The Runahead Network-On-Chip

Zimo Li
University of Toronto
zimo.li@mail.utoronto.ca

Joshua San Miguel
University of Toronto
joshua.sanmiguel@mail.utoronto.ca

Natalie Enright Jerger
University of Toronto
enright@ece.utoronto.ca

ABSTRACT

With increasing core counts and higher memory demands from applications, it is imperative that networks-on-chip (NoCs) provide low-latency, power-efficient communication. Conventional NoCs tend to be over-provisioned for worst-case bandwidth demands leading to ineffective use of network resources and significant power inefficiency; average channel utilization is typically less than 5% in real-world applications. In terms of performance, low-latency techniques often introduce power and area overheads and incur significant complexity in the router microarchitecture. We find that both low latency and power efficiency are possible by relaxing the constraint of lossless communication. This is inspired from internetworking where best effort delivery is commonplace. We propose the *Runahead NoC*, a lightweight, lossy network that provides single-cycle hops. Allowing for lossy delivery enables an extremely simple bufferless router microarchitecture that performs routing and arbitration within the same cycle as link traversal. The Runahead NoC operates either as a *power-saver* that is integrated into an existing conventional NoC to improve power efficiency, or as an *accelerator* that is added on top to provide ultra-low latency communication for select packets. On a range of PARSEC and SPLASH-2 workloads, we find that the *Runahead NoC* reduces power consumption by 1.81× as a *power-saver* and improves runtime and packet latency by 1.08× and 1.66× as an *accelerator*.

1. INTRODUCTION

With increasing on-chip core counts, networks-on-chip (NoCs) are an effective way of communicating between these many components. However, NoCs consume a significant amount of power in modern chip multiprocessors (CMPs) [26, 42], and energy efficiency has been a primary concern for researchers and designers [10, 11]. Reducing the power of the NoC while increasing performance is essential for scaling up to larger systems for future CMP designs.

Minimizing power consumption requires more efficient use of network resources. Though buffers consume a significant portion of network power and area [26], traditional NoC designs tend to provision large amounts of buffers to meet worst-case throughput requirements. Yet large buffers are often unnecessary as single-flit packets represent a high fraction of the total network traffic in real applications [33]. Several bufferless NoC designs have been proposed in the past [18, 23, 35, 37]. These designs achieve significant power savings at a cost of lower saturation throughput compared to conventional buffered routers. NoC channel utilization of single-threaded and multi-threaded CMP workloads tends to be low, with average injection rates of only 5% [5, 22, 25]. Low resource utilization translates into inefficient use of network resources. To address this, several multi-NoC systems have been proposed in the past [1,16,17,19,39,40,44]. Multi-NoCs use total bandwidth more efficiently since they can be designed with heterogeneous physical subnetworks; messages can be categorized and injected into different networks depending on packet type. For example, latency sensitive messages are injected into a low-latency, high-power network, while non-critical messages are injected into a low-power network [1, 40].

Minimizing NoC latency is essential to meet the higher communication demands of future CMPs. Techniques include reducing the number of router pipeline stages through lookahead routing [20] and bypassing via express virtual channels [31]. Non-speculative single-cycle routers allocate router switches in advance of packet arrival [30]. Route predictions can also reduce NoC latency [24,34]. Though these designs improve performance, they come at a cost of increased complexity, power and area.

We propose the *Runahead NoC*[1] which serves as 1) a *power-saver* that exploits heterogenous traffic for more efficient use of network resources, or 2) an *accelerator* that provides lightweight, low-latency communication on top of a conventional NoC. The Runahead NoC is designed for simplicity; it is bufferless with a lightweight router architecture which consumes very little area and power. To accomplish this simplicity, the Runahead NoC is lossy, allowing packets to be dropped in the presence of contention. It is inspired by the "best ef-

[1]The proposed network allows select packets to take a head start compared to the rest of the network traffic, hence the name *Runahead*. The name for our network is also inspired by runahead execution [38] which allows the processor to speculatively prefetch loads from the instruction window to tolerate long latency operations. The Runahed NoC is an orthogonal design that could be easily combined with processor optimizations such as runahead execution.

978-1-4673-9211-2/16/$31.00

fort" concept in internetworking, meaning that there is no guarantee a packet will arrive at its destination.[2] Our design is not meant to be a stand-alone network; it is meant as a plug-and-play network that either replaces resources in an existing NoC to save power or is added on top as an accelerator.

Contributions. We make the following contributions:

- Propose the Runahead NoC, which through its simplicity, provides single-cycle hops and "best effort" delivery for latency-sensitive packets;
- Evaluate the Runahead NoC as a *power-saver* and show that it achieves 1.81× and 1.73× savings in power and active silicon area while still providing 1.33× lower latency (1.05× application speedup);
- Evaluate the Runahead NoC as an *accelerator* and show that it improves packet latency by 1.66× on average (1.08× application speedup), with only 10% and 16% overheads in power and active area.

2. THE RUNAHEAD NETWORK

In this section, we present our Runahead NoC architecture which is lightweight, lossy, and achieves a single-cycle per-hop latency. It must be paired with a regular lossless NoC to provide guaranteed delivery of all packets. The Runahead NoC can be 1) added to a regular NoC as an *accelerator*, providing low-latency transmission of latency-sensitive packets, or 2) integrated into a regular NoC as a *power-saver*, providing power and area savings without harming performance.

As an Accelerator: When used as an accelerator, the configuration of the existing regular NoC is left unchanged and its operation is undisturbed. All packets are injected into the regular NoC, while only latency-sensitive single-flit packets are injected into the Runahead NoC. Multi-flit packets are excluded to minimize the complexity when one or more flits of a packet are dropped. The Runahead network carries all coherence control packets, which are typically single flit. It also carries single-flit data response packets. These packets are sent in response to a cache miss and only contain the critical word (i.e., the initially requested word) of the data block. This is described in Section 2.3. Since the regular NoC is lossless, any packets dropped by the Runahead NoC will still arrive at their destination. The goal of the accelerator is to provide an opportunity for ultra-fast delivery of latency-sensitive packets while incurring low power and area overheads.

As a Power-Saver: When used as a power-saver, the existing regular NoC is under-provisioned to allow for the integration of the Runahead NoC. As in the accelerator case, the Runahead NoC only carries latency-sensitive single-flit packets. The regular NoC still carries all packets to guarantee delivery of any packets that may be dropped. In our experiments, we assume a regular multi-NoC system and replace one of the subnetworks with our Runahead NoC. The Runahead network consumes very little power and area. This is because it is bufferless and consists of a simplified router architecture with no routing tables nor complex arbiters. Despite the increased congestion in the smaller regular NoC, overall application performance is unharmed since latency-sensitive packets are transfered quickly. The goal of the power-saver is to minimize area and power consumption while maintaining comparable or better performance.

Overview. In the following sections, we first give a high-level overview of our Runahead router architecture (Section 2.1). We then describe how routing computation and port arbitration are performed (Section 2.2), enabling the single-cycle per-hop latency. Finally, we discuss critical word forwarding for data response packets (Section 2.3) and how to integrate our Runahead NoC with the regular NoC (Section 2.4).

2.1 The Runahead Routers

To achieve single-cycle hops and ensure low area and power consumption, the routers in the Runahead network need to be simple. In this work, we target a 2D mesh, which is commonplace in modern systems (e.g., Tilera [45] and Intel Terascale [26]). Figure 1 illustrates the design of the Runahead router. It consists of five multiplexers: one for each of the output ports of the four cardinal directions and one for the ejection port. The Runahead routers share the same injection port as the routers in the regular network. Runahead routers are bufferless. Only four latches are needed to store up to four single-flit packets that may come in from the input ports of the 4 cardinal directions at any cycle. Injected packets are stored in the input buffer in the regular router. Header information is extracted from incoming packets at the input ports and directed to the routing computation and port arbitration unit. For clarity, data connections are not shown in the figure.

Lossy Delivery. Our Runahead routers use XY dimension-order routing (DOR), which greatly simplifies routing and arbitration. Port arbitration directs packets from input ports to their corresponding output ports and determines which packets to drop in the event of a conflict. The Runahead router does not collect dropped packets, and the Runahead NoC does not try to forward them again. This is different from prior work, such as SCARAB [23] and BPS [21], where a NACK is sent to the source for packet re-transmission. In the Runahead NoC, the dropped packet will always be delivered by the lossless regular NoC. The Runahead NoC is inherently deadlock-free since packets are dropped instead of blocked upon conflicts. This eliminates the need for complex deadlock prevention mechanisms. Section 2.2 describes the routing computation and port arbitration.

Single-Cycle Hops. Unlike conventional virtual-channel routers with 3 to 5 pipeline stages, the Runahead router delivers packets in a single cycle per hop. Route computation, port arbitration and link traversal are all combined into a single step. The Runahead routers are hardcoded for XY DOR; no routing tables

[2]In contrast, "best effort" in NoC literature usually means that there is no guarantee on bandwidth and latency. Best effort NoCs have been explored in the context of quality of service [2, 32, 43].

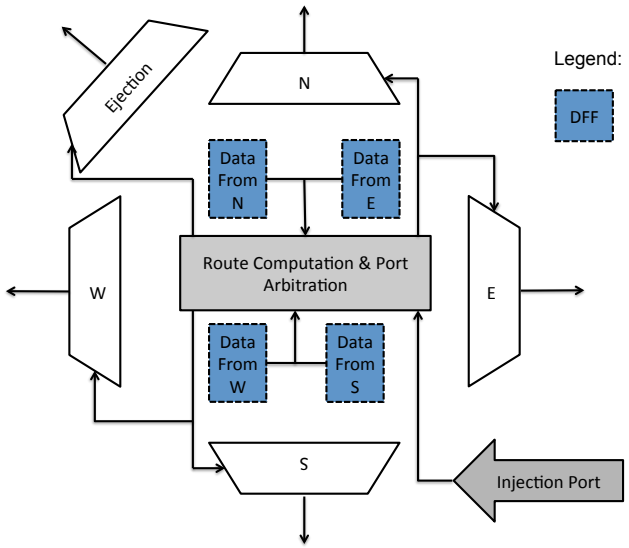


Figure 1: Runahead router design

are necessary. The output port of an incoming packet is quickly determined from the destination information in the header. By allowing for dropped packets, the Runahead router greatly simplifies port arbitration. Our design uses a fixed arbitration scheme where the priority of incoming packets is static for each output port. Specifically, our port arbitration scheme always prioritizes packets going straight over packets turning. Thus the logic that controls the output port multiplexers is very simple, allowing the entire process to fit within one cycle. Since all hops are single cycle, the latency of a packet is fully deterministic from source to destination, assuming it is not dropped along the way. The latency in cycles is equal to the number of hops travelled. Because of this, it is impossible for a packet to arrive on the regular NoC earlier than on the Runahead network.

Single-Flit Packets. Since there are no buffers, the Runahead router does not have virtual channels, just a single physical channel per output port. Handling multi-flit packets introduces too much complexity in the Runahead NoC since at any moment, any flit can be dropped. As a result, a multi-flit packet could arrive at its destination with some missing flits. We would need additional complexity at the cache controllers and memory controllers to support incomplete data packets. Thus, to minimize overheads and complexity, our design only supports single-flit packets.

2.2 Route Computation and Port Arbitration

Route computation and port arbitration are performed together to allow packets to traverse each hop in a single cycle. The destination is encoded in the header as signed X and Y values that indicate the relative number of hops remaining until the destination. The sign indicates the direction of travel. Route computation is a simple matter of determining the correct output port based on these values.

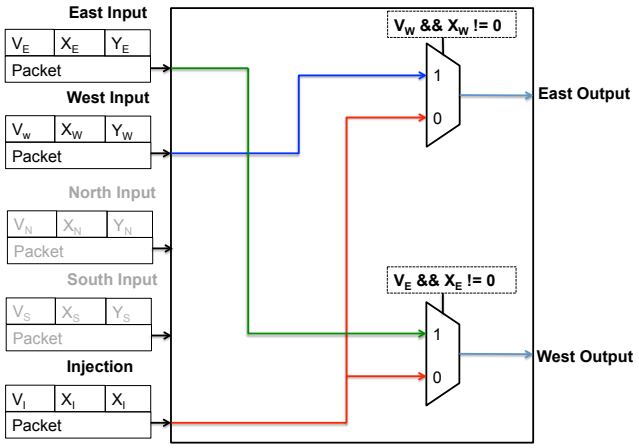
We employ a fixed arbitration scheme for each output port. A packet that is going straight has higher priority than a packet that is turning. If packets from two different input ports are turning towards the same output port, one of the input ports is hardcoded to always take precedence. For example, if both the east and west input ports are contending for the north port, the arbitration always selects the west port. Similarly, for the ejection port, arbitration is hardcoded such that specific input ports always win. This minimizes complexity and allows us to combine the route computation and port arbitration steps into a single cycle.

With XY DOR routing, our fixed arbitration scheme yields only three places where a packet can be dropped: 1) At injection, 2) When the packet is making a turn from the X to Y direction, or 3) At ejection when a routing conflict occurs at the ejection port. This applies to all packets no matter how far they are traveling. Thus the number of places where a packet can be dropped is constant and does not scale with network size.

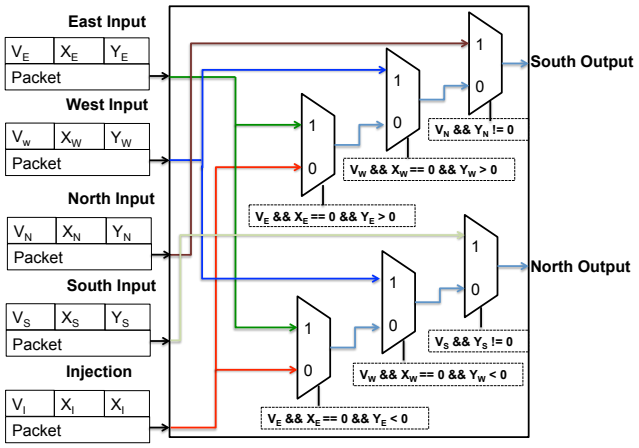
The route computation and port arbitration unit is shown in Figure 2. The inputs are obtained from the header information of the incoming packets at each input port. The required signals from each input packet are denoted by $X_{direction}$, $Y_{direction}$, $V_{direction}$, which correspond to the destination X and Y values and a valid bit; the valid bit indicates that there exists a valid packet waiting at the input port. The figure shows the logic for different output ports. In parallel with route computation and port arbitration, the X or Y value in the packet header is updated for the next hop.

East and West Output: The advantage of using XY DOR is that it simplifies east and west routing and arbitration, as shown in Figure 2a. The east and west direction output ports only need to consider the latches of their opposing input ports, as well as the injection port. Anytime a packet arrives at either the east or west input port with a non-zero X value, it is guaranteed to be forwarded straight since it has the highest priority in our fixed arbitration scheme. It is impossible for a packet to turn on to either the E or W directions. It is also impossible for a packet to be forwarded back to the direction from which it arrived.

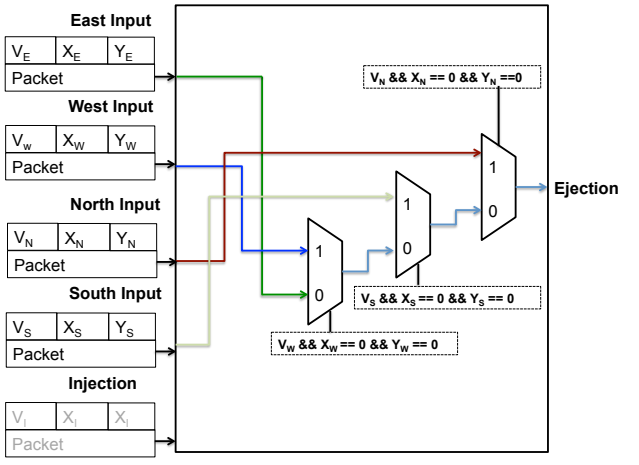
North and South Output: Routing and arbitration are more complicated for the north and south output ports since they need to consider packets that are turning. Figure 2b shows that arbitration is hardcoded such that the outermost multiplexer always chooses the opposing input port if there is a valid incoming packet; this enforces our straight-first arbitration. The logic for this only needs to look at the header's Y value to check that the packet has not reached its destination. If there is no valid packet at the opposing input port, the fixed arbitration scheme first checks to see if the west input port is turning, followed by the east input port, and finally the injection port. In our implementation, a packet at the west input port always takes precedence over a packet at the east port when both of them are trying to turn to the same output port. A packet at the east or west input port is determined to be turning if it contains a zero X value with a non-zero Y value. Note



(a) Structure for E and W output



(b) Structure for N and S output



(c) Structure for Ejection

Figure 2: Route computation and port arbitration.

that a packet traverses at most three 2-to-1 multiplexers from its input port to its output port, keeping the critical path delay low.

Ejection Output: The ejection port, shown in Figure 2c, is similar to that of the north and south output ports. Incoming packets are ranked based on the following order of input ports: N, S, W, E. As with the north and south output ports, an ejecting packet traverses at most three multiplexers. To determine if a packet is destined for the ejection port, both the X and Y values need to be zero. A packet can never be injected with the same source and destination nodes, thus eliminating the need to connect the ejection port to the injection port.

Starvation and Fairness. Hardcoding arbitration leads to some potential unfairness or starvation. However, since all data injected into the Runahead NoC is also injected into the regular NoC, forward progress for the application is guaranteed. Our port arbitration scheme is stateless. There is no mechanism to detect and prevent starvation. For example, when arbitrating for the north output port, packets at the south input port will always take precedence over those at the east and west input ports even if it means the packets at these two ports are always dropped. This keeps complexity at a minimum, allowing for low overhead and ensuring that the design fits within a single clock period. The goal of the Runahead network is not to provide fair communication because the regular NoC would already provide such a platform. In Section 4, we evaluate unfairness; in practice, arrival rates are relatively uniform across all source nodes. Due to low contention in these networks, packets are often delivered successfully, mitigating any concerns about fairness. Without mechanisms to prevent starvation and ensure fairness, the Runahead NoC has less overhead once it is combined with a regular NoC.

2.3 Critical Word Forwarding

The Runahead network is meant to carry only latency-sensitive packets. In a cache-coherent CMP, all control packets (i.e., requests, invalidations, acknowledgements) and data response packets are latency-sensitive. However, the Runahead network is designed for single-flit packets to avoid the complexity of dropped flits in multi-flit packets. As a result, data response packets cannot be carried on the Runahead network. Fortunately, 67% of the critical words are the first word in a cache block in real applications [12]. Also, many modern CMPs can support critical word forwarding. When composing the data response packet, the initially requested (critical) word is sent in the first flit. This way, when the first flit arrives at the L1 cache, the critical word is forwarded directly to the processor so that it can continue executing before the rest of the packet has arrived. Naturally, the critical word is the most latency-sensitive word in the data block. Thus in our implementation, we assume critical word forwarding and inject the first flit of all data response packets (bound for the L1 cache) into the Runahead network.

2.4 Integration into the Regular Network

The Runahead network can be easily integrated with an existing network. Its routers' injection and ejection ports are simply connected to the injection and ejection queues of the regular network. The injection port of each Runahead router connects directly to the head of

the regular input buffer, so that single-flit packets waiting to be injected into the regular network are also injected into the Runahead network. In our experiments, we find that a large portion of packets are dropped at injection, accounting for up to 50% of all dropped packets in the Runahead network. This is because in port arbitration, packets from the injection ports have lowest priority, as explained in Section 2.2. To improve this, we design the Runahead router to try to inject a packet multiple times for as long as it is at the head of the injection queue. If the packet at the head of the queue does not succeed in arbitrating for its output port, we try again in the next cycle if the packet is still at the head (i.e., if the packet has not yet been injected into the regular network either). If the packet is injected into the Runahead network successfully, a flag is set at the input port so that we do not try to inject it again in subsequent cycles.

The ejection ports connect to the regular output buffers. When a packet is ejected from the Runahead network, it is immediately forwarded to the corresponding cache controller or memory controller. It is then stored in a small buffer until the same packet is ejected from the regular network. This ensures that packets that are successfully delivered via the Runahead network are not sent to the controllers twice. Note that a packet will never be ejected from the regular network before the Runahead network, as discussed in Section 2.1. For the applications we have studied, the maximum number of entries that a buffer needs to hold is 15. Conservatively assuming packet IDs of 8 bytes, this buffer would be less than 128 bytes which is small compared to the size of buffers in regular NoC routers. In the unlikely event that the buffer is full, the network interface will discard packets that arrive on the Runahead network. This is safe since any packet that arrives in the Runahead network will also arrive on the regular network.

2.5 Discussion

This section discusses and reiterates some key points in our design. The Runahead network does not compromise correctness in the communication fabric despite being a lossy design, since it serves as a companion to a lossless network. The use of such specialized networks with general-purpose networks is timely in the dark silicon era, providing efficiency gains analogous to accelerators for general-purpose cores. As discussed previously, to ensure correctness, the Runahead network requires buffers at ejection ports for packets that are still in-flight in the lossless network. However, this does not introduce much overhead; the size of these buffers can be fixed and does not need to scale with network size nor network usage. In the rare event that the buffers are full, packets can simply be dropped upon ejection without compromising correctness. The drop rate also does not scale with network size due to the fact that for any given packet, there will always be 3 (and only 3) places where the packet can be dropped: 1) at injection, 2) when turning, and 3) at ejection. Furthermore, the drop rate does not necessarily scale with network usage; the Runahead network can handle high network load by being selective. Our design thrives on the common case of low network load. However, at high load, the Runahead network can simply be more selective when injecting packets, selecting only those deemed to be most latency critical.

Topology	4×4 mesh (8×8 for SynFull)
Channel width	8 byte
Virtual channels	6 per port (4 flit each)
Router pipeline stages	3
Routing algorithm	X-Y dimension-order
Flit size	8-byte
Packet size	1 (Control) / 9 (Data) Flits

Table 1: Baseline network simulation parameters

Topology	4×4 mesh (8×8 for SynFull)
Channel width	10 byte (8B for flit, 2B for other metadata)
Virtual channels	None
Routing algorithm	X-Y dimension-order
Flit size	8-byte

Table 2: The Runahead network simulation parameters

# of Cores/Threads	16/16, 1GHz
Private L1 Cache	16KB 4-way LRU 64Byte blocks
Shared L2 Cache	fully distributed, 8-way LRU, 4 MB total
# of Directories	4 directories located at each corner of the topology
Cache Coherence	MOESI distributed directory

Table 3: Full-system simulation system parameters

3. METHODOLOGY

We evaluate the effectiveness of the Runahead network in conjunction with a baseline lossless network. The configuration parameters for our baseline network are listed in Table 1. The Runahead network simulation parameters are listed in Table 2.

NoC Configurations. We compare our proposed Runahead network against conventional lossless NoCs, some of which are multi-NoC designs. We also compare the Runahead network against two existing designs: Aergia [15], a prioritization scheme, and DejaVu switching [1], a multi-plane NoC design. The configurations are listed below:

- **Baseline64*:** This configuration has a single lossless NoC with 64-bit channels, as in Table 1.
- **Baseline128_Random*:** In this configuration, the NoC is composed of two independent lossless networks, each configured as in Baseline64. Total channel width is 16 bytes (two 64-bit networks). The workload is shared evenly between the two NoCs (i.e., 50% of traffic is randomly injected into each network).
- **Baseline128_Select†:** In this configuration, the

*Used in both full-system and SynFull evaluations.

†Used only in SynFull evaluations.

NoC is configured identically to that of Baseline128_Random. However, instead of sharing the traffic evenly, Network 1 is responsible for latency-sensitive traffic (i.e., packets that we would inject into the Runahead network). This includes single-flit packets and critical words. Network 2 handles all other traffic. Since delivery is guaranteed, single-flit packets are only injected into Network 1 instead of both networks.

- **Aergia**[‡] is a prioritization scheme that uses the notion of slack to determine the priorities of packets [15]. Aergia calculates packet priority based on local slack which is defined to be the number of cycles a packet can be delayed without delaying any subsequent instructions. Aergia uses L2 hit/miss status, number of L2 miss predecessors and number of hops to estimate the packet slack. In our full-system simulations, we use the same setup as the Baseline64 network. We modify Aergia's prioritization scheme for allocations and arbitrations; we conservatively assume a perfect L2 miss predictor for accurate slack calculations.
- **DejaVu Switching**[‡] is a multi-plane NoC design where single-flit control packets and multi-flit data packets are separated in to different planes [1]. Reservation packets are sent out on the control plane to reserve network resources on the data plane routers. The reservations enable the data packets to be forwarded without suffering delays from making routing decisions at every router. In our full-system simulations, the control plane uses the same parameters as the baseline network listed in Table 1. The reservation packets are sent 3 cycles ahead of data packets and they travel in the control plane. We assume the reservation queues in the data plane have infinite size. To model the simplified router design in the data plane of DejaVu switching, we use routers with single cycle router delay and one VC. Also, we forward the critical words to the processor as soon as the head of the data packets arrive. Though DejaVu switching can use a slower data plane for energy savings, we opt not to for a conservative performance comparison against Runahead.
- **Runahead*:** In this configuration, we have a single Baseline64 network, which carries 100% of the injected packets, along with the proposed Runahead network that carries latency-sensitive packets (i.e., single-flit packets and critical words). As delivery is not guaranteed in the Runahead network, duplicate injection of latency-sensitive packets into both networks is required. The total channel width in this case is 18 bytes (8 bytes for the Regular network and 10 bytes for the Runahead network). Note that we allocate two extra bytes to the Runahead network channel width to conservatively to account for any additional metadata for supporting critical word forwarding. This does not give our Runahead network a performance advantage since all packets are single-flit; in fact, it incurs a power and area disadvantage.

As an accelerator, Runahead is evaluated relative to Baseline64 and compared against Aergia. As a power-saver, Runahead is evaluated relative to the Baseline128 configurations and compared against DejaVu Switching.

Synthetic Traffic. We evaluate latency and throughput of the Runahead network under several synthetic traffic patterns covering a wide range of network utilization scenarios. We use a modified version of Booksim, a cycle-level network simulator [28]. All configurations use an 8×8 2D mesh network, and we assume all packets are single-flit.

Full-System Simulation. To evaluate the real system performance of our Runahead network, we use Booksim and Gem5 [8]. The system parameters are listed in Table 3. All network configurations use a 4×4 mesh topology with parameters listed in Tables 1 and 2. The full-system simulation workloads consist multi-threaded workloads from SPLASH-2 [46] and PARSEC [7]. For each multi-threaded workload, we run 16 threads with the simmedium input set until completion. We measure the execution time in the application's region of interest. For full-system simulations, the memory controllers are located at the corners of the mesh network. We keep the cache sizes small to provide greater stress on the network. This does not give the Runahead network an advantage because it drops more packets when there is more network contention.

SynFull Workloads. To further evaluate our Runahead network design in a larger network, we use multi-programmed SynFull traffic workloads [3] with Booksim. SynFull workloads are designed to reproduce the cache coherent behavior of multi-threaded applications from SPLASH-2 [46] and PARSEC [7]. These workloads consist of single-flit control packets and multi-flit data packets. All configurations use an 8×8 2D mesh network. Other network configuration parameters are the same as previous experiments. For each 64-core workload, we run 4 identical instances of a 16-way multi-threaded application. Each instance is assigned a 4×4 quadrant of cores. For SynFull, memory controllers are located at the left and right edge nodes of the 8×8 mesh. All four instances send memory traffic throughout the chip. To keep measurements consistent across all configurations, we only measure the latency of unique packets that are seen by the application. This means that if a packet is injected into both the Runahead and regular lossless networks, we only measure the latency of the packet that arrives first; subsequent arrival of the same packet is discarded by the NoC. For data packets, latency is taken for the entire packet to arrive, not just the critical word. To measure the potential benefit of accelerating critical words, we report the difference in arrival times between the critical word and the rest of the data block. We did not compare with Aergia and DejaVu switching as Aergia relies on information from real system conditions and DejaVu generates extra reservation packets; neither of these can be easily modeled with SynFull.

Power and Area. We model power and area us-

[‡]Used only in full-system evaluations.

ing DSENT [41] and RTL. DSENT results are collected using a 22nm bulk/SOI, low-V process node. Dynamic power is obtained by modelling the system using the average injection rates collected from the SynFull workloads. To ensure the feasibility of the Runahead router, we use an open source RTL router design [6] as a conventional router. The Runahead router is constructed on top of the existing RTL design. We use Synopsys design compiler with TSMC 65nm technology to evaluate the power and area for a single Runahead router.

4. EVALUATION

This section provides performance, power and area evaluations of our proposed Runahead network. We first evaluate latency and throughput under synthetic traffic. We then evaluate the performance improvements in the Runahead network in full-system simulation, followed by a performance evaluation of the Runahead network using real application models from SynFull. We then measure area and power consumption of the Runahead network.

4.1 Latency and Throughput

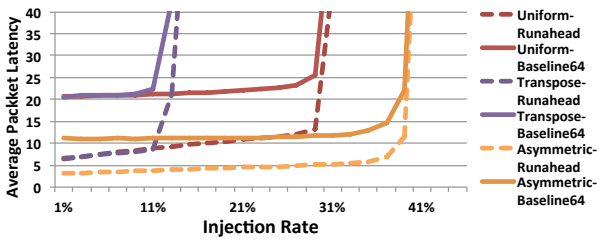
Figure 3 shows the average packet latency for the Baseline64 and Runahead configurations on different synthetic traffic patterns. All simulations are done using single-flit packets, and all packets are injected into the Runahead network. The Runahead NoC shows a significant decrease in average packet latency compared to Baseline64. Note that the packet latency increases faster prior to saturation in the Runahead NoC for several traffic patterns. This is because as injection rate increases, the arrival rate decreases, leading to lower effectiveness of the Runahead network; more packets rely on the regular lossless network for delivery.

Both NoC setups saturate around the same injection rate for most traffic patterns. This is because our Runahead network's injection ports are connected to the same injection queues of the routers in the regular lossless network. If congestion occurs at injection in the lossless network, the Runahead network does not provide any benefit. However, in other traffic patterns such as Bit Reverse where congestion occurs within the network rather than at the injection ports, the Runahead NoC saturates at a higher injection rate.

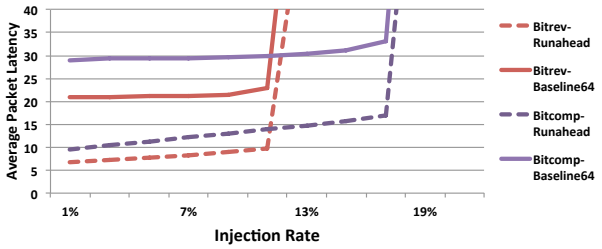
4.2 Full-System Simulation

In this section, we evaluate the performance of Runahead as an accelerator and as a power-saver, compared against various NoC configurations (Section 3). We simulate benchmarks from PARSEC and SPLASH-2 using the Gem5 simulator.

As an Accelerator. We compare speedup between Baseline64, Aergia and Runahead. The speedups are shown in Figure 4. These results are normalized to Baseline64, since the Runahead configuration is essentially our proposed design added on top of the baseline 64-bit lossless network. We first notice that in our experiments, Aergia has little impact on system performance. The reasons are twofold. First, most PARSEC



(a) Uniform Random, Transpose and Asymmetric



(b) BitRev and BitComp

Figure 3: Load-latency curves under synthetic traffic

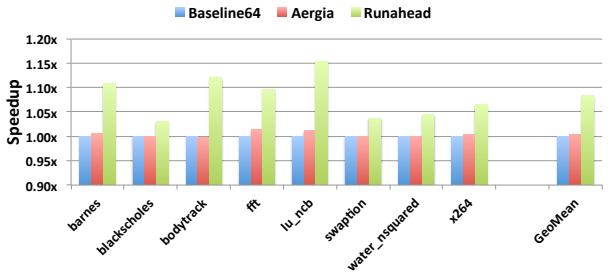


Figure 4: Runahead speedup as an accelerator

and SPLASH benchmarks do not have high L2 miss rates, which is one of the key factors in computing priority in Aergia.[6] Second, the benchmarks have very little congestion. One of our key motivations is that average channel utilization is low in real-world applications. Because of this, prioritization schemes are unlikely to find opportunities to accelerate packets in the absence of congestion. On the other hand, Runahead achieves 1.08× speedup compared to the baseline (Baseline64).

As a Power-Saver. When using the Runahead network as a power-saver, speedup is shown in Figure 5. Unlike in the previous section, these results are normalized to Baseline128_Random, since the Runahead configuration effectively under-provisions the baseline 128-bit lossless NoC to make space for our proposed design. DejaVu has a speedup of 1.035× compared to Baseline128_Random. Runahead delivers a greater speedup of 1.045×, due to its lower per-hop latency for control packets and critical data words. Note that DejaVu has an advantage in situations where applications tend to access other words in the cache line (aside from the crit-

[6]Aergia was originally proposed and evaluated using multi-programmed SPEC workloads.

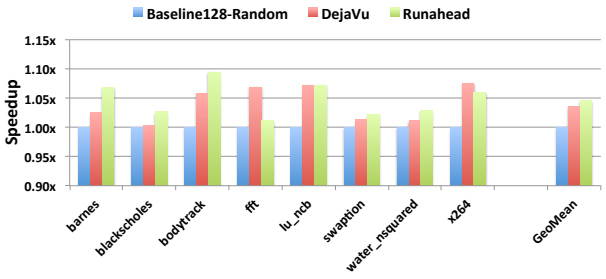


Figure 5: Runahead speedup as a power-saver

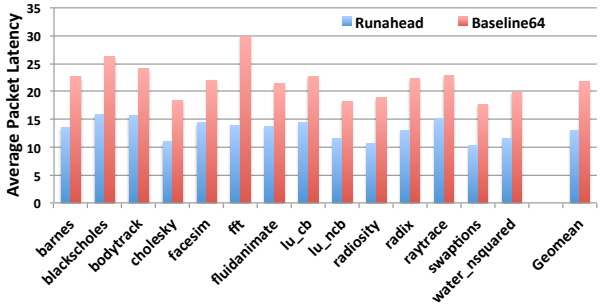


Figure 6: Average packet latency as an accelerator

ical word) much earlier. This is because data packets, as a whole, travel faster in DejaVu due to advanced reservations at data plane routers. The separation of data and control packets in DejaVu causes both networks to be less congested, unlike in Runahead where the smaller lossless network carries all types of packets. Although we forward the critical word in both cases, DejaVu populates lines in the cache sooner than Runahead after unblocking the stalled processor with the critical word. This explains the higher speedups for fft, lu_ncb and $x264$ compared to the Runahead network. Despite this, the Runahead network achieves 1.045× speedup while using fewer network resources compared to both Baseline128_Random and DejaVu switching.

We obtain Runahead network activity by collecting results without the critical word forwarding optimization. Critical word packets are single-flit duplicated packets that travel in the Runahead network; to keep the percentage of single-flit packets accurate, we do not include these packets when collecting the results. Considering that data packets are 9 flits in size, on average, only 23% of all flits that travel through the network are injected into the Runahead network. We observe that applications with a higher percentage of single-flit packets see more performance benefit from the Runahead NoC. Fortunately, since over 72% of packets in the network are single-flit, the Runahead network is still capable of speeding up the majority of network messages despite the fact that it only carries 23% of the flit traffic. The Runahead NoC is most effective at improving performance if a large fraction of packets are successfully delivered; all applications studied have over 95% arrival rate.

4.3 SynFull

In this section, we evaluate Runahead as an accelerator and as a power-saver given a larger network topology, compared against varying NoC configurations (Section 3). We simulate 14 different SynFull applications.

As an Accelerator. Figure 6 compares the average packet latency between Runahead and Baseline64. Recall that in the Runahead configuration, the NoC is configured with a regular lossless network (identical to Baseline64) augmented with our proposed Runahead design, which serves as an accelerator. Given that the our design offers very low per-hop latency, Runahead achieves 1.66× faster packet delivery on average. Note that the performance increase is significant since arrival rates are very high, even with a larger network topology, as we show later in this section.

As a Power-Saver. Figure 7 compares the average packet latency of Runahead against the two 128-bit baselines: Baseline128_Random and Baseline128_Select. The packet latency is normalized to Baseline128_Random. Recall that both of these baselines are configured with two 64-bit lossless subnetworks each; the only difference is that Baseline128_Select selectively injects packets based on latency-sensitivity. Baseline128_Random generally performs better than Baseline128_Select due to better load balancing between subnetworks. Given that both subnetworks are identical, selective injection offers minimal latency improvement. However, the Runahead NoC performs the best across all benchmarks. On average, Runahead delivers packets 1.33× and 1.49× faster compared to Baseline128_Random and Baseline128_Select.

To further investigate the benefits of the Runahead network, we look at the cycle count between the time a critical word arrives and the time when the rest of the data block arrives in Baseline128_Select and the Runahead NoC in Figure 8. The Runahead network on average delivers critical words almost 23 cycles faster than the rest of the cache block.

The arrival rate of packets in the Runahead network for each application is listed in Table 4. The average arrival rate across all applications is over 97% and the average hop count is 3.7. This means that the Runahead network delivers almost all of the single-flit packets that travel through it, with an average latency of 3.7 cycles. In comparison, the lowest latency that can be achieved in the lossless regular network, which has a 3-stage pipeline router plus link traversal, is 14.8 cycles (3.7×(3+1)). As the majority of packets in the network are single flit, the use of the Runahead network enables average packet latency to drop significantly.

To investigate the fairness of our port arbitration scheme, we show the arrival rate per source node for blackscholes (Figure 9a) and fft (Figure 9b). Blackscholes exhibits the lowest arrival rate while fft exhibits the highest hop count, as shown in Table 4. Lower average arrival rate may be an indication that some nodes experience starvation especially when applications have light traffic. On the other hand, as the distances trav-

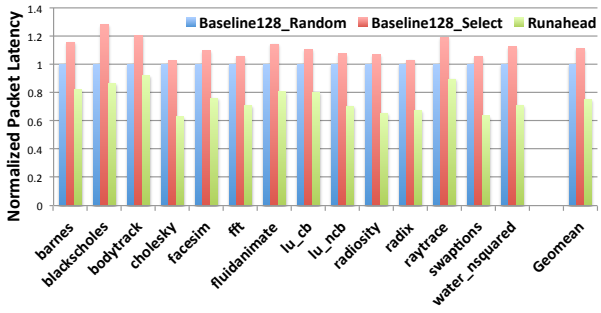


Figure 7: Normalized packet latency as a power-saver

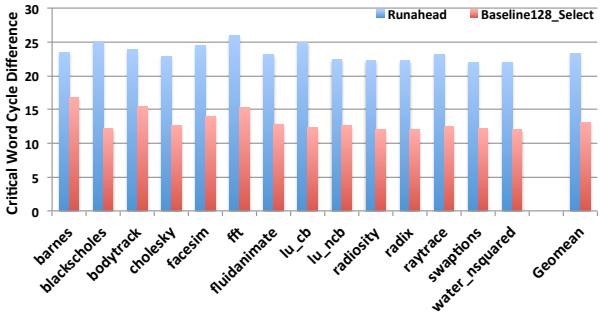


Figure 8: Average time difference between arrival of critical word and corresponding cache block

eled by packets increase, packets stay in the network longer and have a higher chance of causing contention in the Runahead network. For fft, the source node arrival rate is very even across all nodes; we do not see any particular node suffering from low arrival rate. On the other hand, blackscholes exhibits low arrival rate for some nodes due to small variations in packet destination. Packets are destined to only a few nodes, creating congestion around these nodes, which causes the Runahead network to drop more packets. However, the difference in arrival rates is modest, leading us to believe that the unfairness of our arbitration scheme does not have a negative impact on application performance.

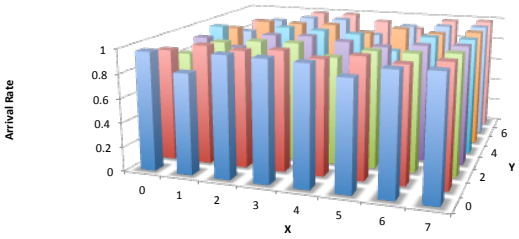
4.4 Power and Area

In this section, we evaluate power and area consumption of the Runahead network. We first evaluate the 8×8 network power and area usage using DSENT. Next, we evaluate the power and area of a single router using RTL modeling.

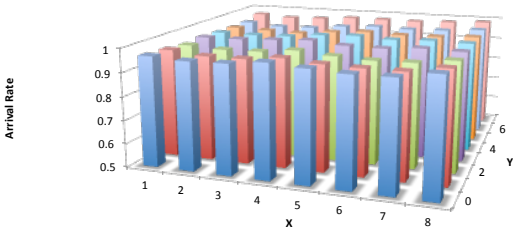
Critical Path: DSENT reports a minimum clock period for the Runahead network of 481.073 *ps* when optimized for a frequency of 2GHz. As a result, we are confident that a Runahead network router can be traversed in a single cycle in a 2GHz system. The critical path reported by Synopsys in 65nm for a single regular router is 1.5 *ns*. However, the RTL results show that the critical path of the Runahead router does not increase compared to the regular lossless router. This ensures that the combined design can still operate in the original clock speed. To ensure that the Runahead router can forward packets in one cycle, we measure the critical path of the Runahead router alone. The Synopsys design compiler reports a critical path of 800 *ps* for the Runahead network logic alone, after timing optimizations. The Runahead network can operate almost twice as fast as the regular router. As a result, we are confident that the Runahead network can function correctly in a single router cycle.

Application	Packet Arrival Rate	Hop Count
barnes	96.29%	3.44905
blackscholes	95.63%	3.61983
bodytrack	95.78%	3.41479
cholesky	98.43%	3.77890
facesim	97.66%	4.10944
fft	97.12%	4.43595
fluidanimate	96.94%	3.57521
lu_cb	97.93%	3.98956
lu_ncb	98.17%	3.56591
radiosity	97.54%	3.52332
radix	97.33%	4.41145
raytrace	96.51%	3.48362
swaptions	98.69%	3.51006
water_nsquared	97.18%	3.45227
Average	97.23%	3.7371

Table 4: Packet arrival rate and hop count



(a) blackscholes

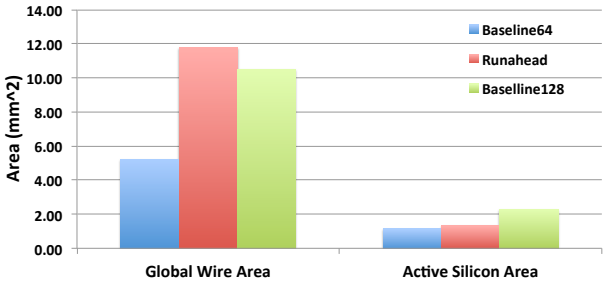


(b) fft

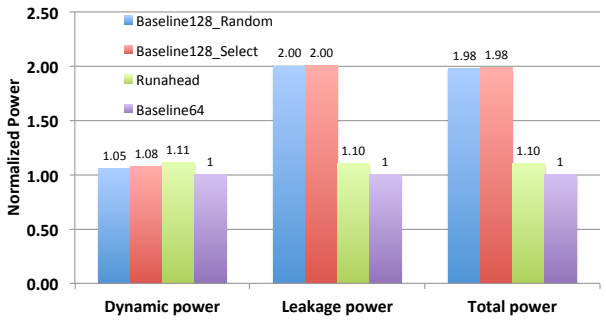
Figure 9: Source node arrival rate

Area: The Runahead network incurs 16% active silicon area overhead compared to the 64-bit baseline NoC (i.e., the Runahead network as an accelerator) and achieves 1.73× area savings compared to the 128-bit baseline NoC (i.e., the Runahead network as a power-saver), as shown in Figure 10a. However, the Runahead NoC has the highest usage of global wire area. This is due to the wider channels used to carry additional metadata, as discussed in Section 3.

Table 6 shows area consumption of the Runahead



(a) Area comparison



(b) Power comparison

Figure 10: Power and area comparison

Configuration	Area(um2)	Difference in Area
Baseline64	218777.398	1
Baseline128	437554.796	2
Runahead	225427.679	1.030

Table 5: RTL router area comparison

network alone as well as total area consumption when added on top of the regular lossless 64-bit baseline. The Runahead subnetwork only accounts for 13.67% of the active silicon area of the total NoC. The Runahead network uses more than half of the global wire area because it has wider physical channels as discussed previously.

The details of the RTL area comparison for a single router can be found in Table 5, for the Baseline64, Baseline128 and Runahead configurations (Section 3). Links are not included in this evaluation. As an accelerator, there is only a 3.2% increase in area usage. This slight increase consists of the additional multiplexers and registers in the Runahead router. As a power-saver, Runahead decreases area usage by almost 1.94×. This is due to the fact that Runahead effectively replaces network resources in the 128-bit baseline NoC with our lightweight design.

Power: Figure 10b shows dynamic and leakage power normalized to Baseline64, obtained using DSENT. As shown, Runahead significantly reduces leakage power compared to the baseline 128-bit NoCs (i.e., the Runahead network as a power-saver) and incurs only a 10% overhead in leakage power compared to the baseline 64-bit NoC (i.e., the Runahead network as an accelerator). When measuring dynamic power, we use the average injection rate obtained in our Syn-

Configuration	Power(mW)	Difference in Power
Baseline64	50.5851	1
Baseline128	101.1702	2
Runahead	52.3573	1.035

Table 7: RTL router power comparison

Full simulations across all benchmarks. In general, the Runahead NoC consumes more dynamic power than the other network setups due to packet duplication in the Runahead network. Unlike in a multi-NoC design (such as the Baseline128 configurations) where the injection rates of each subnetwork is lower, Runahead has an injection rate that is no less than that of Baseline64, incurring additional switching in the Runahead network to carry duplicate latency-sensitive packets. However, in our measurements, dynamic power accounts for only a small portion of total power, ranging from 1.1% for the 128-bit baselines to 2.09% for Baseline64 and Runahead. Leakage power tends to dominate total NoC power consumption. Overall, the Runahead NoC reduces total power usage by 1.81× when used as a power-saver and incurs only a 10% total power overhead when used as an accelerator.

As shown in Table 6, the Runahead subnetwork only accounts for 9.13% of total power usage in the combined network (i.e., Runahead subnetwork on top of the 64-bit baseline). As discussed, much of this is attributed to leakage power due to the wider channels and larger multiplexers to accommodate additional metadata in the Runahead network. Fortunately, the Runahead network generally accelerates application performance, which naturally leads to additional overall energy savings due to shorter runtimes.

RTL results for the individual routers in TSMC 65nm are listed in Table 7. Compared to the baseline 64-bit lossless router, the Runahead router only uses 3.5% additional power as an accelerator. However, as a power-saver, power consumption is decreased by 1.93× compared to the routers of the 128-bit baseline networks. This is expected as the Runahead network effectively replaces the routers in the multi-NoC baselines with our proposed lightweight router microarchitecture.

5. RELATED WORK

In this section, we explore related work in multi-NoCs, bufferless NoCs, low-latency designs and critical word optimizations.

Multi-NoC Designs. Employing multiple NoCs can improve performance while simultaneously improving area and power efficiency [4]. Flit-reservation flow control [39] uses a separate network for reservation messages; these messages reserve buffers and channels for the exact time a data packet will use them. Doing so speeds up message handling, improving performance. Deja Vu switching [1] proposes a two-network design: one network for control and coherence packets and one for data packets. Both of their NoC planes use conventional NoC routers with VCs. They achieve power savings by slowing down the data plane. Flores et al. [19]

	Total Usage	Runahead Usage	Percentage
Dynamic Power	0.0237 W	0.00398 W	16.81%
Leakage Power	1.11 W	0.0995 W	8.97%
Total Power	1.13 W	0.103 W	9.13%
Global Wire area	11.8 mm^2	6.55 mm^2	55.56%
Active Silicon Area	1.34 mm^2	0.183 mm^2	13.67%

Table 6: Power composition of Runahead on top of 64-bit baseline

propose two networks for critical and non-critical traffic. They use heterogeneous networks composed of low-latency wires for critical messages and low-energy wires for non-critical ones. Mishra et al. [36] propose a heterogeneous multi-NoC system where one network has low latency routers, and the other has high bandwidth channels. Multiple networks also provide opportunities for traffic partitioning [44] and load balancing [4]. Catnap [16] is an energy-proportional multi-NoC design; rather than separating the types of traffic sent to each NoC, networks are turned on and off to respond to changes in network load. Enright Jerger et al. [17] propose a hybrid NoC design where a separate NoC exists on a silicon interposer. In their design, the interposer NoC carries memory traffic while the NoC on the chip carries the rest of the traffic.

Prioritization Schemes in NoCs. Traditional NoCs employ simple arbitration strategies like round-robin or age-based arbitration for packets. Bolotin et al. [9] propose prioritizing control packets over data packets in the NoC. They see substantial performance improvement when small control packets are prioritized over data packets. Globally Synchronized Frames (GSF) [32] is proposed as a local arbitration, QoS-oriented prioritization scheme. GSF provides prioritization mechanisms within the network to ensure each application receives equal amount of network resources. Application-Aware Prioritization Mechanism (STC) [14] is proposed as a prioritization scheme to accelerate network-sensitive applications. STC ranks applications at regular intervals based on their network intensity. Aergia [15] uses the notion of slack to prioritize packets. Aergia may increase network throughput if network is congested. However, from our evaluations, we see little performance impact because of the absence of contention in our simulations. Prioritization schemes can best show their full potential when the network carries heavy traffic. On the other hand, the Runahead network performs well with a lack of contention.

Bufferless NoC Designs. Bufferless networks have received significant research attention [18, 23, 35, 37]. In BLESS [37] and CHIPPER [18], packets are deflected until they reach their destination. In SCARAB [23] and Millberg et al. [35], packets are dropped upon contention and a retransmission message is issued to the source. Our Runahead network does not react to dropped packets and does not deflect packets in the face of contention. This keeps the design of the Runahead network routers simple.

Low-Latency NoC Designs. The goal of our network design is to accelerate packet transmission. Similarly, there has been significant research on low-latency NoCs to improve performance. Express virtual channels [31] reduce latency by allowing packets to bypass intermediate routers. A non-speculative single-cycle router pipeline improves performance by allocating the switch in advance of the message arrival [30]. A low-cost router design [29] reduces latency using a simple ring-stop inspired router architecture for fast traversal of packets traveling in one direction; packets changing direction pay additional latency when they are buffered. Lookahead routing [20] is another common technique to reduce the number of pipeline stages in the router. Route predictions can also speed up the network [24,34]. Often these low-latency designs increase complexity, energy and area in order to achieve better performance. SMART [13] is proposed to reduce overall communication latency by allowing packets to travel multiple hops in a single cycle. They observed that the wire delay is much shorter then a typical router cycle. The links in SMART require specialized repeaters to enable multi-hop traversal. Our simple Runahead network achieves performance improvements with minimal power and area overhead.

Critical Word Optimizations. Delivering the critical word as soon as possible can improve application performance. Separating critical data in either main memory [12] or the caches [27] can efficiently deliver critical data faster. NoCNoC [40] proposes a two network design that separates critical words from non-critical ones in a cache line. It saves power by DVFS for the non-critical network.

6. CONCLUSION

In this paper, we propose the *Runahead NoC*, which can serve as either a *power-saver* for more efficient use of network resources, or as an *accelerator* that provides lightweight, low latency communication on top of a conventional NoC. The Runahead NoC is designed to provide single-cycle hops across the network. To accomplish this, the network is lossy in nature, dropping packets when contention occurs. We present the design of the Runahead NoC router architecture that combines route computation and port arbitration with link traversal. From experiments with SynFull workloads, we find that the Runahead network can maintain over 97% packet arrival rate on average. As an accelerator, the Runahead network reduces average runtime and packet latency by 1.08× and 1.66× with only 10% overhead. As a power-saver, Runahead achieves 1.73× and 1.81× savings in active area and power respectively.

Acknowledgements

The authors thank the anonymous reviewers for their insightful feedback. This work is supported by a Queen Elizabeth II Scholarship in Science and Technology, the Natural Sciences and Engineering Research Council of Canada, the Canadian Foundation for Innovation, the Ministry of Research and Innovation Early Researcher Award and the University of Toronto.

7. REFERENCES

[1] A. Abousamra *et al.*, "Deja vu switching for multiplane NoCs," in *NOCS*, May 2012, pp. 11–18.

[2] P. Avasare *et al.*, "Centralized end-to-end flow control in a best-effort network-on-chip," in *Proceedings of the 5th ACM International Conference on Embedded Software*, ser. EMSOFT '05, 2005, pp. 17–20.

[3] M. Badr and N. Enright Jerger, "SynFull: Synthetic traffic models capturing cache coherent behaviour," in *ISCA*, 2014.

[4] J. Balfour and W. J. Dally, "Design tradeoffs for tiled cmp on-chip networks," in *ICS*, 2006.

[5] N. Barrow-Williams *et al.*, "A communication characterisation of splash-2 and PARSEC," in *IEEE International Symposium on Workload Characterization*, Oct 2009, pp. 86–97.

[6] D. U. Becker, "Efficient microarchitecture for network-on-chip routers," Ph.D. dissertation, Stanford University, 2012.

[7] C. Bienia *et al.*, "The PARSEC benchmark suite: Characterization and architectural implications," in *PACT*, 2008.

[8] N. Binkert *et al.*, "The gem5 simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.

[9] E. Bolotin *et al.*, "The power of priority: NoC based distributed cache coherency," in *NOCS*, May 2007, pp. 117–126.

[10] S. Y. Borkar, "Future of interconnect fabric: a contrarian view," in *Proc. Int. Workshop on System Level Interconnect Prediction*, 2010.

[11] S. Borkar, "Thousand core chips: A technology perspective," in *DAC*, 2007.

[12] N. Chatterjee *et al.*, "Leveraging heterogeneity in dram main memories to accelerate critical word access," in *MICRO*, Dec 2012, pp. 13–24.

[13] C.-H. O. Chen *et al.*, "Smart: a single-cycle reconfigurable noc for soc applications," in *DATE*, 2013.

[14] R. Das *et al.*, "Application-aware prioritization mechanisms for on-chip networks," in *MICRO*, 2009.

[15] ——, "Aergia: exploting packet latency slack in on-chip networks," in *ISCA*, 2010.

[16] ——, "Catnap: Energy proportional multiple network-on-chip," in *ISCA*, 2013.

[17] N. Enright Jerger *et al.*, "NoC architectures for silicon interposer systems," in *MICRO*, 2014.

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

[19] A. Flores *et al.*, "Heterogeneous interconnects for energy-efficient message management in cmps," *Computers, IEEE Transactions on*, vol. 59, no. 1, pp. 16–28, Jan 2010.

[20] M. Galles, "Spider: A high-speed network interconnect," *Micro, IEEE*, vol. 17, no. 1, pp. 34–39, 1997.

[21] C. Gómez *et al.*, "An efficient switching technique for nocs with reduced buffer requirements," in *ICPADS*, 2008.

[22] P. Gratz and S. W. Keckler, "Realistic workload characterization and analysis for networks-on-chip design," in *The 4th Workshop on Chip Multiprocessor Memory Systems and Interconnects (CMP-MSI)*, 2010, pp. 1–10.

[23] M. Hayenga *et al.*, "SCARAB: A single cycle adaptive routing and bufferless network," in *MICRO*, 2009.

[24] Y. He *et al.*, "Predict-more router: A low latency NoC router with more route predictions," in *Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2013 IEEE 27th International*, May 2013, pp. 842–850.

[25] R. Hesse *et al.*, "Fine-grained bandwidth adaptivity in networks-on-chip using bidirectional channels," in *NOCS*, 2012.

[26] Y. Hoskote *et al.*, "A 5-GHz mesh interconnect for a Teraflops processor," *Micro, IEEE*, 2007.

[27] C.-C. Huang and V. Nagarajan, "Increasing cache capacity via critical-words-only cache," in *ICCD*, 2014.

[28] N. Jiang *et al.*, "A detailed and flexible cycle-accurate network-on-chip simulator," in *ISPASS*, 2013.

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

[30] A. Kumar *et al.*, "A 4.6tbits/s 3.6GHz single-cycle NoC router with a novel switch allocator in 65nm CMOS," in *ICCD*, 2007.

[31] ——, "Express virtual channels: Towards the ideal interconnection fabric," in *ISCA*, 2007.

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

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

[34] H. Matsutani *et al.*, "Prediction router: Yet another low latency on-chip router architecture," in *HPCA*, 2009.

[35] M. Millberg *et al.*, "Guaranteed bandwidth using looped containers in temporally disjoint networks within the nostrum network on chip," in *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, vol. 2, Feb 2004, pp. 890–895 Vol.2.

[36] A. K. Mishra *et al.*, "A heterogeneous multiple network-on-chip design: an application-aware approach," in *DAC*, 2013.

[37] T. Moscibroda and O. Mutlu, "A case for bufferless routing in on-chip networks," in *ISCA*, 2009.

[38] O. Mutlu *et al.*, "Runahead execution: an alternative to very large instruction windows for out-of-order processors," in *HPCA*, 2003.

[39] L.-S. Peh and W. Dally, "Flit-reservation flow control," in *HPCA*, 2000.

[40] J. San Miguel and N. Enright Jerger, "Data criticality in network-on-chip design," in *NOCS*, 2015.

[41] C. Sun *et al.*, "DSENT - a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling," in *NOCS*, 2012.

[42] M. Taylor *et al.*, "The Raw microprocessor: a computational fabric for software circuits and general-purpose programs," *Micro, IEEE*, vol. 22, no. 2, pp. 25–35, Mar 2002.

[43] J. van den Brand *et al.*, "Congestion-controlled best-effort communication for networks-on-chip," in *Design, Automation Test in Europe Conference Exhibition, 2007. DATE '07*, April 2007, pp. 1–6.

[44] S. Volos *et al.*, "CCNoC: Specializing on-chip interconnects for energy efficiency in cache-coherent servers," in *NOCS*, May 2012, pp. 67–74.

[45] D. Wentzlaff *et al.*, "On-chip interconnection architecture of the tile processor," *IEEE Micro*, 2007.

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