly as a 2D mesh.

We also created synthetic *directed traffic* to gain more insights into the strengths and limitations of our scheme. Our synthetic directed workloads consist of 20 phases, each of which lasts 50 epochs and has a number of frequently communicating pairs (FCPs). The new FCP set is randomly selected after each phase. Other network nodes produce traffic at low injection rate (0.005 flits/node/cycle). Figure 5 compares the average latency of the topologies under consideration with directed traffic using 15 FCPs. On the x-axis we sweep the injection rate for the FCPs.

To compare latency improvements, we define three traffic load levels for the FCPs: low, medium and high. Low traffic corresponds to 0.1 flits/node/cycle, and it is the lowest injection rate used in our experiments. The medium and high injection rates are defined as the injection rates where the network latency for the 2D mesh is 1.5× and 2× that of the low-load latency. We observe that the latency improvement over 2D mesh for the adaptive 3D torus is 22.7%, 29.3% and 36.9% for low, medium and high injection rates, respectively, while the corresponding latency improvements for the adaptive flattened butterfly are slightly better at 30.8%, 35.6% and 37.8%. This experiment proves HiROIC's potential in providing significant reduction in network latency in the presence of traffic imbalance.

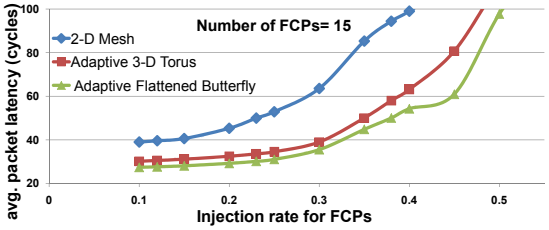

Figure 5: **Average network latency with directed traffic.** The plot compares the average network latency for three topologies under directed traffic with increasing injection rate for the FCPs. HiROIC provides low-latency paths between FCPs, resulting in significant overall latency improvements.

5.2 Multiprogrammed Workloads

We also evaluated our proposed scheme with a set of multi-programmed workloads consisting of 35 applications from the SPEC CPU 2006 benchmark suite. The experiments were conducted across 60 multi-programmed workloads, with each workload consisting of 15 copies each of 4 unique applications. The studied applications exhibit a wide range of cache misses per kilo instructions (MPKI) values: the MPKI metric directly correlates to the amount of traffic sent through the NoC. Some workloads use applications with similar MPKI values, causing all cores to inject similar amount of traffic on the NoC. We further divide such workloads into two categories: the *LL* category workloads use applications with *low* MPKI, while the *HH* category workloads use applications with *high* MPKI. We also use imbalanced workloads, in which the MPKI values among the applications differ substantially. We group such workloads under the *LH* category, as they use applications with both *low* and *high* MPKI.

Figure 6 compares the network latency of a 2D mesh against the HiROIC-enabled adaptive 3D torus and adaptive flattened butterfly under multi-programmed workloads. As expected, HiROIC provides the highest latency improvements for LH category workloads: average latency reduction over 2D mesh is 21.4% and 21.2% for adaptive 3D torus and adaptive flattened butterfly, respectively. HiROIC also provides good improvements for workloads in the *LL* category: 18.5% and 17.0% latency reduction over 2D mesh for adaptive 3D torus and adaptive flattened butterfly, respectively. This is because network transmissions are scarce in such workloads, and only a small subset of applications produce significant traffic within a given computational epoch (in contrast to all applications producing traffic all the time). Therefore, HiROIC optimizes the NoC topology to provide short communication paths to and from the active subset. Finally, we observe the smallest gains for workloads in the *HH* category, as most nodes generate heavy traffic, leading to a larger than optimal FCP set. For workloads in the *HH* category, the average latency improvement for adaptive 3D torus and adaptive flattened butterfly over 2D mesh, is 17.0% and 16.7%, respectively.

Our evaluations with multi-programmed workloads do not yield results as promising as the directed traffic evaluation of Section 5.1. The primary reason for this is the organization of the underlying CMP system. Our baseline CMP uses a shared and distributed L2 cache architecture, and therefore L1 cache misses are uniformly distributed over the entire CMP. As a result of this distribution, the majority of the traffic in the NoC is uniform, and HiROIC is not able to optimize communication paths aggressively, leading to sub-optimal results. However, with a growing adoption of application-specific accelerators on-chip [10], only a small subset of which will be active at any point in time, we expect a greater imbalance in communication for future architectures. The scenario is expected to be similar to our synthetic directed traffic experiments from Section 5.1 and thus we expect HiROIC to provide even better benefits in future architectures.

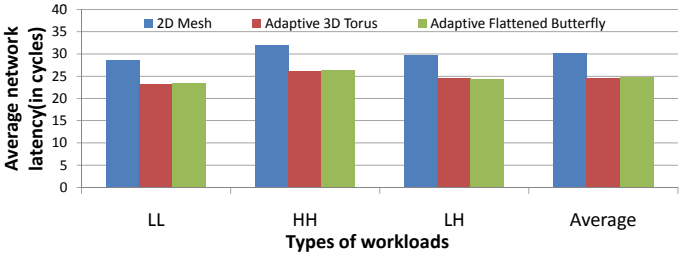

Figure 6: **Average network latency under multi-programmed workloads.** The results are presented for 2D mesh, adaptive 3D torus and adaptive flattened butterfly topologies under three different types of workloads. HiROIC is most effective for workloads in the *LH* category due to high traffic imbalance. Both adaptive 3D torus and adaptive flattened butterfly show similar latency improvements over 2D mesh.

5.3 Overheads

Area and Power Overhead. The addition of small multiplexers (typically 2:1 or 3:1) around the router core is significantly cheaper than increasing the number of ports. Apart from that, Ariadne-style route-reconfiguration can be implemented at a cheap cost of 2%. However, this cost can be neglected if the CMP already accounts for fault-tolerance and has Ariadne-style route-reconfiguration logic. All other HiROIC components are extremely lightweight, with small counters, comparators and adders added to each router. A lightweight distributed implementation of our reconfiguration algorithm is discussed in [7]. Compared to modern routers, with deep buffers and many virtual channels, HiROIC's controller logic overhead is negligible.

The addition of multiplexers and additional wires can lead to an increase in power dissipation. However, wires that are not used during an epoch of execution, can be power-gated to eliminate any additional leakage power consumption. In addition, the amount of dynamic power overhead due to longer wires is negligible, compared to the dynamic power spent in reading and writing buffers, traversing the crossbar, *etc*. Fortunately, HiROIC reduces the amount of dynamic energy spent on each packet traversal by reducing the average packet hop count. Thus, we argue that HiROIC more than compensates for the increase in power dissipation due to multiplexers and long wires.

Transition between topologies. Draining all packets in-flight during topology transition is essential because wormhole packets can be physically distributed over multiple port buffers connected via links. In our evaluation, the average number of cycles required to drain packets per topology reconfiguration across all the workloads is 35 for the adaptive 3D torus topology. This overhead is small compared to our computational epoch length of 10,000 cycles, and we account for this overhead in our results.

6. CONCLUSION

HiROIC provides performance similar to high-radix (>5 ports) NoC topologies using resources comparable to low-radix topologies ($<=5$ ports) by optimizing for critical high-volume communication paths at runtime. In HiROIC, links are deployed abundantly for rich connectivity as in high-radix topologies, while the number of router ports is kept low. Router ports bind to links at runtime in accordance to a distributed traffic analysis heuristic implemented at each router. Our experiments show that HiROIC reduces average network latency by 19% compared to an area- and power- comparable mesh on SPEC2006 multiprogrammed workloads. When using non-uniform synthetic traffic, the latency reduction is in the 30-38% range.

Acknowledgements: This work was supported in part by C-FAR, one of the six SRC STARnet Centers, sponsored by MARCO and DARPA, and NSF grant #0746425.

7. REFERENCES

[1] K. Aisopos, A. DeOrio, L.-S. Peh, and V. Bertacco. ARIADNE: Agnostic reconfiguration in a disconnected network environment. In *Proc. PACT*, 2011.
[2] J. Balfour and W. Dally. Design tradeoffs for tiled CMP on-chip networks. In *Proc. ICS*, 2006.
[3] R. Das, S. Narayanasamy, S. Satpathy, and R. Dreslinski. Catnap: energy proportional multiple network-on-chip. In *Proc. ISCA*, 2013.
[4] A. Dhodapkar and J. Smith. Comparing program phase detection techniques. In *Proc. MICRO*, 2003.
[5] M. Faruque, T. Ebi, and J. Henkel. Configurable links for runtime adaptive on-chip communication. In *Proc. DATE*, 2009.
[6] B. Fu, Y. Han, J. Ma, H. Li, and X. Li. An abacus turn model for time/space-efficient reconfigurable routing. In *Proc. ISCA*, 2011.
[7] A. Jain, R. Parikh, and V. Bertacco. High-radix on-chip networks with low-radix routers. In *Proc. ICCAD*, 2014.
[8] A. Kahng, B. Lin, K. Samadi, and R. Ramanujam. Trace-driven optimization of networks-on-chip configurations. In *Proc. DAC*, 2010.
[9] T. Krishna, C.-H. Chen, W. Kwon, and L.-S. Peh. Breaking the on-chip latency barrier using SMART. In *Proc. HPCA*, 2013.
[10] M. J. Lyons, M. Hempstead, G.-Y. Wei, and D. Brooks. The accelerator store: A shared memory framework for accelerator-based systems. *ACM Trans. Archit. Code Optim.*, 8(4), 2012.
[11] H. Matsutani, M. Koibuchi, H. Amano, and D. Wang. Run-time power gating of on-chip routers using look-ahead routing. In *Proc. ASPDAC*, 2008.
[12] R. Mullins, A. West, and S. Moore. Low-latency virtual-channel routers for on-chip networks. In *Proc. ISCA*, 2004.
[13] M. Palesi, R. Holsmark, S. Kumar, and V. Catania. *IEEE Trans. Parallel and Distributed Systems*, 20(3), 2009.
[14] R. Parikh and V. Bertacco. uDIREC: unified diagnosis and reconfiguration for frugal bypass of NoC faults. In *Proc. MICRO*, 2013.
[15] L.-S. Peh and W. Dally. A delay model and speculative architecture for pipelined routers. In *Proc. HPCA*, 2001.
[16] M. Schroeder et. al. Autonet: A high-speed, self-configuring local area network using point-to-point links. *IEEE Trans. Selected Areas in Communication*, 9(8), 1991.
[17] M. Stuart, M. Stensgaard, and J. Sparsø. The ReNoC reconfigurable network-on-chip: Architecture, configuration algorithms, and evaluation. *ACM Trans. Embed. Comput. Syst.*, 10(4), 2011.