FlexiBuffer: Reducing Leakage Power in On-Chip Network Routers

Gwangsun Kim, John Kim
Department of Computer Science
KAIST
Daejeon, Korea
{gkim, jjk12}@cs.kaist.ac.kr

Sungjoo Yoo
Department of Electronic and Electrical Engineering
POSTECH
Pohang, Korea
sungjoo.yoo@postech.ac.kr

ABSTRACT

The increasing number of integrated components on a single chip has increased the importance of on-chip networks. A significant part of on-chip network routers is the buffer, as it occupies a large area and consumes a significant amount of power. In this work, we propose FlexiBuffer, a microarchitecture in which we minimize buffer leakage power by using fine-grained power gating and adjusting the size of the active buffers adaptively. We propose two microarchitecture techniques to support fine-grained power gating – *early* credit in credit-based flow control and new buffer organizations to overcome the limitation of circular buffers. Our results show that, with minimal loss in performance, we can reduce the leakage power of on-chip network router buffers by up to 61% and overall router power consumption by up to 39%.

Categories and Subject Descriptors

C.1.2 [**Computer System Organization**]: Multiprocessors—*Interconnection architectures*

General Terms

Design

Keywords

Leakage power, Buffer Organization, Power gating, On-chip networks, Routers

1. INTRODUCTION

The on-chip network that interconnects the different components together is becoming more critical as on-chip network size increases. One of the key components of an on-chip network router is the buffer because of its large area and high power consumption [8]. Buffers are necessary to provide high performance in an on-chip network; however, we make the observation that, even when the network is saturated, not all of the buffers in the network are fully utilized. As a result, power gating can be leveraged to turn off the unused buffers and minimize its leakage current. By leveraging power gating, we propose to achieve an *energy-proportional* [1] on-chip network.

In this work, we propose a fine-grained power-gating of the on-chip network router's input buffers at the per-entry granularity. However, fine-grained power gating of buffers presents several challenges in the design of an on-chip network, including how to manage the activation and deactivation of buffer entries and how to account for the wakeup latency. We modify conventional credit-based flow control and propose *early* credit such that buffers can be turned on/off with minimal change to the flow control. In addition, we show how conventional circular buffer organization is problematic with fine-grained power gating because of its energy overhead. To overcome this overhead, we propose a modified linked-list based organization and a novel *split queue* buffer management that reuses the physical buffer entries without requiring a complex buffer management scheme.

DAC'11, June 5-10, 2011, San Diego, California, USA

2. BACKGROUND / MOTIVATION

In this section, we provide a background on the role of buffers in on-chip networks and discuss related work on minimizing the negative impact of buffers. We also show how buffers impact performance but are not fully utilized, which motivates the need to leverage fine-grained power gating to turn off unused buffers entries.

2.1 Buffers in On-Chip Networks

Buffers are commonly used in interconnection networks to decouple the allocation of channel resources [4]. It is well known that increasing the size of buffers provides higher performance by increasing network throughput. A latency-throughput plot is shown in Figure 1 as the amount of input buffers is varied for uniform random traffic pattern. We assumed an aggressive single-cycle router with a single-cycle wire delay and simulated a 64-node, 8×8 2D mesh network. Increasing the size of buffers from 1 to 4 entries provides a significant improvement in overall performance as sufficient buffer depth is provided to cover the credit round-trip latency. However, continuing to increase the size of buffers beyond 8-16 entries provides a very small increase in performance; thus, we assume a baseline router with 8 entries per buffer (or virtual channels) in this work.

Buffers in on-chip networks occupy a significant portion of the area and power. To reduce the negative impact of buffers in on-chip networks, *bufferless* on-chip networks have been recently proposed [12, 5, 11]. Since there are no buffers to store packets in the network, bufferless networks handle contentions with either deflection routing [12] or dropping packets and retransmitting them [5]. Although bufferless networks eliminate the need for buffers, they

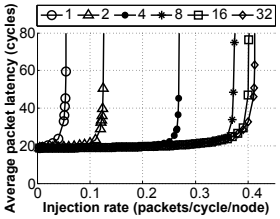
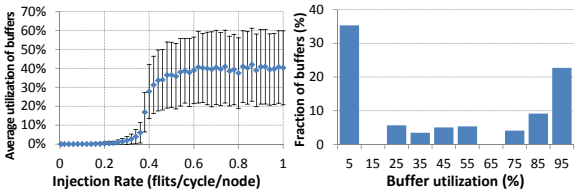


Figure 1: Impact on performance as the number of buffer entries is varied.



(a) (b)

Figure 2: (a) Buffer utilization under uniform random traffic as offered traffic is varied. and (b) buffer utilization histogram in saturation.

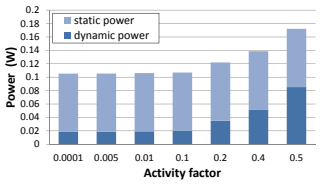


Figure 3: Power consumption of a single router as activity factor is varied.

also introduce additional overhead. For example, they need reorder buffers at the destination or additional buffering at the source to retransmit the packets. Recent study [11] showed that if all these overheads are properly accounted for, the benefits of bufferless can be reduced. To provide the benefits of both bufferless and buffered flow control, adaptive flow control (AFC) [7] has been proposed, which adapts between the two flow controls. This work leverages power gating to disable all buffer entries at an input port when in bufferless mode. Since bufferless mode uses deflection routing, which requires reorder buffers, AFC leverages the existing MSHR buffers. As a result, their flow control is restricted to cache coherence protocols that preallocate entries in the MSHR for all network requests.

2.2 Buffer Utilization

As network load increases, more packets arbitrate for the shared channel resources and packets that lose arbitration are stored in the buffers until they win arbitration. Thus, higher network loads or higher rates of traffic injection lead to higher buffer utilization. In Figure 2(a), we plot the average buffer utilization across all the buffers in a 2D mesh network and the standard deviation of buffer utilization for uniform random traffic. [1] When the injection rate is low, the utilization of buffers is very low (under 1% utilization) with very small variance across the network. As the network load approaches saturation, both the buffer utilization and the variance increase. However, simulations results show that even when the network is in saturation, not all of the buffers are fully utilized or occupied. Network saturation is defined as the load when *some* resource in the network is saturated [4]. Thus, even if the network is saturated, not all of the buffer resources are fully utilized.

To better understand the distribution of buffer utilization, a buffer utilization histogram of all the buffers in an on-chip network is shown in Figure 2(b). The y-axis represents the fraction of the input buffers [2] and the x-axis represents the utilization or occupancy of the buffers – i.e., what fraction of the buffer entries within a router input port is currently occupied by packets and not empty. This data was collected after network saturation and is a *snapshot* of the buffer occupancy at a given time. The results show that not all of the channels (and thus, all of the buffers) are saturated when the network becomes saturated, thereby resulting in a significant portion of the buffers being not utilized. However, the actual buffers that are not utilized vary with different traffic patterns and network load; thus, removing unutilized buffers is not a practical option while reducing the overall buffer space can cause performance degradation, as shown earlier in Figure 1. In this work, we propose the use of fine-grained power gating in on-chip network buffers to exploit the low utilization of buffers and reduce the buffer leakage power.

As energy consumption is a significant factor in digital systems, it is not only important to reduce energy consumption, but also to achieve *energy-proportional* computing [1] – i.e., use energy proportional to the amount of work done. This concept was originally proposed in datacenter computing and extended to datacenter networks. In this work, we attempt to achieve similar energy-proportionality in on-chip networks through better buffer management. In Figure 3, we plot the power consumption of an on-chip network router from Orion 2.0 [8] for a 32nm technology. The activity factor of the network is varied, and the total power consumption for a 5-port router is shown. At a very low activity factor, a significant portion (over 95%) of the total power is from static (or leakage) power. Even for high activity factor, the static power consumption still represents 50% of the total power consumption. Thus, it is critical to minimize the number of active buffer entries, especially when the buffers are not utilized.

2.3 Related Work

The bufferless routers share similar goals to our work, and different bufferless flow control approaches were discussed earlier in Section 2.1. A low-cost router microarchitecture [9] was proposed to reduce the complexity of on-chip networks by simplifying the router microarchitecture. The size of the input buffers was minimized while intermediate buffers were required between the dimension-sliced switches. The techniques described in this work can be extended to intermediate buffers to minimize their leakage power. Power-aware buffers [2] were proposed to minimize leakage power, and they described different design spaces of power-aware buffers. Our microarchitecture can be classified as *predictive* according to their terminology [2]. However, they do not provide details of implementing different power-aware buffers, and their study focused on large-scale, off-chip networks, whereas our work focuses on on-chip networks which have different constraints [3]. Matsutani et al. [10] proposed ultra fine-grained power gating where each component in a router is individually managed. Their work includes the power-gating of buffers but at a granularity of the entire virtual channel buffer, whereas our work uses the power-gating of each buffer entry granularity.

3. FLEXIBUFFER

We propose FlexiBuffer, which reduces the leakage power by leveraging fine-grained power gating of buffer entries and, thus, flexibly manages the size of active buffers. With power-gating, each buffer entry can be in one of two states: an *active* or ON state or a *sleep* or OFF state. We also define the *active* window size within each input buffer of a router as the number of buffer entries that are active or ON.

[1] Other traffic pattern causes a more imbalance in buffer utilization while uniform random traffic creates the most random (or load-balanced) traffic.

[2] For the routers on the edge of the 2D mesh network, some of the ports are not connected and we ignore the impact of those buffers in this work and assume they are always power-gated.

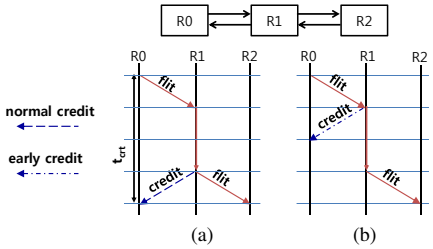


Figure 4: Timeline of credit-based flow control with (a) a normal credit and (b) an early credit.

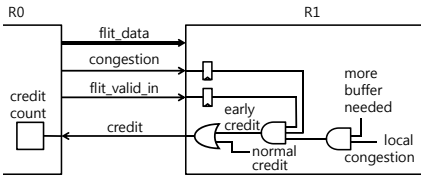


Figure 5: Block diagram for congestion detection.

3.1 Modified Credit-based Flow Control

Backpressure in the on-chip network is often managed through credit-based flow control. In conventional credit-based flow control, credit counts are maintained for the downstream router's buffers. For each flit sent downstream, the credit count is decremented. Once the flit leaves the downstream router, a credit is sent back upstream to signify to the upstream router that a buffer entry has become available. We modify credit-based flow control to prevent any flits being written to OFF entries and, as needed, turn on OFF entries and increase the number of ON entries.

To support Flexibuffer, we modify the initialization of the credit count in a conventional credit-based flow control. If a downstream buffer has b buffer entries, upstream routers initialize their credit count with a value of b. However, in Flexibuffer, the credit count is initialized with b_{min}, which represents the minimum number of active or ON buffer entries needed to prevent stalls caused by the lack of available buffer entries. The rest of the buffer entries ($b - b_{min}$) are in the sleep or OFF state. The value of b_{min} can be calculated as

$$b_{min} = max(t_{wakeup}, t_{crt}),$$

where t_{crt} is the credit round-trip latency and t_{wakeup} is the number of cycles needed to wakeup a buffer entry. If $t_{crt} > t_{wakeup}$, the t_{wakeup} latency is completely hidden in the router microarchitecture.

In addition, we define two different types of credits for our flow control.

- **Normal credit**: credit sent upstream corresponding to a flit that leaves for a downstream router (Figure 4(a)).
- **Early credit**: credit sent upstream immediately upon receipt of a flit when congestion is detected. While early credit is sent upstream and a corresponding flit is sent downstream, the local router activates (or turns on) another buffer entry (Figure 4(b)).

According to these definitions, conventional credit-based flow control only uses normal credit while Flexibuffer also requires early credit. For an incoming flit, an early credit is immediately sent upstream if the following two conditions are met: 1) there are buffer entries that are in the OFF state (i.e., buffer entries can be turned on) and 2) congestion exists in both the current buffer and the upstream router. [3]

num. of credits	early credit	normal credit	window size	description of condition
0	No	No	decrease	congestion decreased
1	Yes	No	same	no congestion
1	No	Yes	same	no congestion
2	Yes	Yes	increase	congestion increased

Table 1: Different combination of credits transmitted in FlexiBuffer.

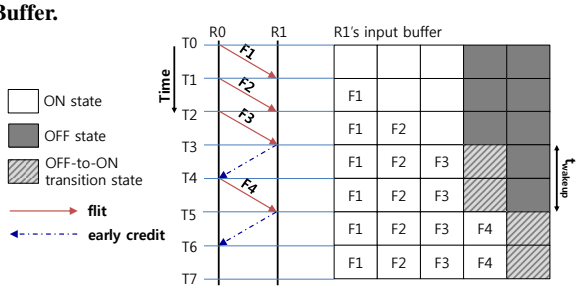


Figure 6: An example of active window management with early credit under extreme congestion.

For an outgoing flit, a normal credit is always sent except for when the available number of entries, which must be ON and empty, is larger than t_{wakeup}. To summarize, different types and numbers of credits can be sent upstream depending on the congestion of the networks as shown in Table 1. In some case, no credit will be sent for a given flit if there is no congestion and the current active window size is greater than the b_{min}. This condition represents when congestion exists but is being reduced. Therefore, the active window size will be reduced and an ON entry will be turned off.

3.2 Congestion Detection

In addition to the modified initial credit count value, the additional changes needed in the router microarchitecture are the congestion detector in the upstream router and the logic to generate the early credit signal (Figure 5). To determine how the active window size has to be changed, an input port control needs to detect both upstream and local congestion. The upstream router ($R0$ in Figure 5) generates a congestion signal if the *contention degree* for this particular output is greater than or equal to a threshold value. In our evaluation, we use a threshold value of 2 – i.e., there is at least one additional flit that needs to be transmitted from the upstream router. This congestion signal is transmitted with the flit data to the current router. The local router ($R0$ in Figure 5) generates its own congestion signal depending on whether the front of the queue will be removed or not. If it is not removed, it indicates congestion in the current router – i.e., the input port loses arbitration to another input port. Since early credit is sent only when there is an incoming flit, indication of incoming flit ($flit\_valid\_in$ in Figure 5) from the upstream router is AND'ed with this congestion signal (both upstream and local congestion signal) to decide if an early credit needs to be sent. For each early credit sent upstream, an additional buffer entry is turned on.

3.3 Buffer Window Example

Figure 6 shows an example of how FlexiBuffer works. Transmission of flits (from $R0$ to $R1$) and credits (from $R1$ to $R0$) are shown on the left, and the occupancy of the input buffer for $R1$ is

[3]Discussion of how congestion is determined is discussed in Section 3.2.

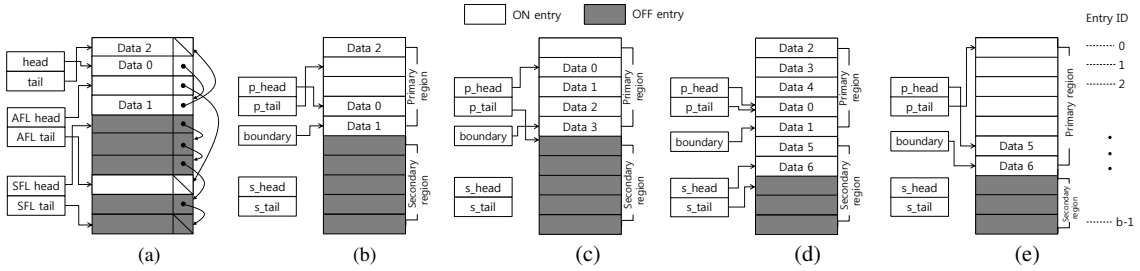


Figure 7: Block diagram of a (a) modified linked-list buffer management (MLL) and split queue (SQ) buffer management in ((b), (c), and (e)) unified mode and (d) split mode. Usually, SQ is in (b) unified mode and the primary region works similar to a circular queue. (c) is when the primary region can expand, but in (d), it couldn't expand and switched to split mode. Later, it will return to (e) unified mode after primary region is emptied, and the primary region will be set to all ON entries.

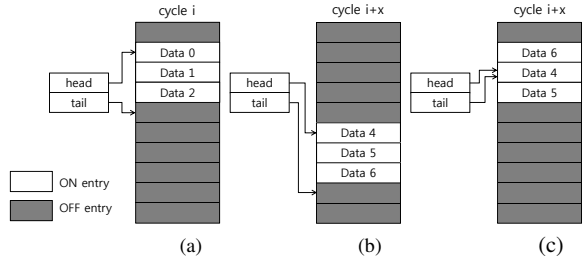


Figure 8: (a) Initial buffer occupancy with baseline, buffer occupancy after x cycles with (b) baseline circular buffer and (c) an ideal buffer management.

shown on the right. In this example, we assume that R0 has four flits (F1 – F4) and three credits, and $b_{min} = 3$ and $t_{wakeup} = 2$ are used as parameters. Initially, while two flits ($F1, F2$) are sent from $R0$ to $R1$, no congestion is detected and no early credit is sent. As a result, at $T3$, $R0$ has no credit and cannot send $F4$. At the same time, an early credit is sent upstream and arrives at $R0$ at $T4$, and then, $R0$ can send $F4$. As denoted by the shaded squares in Figure 6, while the early credit is sent upstream and a corresponding flit ($F4$) is sent downstream, the downstream router activates an OFF entry and $F4$ can be stored in this buffer entry.

4. BUFFER MANAGEMENT

4.1 Circular Buffer Problem

Buffers are commonly managed as circular buffers with a head and a tail pointer [4]. After every write to the buffer, the tail pointer is incremented, and after every read, the head pointer is incremented. When a head or a tail pointer reaches the last buffer entry, it is wrapped around and points to the first entry. However, using this buffer management with fine-grained power gating can cause more power consumption by continuously turning buffer entries on and off. For example, if we assume an active window size of one and there is no congestion (i.e., the active window size will not change), only one buffer will be active at any given cycle. However, the actual physical buffer entry that will be utilized will continuously change. As a result, although inactive buffer entries will continue to be in the OFF state, additional power overhead is required to turn buffer entries on and off when logically, only one buffer entry is actually being utilized.

An example of circular buffer management is shown in Figure 8. Current buffer occupancy is shown in Figure 8(a) with three buffers in the ON state and the rest of the buffer entries in the OFF state. Assuming the traffic load is constant and no additional congestion occurs in the network, at x cycles later, the head and tail pointers would have incremented such that three different buffer entries will have been turned on while the previous three buffer entries are turned off (Figure 8(b)). Although this still maintains the maximum number of buffer entries in the OFF state, it requires continuous activation and deactivation of buffer entries. Instead, an *ideal* buffer management would attempt to reuse the same *physical* buffer entries, as shown in Figure 8(c). To overcome this limitation with the circular buffer management, we propose two buffer management mechanisms, extending a conventional linked-list based approach and a novel, split queue organization.

4.2 Modified Linked-List (MLL) Buffer Management

In conventional linked-list based buffer management [4], there is a *head* and a *tail* pointer which point to the front and end of the logical FIFO queue, respectively, and each buffer entry has a pointer to the next buffer entry. A *free* pointer also exists that is used to keep track of free buffer entries. This baseline linked-list buffer management manages the buffers as a circular queue. To overcome the problem of excessively turning on and off buffer entries, additional head and tail pointers are used to differentiate between an *active* free list (AFL) and a *sleep* free list (SFL) (Figure 7(a)).

Initially, b_{min} entries are in the active state and present in AFL. As flits arrive and get buffered, entries from the AFL are popped and transferred to the linked list. When flits depart from the buffer, the buffer entry is added to the SFL if the entry is turned off; otherwise, the buffer entry is added to the AFL. Only when the active window size needs to be increased is an entry popped from SFL and pushed back to AFL. While an entry is being woken up and transitions from the sleep state to active state, modified credit-based flow control will prevent any flits being written on this entry. To manage both AFL and SFL, separate head and tail pointers are needed for each list in addition to the linked list's head and tail pointers.

The modified linked-list approach provides an efficient mechanism to manage buffers, as each entry can be properly managed. However, additional complexity is introduced, as a multi-ported register file is needed to allow simultaneous access. In addition, each buffer entry has storage overhead for the pointer, and these additional bits cannot be power gated.

4.3 Split Queue (SQ)

To overcome the limitation of the modified linked-list approach and simplify buffer management, we propose a novel *split queue* organization. The buffer is separated into two regions – the *primary* and *secondary* region. There is a *boundary* pointer that separates the two regions and points to the last entry of the primary region. Each region has its own $head$ and $tail$ pointers, and sim-

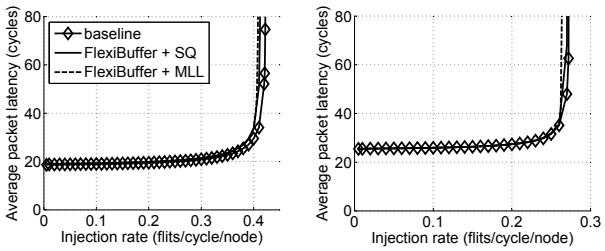


Figure 9: Performance comparison of baseline and FlexiBuffer for (a) uniform random and (b) tornado traffic pattern.

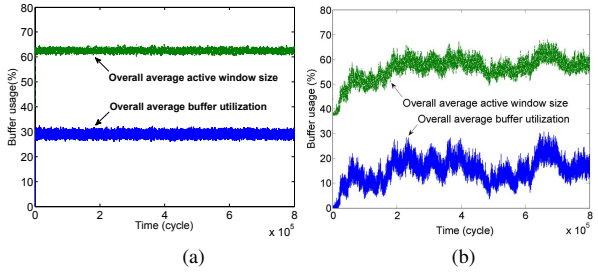


Figure 10: Average active window size with (a) uniform random and (b) bursty traffic patterns.

ilar to other buffer management schemes, they point to the logical beginning and end of the buffer that is currently being occupied, respectively. With these two regions, the split queue can operate in either a *unified* mode or a *split* mode. In the unified mode, only the primary region is used while in split mode, both regions are utilized.

Initially, a buffer is in unified mode and has b_{min} active entries in the primary region as buffer entries labeled 0 to $b_{min} - 1$ are in the active state. The pointers for the secondary region (s_head, s_tail) are not used in the unified mode. If there is no congestion, the b_{min} active entries in the primary region operate as a circular queue and flits are written and read from the primary region as shown in Figure 7(b). However, as the size of active window increases (or additional buffer entries need to be turned on), the buffer either remains in a unified mode or switches to a split mode, depending on the location of the p_head and p_tail pointers.

If $p\_head < p\_tail$ (i.e., the buffer entry ID [4] addressed by p_head is smaller than the buffer entry ID addressed by p_tail as shown in Figure 7(c)), the buffer remains in the unified mode and the size of the primary region increases downwards as more buffer entries are needed. Since the buffer continues to remain in the unified mode, the boundary buffer is also increased appropriately. However, as congestion decreases and the active window size needs to be reduced, the boundary pointer is incremented to reduce the size of the primary region. If the buffer entry being pointed to by the $boundary$ pointer is occupied, the active window does not shrink immediately but needs to wait until the data flit is removed from the buffer. However, if $p\_head \geq p\_tail$ as shown in Figure 7(d), extending the primary region downwards will complicate the pointer management. Since the buffer is logically a FIFO with packets being read out in the order that it was written, continuing to increase the primary region when $p\_head \geq p\_tail$ would require multiple head and tail pointers to properly manage the order of flits – or pointers similar to linked list, since the last entry of the primary region will need to point to the first entry of the secondary region. To avoid this complex pointer management, we change the buffer from unified mode to split mode and buffers in the secondary region are turned on, one by one as needed. The size of the two regions are fixed in the split mode to avoid complicating pointer management and the secondary region pointers are initialized to $s\_head = s\_tail = boundary + 1$. Once the primary region is full, the additional flits are written into the secondary region. The flits are first read out from the primary region and once it has been drained, the flits from the secondary region are read out. In addition, once the primary region is drained, the buffer switches back from split mode to unified mode (Figure 7(e)), and the ON entries in the secondary region are added to the primary region. The secondary region shrinks accordingly and the $boundary$ pointer is established again between the two regions. One limitation of the split queue design is that while the primary region is being drained, no additional flits can be written to this region and thus, the active window size is effectively reduced until the buffer returns to the unified mode.

[4]The buffer entry ID is shown on the right in Figure 7.

5. EVALUATION

In the evaluation , we used a cycle accurate simulator [4] and evaluated FlexiBuffer on a 8×8 2D mesh topology using dimension-ordered (XY) routing with $b = 8$, $v = 4$ (number of virtual channels), and an aggressive single-cycle router and $t_{wakeup} = 2$ parameters. The leakage power of the alternative microarchitectures is estimated using Orion 2.0 [8]. We modified Orion2.0 such that reduction of buffer leakage power and the overhead of turning on OFF entries are taken into account when the power consumption is calculated. We assumed 32nm technology with 1.5 GHz clock frequency, 1.0V VDD, and a register file implementation of the buffers. The energy overhead of turning on a buffer entry is assumed to be equivalent to the leakage power savings of turning off a buffer entry for 10 cycles [6].

5.1 Performance

Performance of FlexiBuffer with modified linked-list (FlexiBuffer +MLL) and split queue (FlexiBuffer+SQ) implementations are compared with baseline router microarchitecture which does not have any power gating. As shown in Figure 9, there is minimal loss in performance, approximately only 3% degradation in throughput while the same zero-load latency is achieved. By maintaining an active window size of b_{min}, there is minimal loss in throughput while obtaining a significant benefit in reduction of power (Section 5.2). The results in Figure 9 are shown for only two particular traffic patterns (uniform random and tornado traffic) but results for other traffic pattern also follow similar trend – minimal loss in throughput for FlexiBuffer compared to the baseline microarchitecture.

With steady-state traffic patterns, the active window size remains relatively constant throughout the simulation. For example, the active window size (and the actual buffer utilization) for uniform random traffic at an injection rate of 1.0 is shown in Figure 10(a). The values in Figure 10 are averaged across all the router buffers in the network. Both the buffer utilization and the active window size remains relatively constant since the traffic rate or the traffic pattern does not change. Thus, in addition to steady-state traffic pattern, we also evaluated a bursty traffic pattern [4]. The latency-throughput curve trend comparison for the bursty traffic also follows similar trend as shown in Figure 9 but the active window size changes as shown in Figure 10(b). As the traffic load changes, the active window size increases or decreases to adapt flexibly to the network load.

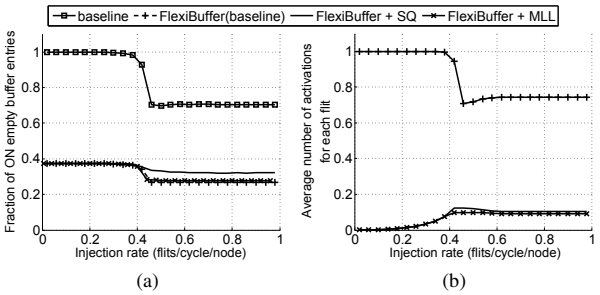


Figure 11: (a) Fraction of active empty entries across all the buffers and (b) average number of entries activated for each flit is shown for uniform random traffic pattern.

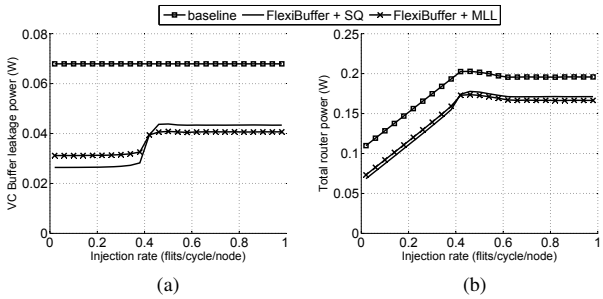


Figure 12: (a) Leakage power consumption of a VC buffer and (b) total router power are compared as injection rate is varied with uniform random traffic pattern.

5.2 Leakage Power Reduction

As described earlier in Section 2, significant number of buffer entries can be empty, even at high load, and results in high leakage power consumption. FlexiBuffer minimizes this by using power gating to turn off unused buffer entries.

Figure 11(a) shows the fraction of active or ON buffer entries that are empty and not occupied by a flit. With the baseline microarchitecture, at low load, most of the buffers are not being occupied and thus, this fraction approaches one. Even at high load (near saturation), a significant portion of the buffers (approximately 70%) is still empty but in an active or ON state since no power gating is used. In comparison, split queue reduces the number of active empty buffer entries by approximately 63% near zero-load and by approximately 54% at high load. As a result, the average leakage power of buffers is reduced by 61% near zero-load, and 36% at high load as shown in Figure 12(a). Compared with split queue (SQ), the modified linked-list (MLL) consumes 15% more leakage power near zero-load, because of its storage overhead by pointers, which cannot be power-gated. However, the modified linked-list is able to further reduce leakage power at high load (by 12%) because it does not have the limitation of split queue, where entries in the primary region cannot be turned off in split mode.

In addition to the reduction in the number of buffers that are turned off, we compare the number of times that flit buffer entries are unnecessarily turned on. In Figure 11(b), we plot the average number of buffers that needs to be activated for the arrival of each flit. Ideally, this value should be close to zero as a value of one means that for the arrival of every flit, another buffer entry needs to be turned on. For the baseline circular buffer approach, number of times buffer entries are turned ON for each flit is approximately one at low load since new physical buffer entries need to be activated with a circular buffer organization. In comparison, both the MLL and the SQ approach have a value near zero at zero-load and only reach a value of approximately 0.1 at saturation. Unlike circular buffer organization, only few buffer entries need to be activated with both MLL and SQ as the same physical buffer entries in the active window are reused. Based on the advantages of the new flow control and the split queue organization, the energy consumption with FlexiBuffer is shown in Figure 12(b). Compared with the baseline, FlexiBuffer is shown to achieve energy savings of approximately 39% at low utilization and approximately 13% at high network traffic load.

6. CONCLUSIONS

In this work, we showed how not all on-chip network router buffers are fully utilized, and they provide an opportunity to turn-off the buffer entries to minimize leakage power. We proposed FlexiBuffer, a flexible and adaptable buffer management that leverages fine-grained power gating to adjust the number of active buffer entries in each router. We modified credit-based flow control by introducing *early* credits to adjust the active window size of the buffer appropriately. We also identified how circular buffer management creates a significant problem as different physical buffer entries need to be continuously turned on and off. To overcome this limitation, we proposed a novel split queue organization that allows the same physical buffer entries to be reused. Our results show that with minimal loss in performance, FlexiBuffer can reduce the leakage power consumption of on-chip network buffers by up to 61%.

7. REFERENCES

[1] L. A. Barroso and U. Holzle. The case for energy-proportional computing. *Computer*, 40:33–37, 2007.

[2] X. Chen and L.-S. Peh. Leakage power modeling and optimization in interconnection networks. In *Proc of the international symposium on Low power electronics and design*, pages 90–95, Seoul, Korea, 2003.

[3] W. J. Dally and B. Towles. Route packets, not wires: on-chip inteconnection networks. In *Proc. of the Design Automation Conf (DAC)*, pages 684–689, Las Vegas, NV, 2001.

[4] W. J. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, San Francisco, CA, 2004.

[5] M. Hayenga, N. E. Jerger, and M. Lipasti. Scarab: a single cycle adaptive routing and bufferless network. In *Proc of IEEE/ACM International Symp. on Microarchitecture (MICRO)*, pages 244–254, New York, NY, 2009.

[6] Z. Hu, et al, Microarchitectural techniques for power gating of execution units. In *Proc. of international symposium on Low power electronics and design*, pages 32–37, Newport Beach, CA, 2004.

[7] S. A. R. Jafri, Y.-J. Hong, M. Thottethodi, and T. N. Vijaykumar. Adaptive flow control for robust performance and energy. In *Proc. of IEEE/ACM Intl Symp. on Microarchitecture*, Atlanta, GA, 2010.

[8] A. Kahng, B. Li, L.-S. Peh, and K. Samadi. Orion 2.0: A fast and accurate noc power and area model for early-stage design space exploration. In *Proc of Design Automation and Test in Europe (DATE)*, Nice, France, 2009.

[9] J. Kim. Low-cost router microarchitecture for on-chip networks. In *Proc. of IEEE/ACM Intl Symp. on Microarchitecture*, pages 255–266, New York, NY, 2009.

[10] H. Matsutani, et al, Ultra fine-grained run-time power gating of on-chip routers for cmps. In *Proc of International Symp. on Networks-on-Chip (NOCS)*, pages 61–68, 2010.

[11] G. Michelogiannakis, D. Sanchez, W. J. Dally, and C. Kozyrakis. Evaluating bufferless flow control for on-chip networks. In *Proc of International Symp. on Networks-on-Chip (NOCS)*, pages 9–16, 2010.

[12] T. Moscibroda and O. Mutlu. A case for bufferless routing in on-chip networks. In *Proc. of the Intl Symp. on Computer Architecture (ISCA)*, pages 196–207, Austin, TX, 2009.