High Radix On-Chip Networks at Incremental Reconfiguration Costs

Animesh Jain, Ritesh Parikh and Valeria Bertacco

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

ABSTRACT

Networks-on-chip (NoCs) have become increasingly widespread in recent years due to the extensive integration of many components in modern multicore processors and SoC designs. One of the fundamental tradeoffs 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. Today, NoC designs take significant silicon area and may consume up to 30% of the entire chip power budget; thus, naïvely deploying an expensive high-radix network is no longer possible.

In this work, we present HiROIC [1] (**Hi**gh **R**adix **O**n-chip Networks at **I**ncremental re-**C**onfiguration Cost), to provide 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. To this end, HiROIC proposes a flexible topology reconfiguration infrastructure where the abundantly available links between routers (in accordance to a high-radix topology) are decoupled from scarcely available router ports (similar to a low-radix topology). The link-to-port binding is done at runtime, based on traffic patterns, using low-overhead multiplexers. HiROIC employs a statistics collection and decision-making framework to orchestrate the topology modifications that maximize performance. While our solution may require some additional and/or longer links, we observe that links are not the timing bottlenecks in contemporary router pipelines and links that are not coupled to ports could be power-gated to save power. HiROIC ensures a globally connected and deadlock-free network at all times. Our experiments on a 64-node CMP, running multi-programmed workloads, show that HiROIC reduces average network latency by 21% over an area- and power- comparable mesh NoC.

1. INTRODUCTION

As a result of increasing integration of components into CMP and SoC architectures, networks-on-chip (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 the topology, that is, the number of I/O ports that a router provides to connect links to adjacent routers. High-radix routers enable low-diameter topologies, and allow the processing nodes to be connected closely, with packets traversing just a few routers to reach their destination. On the down side, 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. A popular alternative are topologies deploying low-radix routers, such as meshes. Typically, routers up to a radix of five (*e.g.*, mesh) are considered low radix, while larger ones are considered high-radix. Low-radix routers can be clocked at a substantially higher clock rate than their high-radix counterparts. For example, according to the models provided in [17], a radix-7 router has a 4.1% higher cycle time compared to a radix-5 router. Unfortunately, for many-core architectures, low-radix topologies could lead to large network diameters and prohibitively high hop counts. Using low-radix routers particularly hurts performance when applications do not have sufficient memory-level parallelism (MLP) to hide the higher latency. The radix of the router is therefore an important design decision that directly affects latency, area and power targets.

With HiROIC 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. Therefore HiROIC provides, on average, a small hop count for packets traversing the network, similar to high-radix topologies. Naturally, the greater the disparity in communication load between source-destination pairs, the greater is HiROIC's effectiveness. 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 link-port decoupling. HiROIC's routers do not statically bind their ports to links, unlike traditional routers. Rather, this binding is applied at runtime, depending on the communication demands of the application. HiROIC deploys links abundantly, in accordance to a high-radix topology, to *potentially* provide short paths between any source-destination pair. However, HiROIC's routers still maintain the internal port-count of low-radix networks. HiROIC deploys an additional layer of glue-logic to bind ports to links at runtime. These bindings are used for one epoch of execution, after which HiROIC evaluates whether the current topology is suitable for upcoming traffic patterns. If not, binding decisions are re-evaluated to globally optimize for the new communication patterns. In essence, our infrastructure's ability to realize many irregular or regular topologies is leveraged to adapt to application's demands at runtime. While HiROIC's wiring overhead is greater than conventional topologies like meshes, due to longer and additional wires, we observe that wires are never the timing bottleneck in conventional router pipelines [10]. In addition, unused wires can be power-gated once the port-to-link binding decisions are finalized.

Note that, in typical NoCs, routers have one *local* port (sometimes more) connecting to the processing node(s). Since the connection to the processing node is essential, HiROIC uses a fixed port-link binding for *local* ports. In the rest of this paper we exclude the local port(s) when reporting the radix of the router.

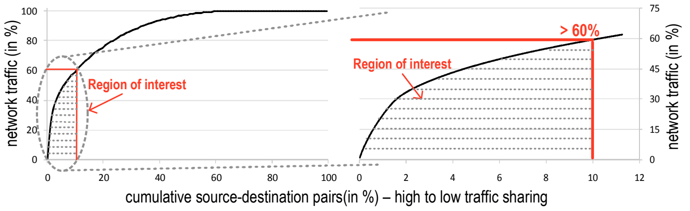

Figure 1: **Fraction of traffic load shared by the most exercised source-destination pairs.** The top 10% source-destination pairs transfers more than 60% of total network traffic between them, therefore, HiROIC targets this pool for topology optimization.

It is essential for HiROIC to have a high variation between high-usage source-destination pairs and other source-destination pairs. To this end, we conducted a study whose findings are plotted in Figure 1. The plot shows the contribution of traffic flowing between each source-destination pair. Our testbed consisted of an 8x8 mesh CMP running a multiprogrammed mix of applications from the SPEC CPU2006 suite. Source-destination pairs are sorted by decreasing traffic activity during the execution, and the plot on the left indicates what fraction of network traffic (Y axis) was carried out by a given fraction of sorted source-destination pairs. The plot on the right is an enlargement of the contribution by the top 12% source-destination pairs: less than 10% of the source-destination pairs shares as much as 60% of the traffic load on average. Beyond the tenth percentile of utilization, this disparity is no longer obvious. Thus, HiROIC's goal is to identify and leverage the 10% most used source-destination pairs to provide short and high-bandwidth paths between them. This, in turn, minimizes the effective

[1] ROIC is a popular Economics acronym for 'Return on Invested Capital.' HiROIC is synonymous to *high-ROIC*, since we provide a high performance return for a given area/power budget dedicated to NoCs.

network hop count.

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 software-based 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

Extensive research has been dedicated in the past towards: i) reducing the number of pipeline stages within the router [13, 17], and ii) increasing the frequency of a router's operation [2]. Certain other works have optimized routers for a particular topology and flow control [9, 18]. Our technique, HiROIC, is orthogonal to such approaches, as HiROIC aims at decreasing the effective hop count of packets.

A considerable body of previous work targets application-specific NoC synthesis. In these works, before the NoC design phase, all applications that are expected to run on the system are characterized. This characterization then drives the optimization of a variety of design aspects: (i) topology [3, 21], (ii) routing [4, 14], (iii) buffer sizing [8], *etc.* However, these techniques typically produce static configurations and the resulting NoC cannot adapt to changing application behavior at runtime. In contrast, HiROIC can quickly adapt by predicting near-future communication patterns and reconfiguring the topology accordingly.

A few runtime reconfiguration approaches have also been proposed to optimize for either power or performance. To reduce the NoC's leakage power, researchers have proposed power-gating network resources at a coarse [19, 11] or fine [12] granularity during periods of inactivity. Other works, as in [7], reconfigure the routing algorithm at runtime to provide greater routing adaptivity at hot-spots, or adapt the channel bandwidth at runtime to optimize for performance [6]. All these techniques are orthogonal to our approach as they adapt the routing algorithm or channel bandwidth, while HiROIC reconfigures the topology; thus, they have the potential to provide extra gains.

Most of the previous runtime topology reconfiguration solutions have been targeting silicon reliability. Ariadne [1] and uDIREC [15] propose reconfiguration algorithms for faulty NoCs that update the routing function to bypass the malfunctioning components. To the best of our knowledge, HiROIC is the first technique that aims at reconfiguring the topology at runtime for performance gains.

3. METHODOLOGY

HiROIC leverages link-port decoupling to mimic high-radix topologies by utilizing resources comparable to a low-radix topology. HiROIC optimizes a topology for transferring traffic efficiently between heavy load source-destination pairs, using three major components: i) a statistics collection framework to predict upcoming traffic patterns, ii) a decision engine to determine whether to invoke a topology reconfiguration, and iii) a centralized software-based reconfiguration algorithm that determines the link-port bindings for each epoch of execution.

3.1 Link-Port Decoupling

In conventional routers, ports have a one-to-one mapping with links. Therefore, each port has a dedicated link connected to it. As mentioned the Section 1, we propose to add more links to low-radix routers and decouple the traditional one-to-one connection between ports and links. With the help of a few multiplexers, we can still have the same number of ports as in a low-radix router, while providing more links to choose from in forming a connection. The micro-architecture of the low-radix routers remains unchanged because each port is still connected to only one link. Upon closer examination, only the routing function within the router must be updated to reflect changes in the topology caused by new port-link bindings.

The proposed design provides flexibility to the network to adapt to changing communication needs. It enables binding a same port to different links during different phases of execution. In Figure 2, we show

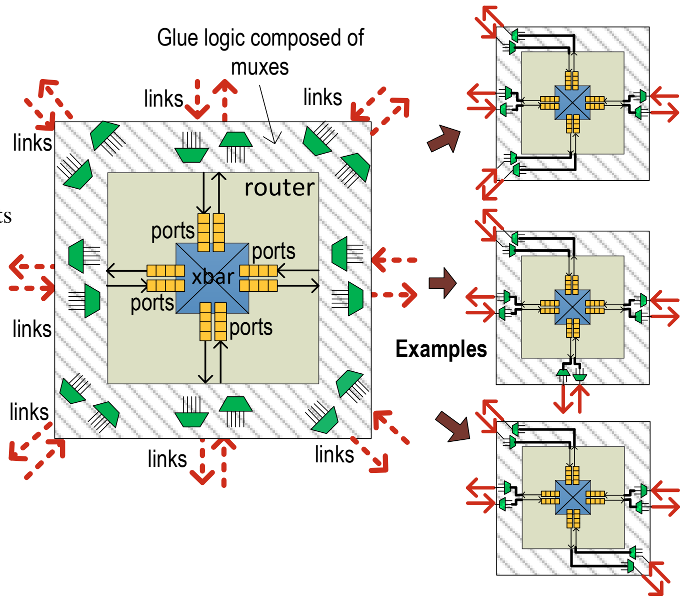

Figure 2: **Port-link decoupling.** The router in the figure can connect up to four router ports by selecting among eight links. Depending on the application's traffic demands, it can adopt different port-binding configurations, by simply assigning select signals of the multiplexers. The right part of the figure shows three such possible bindings.

a HiROIC-enabled router with four ports that has a choice to connect to eight different links. The figure shows three such possible bindings for the router. These link-port bindings are enforced with the help of multiplexers (represented by the glue logic in the Figure), which are configured depending on the application's demands. The size and number of multiplexers are heavily dependent on the baseline topology and the flexibility of port-link bindings desired. This trade-off is detailed in Section 4. We also provide a statistics collection framework to monitor the traffic transferred between each pair of nodes. This framework provides the information necessary to predict the quasi-optimal port-link bindings for upcoming execution phases.

3.2 Execution Flow

Figure 3 provides a high level overview of HiROIC's execution flow. An application's execution is divided into epochs of fixed clock cycle length. The NoC statistics collection framework records the communication patterns of each router for every epoch. When the epoch ends, the NoC framework transfers this information to one of the cores. This core drives the topology reconfiguration phase and it is therefore referred to as the *supervisor core*. The supervisor core makes the decision of whether to modify the topology based on the current topology and the recorded patterns in the previous epoch. If the supervisor core decides to change the topology, then traffic injection is suspended while the supervisor runs the reconfiguration algorithm. Meanwhile, all the packets that were in flight at the end of the previous epoch are allowed to drain. The supervisor node then transmits the new link-port bindings to each router through dedicated wires. Once the topology and routing reconfiguration completes, the NoC resumes its normal operation with the new reconfigured topology.

Statistics Collection Framework. Knowledge of traffic statistics is necessary for the supervisor node to decide whether the current topology suits the traffic demands of the application. This information helps the supervisor node predict an application's communication needs for the next epoch. The insight is that if few nodes are generating most of the traffic in one epoch, then the same nodes are expected to generate high volume traffic in the next epoch, too. Thus, the prediction is history-based, as it depends on the communication patterns observed in last epoch. Specifically, our framework collects the following two values: i) number of packets sent between each source and destination, and ii) each router's maximum buffer occupancy [5] averaged over the epoch duration. Note that we utilize the maximum buffer occupancy metric as an indicator of congestion in NoC's paths. The HiROIC scheme leverages this information to reduce congestion in the reconfigured topology by allowing only a few packets to use the congested paths.

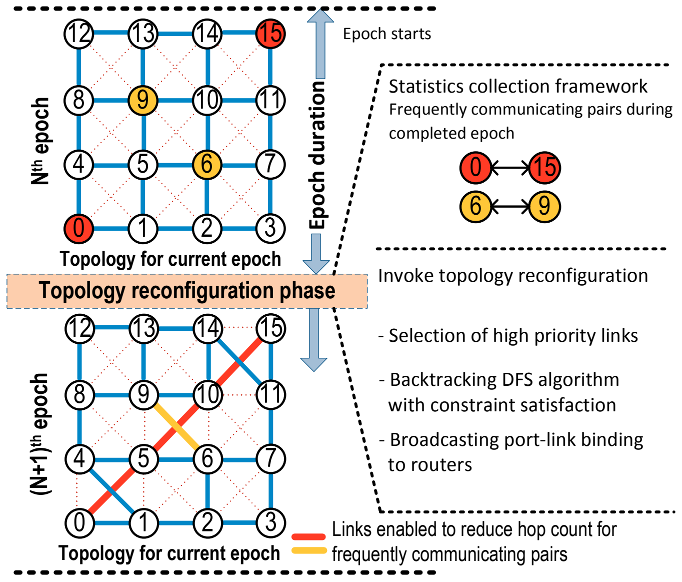
Epoch starts
Nth epoch
Epoch duration
Statistics collection framework
Frequently communicating pairs during completed epoch
Topology for current epoch
Topology reconfiguration phase
Invoke topology reconfiguration
- Selection of high priority links
- Backtracking DFS algorithm with constraint satisfaction
- Broadcasting port-link binding to routers
(N+1)th epoch
Topology for current epoch
Links enabled to reduce hop count for frequently communicating pairs

Figure 3: **HiROIC execution flow.** The application's execution is divided into epochs. A NoC statistics collection framework identifies the frequently communicating pairs. Once the epoch ends, the topology reconfiguration phase is triggered. During this time, NoC traffic is suspended. The newly reconfigured topology aims at reducing the hop count between frequently communicating node-pairs.

```
FCP_set = find_fcp(info_by_statistics_collection);

/* Pre-selection of high priority links */
for (PAIR in FCP_set);
| path = up_down_routing_algo(PAIR);
| enable_links(path);
| if(check_constraints() == FAIL)
| | disable_links(path);

/* Backtracking algorithm for constraint satisfaction*/
for (NODE in NoC);
| for (BINDING in remaining_bindings(NODE))
| | enable_binding(BINDING);
| | if(check_constraints() == FAIL)
| | | disable_binding();

broadcast_bindings_to_routers();
compute_up_down_routes_with_new_bindings();
```

Figure 4: **HiROIC's centralized software-based topology reconfiguration algorithm.** First, the supervisor node pre-selects the high priority links for the the FCPs. The remaining topology is generated by running a backtracking DFS algorithm. Remaining ports at each router are randomly bound to links one-by-one, while the algorithm backtracks if any such binding leads to a violation of the constrains imposed by the router's radix and the glue logic.

The **Decision Engine** is used to determine when to invoke a topology reconfiguration. As discussed previously, the supervisor node is responsible for determining when to trigger the topology reconfiguration phase. Initially, it identifies which are the top 10% pairs that are generating the highest amount of traffic in the NoC. In the rest of this paper, we will refer to such source-destination pairs as the **Frequently Communicating Pairs (FCP)**. At the end of each epoch, the supervisor node calculates the new FCP set. Topology reconfiguration is invoked when the following conditions are met: i) FCPs share more than a threshold percentage (T_{th}) of total traffic, ii) and the new FCPs are significantly different from the FCPs of the previous epoch. The first condition determines HiROIC's sensitivity to changing application's behavior. If the usage threshold T_{th} is too high, then the topology will not adapt quickly enough to the NoC's communication needs. However, if the threshold T_{th} is kept too low, then the topology reconfiguration phase will be triggered too frequently, causing traffic to suspend often and resulting in net slowdown. Our experiments show that setting the threshold T_{th} to 60% provides a suitable trade-off. The second condition makes sure that a topology reconfiguration is not triggered when the current topology is already capable of handing the application communication needs for the next epoch. These two conditions together guarantee that the topology reconfiguration is triggered only when the current topology is not fit for the current and near-future communication patterns.

HiROIC also utilizes the maximum buffer occupancy metric to attain a more effective reconfiguration. Based on the number of packets metric, HiROIC can only optimize the average hop count, without taking congestion into account. Congestion could lead to longer packet wait times and, hence, application slowdowns. Therefore, HiROIC tries to balance the load on all links using the maximum buffer occupancy metric. 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 supervisor node reverts the NoC back to the 2D mesh topology. Note that this situation is a special case of topology reconfiguration, where the port-link bindings lead to a 2D mesh network. Therefore, HiROIC always ensures above-par performance compared to a baseline 2D mesh.

Centralized Software-Based Topology Reconfiguration Algorithm. Once the decision engine triggers a topology reconfiguration event, the supervisor core runs the algorithm outlined in Figure 4. The topology reconfiguration problem falls under the category of constraint satisfaction problems (CSP). It can be modelled as a depth-first search (DFS), using a backtracking algorithm to find valid configurations. At each step of the decision process, the current selection is validated against a set of constraints, while the algorithm strives to optimize a target metric. Specifically, HiROIC tries to minimize the hop count between the FCPs, while constraining the number of links bound to a router to be less than or equal to its radix. The topology reconfiguration algorithm involves the following steps:

i) **Selection of High Priority Links.** Initially, all links are considered disabled. Then data received from the statistics framework is analysed and links that are on the shortest paths between FCPs are enabled. We use the up*/down* routing algorithm to route packets in the NoC, and for identifying the shortest path between FCPs. The link selection is completed step-by-step for each frequently communicating pair. A check is performed after each step to determine that the constraints are not violated. The exact set of constraints depend on the layout of links and router ports and the glue logic. We outline the set of constraints for our representative system in Section 4. If any constraint is violated for any pair, then all the links for that particular pair are released. This completes the first phase of our reconfiguration algorithm, as shown in Figure 4.

ii) **Backtracking DFS Algorithm with Constraint Satisfaction.** Once the links connecting the FCPs are selected, the DFS algorithm traverses the NoC router by router and selects random port-link bindings. Every binding is validated against the same set of constraints. If constraints are not met, then the binding is disabled and another binding is tried. Once all the routers have been visited, a connectivity check is performed to ensure that all routers are globally connected. If this check fails, then the algorithm is run again with a different initial router. We ran several Monte Carlo simulations to evaluate the frequency of this connectivity check failure for the first root node, and found out that it is, in practice, a rare event, occurring only 5 times in 1,000 runs. The corresponding pseudo-code is depicted in Figure 4. We relaxed some constraints that are not vital to the functional correctness of the system in order to ensure full connectivity. This aspect is discussed in Section 4.

iii) **Broadcasting port-link bindings to routers**. Once the software reconfiguration routine terminates, the new port-link bindings are communicated to routers using dedicated wires. The routers modify the control signals of the multiplexers within the glue logic to reflect the new port-link bindings.

To route packets through the network, once the topology has been updated, we employ the up*/down* routing algorithm [1]. The up*/down* routing algorithm best suits our needs because HiROIC's runtime topologies are often irregular. The up*/down* algorithm, by construct, is deadlock-free and guarantees connectivity if the network is fully connected. It works by assigning a direction, *up* or *down*, to all links. Then all the routes that contain a down-link followed by an up-link are forbidden. By forbidding such routes, the algorithm breaks the abstract cycles that can potentially lead to deadlock. We use the Ari-

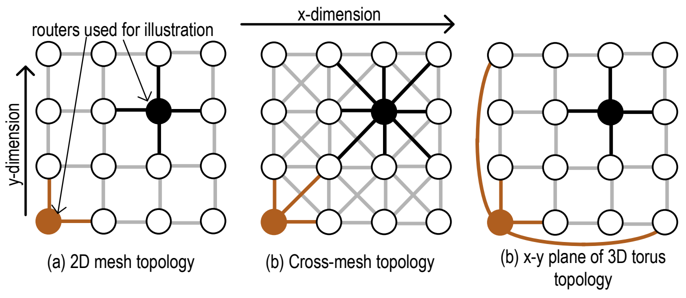

Figure 5: **Organization of links and routers in our proposed physical topologies.** We consider two different topologies for HiROIC: a cross-mesh and a 3D Torus. For simplicity of illustration, the figure shows the x- and y- dimension connections only for the bold colored routers. A cross-mesh router has eight connections around a router, while a 3D torus router has two connections in each dimension resulting in a total radix of six.

adne [1] reconfiguration algorithm to update the routing tables for all routers, once the topology has been updated. In a network of N nodes, Ariadne takes N^2 cycles for updating the routing table of all routers. Finally, we enhance the up*/down* routing algorithm to ensure that the number of hops between the frequently communication nodes is kept as low as possible.

4. PHYSICAL TOPOLOGIES

The key idea behind HiROIC is to emulate a router with more links than available ports. We refer to a particular arrangement of links, ports and binding logic as a *physical* topology. In *physical* topologies, links are organized in accordance to a high-radix topology (*e.g.*, 3D torus), while router ports are organized as in a low-radix topology (*e.g.*, 2D mesh). Naturally, the effectiveness of HiROIC greatly depends on the arrangement of links and ports within the *physical* topology. Specifically, we evaluate HiROIC with two *physical* topologies: both of them use routers with only four ports, as in a mesh. We argue that, due to the similar internal structure of routers, both *physical* topologies will have power and area characteristics similar to a 2D mesh network. Therefore, all the presented performance comparisons are against a 2D mesh topology. Notice that a traditional 2D mesh has a one-to-one binding between ports and links, as depicted in Figure 5(a). We evaluate HiROIC with the following proposed topologies:

Adaptive Cross-Mesh. A cross-mesh topology is a high-radix topology that is a direct extension of the 2D mesh topology. As shown in Figure 5(b), in this topology routers have a radix of eight. In our proposed *physical* topology, referred to as adaptive cross-mesh topology, routers have a radix of four, while still having eight links to select from in building connections. Depending on application's demands, the routers dynamically choose the four links to bind to. Figure 3 shows one configuration of an adaptive cross-mesh topology.

Adaptive 3D Torus. Any HiROIC *physical* topology should be able to provide short paths between particular source-destination pairs. The average hop count between the nodes in a 3D torus topology is substantially smaller than a 2D mesh. This is because of the higher (six) radix of its constituent routers. Figure 5(c) shows that the 3D torus topology has 2 connections in each dimension and thus provides richer connectivity than a 2D mesh. We therefore build our proposed adaptive 3D torus topology by organizing links in a similar fashion as the 3D topology. Once again, the routers in the adaptive 3D torus topology have a radix of four with six links to choose from in building connections.

As mentioned in the previous section, the glue logic incorporates multiplexers to decouple the traditional port-link binding. The size and the number of multiplexers affects how many links a particular port in the router can choose from. For instance, consider the scenario where a router in an adaptive cross-mesh topology is provided full flexibility to connect any link to any port. In such a scenario, each output link can be connected to any of the four output ports, requiring eight 4:1 multiplexers. On the input side though, each input port can receive the data from any of the eight links. Thus, it results in the addition of four 8:1 multiplexers. Therefore, full flexibility results in the addition of eight 4:1 and four 8:1 multiplexers in total. In addition, the mixing of ports and links might lead to layout challenges. We experimentally concluded that giving full port-link binding flexibility is not beneficial, considering the extra logic overhead. Therefore, to create a balance between flexibility in port-link bindings and size/number of multiplexers, we decided to limit the number of links a port can choose from. Specifically, we used the following arrangement of multiplexers in the router's glue logic.

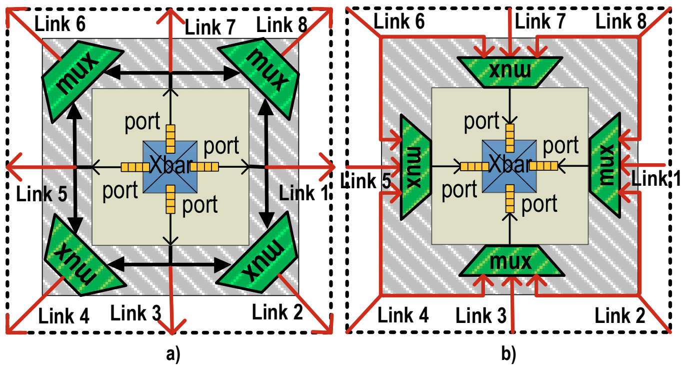

Figure 6: **Logic diagram of the glue logic for an adaptive cross- mesh router.** a) shows the output glue logic comprising four 2:1 multiplexers while b) shows the input glue logic comprising four 3:1 multiplexers. Note that only four links will be connected to the ports at any one time. At the end of the topology reconfiguration phase, the supervisor node provides the control signals to perform the port-link bindings.

In an adaptive cross-mesh topology, each output port of the router has only three fixed output links to choose from. As shown in Figure 6(a), four out of the eight links can connect to only one port (for example, Link 7 can connect only to the North port), whereas the other four links can connect to one of two ports (for example, Link 8 can connect to North or East port). This results in the addition of four 4:1 multiplexers for the output glue logic. For the input glue logic, an input port can recieve data from three input links, requiring four 3:1 multiplexers, as shown in Figure 6(b).

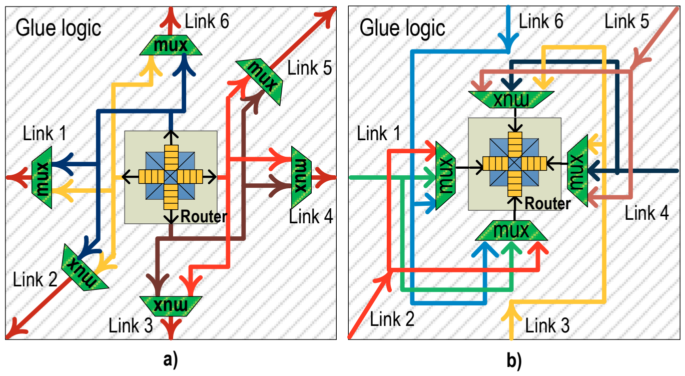

Figure 7: **Logic diagram of the glue logic for an adaptive 3D torus router.** a) shows the output glue logic comprising six 2:1 multiplexers while b) shows the input glue logic comprising four 3:1 multiplexers. Note that only four links will be connected to the ports at any one time. At the end of the topology reconfiguration phase, the supervisor node provides the control signals to perform the port-link bindings.

In the adaptive 3D torus topology, each output port of the router can connect to one of three fixed output links. This constrains the router to have at most two links in each dimension. In one possible configuration, it can have two links in the x-dimension, while having one link each in y- and z- dimensions. In another possible configuration, it can have two links each in x- and y- dimensions, with no link in the z-dimension. Such glue logic design results in a suitable trade-off between flexibility and size/number of multiplexers. As shown in Figure 7a), each output link can connect to one of two ports, and therefore six 2:1 multiplexers are required for the output glue logic. For the input glue logic, each input port can get data from one of three links, resulting in the addition of four 3:1 multiplexers, as shown in the Figure 7b).

The restrictions we place on the port-link bindings translate to constraints in the backtracking search algorithm that was discussed in Section 3.2. Since each router can choose among more links than the number of ports, the most basic constraint is that the radix of the router should never be greater than the number of ports (four in our physical topologies). Other constraints are physical topology specific and arise from the limited flexibility of the glue logic: an example constraint for the adaptive cross-mesh topology is that its routers cannot connect simultaneously to links 1, 2, 3 and 4 from Figure 6. Such a binding would violate the restrictions placed on the link-port binding.

As it is common in constraint satisfaction problems, some conditions might not result in a solution satisfying all the constraints. In order to handle such scenarios, some of the constraints that are not vital for correct functionality, are relaxed. For *e.g.*, we can relax the constraint that the radix of each router should be exactly four. The relaxed version of this constraint accepts solutions with routers of radix less than four. This is reasonable as one port can easily be disabled and power-gated.

Obviously, constraint relaxation can result in routers with fewer active ports, and thus can increase average the hop count between nodes. To evaluate the effect of this constraint relaxation, we conducted Monte Carlo simulations to determine the number of routers that do not achieve a radix of four upon topology reconfiguration. In this experiment, only a fixed number of links were enabled initially, then the topology reconfiguration algorithm was run to build the complete topology. Table 1 shows that even after pre-selecting 50 links at the start of the reconfiguration in an adaptive 3D torus topology, 94% of the routers are able to achieve the ideal radix of four. This provides a quantitative indication that relaxing the radix constraint does not drastically degrade the topology properties.

(a) Adaptive 3D torus topology

Number of pre-selected links	% of routers with radix-4
10	96.46
20	95.67
30	95.18
40	94.65
50	94.36

(b) Adaptive cross-mesh topology

Number of pre-selected links	% of routers with radix-4
10	99.95
20	99.76
30	99.45
40	99.16
50	98.72

Table 1: **Impact of low (<4) active ports in routers.** The Monte Carlo analysis shows that even when relaxing the number of active ports constraint, almost all routers still achieve an ideal radix of four. Even with 50 pre-selected links, 94% (99%) of the routers achieve an ideal radix of 4 for adaptive 3D torus (adaptive cross-mesh) topology.

5. EXPERIMENTAL RESULTS

We evaluated HiROIC on a cycle-accurate trace-driven multi-core simulator. Table 2 shows the characteristics of the processors and the NoC we evaluated. We ran all experiments considering a 64 core system as a baseline. The traces for applications were obtained using the PIN [16] instrumentation tool. The simulator further incorporates a detailed model for the NoC with 3-stage pipelined routers. We implemented the HiROIC scheme over the adaptive cross-mesh and adaptive 3D torus topologies discussed in Section 4. All our comparisons are against a baseline 2D mesh topology. Finally, we used the up*/down* routing algorithm [20] to navigate the packets within the NoC.

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 values for cache misses per kilo instructions (MPKI): 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 in 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.

(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 2: **Experimental CMP: configuration of processor and network.**

Area Overhead. As discussed previously, the HiROIC scheme is implemented by glue logic surrounding the core router logic. The glue logic comprises only multiplexers, which are very small compared to the size of the router, rendering HiROIC's area overhead negligible. The adaptive 3D torus topology has slightly higher area overhead compared to the adaptive cross-mesh topology (due to more multiplexers in the glue logic), but the overall overhead is still small and it is not a vital factor in choosing a physical topology.

Power Overhead. The addition of multiplexers can lead to an increase in power dissipation. Longer (and more abundant) wires in HiROIC also lead to a slight 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. Same radix routers spend the same amount of dynamic energy per packet transmission. Therefore, by reducing the number of routers that packets hop through to reach the destination node, HiROIC also decreases the dynamic energy consumption. Thus, we argue that HiROIC more than compensates for the increase in power dissipation due to multiplexers and long wires.

Reconfiguration Duration. Whenever we trigger a topology reconfiguration phase, traffic is suspended and packets in flight are allowed to drain. The average number of cycles required to drain packets for each topology reconfiguration across all workloads is approximately 46 for the adaptive 3D torus topology. This overhead is small as the computational epoch is 10,000 clock cycles long. Apart from this overhead, the topology reconfiguration algorithm is executed in software on the supervisor node, potentially taking up to a few milliseconds to calculate the new topology and routing function. We expect to evaluate the reconfiguration duration as part of our future work, while also trying to develop techniques to reduce the reconfiguration duration. We also argue that the reconfiguration process, no matter how slow, can be masked by running in background, while still using the old topology and routing configuration. After the new topology and routing configuration is computed, the old configuration is discarded and normal operation can resume after packet drain. This reduces HiROIC's responsiveness to application behavior, but still allows it to capture traffic behavior in longer phases. Note that we have taken the packet drain time into account in our experimental evaluation.

Frequency of Reconfiguration. The HiROIC scheme tries to adapt to communication needs quickly. Therefore, for high-imbalance workloads, the topology reconfiguration happens more frequently. For workloads with only slightly imbalanced applications, the number of topology reconfigurations are fewer. The reason behind this is that for such workloads the traffic originating and terminating at all nodes is more or less the same, and HiROIC decides against topology reconfiguration due to lack of favorites. Quantitatively speaking, we found out that the average number of topology reconfigurations per workload is 102, where each workload was executed for 1,000 computational epochs.

Suitable T_{th} Value to invoke the topology reconfiguration. As discussed in Section 3, one of the conditions for triggering the topology reconfiguration is that FCPs share more than a preset fraction of total network traffic. We conducted the experiments by simulating all the workloads on an adaptive 3D torus topology with different T_{th} values, and comparing the results on the basis of average packet latency. We found out that the optimal T_{th} to trigger a topology reconfiguration is 60% for our baseline system. It gives the best trade-off between the capability to quickly adapt to communication needs and the frequency of topology reconfiguration. If the value of T_{th} is higher than 60%, then HiROIC is not able to adapt to changing application needs and the workloads do not see much latency improvement. However below 60%, topology reconfigurations happen more frequently, often causing unnecessary suspension of traffic, which in turn, nullifies the benefits of HiROIC. Figure 8 shows the average packet latency averaged across all the workloads for different values of T_{th}.

Reduction in Average Network Latency. Since HiROIC optimizes the hop count for a subset of communication paths, it provides solid

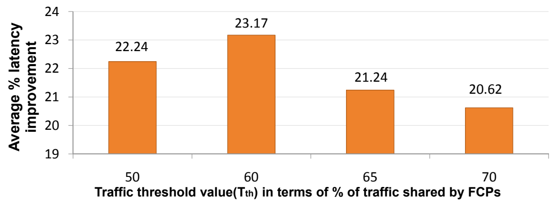

Figure 8: **Comparison of different T_{th} values on the basis of average packet latency.** The plot shows that a 60% T_{th} provides the best trade-off between application adaptivity and time spent in draining packets upon a reconfiguration trigger.

latency improvements for LH category workloads. By adapting to the communication needs, it facilitates the communication between the frequently communicating pairs and sends more amount of packets on the NoC, as compared to 2D mesh in the same time. We found that for LL category workloads, HiROIC still shows good improvement. This is because network transmissions are scarce for such workloads, and only a small subset of applications produce traffic within a certain 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 this active subset. Consequently, the network observes significant decrease in average packet latency.

However, in the case of HH workloads, HiROIC does not provide exciting latency improvements, as it reverts back to the better-balanced (baseline) mesh topology, as discussed in Section 3. Remember from Section 3 that HiROIC leverages congestion information for the decision to revert to a mesh topology. For an exhaustive analysis, we disabled this aspect of HiROIC, and monitored the latency characteristics of the system. In such a case, the latency degradation was as much as 20% in the worst case for low-imbalance high-load workloads. However, with the congestion detection and decision framework in place, HiROIC ensures that the adaptive topologies are never congested, by switching the topology back to a 2D mesh upon indications of congestion.

The average latency improvement for the adaptive 3D torus topology across LH and LL workloads is 23%. For adaptive cross-mesh, the average latency improvements across the same workloads is 6%. We conducted these experiments by setting the T_{th} to 60%. Even for the case of HH category workloads, we found that adaptive 3D torus provides a latency improvement of 19% on average. We further observe that the HiROIC-enabled adaptive 3D torus provides better latency improvements than the HiROIC-enabled adaptive cross-mesh. Upon closer inspection, we found that the reason for such behavior is two-fold: i) the average hop count between nodes for adaptive 3-D torus is 4.7, which is less than the average hop count of 5.4 for adaptive cross-mesh, and ii) for adaptive cross-mesh, reconfiguration results in some links sharing more traffic than others, leading to congestion. The adaptive 3D torus has more potential minimal paths between any two nodes, providing greater path diversity, and therefore, it handles congestion by better distribution of traffic.

Figure 9 shows the latency improvements for different categories of workloads. The chart shows that the adaptive 3D torus quickly adapts to the communication needs of the network and it never performs worse than a 2D mesh. Adaptive cross-mesh results show little benefits for some of the workloads because of the reasons described above. The average latency improvements across all workloads for adaptive 3D torus and adaptive cross-mesh are 22% and 5%, respectively.

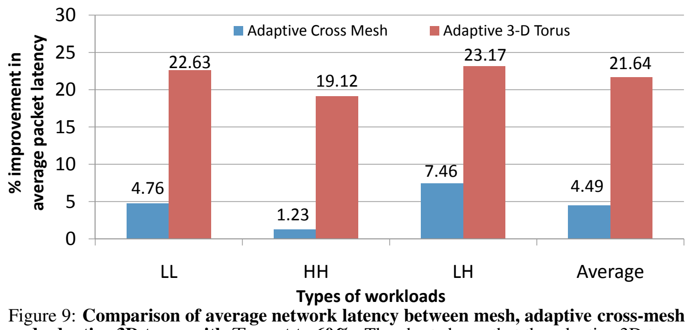

Figure 9: **Comparison of average network latency between mesh, adaptive cross-mesh and adaptive 3D torus with T_{th} set to 60%.** The chart shows that the adaptive 3D torus adjusts to communication needs quickly, and thus reduces the average hop count between busy nodes.

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. HiROIC proposes a topology reconfiguration infrastructure that decouples dynamic router ports from links in the network topology. In HiROIC, links are deployed abundantly for rich connectivity as in high-radix topologies, while the number of router ports are kept low. Router ports bind to links at runtime based on a traffic analysis heuristic that runs periodically at a central node. Unused links are power-gated to avoid any excess power dissipation. Our experiments with multi-programmed workloads on a 64-node CMP show that HiROIC reduces average network latency by 21% compared to an area- and power- comparable mesh.

Acknowledgements: This work was partially supported by NSF grant #0746425 and CFAR, within STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

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] D. Bertozzi, A. Jalabert, S. Murali, R. Tamhankar, S. Stergiou, L. Benini, and G. Micheli. NoC synthesis flow for customized domain specific multiprocessor systems-on-chip. *IEEE Trans. Parallel and Distributed Systems*, 16(2), 2005.

[4] J. Cong, C. Liu, and G. Reinman. Aces: Application-specific cycle elimination and splitting for deadlock-free routing on irregular network-on-chip. In *Proc. DAC*, 2010.

[5] R. Das, S. Narayanasamy, S. Satpathy, and R. Dreslinski. Catnap: energy proportional multiple network-on-chip. In *Proc. ISCA*, 2013.

[6] M. Faruque, T. Ebi, and J. Henkel. Configurable links for runtime adaptive on-chip communication. In *Proc. DATE*, 2009.

[7] 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.

[8] A. Kahng, B. Lin, K. Samadi, and R. Ramanujam. Trace-driven optimization of networks-on-chip configurations. In *Proc. DAC*, 2010.

[9] J. Kim. Low-cost router microarchitecture for on-chip networks. In *Proc. MICRO*, 2009.

[10] T. Krishna, C.-H. Chen, W. Kwon, and L.-S. Peh. Breaking the on-chip latency barrier using SMART. In *Proc. HPCA*, 2013.

[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] H. Matsutani, M. Koibuchi, D. Ikebuchi, K. Usami, H. Nakamura, and H. Amano. Ultra fine-grained run-time power gating of on-chip routers for cmps. In *Proc. NoCs*, 2010.

[13] R. Mullins, A. West, and S. Moore. Low-latency virtual-channel routers for on-chip networks. In *Proc. ISCA*, 2004.

[14] M. Palesi, R. Holsmark, S. Kumar, and V. Catania. *IEEE Trans. Parallel and Distributed Systems*, 20(3), 2009.

[15] R. Parikh and V. Bertacco. uDIREC: unified diagnosis and reconfiguration for frugal bypass of NoC faults. In *Proc. MICRO*, 2013.

[16] H. Patil, R. Cohn, M. Charney, R. Kapoor, A. Sun, and A. Karunanidhi. Pinpointing representative portions of large intel itanium programs with dynamic instrumentation. In *Proc. MICRO*, 2004.

[17] L.-S. Peh and W. Dally. A delay model and speculative architecture for pipelined routers. In *Proc. HPCA*, 2001.

[18] E. Rijpkema et. al. Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip. In *Proc. DATE*, 2003.

[19] P. Salihundam, S. Jain, T. Jacob, S. Kumar, V. Erraguntla, Y. Hoskote, S. Vangal, G. Ruhl, and N. Borkar. A 2 TB/s 6x4 mesh network for a single-chip cloud computer with dvfs in 45 nm cmos. *IEEE Journal of Solid-State Circuits*, 46(4), 2011.

[20] 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.

[21] 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.