Muffin: Minimally-Buffered Zero-Delay Power-Gating Technique in On-Chip Routers

Hossein Farrokhbakht  
Department of ECE  
University of Toronto  
Toronto, CA  
h.farrokhbakht@mail.utoronto.ca

Hadi Mardani Kamali  
Department of ECE  
George Mason University  
Fairfax, USA  
hmardani@gmu.edu

Natalie Enright Jerger  
Department of ECE  
University of Toronto  
Toronto, CA  
enright@ece.utoronto.ca

Abstract—Although conventional Network-on-Chip (NoC) designs provide high bandwidth, many modern applications for many-core architectures have significant periods of low NoC utilization. Highly provisioned NoCs provide the required performance during periods of high activity; yet, large NoC designs come with high power costs. Furthermore, as technology shrinks, the contribution of static power increases. Hence, numerous NoC power-gating techniques have been proposed to alleviate the growing contribution of static power. However, the efficiency of power-gating techniques decreases due to sporadic packet arrivals across a range of injection rates. In this paper, we propose Muffin, which improves static power consumption by an average of 95.4%, while improving the average packet latency by 73.7%.

Index Terms—Power-gating, Network-on-Chip, Bypass, router micro-architecture

I. INTRODUCTION

Networks-on-Chip (NoCs) handle high volume communication and provide a high-performance, cost-effective, and scalable interconnection network for many-core architectures. However, decreasing technology size leads to concerns regarding NoC power consumption. In fact, NoCs not only contribute a high fraction of a chip’s total power consumption [1], their contributions to static power are getting worse as technology sizes shrink. A recent power breakdown demonstrates that static power represents 74% of total NoC power at 22nm [2] and static power dissipation grew by a factor of 3.3× (from 11.2% at to 33.6%) as technology size decreased from 65nm to 32nm [3].

Large buffers are a major contributor to NoC power. Routers are provisioned with larger buffers to meet worst-case throughput requirements. These buffers enable the NoC to handle peak loads close to the saturation point with no performance degradation. However, network utilization is significantly lower than the saturation point in many real applications. For example, SPLASH-2 benchmarks [4] have an average router utilization of less than 20% [5]. This low network utilization results in large aggregate idle time in routers [6]–[10]. Low utilization has spawned a wealth of research in lightweight, low-cost, and bufferless router micro-architectures. These works dramatically reduce the amount of buffering [11], [12]; however, energy benefits of these approaches are minimal and buffered designs have superior latency and bandwidth [13]. The low efficiency of these low-cost router architectures coupled with the large aggregate idle time, motivates researchers to focus on designing efficient power-budgeting techniques to alleviate cost, especially static power dissipation, of router micro-architecture. However, power-budgeting techniques must not sacrifice the main advantages of NoCs: performance and scalability.

The two main categories of power-budgeting solutions are: (1) policy-based, including techniques such as Dynamic Voltage and Frequency Scaling (DVFS) to save dynamic power and power-gating to alleviate static power and (2) Structural which involve modifications to the NoC topology and architecture. Unlike the former techniques that are dynamic and traffic dependent, structural techniques are inherent to the design and always present [14]. We focus on power-gating techniques. Power-gating promises to significantly mitigate static power dissipation in circuit-level designs [15], but faces challenges when applied to on-chip routers. Powering on a power-gated router imposes wake-up latency. A typical wake-up requires ~8 cycles at 2 GHz [16]. Considering typical NoC per-hop delays of 2-4 cycles, this 8-cycle wake-up per router can add significant delay to total packet latency. At low NoC utilization, there is high probability that a packet will encounter multiple power-gated routers which further increases packet delay. Moreover, the power overhead of the wake-up process is not negligible. As a result, the idle period must be long enough to save more power than is spent on wake up. Despite low NoC utilization, idle periods are typically short; as a result, conventional power-gating techniques which power gate the router immediately after becoming idle, suffer significantly from these wake-up overheads.

An efficient power-gating technique must:

1) Decrease the delay of encountering power-gated routers;  
2) Compensate for the power overhead of wake-up.

Most prior art focuses on addressing the wake-up delay. However, investigations into typical traffic flow demonstrate that most real applications have aperiodic traffic that significantly decreases the probability of finding sufficiently long idle periods, making power-gating techniques less efficient [17]. To mitigate the challenges associated with wake-up overhead, we focus on the route packets will take through each router. A high proportion of flits travel straight through each router. As a result, the resource allocation for other types of flits, i.e., turn, inject, and eject, is relatively low. This observation inspires us to design a power-gating scheme, in which small shared buffers provide a simple bypassing mechanism for each type of flit, i.e., inject, eject, straight, and turn. All routers are initially power gated; power-gated routers can handle all types of flits without need for powering on. An upper bound threshold on shared buffer utilization determines when to power on each router in response to increasing network utilization. Similarly, a lower bound utilization threshold determines when to power gate each router. These extra buffers increase the
number of power-gated routers and the power-gated periods in the NoC significantly. Compared to a conventional power-gating technique, the proposed technique saves 95.4% power consumption on average and reduces latency overhead by 73.7% for SPLASH-2 benchmarks [4].

II. BACKGROUND AND RELATED WORK

In power-gating techniques, the unused (idle) components are gated (disconnected) from the power supply. They are reconnected when the power-gated component must be re-activated. Prior work on power-gating in NoCs can be classified into two sets: Coarse-grained and Fine-grained. The granularity refers to the size of targeted components that will be power gated. In coarse-grained, a large component such as whole router will be connected/disconnected to/from power supplies. For both sets, applying power-gating requires handshaking between a power-gated component, such as the router, and its neighbors, to control and verify the correctness of control flow. As shown in Fig. 1, a router controller is equipped with two new signals, a power-gate (PG) signal that notifies neighbors about the status of the power-gated router, and a wake-up (WU) signal which allows the neighbors to wake up power-gated routers.

Most power-gating techniques are coarse grained, in which the power switch is located between the router and its power/ground I/O cells to power on/off the whole router [3], [5], [18]–[21]. Catnap [22] switches off the power for a set of routers to scale up and down the available bandwidth. The power-gating status of cores can be used to make decisions about power gating the connected routers [3]. This approach uses a connection-aware method to mitigate the delay imposed by detouring. However, detouring limits the scalability of the technique. NoRD uses a bypass mechanism to avoid powering on routers [19]; however NoRD must detour packets which adds latency and hurts performance. TooT [20] uses bypassing for straight packets; however routers must be powered on when turn or inject packets arrive which incurs wake-up delay. Many of these coarse-grained scheme focus on reducing wake-up delay through (1) run-ahead early wake-up notification [16], [18], (2) traffic-driven, application-specific mapping and routing techniques [5], (3) decreasing the number of wake-up processes by avoiding powering on routers for specific types of flits [20], or (4) detouring flits towards pivot routers which results in increasing the average number of power-gated routers during execution time [21]. Major differences between these techniques has been demonstrated in Table I.

In fine-grained power-gating techniques [23], small-scale components, such as buffers or virtual channels (VCs), are targeted for power gating. Multiple router components can be targeted for power gating. Hence, the wake-up process requires different considerations, resulting in more complicated schemes compared to coarse-grained techniques [24]–[26].

We propose Muffin, a coarse-grained power gating techniques. In contrast with the aforementioned work, Muffin virtually eliminates the delay overhead imposed by the wake-up process. This is achieved through a comprehensive bypass mechanism that can efficiently handle all types of flits, i.e., inject, eject, straight, and turn rather than only handling one or two types of flits. Furthermore, Muffin works well with adaptive routing unlike most other techniques; only TooT [20] and NoRD [19] support adaptive routing. Muffin exhibits good scalability by avoiding detouring and not requiring any additional inter-router wiring. In the next section, we explain our technique in more detail.

III. PROPOSED TECHNIQUE: MUFFIN

There are two major shortcomings in prior power-gating techniques: (1) power overhead of wake-up due to aperiodicity of traffic flow and (2) The delay overhead of power-gating due to detouring or wake-up delay. Muffin adds only 5 flit-size buffers and a lightweight controller. These changes enable Muffin to efficiently handle a high proportion of flits, regardless of type, i.e., inject, eject, straight, and even turn, without powering on the router. Since a high proportion of flits can traverse power-gated routers, Muffin significantly decreases the number of wake-ups which reduces the delay incurred by blocking flits requesting power-gated routers while extending the length of the power-gated periods for each router. Muffin achieves performance benefits; packets using the extra buffers in Muffin, effectively bypass the pipeline stages of the power-gated router to achieve lower latency.

A. Enabling Bypass for All Flit Types

To avoid providing on the router for all flit types, Muffin requires 5 flit-sized buffers, a small amount of logic and a lightweight power-gating controller in each router. Muffin controller handles buffer management using Free to Forward signals, i.e., $FF_{dir}$ where dir is one of the cardinal directions: N, S, E or W. Fig. 2 depicts the Muffin router microarchitecture. The basic router functionality is highlighted in gray with dotted lines and remains unchanged.

When a flit arrives and the router is powered on (i.e., $en_t = 1$), the flit enters the input buffers of the basic router through the upstream multiplexers, i.e., $MUX_{UP_{dir}}$. When the router is in a power-gated mode (i.e., $en_t = 0$), the incoming flits from the upstream router and the network interface (NI) are buffered in $ByP_{dir}$, and interject buffer, respectively. Thereafter, the Muffin controller calculates the control signals for all multiplexers to bypass flits. To do that, Muffin controller determines the flit type, i.e., inject, eject, straight, and turn, using the destination information which is stored in the flit header. The logic for this is very similar to the routing logic; due to its simplicity, it can be completed in a single cycle to enable the proper handling of flits when the router is power-gated.

1) Straight: Four out of five flit-size buffers are located at each of the four outgoing ports of each side, called $ByP_{dir}$. These buffers bypass straight flits. For example, the flits from upstream link of the E port, $UF_E$, are directly connected to $ByP_E$. The Muffin controller sets the selectors of $MUX_{dir}$ to bypass this flit to downstream link of port W. Similarly, the remaining three ports handle straight flits using these buffers without powering on the router. Note that one of the inputs of the downstream multiplexers ($MUX_{dir}$) comes from the basic router when the router is powered on. When the router is powered on, Muffin controller selects the output of basic router.
2) **Eject**: These four flit-size buffers also handle eject flits. Suppose an incoming flit must be ejected to the NI. In this case, the flit will be buffered in corresponding ByP buffer and forwarded to the NI. As shown in Fig. 2, we add a 5-to-1 multiplexer ($MUX_{N1}$) to provide this capability. For periods when the router is powered on, the 5th input of MUX$_{N1}$ connects to the NI port of the basic router.

3) **Turn**: Turn packets require at least two clock cycles to traverse the proposed router. To handle turn flits, we add another flit-size buffer, called *interject*. All flits arriving from upstream ports are buffered in bypass buffers. After buffering the incoming flit, the Muffin controller determines the outgoing port for this flit. If it is a turn flit, the controller sends this flit to the interject buffer to turn the flit towards its destination. Accordingly, the select of MUX$_{inter}$ determines which bypass buffer should be stored in the interject buffer. After buffering the flit into the interject buffer, the Muffin controller allocates the link and releases the buffer. Since the downstream router must be checked for turn flits before link allocation by the controller, it is not possible to handle turn packets without having the interject buffer.

4) **Inject**: The interject buffer also handles injecting flits without needing to power on the router. The inject input port is one of the inputs of MUX$_{inter}$. So, inject flits will be buffered in interject when there is no turn flit in it. The Muffin controller sets the select of corresponding MUX$_{dir}$ to inject this flit.

### B. Priority Arbitration for Different Flit Types

Since we use shared buffers and additional multiplexers in Muffin, its controller must determine the priority of different types of flits. The first arbitration is for the downstream multiplexers (MUX$_{dir}$). Each MUX has three inputs, the ByP buffer that contains a straight flit, the Interject buffer, and the basic router when it is powered on. Since the third one is only used when the router is powered on, Muffin arbitration happens between the first two inputs. The controller grants higher priority to the ByP buffer. The interject buffer holding either a inject or turn flit must wait until there is no straight flit in ByP buffer. The second arbitration is for the multiplexer of interject buffer (MUX$_{inter}$). Here, flits stored in ByP buffers (turn flits), have lower priority than an inject flit when they arrive simultaneously. However, if the interject buffer currently holds a turn flit, the inject flit must wait until the turn flit finishes. If multiple ByP flits request the interject buffer simultaneously, the Muffin controller always grants the priority to ByP$_N$, ByP$_S$, ByP$_E$, and ByP$_W$, respectively. Similarly, for the arbitration of the multiplexer of $N_{out}$, the controller ranks the ByP$_N$ buffers in the aforementioned order. This statically defined order allows us to make the Muffin controller simple and lightweight. In Sec. III-D, we discuss our congestion and starvation avoidance mechanism. This rank-based arbitration in Muffin is easily implementable using cascading multiplexers.

#### C. No Restrictions on Routing Algorithm

Muffin supports both deterministic and adaptive routing algorithms. For XY dimension-order routing (DOR), we only need to support two turns: $W$ to $N/S$ or $E$ to $N/S$. Hence, we can remove ByP$_N$ and ByP$_S$ wires from MUX$_{inter}$, and make this multiplexer smaller. To support all possible turns, we connect $It_{by}$ to all downstream multiplexers.

Muffin does not impose any restriction on the capability of the router for handling different scenarios while the router is power-gated compared to the baseline router. Fig. 3 reflects possible scenarios of traffic in a Muffin router in a $3 \times 3$ mesh network. We assume that all routers are power-gated in these examples to show the efficiency of Muffin in handling different scenarios. With no contention (Fig. 3(a)), flits travel to their destinations with ideal latency. The ideal latency is only one cycle for inject, eject, and straight flits, and two cycles for turn flits. In this example, the flit takes only 6 cycles. However, for an optimized baseline router with two pipeline stages and no power-gating, it takes 10 cycles. This gives Muffin a performance advantage. Straight flits from different dimensions (Fig. 3(b)) do not affect each other; similar to the first scenario, the latency is ideal. When a turn flit and straight flit must traverse the same downstream link (Fig. 3(c)), contention occurs; straight flits are prioritized over turn flits in downstream multiplexers. For two turn flits with the same downstream link (Fig. 3(d)), prioritization is based on the following order: $N$, $S$, $E$, and $W$. The winning turn flit accesses the interject buffer, while the other turn flit must wait in its ByP buffer. The contention in Fig. 3(c, d) also occurs in the baseline router; the difference is how the contention is resolved. Since we use a shared buffer for all turns in a router (interject buffer), there is a specific type of contention that occurs only in Muffin (Fig. 3(e)). In this scenario, there are two turn flits from different directions towards different directions. However,

---

**TABLE I: Comparison of Different Power-Gating Schemes.**

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Routing</th>
<th>Extra Wiring</th>
<th>Extra Buffers</th>
<th>Threshold-based</th>
<th>Changing NI</th>
<th>Handling BET*</th>
<th>Early Wake-Up</th>
<th>Flit-Type-based</th>
<th>Detouring</th>
</tr>
</thead>
<tbody>
<tr>
<td>PowerPunch [18]</td>
<td>XY</td>
<td>784 bits for $8 \times 8$</td>
<td>none</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>none</td>
<td>✓</td>
</tr>
<tr>
<td>NoRD [19]</td>
<td>XY + Adaptive</td>
<td>none</td>
<td>1 multi-flit</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TooT [20]</td>
<td>XY + Adaptive</td>
<td>none</td>
<td>4 one-flit</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SMART [5]</td>
<td>XYX Deterministic</td>
<td>none</td>
<td>none</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SPONGE [21]</td>
<td>XY</td>
<td>32 bits for $8 \times 8$</td>
<td>4 one-flit</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Muffin</td>
<td>XY + Adaptive</td>
<td>none</td>
<td>5 one-flit</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

*BET (Break-Even Time): Minimum number of consecutive cycles in power-gated mode to compensate the wake-up energy overhead.*

![Muffin Router Micro-Architecture](image_url)

---

Fig. 2: Muffin Router Micro-Architecture.
the prioritization for interject buffer resolves this contention with a throughput degradation of 1/2 compared to baseline router. However, as Muffin operates with higher throughput in other cases, this throughput degradation is compensated for completely.

D. Discussion

1) Congestion Avoidance: In power-gated mode, Muffin can handle a high proportion of packets. However, increasing the injection rate increases the probability of congestion in the network when the routers are power-gated. A metric is required to estimate congestion. For powering on routers, we define a time-based metric and a threshold for the added buffers (interject and ByP buffers). The time-based metric determines the number of cycles that a flit is waiting in one of these buffers. The threshold is the maximum feasible value for this metric to keep the router in power-gated mode. If the number of cycles that a flit is stored in one of these buffers exceeds this threshold, it means that the waiting flit could not access the resource after a long period. This triggers the WU signal by the controller to change the state of this router to powered on. Implementing this threshold only requires a few counters with negligible overhead. In Sec. IV, we use a threshold of 8 cycles.

For low injection rates, we need another metric and threshold to determine when router utilization is low while powered-on. At this threshold, it is more efficient to change its state to powered on. Implementing this threshold only requires a few counters with negligible overhead. In Sec. IV, we use a threshold of 8 cycles.

2) Starvation Avoidance: To minimize the area overhead, we use a statically defined order for the arbitration of the multiplexer of $N_{out}$ and $MUX_{inter}$. Despite this static order, there is no starvation. If a low-priority flit in the arbitration order is being starved by higher priority flits, the power-on threshold will be triggered to wake-up the router and allow all flits to make forward progress. To avoid potential unfairness, the hard-coded arbitration order can be different at each router. For example, one router with priority order: $N$, $S$, $E$, and $W$, and another router with priority order: $E$, $W$, $S$, and $N$.

3) Protocol-level Deadlock Avoidance: Coherence protocols have several message classes. To avoid resource dependences between messages of different classes, which can lead to deadlock, NoCs separate message classes into different virtual channels. Muffin uses single flit buffers at each port; this buffer can mix traffic from different message classes. However, protocol-level deadlock is avoided because congestion due to stalled flits will trigger the power-on threshold in the router which will allow packets to flow into the VCs of their respective message classes, preventing an actual protocol-level deadlock from forming. Flits in the bypass buffers are transferred to the buffers of the current basic routers after powering on. As it can be seen in Fig. 4, two MUXes ($MUX_{P_{out}}$) are added in front of input ports of basic router (inputs coming from the upstream routers). The Muffin controller selects the $ByP$ buffers to transfer into the basic router when powered on. The flit stored in interject buffer will be transferred to the next router. Since upstream links are directly connected to bypass flits, the Muffin controller sends the flit stored in interject buffer to the next router. Then, in the next router, this flit will be stored in these bypass buffers (if the router is power-gated) or in buffers of basic router (if the router is powered-on). Note that if the state of the next router is similar to the current one, i.e., it is in the wake-up process, the controller of the current router waits until the next router is powered-on, then releases the flit of interject router. After releasing the flit of this one-flit buffer, the current router will be powered on.

IV. EXPERIMENTAL RESULTS

We implement Muffin in Booksim [28] and run traces gathered from SPLASH-2 benchmarks [4]. We generate traces using gem5 [29] in syscall emulation mode. The traces consist of traffic over 10 million cycles after thread creation and initialization. We also use Bernoulli-based uniform synthetic workload and a Markov-based injection process [28] to evaluate the efficiency of Muffin. The simulator is warmed-up for 30,000 cycles, and then network statistics are collected for one million cycles. Table II lists all configuration parameters for our evaluation. Although the fundamental insights of Muffin can be applied to a wide range of topologies that have turning and straight flits, we focus our evaluation on a mesh as it is the most commonly used NoC topology [30], [31] and makes for a straightforward comparison with prior power-gating techniques [5], [18]–[21]. We use DSENT [2] to model static and dynamic power for a 45nm process. The area overhead of Muffin has been evaluated using Synopsis™ Design Compiler 45nm Open Cell Library. The extra hardware for the bypass mechanism added in Muffin only incurs a 7.2% area overhead per router compared to conventional power-gating. Experimental results are compared to the state-of-the-art power-gating techniques:
1) **No-PG**: Baseline with no power-gating technique.
2) **Conv-PG**: Simple power-gating technique (immediate power-gating after idle detection).
3) **PG-ConvOpt**: Optimized power-gating endowed with a prediction and early wake-up mechanism [16].
4) **PowerPunch, TooT, SMART, and SPONGE**: Highly-efficient state-of-the-art power-gating techniques [5], [18], [20], [21].

Fig. 4 shows the power dissipation of all techniques normalized to No-PG. Total power consumption includes dynamic power, static power and the overhead of power gating (wake-ups and power-gating controller overhead) which contributes to the total static power. Since **Muffin** handles all types of flits without powering on the router, it alleviates static power dissipation significantly. SPLASH-2 benchmarks typically have very low traffic volumes; for 7 benchmarks, all routers are power-gated throughout execution using **Muffin**. Two applications, **raytrace** and **water_spatial**, had only 4 and 6 of the 64 routers powered on. Compared to No-PG, **Muffin** reduces router static power consumption by 97.33% on average, while **PG-ConvOpt**, **PowerPunch**, **SMART**, **TooT**, and **SPONGE** save 25.4%, 26.2%, 42%, 73.1%, and 86.27% of static power.

Although **Muffin** significantly reduces static power compared to all enumerated techniques, its main advantage is effectively zeroing out the delay overhead of power-gating. In fact, it even improves the average packet latency compared to an optimized router architecture with pipeline bypassing that has only two stages and no power-gating technique. Fig. 5 shows the average packet latency of different power-gating techniques. Although **SPONGE** improves the packet latency through bypassing compared to No-PG, comparing to 4-stage pipeline router architecture is unfair. Since **Muffin** uses a bypass mechanism, we compare the packet latency to a router with 2-stage bypassing pipeline; even here, **Muffin** improves packet latency by 7.5%. Compared to the 2-stage optimized router, **SPONGE** and **TooT**, which also employ bypassing incur 30.1% and 85.6% delay overhead due to detouring and wake-up delay. **Muffin** forwards **straight, eject**, and **inject** flits in only one cycle, and only **turn** flits take at least two cycles. However, a 2-stage router pipeline needs 2 cycles for all packet types. Since only ~17.5% of flits are **turn**, **Muffin** improves the average packet latency compared to a router with a 2-stage bypassing pipeline.

Fig. 6 shows the Power Delay Product (PDP) for each technique. PDP of PowerPunch and **SMART** is 31.9% and 17.7% higher than that of No-PG. However, for **TooT** and **SPONGE**, PDP improves by 51.5% and 82.8%. **Muffin** improves PDP by 96.7%, outperforming all of these power-gating techniques.

Fig. 7 shows the behavior of **Muffin** across the full range of network loads. The power dissipation of **Muffin** is low at low injection rates. Since **Muffin** handles all flits without powering on the router while the traffic volume is low, its efficiency is high at low injection rates. However, as injection rate increases, routers must be powered on and the power dissipation approaches that of No-PG. Similarly, the average packet latency of **Muffin** is less than all other methods for injection rates less than 0.06. However, similar to power, the delay approaches that of No-PG for higher injection rates. The synthetic traffic used in Fig. 7 is a **Bernoulli**-based uniform workload. We also evaluate a **Markov**-based injection process to mimic bursty traffic; we find that **SPONGE** cannot tolerate bursty traffic since it detours packets to the predetermined column to handle **turn** packets. Bursty traffic does not have any negative effects on the other methods including **Muffin** in terms of latency and power saving.

Another advantage of **Muffin** is its ability to support adaptive routing. To evaluate this, we use Duato’s protocol and maximize...
the number of turns for each packet [32]. TooT also supports adaptive routing but does not efficiently handle turn flits. As the number of turns increase so do its packet latency and static energy consumption. Figs. 8 and 9 show the power consumption and packet latency of Muffin compared to No-PG, and TooT. Since the number of turn flits is maximized, Muffin experiences a 5% delay overhead compared to No-PG due to contention for the interject buffer.

V. CONCLUSION

We propose a novel Minimally-Buffered Router Infrastructure (Muffin) for NoCs, which efficiently handles all types of flits, i.e., inject, eject, and turn, and straight without powering on the router. Muffin requires very low overhead—only five flit-size buffers and a lightweight controller that imposes only 7.2% area overhead compared to a conventional power-gating technique. It improves static power consumption and average packet latency by 95.4% and 73.7%. In addition, by using two simple thresholds that have a lightweight implementation, powering-on and power-gating are controlled to avoid any starvation and under-utilization in the network.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers and members of the NEJ Group for their helpful feedback on this work. This work was supported in part by the Natural Science and Engineering Research Council of Canada, the University of Toronto and the Percy Hart Early Career Professorship.

REFERENCES