Power Punch: Towards Non-blocking Power-gating of NoC Routers

Lizhong Chen[1], Di Zhu[2], Massoud Pedram[2], and Timothy M. Pinkston[2]
[1]*School of Electrical Engineering and Computer Science, Oregon State University, USA*
[2]*Department of Electrical Engineering, University of Southern California, USA*
chenliz@eecs.oregonstate.edu, {dizhu, pedram, tpink}@usc.edu

Abstract

As chip designs penetrate further into the dark silicon era, innovative techniques are much needed to power off idle or under-utilized system components while having minimal impact on performance. On-chip network routers are potentially good targets for power-gating, but packets in the network can be significantly delayed as their paths may be blocked by powered-off routers. In this paper, we propose Power Punch*, a novel performance-aware, power reduction scheme that aims to achieve non-blocking power-gating of on-chip network routers. Two mechanisms are proposed that not only allow power control signals to utilize existing slack at source nodes to wake up powered-off routers along the first few hops before packets are injected, but also allow these signals to utilize hop count slack by staying ahead of packets to "punch through" any blocked routers along the imminent path of packets, preventing packets from having to suffer router wakeup latency or packet detour latency. Full system evaluation on PARSEC benchmarks shows* Power Punch *saves more than 83% of router static energy while having an execution time penalty of less than 0.4%, effectively achieving near non-blocking power-gating of on-chip network routers.*

1. Introduction

A significant challenge for the design of current and future many-core chips is how to provide fast and efficient on-chip communication. While network-on-chip (NoC) offers a potentially scalable solution, current designs consume substantially more power than may be needed (e.g., up to 28% in Intel Teraflop [16] and 19% in Scorpio [10]), with a large percentage of static power (over 60% even for simpler router designs) due to relatively low average traffic load of real applications. Static power consumption is exacerbated as transistor feature sizes continue to shrink. Meanwhile, with more cores being integrated on chips, there is an increasing demand for low latency NoCs due to the longer core-to-core hop distance. Thus, it is of paramount importance to devise effective NoC static power saving techniques that do not compromise NoC performance.

Power-gating is a very promising technique that can reduce the static power component dramatically. When applied to on-chip network routers, however, power-gating is prone to incur a significant increase in packet latency due to blocking. When a router is powered off, it essentially blocks all paths that intersect with the router. Packets thus have to wait for the router to wake up before proceeding and experience wakeup latency multiple times before delivery in cases where many routers along the path are powered off. This leads to large cumulative delay that is pronounced even when common optimizations are applied which are designed to hide the wakeup latency, at least partially. Another approach to mitigate this blocking problem in power gating is to deflect packets via routing and topology reconfiguration. Nevertheless, existing reconfiguration methods either are too slow (~10K cycles) to capture the short but exploitable router idle periods (~10-100 cycles) or can cause a large number of packet detours due to simplified reconfiguration algorithms. Moreover, dynamic reconfiguration unnecessarily complicates the already complex designs of on-chip networks.

In addressing this blocking problem comprehensively, in this work we propose *Power Punch*, a novel performance-aware power-saving scheme that aims to achieve non-blocking power-gating of on-chip network routers. The basic idea of Power Punch is to always send power control signals ahead of packets to "punch through" any blocked routers in power-gated mode along the imminent path of packets so that packets can be transported without having to suffer any router wakeup latency or packet detour latency.

Power Punch consists of two mechanisms that solve major challenges in sending punch-through power control signals. The first challenge concerns the tension between the amount of power control information needed to be propagated across multiple hops ahead of packets and the tight constraints in allocating control wires for this purpose. Using dedicated wires for each wakeup control signal incurs prohibitive hardware cost whereas sharing wakeup signals introduces serious contention delay that could defeat the purpose of sending wakeup signals in advance. The first proposed mechanism utilizes the properties and constraints of the network to minimize the needed control information and is able to merge all the signals arriving at a router in the same cycle, thereby propagating punch-through power control signals in a low-cost and contention-free fashion. In addition, the multi-hop power control signals forewarn routers to know precisely whether there will be packets arriving in the next few cycles. This helps to filter out short counter-productive idle periods (i.e., less than the break-even time), and ensure the resulting router wakeups are accurate and necessary. The second challenge concerns how to punch through powered off routers that are close in vicinity to the injection node and, therefore, do not have enough remaining hops (i.e., hop count slack) for sending wakeup control signals to cover the wakeup latency. Our second proposed mechanism holistically exploits existing slack in the network interface at injection nodes. In essence, the mechanism allows wakeup control signals to be sent before packets are

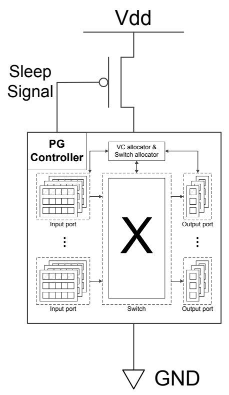


Figure 1: Power-gating technique.

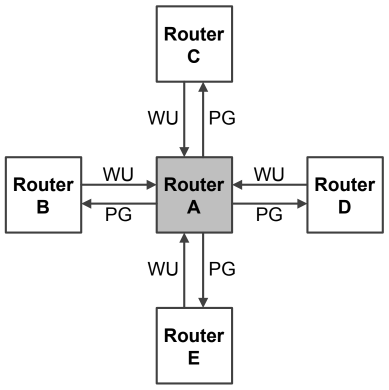
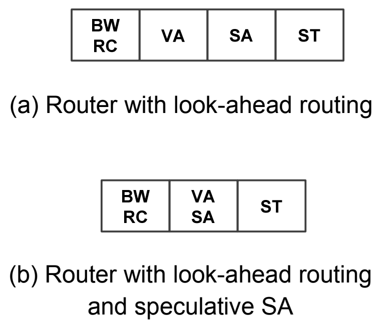


Figure 2: Power-gating handshaking.



BW RC
VA
SA
ST
(a) Router with look-ahead routing
BW RC
VA SA
ST
(b) Router with look-ahead routing and speculative SA

Figure 3: Router pipeline designs.

generated, thus compensating for the otherwise insufficient slack in hop count.

Full-system simulations show that, compared to an optimized conventional power-gating technique applied to on-chip network routers, Power Punch achieves a reduction of 61.2% in network performance penalty while also saving 3.8% more router energy. When compared with not using power-gating, Power Punch reduces router static energy by 83.7% while incurring only 0.4% increase in execution time, essentially achieving near non-blocking power-gating which is the goal.

This research increases our understanding of how to maintain performance in the presence of power-gating of on-chip network routers and provides key insights on the viability of achieving non-blocking power-gating. The rest of the paper is organized as follows. Section 2 provides more background on power-gating and its blocking issue. Section 3 describes the rationale of Power Punch and discusses challenges in achieving it. Section 4 provides details of the proposed Power Punch scheme. Section 5 presents our evaluation methodology, and Section 6 provides simulation results. Finally, related work is summarized in Section 7, and Section 8 concludes the paper.

2. Background and Motivation

2.1 High Static Power in On-chip Network Routers

On-chip networks provide a scalable approach for supporting parallel communication in many-core CMPs. They should be designed so as not to incur considerable overhead in chip area and power. While the area overhead becomes less of a concern as more and more transistors are being integrated on a chip, the NoC power problem has been escalating across each technology node due to ever tighter power constraints. Consequently, despite numerous previously proposed novel topologies and routing algorithms, most taped-out commercial and research many-core chips adopt planar mesh-based topologies with dimension-order routing in practice to reduce NoC overhead (e.g., Intel SCC [17], TRIPS [13], Scorpio [10], Adapteva Epiphany family [1], Tilera TILE-Pro and TILE-Gx families [30]). Even with these simpler implementations, the NoC still consumes substantially more power than necessary, with a large amount consumed by its static power component.

To illustrate, we conduct full-system simulations using gem5 [4] and multi-threaded PARSEC benchmarks [3] on an 8x8 mesh network with XY routing and wormhole switching. The number of virtual networks is configured to be three, the minimum number needed for correctly running the MESI coherence protocol without deadlocks. Each virtual network also has a relatively small buffer configuration, with two 3-flit sized data virtual channels (VCs) and one 1-flit sized control VC. Simulation results from the DSENT [29] NoC power simulator integrated in gem5 show that, under this simple NoC design and minimal resource configuration, router static power still accounts for nearly 64% of the total router power consumption assuming 45nm technology. This is because router components other than buffers also consume noticeable power [6] and because the average network utilization in real benchmarks is relatively low. As chip designs penetrate further into the dark silicon era, the static power component of NoCs will only get worse as more processing cores can be powered off or are operated at lower frequencies, thereby generating less network traffic and leading to higher static power percentage.

2.2 Applying Power-gating to On-chip Network Routers

One way that can dramatically reduce static power is to apply power-gating techniques to each NoC router. This can be very effective as it exploits the idleness exhibited in each router while reducing the static power of all components in a router. As depicted in Figure 1, power-gating is implemented by inserting an appropriately sized header transistor(s), usually a high threshold and non-leaky "sleep switch", between Vdd and the router. When the sleep signal is asserted by the power-gating controller (which is a small hardware component that is always powered on), the supply voltage to the router is cut off, thus eliminating the leakage currents in both the subthreshold conduction and reverse-biased diodes.

Different from other system components, when applying power-gating to on-chip network routers, extra handshaking signaling is needed between neighboring routers to ensure the correct delivery of packets. As shown in Figure 2, besides generating the sleep signal, the power-gating controller also monitors the emptiness of the router datapath and the wakeup signals from neighbors. When the datapath of a router, for example, router *A* is empty (i.e., input buffers, output registers and crossbar are empty) and no wakeup signals (*WU*) come from neighbors, the controller in router *A* asserts the sleep signal after a timeout period[1] to put router *A* into gated-off state and notifies its neighbors by asserting the *PG* signal. Upon detecting the asserted *PG* signal, neighboring routers mark the corresponding output ports as unavailable in their switch allocator. Later, if a packet in router *B* or in other neighbors of router *A* needs to be forwarded to router *A*, a *WU* signal will be asserted which triggers the controller in router *A* to de-assert its sleep signal. The packet is then stalled in router *B* until router *A* is fully awoken and the PG signal is cleared. Hence, the wakeup latency of router *A*, including the blocking latency due to being powered off, is directly part of the overall latency of the packet forwarding process.

2.3 Blocking Problem in Conventional Power-gating

As can be seen, the primary concern with the above conventional way of applying power-gating to routers is the negative performance impact caused by wakeup latency. Essentially, when a router is powered off, it blocks all the paths of a packet that overlap with the router (i.e., forwarding path from any of the router's input ports to its output ports). In what follows, we use the term *blocking power-gating* to refer to this phenomenon.

The blocking problem in conventional power-gating can be prohibitive. Prior works [6, 7, 9, 24, 28] as well as results given in this work show that the wakeup latency of on-chip network routers is around 6-12 cycles depending on implementation, which is quite sizable. More importantly, a packet may experience wakeup latency multiple times in the network as more than one router along the packet's path could be powered off. Thus, the cumulative delay caused by blocking power-gating is comparable to the zero-load packet latency which also is on the order of tens of cycles. As on-chip network latency is very critical to the overall system performance, the blocking problem must be addressed adequately before power-gating can be applied most effectively.

Several approaches have been proposed so far to combat blocking power-gating, but they have various degrees of effectiveness. One approach is to *deflect* packets when their current paths are blocked by powered-off routers. This needs to be achieved through intricate reconfiguration of routing, topology, or both. For example, a fast reconfiguration method [6] that uses pre-determined paths to bypass gated-off routers reconfigures quickly and can capture both long (i.e., >100 cycles) and short idle intervals (i.e., 10-100 cycles). However, it introduces considerable packet detours and degrades system performance. More extensive but complex reconfiguration algorithms [27, 28] use dynamic information to minimize detours. However, they are slow by comparison and, as a result, reconfigure only on a per-epoch basis (~10K cycles for an epoch) to capture idleness on a very coarse granularity. Also, their uses are limited to scenarios in which a couple of cores and the co-located caches are completely idle with no communication with other cores and caches, which might be impractical for typical shared cache architectures. Additionally, the routing and topology reconfiguration due to power-gating unnecessarily complicates the simple design of deterministic routing in mesh networks.

Another approach to deal with blocking power-gating is to control powered-off routers more effectively using techniques directly aimed at reducing waiting time. A common technique is to send the wakeup signal early to the next router as soon as the output direction is computed [24]. This hides a few cycles, equivalent to the number of router pipeline stages but typically is not sufficient to cover the entire wakeup latency. Another technique is to apply a timeout mechanism after a router is detected as being idle. This technique intends to filter out short idle periods that are less than the *break-even time*[2] to reduce the possibility of encountering a powered-off router. However, the timeout value cannot be too long (around 4 cycles [7, 9]) as false filtering essentially wastes the remaining idle cycles that can be exploited by power-gating. Finally, several techniques have been proposed to power-gate individual components within a router [24, 25, 26]. This approach reduces the chance of encountering powered-off router components at the cost of substantially higher implementation complexity. Yet, it still cannot entirely remove the blocking when a powered-off component is needed for packet forwarding, and it does not mitigate the cumulative wakeup latency problem either.

Therefore, given the severe performance penalty that can be caused by wakeup latency and the many limitations in existing approaches, it is imperative for a novel scheme to be devised that can solve the blocking problem comprehensively, ideally achieving *non-blocking power-gating*.

3. Challenges in Achieving Non-blocking Power-gating

Non-blocking power-gating of on-chip network routers can be achieved by pre-powering up routers along network paths taken by packets in advance of packet arrival. To accomplish this, fundamental challenges must be adequately addressed. One major challenge concerns how to completely hide router wakeup latency across multiple hops with minimal overhead; another concerns how to wake up routers located at or neighboring nodes that inject packets into the NoC at the source. Below, these challenges are discussed in detail.

In order to hide wakeup latency of T_{wakeup} cycles completely, the number of hops that a wakeup signal should be sent ahead of a packet is determined by $\lceil T_{wakeup}/T_{router} \rceil$, assuming the packet takes ($T_{router}+T_{link}$) cycles per hop, whereas the wakeup signal takes T_{link} cycles per hop. For example,

[1] A minimum of two-cycle timeout is needed to allow packets that already left upstream routers to be received correctly. Additional timeout cycles can be used to filter short idle periods. More details are in next subsection.

[2] Break-even time (BET) is the minimum number of consecutive cycles that a gated circuit block needs to remain idle before waking up to offset the energy overhead of one power-gating process (e.g., charge capacitance, distribute sleep signal). BET is around 10 cycles for on-chip routers [7].

Figure 3 shows two common router pipeline designs [20]. The router design in Figure 3(a) uses look-ahead routing, resulting in a 4-stage pipeline of buffer writing (BW), VC allocation (VA), switch allocation (SA) and switch traversal (ST). The router design in Figure 3(b) further optimizes with speculative switch allocation, which reduces another pipeline stage if the speculation is successful. With a T_{wakeup} of 8 cycles, wakeup signals need to be sent at least 2 hops in advance for the case in Figure 3(a) and 3 hops in advance for the case in Figure 3(b) in order to hide the entire T_{wakeup}.

However, for a given router, the total number of routers that needs to be monitored may quickly become very large even within a short hop distance. Figure 4 shows an example of an 8x8 mesh network. There are 24 routers within 3 hops of router 27 (denoted as *R27* hereafter), which accounts for nearly 38% of all routers on the chip. This means that, assuming the 3-stage speculative router pipeline design shown in Figure 3(b), *R27* needs to monitor the wakeup signals from all the 24 routers that are either sent to wake up *R27* (e.g., from *R3* to *R27*) or sent to wake up other routers but need to be relayed at *R27* (e.g., from *R26* to *R29*). This makes it very challenging to monitor and propagate effectively all the needed wakeup signals in the network.

A straightforward way to achieve this is to allocate dedicated wire channels for every wakeup control signal between a router and its monitored routers (e.g., 24 separate incoming wire channels are needed for *R27*). Note that it is not enough to have only 1-bit wire channels for the wakeup control signal. For instance, if the wire channel from *R26* to *R27* is only 1 bit, *R27* will have no idea about if and where this signal should be forwarded after it arrives as the router cannot distinguish whether this 1-bit wakeup signal is intended to wake up *R29*, or *R43*, or any of the 9 routers that are within 3 hops of *R26* with the first hop being *R27*. Hence, to allow correct relay of wakeup signals, minimally 4 bits are needed to distinguish the 9 different cases, totaling 96 bits of wire channels in this example which is prohibitively high, e.g., by comparison, packet payload channel widths typically are 128 bits or 256 bits.

An alternative approach is to share wire channels for the wakeup control signals. This solution may be more viable from the perspective of hardware cost, but it immediately brings to fore the critical issue of possible contention among wakeup signals. In the case of Figure 4, a single wire channel such as the one from *R27* to *R28* may be shared by up to 9 wakeup signals. If only one of them can be transmitted in a given cycle, other wakeup signals arriving at the same cycle will be unavoidably delayed. As any delayed cycle is translated directly into delayed wakeup of the needed router, this can seriously degrade the effectiveness of sending wakeup signals in advance unintentionally, causing blocking to persist. To avoid such contention issues, multiple wakeup control signals must be allowed to be transmitted simultaneously. Nevertheless, merging wakeup signals is very difficult due to the large number of distinct cases. We illustrate by continuing to use as an example the wire channel from *R27* to *R28*. Nine routers (i.e., *R11*, *R18*, *R19*, *R25*, *R26*, *R27*, *R34*, *R35*, and *R43*) may send wakeup signals that use this wire channel, and different signals may be intended to

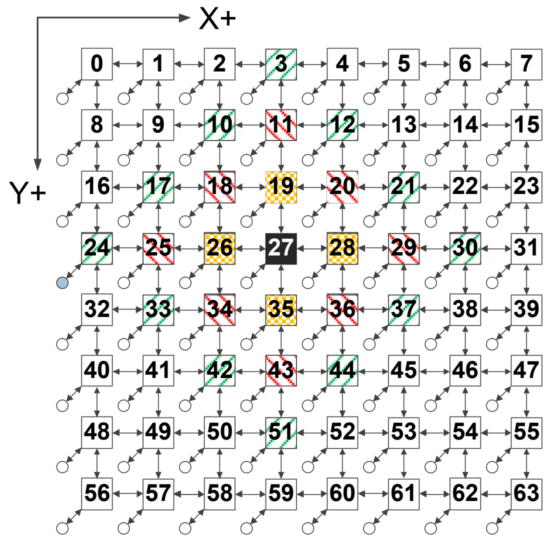


Figure 4: Power Punch challenges and solutions.

wake up different routers leading to a prohibitively large number of combinations. For example, in one cycle, *R27* may need to merge the wakeup signal from *R26* to *R36* and the wakeup signal from *R27* to *R21*. In another cycle, *R27* may need to merge wakeup signals from *R26* to *R20* and from *R27* to *R37*. The merged results are different in these two cases and, therefore, need to be distinguishable in the wire channels. As a result, wire channels need to be wide enough to have different values that can differentiate between all the various combinations of needed wakeup signals. This could lead to very wide wire channels comparable in size to the aforementioned dedicated wire approach. In the next section, we show how this challenge can be addressed by presenting an innovative mechanism that can collectively propagate wakeup signals in a contention-free manner while requiring narrow wire channels.

The other major fundamental challenge arises from the situation of there not being enough routing hop slack to send wakeup signals in advance to fully cover the wakeup latency. This problem is most severe for routers located at or neighboring injection nodes. For instance, if *R24* in Figure 4 is powered off, the associated local node will experience the entire T_{wakeup} latency before being able to inject packets. Our evaluations using PARSEC benchmarks show that, on average, more than 13% of packets received by routers come from local nodes, causing the above performance penalty to occur when those routers are powered off. To address this challenge, other time slack opportunities that holistically include exploiting behavior at injection nodes need to be explored to increase the effectiveness of hiding router wakeup latency, as is achieved with our proposed Power Punch.

4. Power Punch

In this section, we present a detailed description of the proposed Power Punch, a novel scheme that incorporates innovative mechanisms for addressing aforementioned challenges to achieving near non-blocking power-gating. The key rationale for Power Punch is the following: if wakeup information can be cleverly generated and transmitted sufficiently early, power control signals can be sent to "punch through" the network ahead of packets to power up needed routers along the path of packet destinations. From the perspective of packets, transport in the network can be accomplished without having to suffer any router wakeup latency or packet detour latency, as if all NoC routers were virtually always powered on.

In merging wakeup signals, the main obstacle is the tension between the amount of power control information needed to be propagated and the limited power control bandwidth available. The basic idea behind the mechanism for addressing this concern is to utilize the routing and topological properties of the network to minimize the needed information and reduce the width of the merged signals via clever encoding. The proposed mechanism allows all of the wakeup signals arriving at a router in the same cycle to be efficiently merged and relayed, thereby eliminating contention delay. In addressing the challenge of not having enough slack in hop count at or near injection nodes, the basic idea behind the mechanism that addresses this concern is to exploit existing slack at the network interface (NI) from when information for generating wakeup control signals is available and when a packet is generated and ready for injection. This allows wakeup signals to be sent ahead to the source and neighboring routers well enough in advance of packet injection, thus compensating for the otherwise insufficient hop count slack.

Collectively, these mechanisms work in tandem to enable power control signals to "punch though" blocked routers along the entire path of packet destinations, thereby allowing packets to be transported in the network in a near non-blocking fashion. The following subsections describe these mechanisms in further detail.

4.1 Low-cost and Contention-free Multi-hop Punch

To merge and relay wakeup signals across multiple hops, a five-step encoding process can be used to minimize hardware implementation. In this subsection, we explain these steps using the example of sending a wakeup signal 3 hops in advance (i.e., for the speculative router pipeline shown in Figure 3(b)). If needed, a simplified 2-hop and an extended 4-hop design can be derived using similar procedures. It is important to note that wakeup signals should not be sent too early, as this would wake up routers before they are actually needed and, thus, squander powered-off cycles. In practice, 3 hops of slack typically is able to cover router wakeup latency (e.g., hide T_{wakeup} up to 9 cycles for 3-stage routers and up to 12 cycles for 4-stage routers). To facilitate discussion, the term *targeted router* is used to refer to the router that is 3 hops away from the current router (e.g., in Figure 4, if a packet has source *R0*, destination *R7* and is currently in *R3*, then *R6* is the targeted router for the wakeup signal from *R3*). The term *punch signal* is used to refer to the final merged wakeup signals encoded using the proposed Power Punch scheme.

Without loss of generality, we implement Power Punch assuming a 2D mesh network with XY routing. The rationale is the following. First, most commercial and research chips use dimension-order routing to keep NoC overhead low as mentioned in Section 2.1, therefore the proposed scheme can be readily adopted and have high practical impact. Second, the advantage of adaptive routing over deterministic routing becomes prominent only when traffic load approaches saturation. Power-gating is best applied when traffic load is low to medium, a region where there is little distinguishable performance difference between the two routing methods in the absence of power-gating. Third, if certain many-core applications require very high throughput that is not sustainable by deterministic routing, previous works have proposed theory and methodology (e.g., [11, 22]) that allow the routing algorithm to switch from deterministic to adaptive, and vice versa, according to prevailing network loads. This enables the use of Power Punch to save static power under low load conditions while not compromising throughput under high load conditions. Below are the steps.

(1) Determine targeted router based on network topology and routing algorithm

To merge multiple wakeup signals into one punch signal, the targeted router for each wakeup signal must first be determined. We assume a mesh NoC with deterministic routing. The main advantage of mesh XY routing is that, at any given router, the targeted router of the wakeup signal can be easily determined using destination information stored in the packet header. For instance, in Figure 4, a packet currently at *R26* with destination *R31* knows precisely that the targeted router is *R29*, and a wakeup signal will be sent to notify that router (the actual sent wakeup signal will be merged with other wakeup signals into one encoding, as described shortly).

(2) Reduce information for waking up intermediate routers

With an identified targeted router and mesh XY routing, no additional information is needed in the wakeup signal for any intermediate routers along the path to the targeted router that need to be notified (e.g., *R27* and *R28* are along the path from *R26* to *R29*; thus they are implicitly notified if the targeted router is *R29*). This helps to reduce the information needed in wakeup signals and the number of bits of encoding for the final merged punch signals.

(3) Reduce the number of wakeup signals

After reducing the information in each wakeup signal, the next step is to reduce the number of wakeup signals that need to be merged at a given router. Take *R27* as an example. In its X+ direction (i.e., from *R27* to *R28*), wakeup signals from up to 9 routers (i.e., *R11*, *R18*, *R19*, *R25*, *R26*, *R27*, *R34*, *R35*, and *R43*) may need to be relayed from *R27* to *R28* in general. However, due to restrictions in XY routing for avoiding deadlock, only three out of these 9 routers are possible, namely *R25*, *R26* and *R27*. Packets from the remaining six routers do not use the link from *R27* to *R28*, hence do not send wakeup signals along that link (e.g., path *R19*→*R27*→*R28* is not valid as Y+ to X+ turns are illegal). Similarly, in the X- direction of *R27*, only three routers can

Table 1: All possible sets of targeted routers in the X+ direction of *R27* ("||" means or; "&" means and).

#	Represented Wakeup Signals	Set of Targeted Routers	Punch Signal
1	27→28 || 26→28	{ 28 }	00000
2	27→12 || 27→12 & 26→20 || 27→12 & 26→28	{ 12 }	00001
3	27→21 || 27→21 & 26→29 || 27→21 & 26→28	{ 21 }	00010
4	27→30 || 27→30 & 26→29 || 27→30 & 26→28	{ 30 }	00011
5	27→37 || 27→37 & 26→29 || 27→37 & 26→28	{ 37 }	00100
6	27→44 || 27→44 & 26→36 || 27→44 & 26→28	{ 44 }	00101
7	27→20 || 26→20 || 27→20 & 26→20 || 27→20 & 26→28	{ 20 }	00110
8	27→29 || 26→29 || 27→29 & 26→29 || 27→29 & 26→28	{ 29 }	00111
9	27→36 || 26→36 || 27→36 & 26→36 || 27→36 & 26→28	{ 36 }	01000
10	27→12 & 26→29	{ 12, 29 }	01001
11	27→12 & 26→36	{ 12, 36 }	01010
12	27→21 & 26→20	{ 21, 20 }	01011
13	27→21 & 26→36	{ 21, 36 }	01100
14	27→30 & 26→20	{ 30, 20 }	01101
15	27→30 & 26→36	{ 30, 36 }	01110
16	27→37 & 26→20	{ 37, 20 }	01111
17	27→37 & 26→36	{ 37, 36 }	10000
18	27→44 & 26→20	{ 44, 20 }	10001
19	27→44 & 26→29	{ 44, 29 }	10010
20	27→20 & 26→29 || 27→29 & 26→20	{ 20, 29 }	10011
21	27→20 & 26→36 || 27→36 & 26→20	{ 20, 36 }	10100
22	27→29 & 26→36 || 27→36 & 26→29	{ 29, 36 }	10101

be the source for wakeup signals as well. This greatly reduces the number of combinations for targeted routers. As for the Y+ and Y- directions of *R27*, while all nine routers can still be the source of wakeup signals, their targeted routers are limited, so it is straightforward to optimize as explained in the next step.

(4) Reduce combinations of targeted routers

Next, the minimal number of bits needed for encoding the punch signals in order to merge all possible wakeup signals arriving at a router in the same cycle should be determined. As mentioned in Section 3, different sets of targeted routers require distinct punch signals. For instance, a punch signal from *R27* to *R28* with an encoding of "01100" can be used to represent the merged result of wakeup signals from *R26* to *R36* and from *R27* to *R21* (i.e., the set of targeted routers is {36, 21}; see Table 1, entry 13). A different encoding for the punch signal, such as "01111", is needed to merge a different combination of wakeup signals from *R26* to *R20* and from *R27* to *R37* (i.e., the set of targeted routers is {20, 37}), even though the source routers are the same in both cases. However, if the targeted router of one wakeup signal is along the path of the targeted router of another wakeup signal (e.g., *R26* to *R29* is along the path from *R27* to *R21*), the same punch signal encoding of "00010" can be used as in the case where there is only one wakeup signal from *R27* to *R21*. In other words, the width of the punch signal can be optimized to be just wide enough to distinguish all distinctive sets of targeted routers, and no wider.

Based on the results from step (3), up to three routers can be the source of wakeup signals in the X+ direction. *R27* has 9 possible targeted routers (i.e., *R12*, *R20*, *R21*, *R28*, *R29*, *R36*, *R37*, and *R44*); *R26* has 4 (i.e., *R20*, *R28*, *R29*, and *R36*) and *R25* has 1 (i.e., *R28*). Table 1 lists all the distinctive sets of targeted routers for the X+ direction of *R27* (the third column) and the corresponding wakeup signals represented (the second column). The wakeup signal from *R25* is not listed for clarity, as the targeted router for this wakeup signal is always *R28*, which is along the path of a targeted router in any table entry. In total, due to the reduction through previous steps and the use of targeted routers to remove the cases where other targeted routers are implicitly contained, there are only 22 different sets. Therefore, only 5 bits are needed in the punch signal of the X+ direction to distinguish between these sets. Similarly, the width of the punch signal for the X- direction is also 5 bits. Note that we do not use the targeted router numbers in punch signals directly which would otherwise cost 8 bits of encoding each.

For the Y+ and Y- directions, although there are up to 9 routers for sending wakeup signals, the targeted routers have only 3 possibilities in each direction due to the illegal turns from Y to X dimensions (e.g., only *R35*, *R43* and *R51* for Y+). The combination of the three targeted routers result in three distinctive sets in each direction, e.g., {*R35*}, {*R43*}, {*R51*} in Y+ (note that if both *R35* and *R51* are the targeted routers, the resulting set is {*R51*} as *R35* is implicitly contained). Therefore, only 2 bits of encoding are needed for the punch signals in each of the Y directions.

(5) Punch signals from/to neighbors

Figure 5 depicts the resulting punch signals and their widths between neighboring routers. Each cycle, up to four sets of punch signals can arrive from neighbors in the two dimensions with values reflecting the targeted router(s) that need to be controlled. Along with additional targeted routers generated from the local router, the power-gating controller sends newly generated punch signals to at most the four neighbors. As multiple targeted routers can be merged and communicated through a punch signal in one cycle, no contention delay is incurred, and targeted routers can always receive the notification as scheduled and wake up in time. In addition, this mechanism requires only 5 bits of encoding for X directions and 2 bits for Y directions in the case of 3-hop wakeup signal slack. Therefore, the hardware cost of punch signals and the power-gating control logic is very low, particularly compared to the main datapath and control path in routers that operate at the size of flits with 128 bits or 256 bits. It can be shown that, for the case of 4-hop wakeup sig-

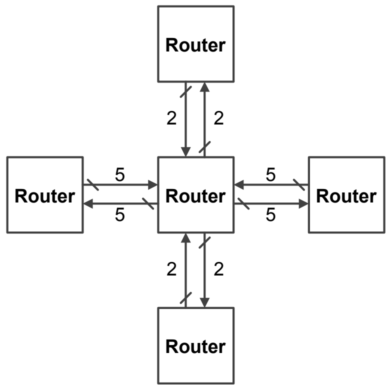


Figure 5: Power Punch signals.

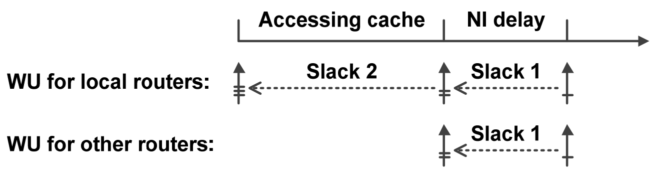


Figure 6: Exploiting slack at injection nodes.

nal slack, the width of punch signals is 8-bit for the X directions and 2-bit for the Y directions, which is still relatively small. Sending wakeup signals with 5 hops or more would be counter-productive as the wakeup latency is not that high (~20 cycles) and more powered-off cycles would be wasted.

4.2 Injection Node Punch

Power Punch has an additional mechanism to address the blocking issue at routers due to there not being enough hop count slack at the injection node to fully cover the wakeup latency of the local router. Our proposed mechanism holistically exploits two potential sources of existing slack at injection nodes to send punch signals before packets are generated.

Figure 6 shows the timeline for generating and sending a packet at an injection node. Normally, after accessing local resources (e.g., cache, directory) and generating a message, several operations are performed in the network interface. This includes encapsulating the message into packets and flits, arbitrating among multiple ready VCs (as only one VC from all virtual networks can send a flit through the physical link in each cycle), checking the availability of the connected input port of the local router, and passing the packet to the VC buffer of the input port. If the local router is found to be powered off when checking the availability, a wakeup signal is sent to the power-gating control of the router. Also, additional wakeup signals (in the form of punch signals as described before) are sent to other non-local routers one or more hops away based on packet destinations.

As can be seen, packets at the NI need to wait for the entire wakeup latency before being injected into the router input port. However, there is slack between the time that the destination is known (as part of the message passed to the NI) and the time that the availability of the router is checked that can be exploited. Therefore, both types of wakeup signals can be sent at the beginning of NI delay instead of at the end of NI delay, as shown in Figure 6 “slack 1”. With this slack, several cycles equivalent to the NI latency (usually 3 or 4 cycles) can be hidden from T_{wakeup}.

The above “slack 1” extends only up to the beginning of the NI as the destinations may not be known earlier. For example, sharers of a cache line for an invalidation coherence message cannot be known without accessing the cache directory on the home node. Therefore, wakeup signals to non-local routers, which rely on destination information to determine the targeted routers, cannot be generated earlier. In contrast, for the local router, as long as there is a packet that needs to be sent from this node, the local router will always be used, even though the destination is not known. This is represented as “slack 2” in Figure 6. Consequently, it is possible to send the wakeup signal for the local router at the beginning of accessing L2 cache or directory when it is known for sure that a packet will be generated and the local router will be used. This hides several additional cycles from T_{wakeup} (e.g., hiding ~6 cycles if accessing L2). One limitation, however, is that it might not always be possible to send the wakeup signal before accessing L1 as not all accesses result in non-local packets. A straightforward solution to this simply is not to exploit “slack 2” for L1 cache accesses. A valid bit is added for each type of resource to signify whether that resource type uses this slack (i.e., “1” for L2 cache and directory, and “0” for L1 cache). Note that “slack 2” is relatively long, so it is very effective in hiding wakeup latency when this slack is used.

4.3 Putting It All Together: Power Punch Impact

To summarize, Power Punch enables punch signals to utilize existing slack at source nodes to wake up powered-off local and neighboring routers along the first few hops even before packets are injected. It also enables punch signals to utilize hop count slack by propagating ahead of packets to “punch through” any blocked routers along the imminent path of packets, waking them up along the way. The contention-free signal propagation mechanism guarantees on-time delivery of wakeup signals and the on-time wakeup of routers, essentially hiding the wakeup latency from packets and not requiring packet detours. In addition, with multi-hop wakeup signals, a router knows exactly

Table 2: Key parameters for simulation.

Network topology	4x4, 8x8, 16x16 mesh
Input buffer depth	3-flit for data VC, 1-flit for control VC
Link bandwidth	128 bits/cycle
Router	3-stage and 4-stage
Private I/D L1$	32KB, 2-way, LRU, 1-cycle latency
Shared L2 per bank	256KB, 16-way, LRU, 6-cycle latency
Cache block size	64Bytes
Virtual channel	2 VCs/VN, 3 VNs
Coherence protocol	Two-level MESI
Memory controllers	4, located one at each corner
Memory latency	128 cycles

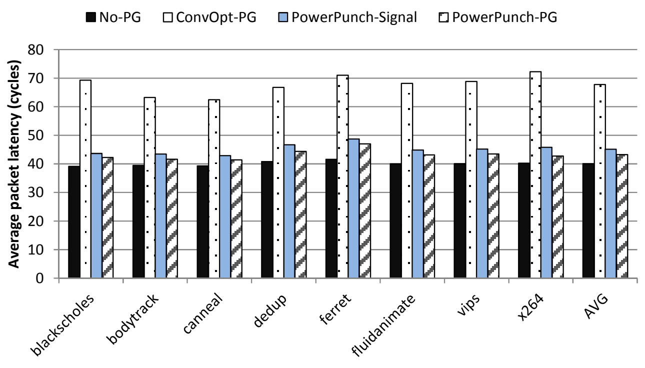


Figure 7: Average packet latency.

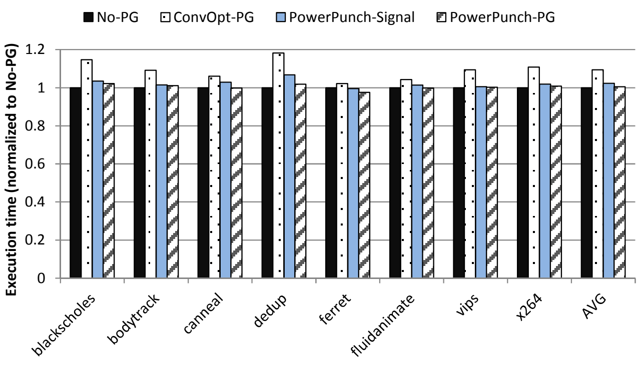


Figure 8: Execution time.

whether there is any incoming packet in next few cycles, thus avoiding power-gating short idle periods. This is superior to timeout techniques as no false filtering can happen and the filtering length is considerably longer. These features allow Power Punch to have significant static energy savings with minimal performance penalty, as shown in the following evaluation.

5. Evaluation Methodology

Power Punch is evaluated under full system simulation with the combined use of architecture-level and circuit-level simulators. The cycle-accurate gem5 [4] simulator enhanced with GARNET [2] is used for detailed timing simulation of the processor, memory and on-chip network. We also integrate the latest DSENT [29] NoC power tool with gem5 and GARNET to obtain runtime network activity statistics and estimate router power consumption more accurately. Significant effort has been made to implement various power-gating functionalities in the previous simulation settings. Besides regular power-gating components, we also modify the simulators to model all the key additional hardware in Power Punch, such as punch signals in all directions, extra logic in the power-gating controller for punch signal relay and handshaking, wakeup signals in network interface and cache controllers for holistically exploiting slack, and so on. The cache architecture assumes a 32KB I/D private L1 cache and 16MB shared L2 cache. The coherence protocol uses two-level MESI implemented with 3 logically separated virtual networks to avoid message-dependent deadlock. An 8x8 mesh network is used for most of the simulations while both 4x4 and 16x16 meshes are used for scalability analysis. All the NI operations are packed compactly in three cycles, although other loosely packed NI designs with longer latency would give Power Punch larger advantage due to increased slack. Table 2 lists other key configuration parameters.

Router wakeup latency is estimated using a standard VLSI design flow with 45nm technology. Parasitic extraction is performed on a 451um by 451um layout, and the extracted data is fed into a SPICE RC model. The wakeup latency is estimated to be 8 cycles. Additional sensitivity studies on various wakeup latencies are also conducted to demonstrate the applicability of Power Punch over a practical range of alternatives. The break-even time is 10 cycles and the timeout is 4 cycles, consistent with prior works [6, 7, 9].

We compare the following four schemes: (1) No-PG: baseline design with no power-gating; (2) ConvOpt-PG: conventional power-gating optimized with timeout and sending of wakeup signals early (these techniques partially hide the wakeup latency and avoid powering off short idle periods); (3) PowerPunch-Signal: proposed scheme with multi-hop punch signal only (no use of NI slack); (4) PowerPunch-PG: comprehensive scheme with multi-hop and NI punch signals.

6. Results and Analysis

6.1 Effect on Performance

We first evaluate one of the primary objectives of Power Punch for reducing the performance penalty of power-gating. Figure 7 compares the average packet latency, and Figure 8 shows the execution time of the four schemes for the multi-threaded PARSEC benchmarks [3]. Results are normalized to the No-PG scheme which generally provides a lower bound for average packet latency and execution time. As can be seen from Figure 7, even with the timeout and early-wakeup optimizations, ConvOpt-PG still increases the average packet latency substantially, by 69.1% on average compared with No-PG. This large penalty mainly comes from the fact that powered-off routers in conventional power-gating essentially block the path of packets and traditional optimization techniques are far from being able to cover the wakeup latency. In contrast, PowerPunch-Signal uses multi-hop punch signals to wake up the needed routers in advance, completely hiding the wakeup latency when there are enough hops. Consequently, PowerPunch-Signal reduces average packet latency significantly, with only 12.6% increase on average. By exploiting slack to compensate for the cases where there are not enough hops, the PowerPunch-PG achieves an additional 4.7% reduction, resulting in an average of only 7.9% increase in packet latency compared to No-PG. This amounts to a remarkable 61.2% improvement compared to ConvOpt-PG.

Similar trends are also observed in execution time. As shown in Figure 8, while the degree of reduction in execution time may vary among benchmarks due to their different sensitivities to network latency, Power Punch always has

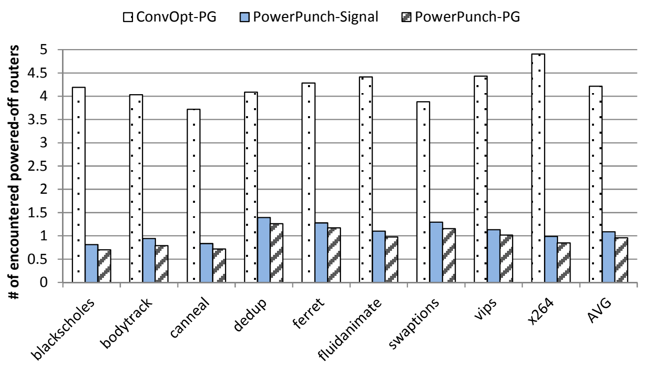


Figure 9: Number of encountered powered-off routers.

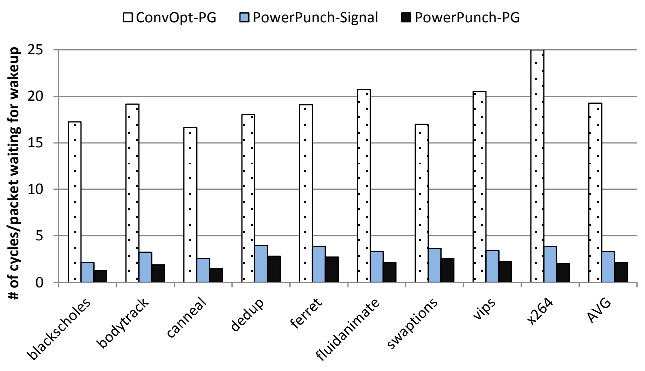


Figure 10: Number of cycles waiting for router wakeup.

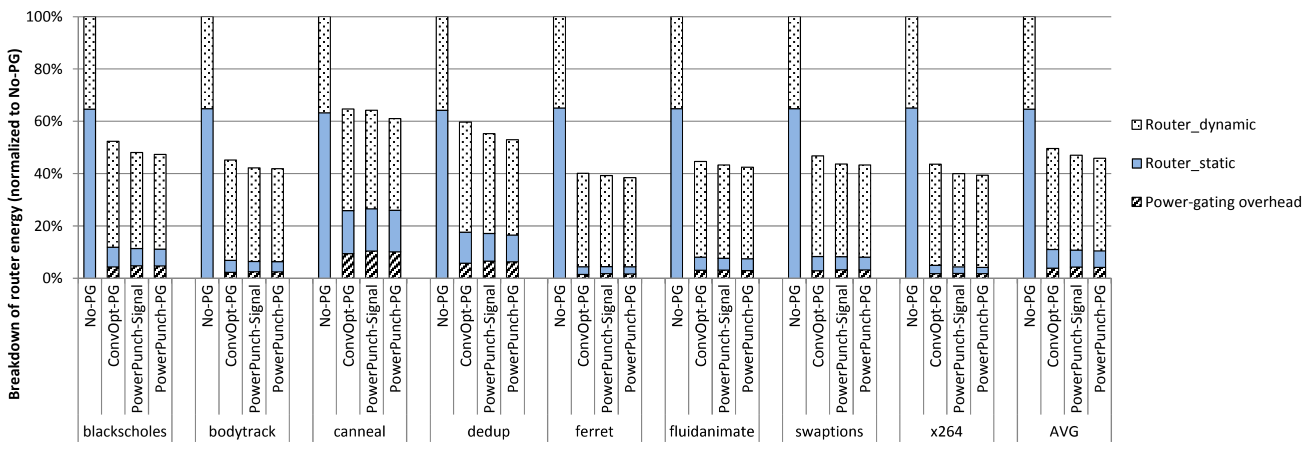


Figure 11: Breakdown of router energy.

the lowest performance penalty (for ferret, it is not clear why PowerPunch-PG actually has a slight decrease in execution time compared to No-PG, but likely causes are changes in thread criticality and synchronization traffic from altered packet timing). On average, PowerPunch-Signal and PowerPunch-PG have only 2.3% and 0.4% increase in execution time, respectively, essentially achieving non-blocking power-gating.

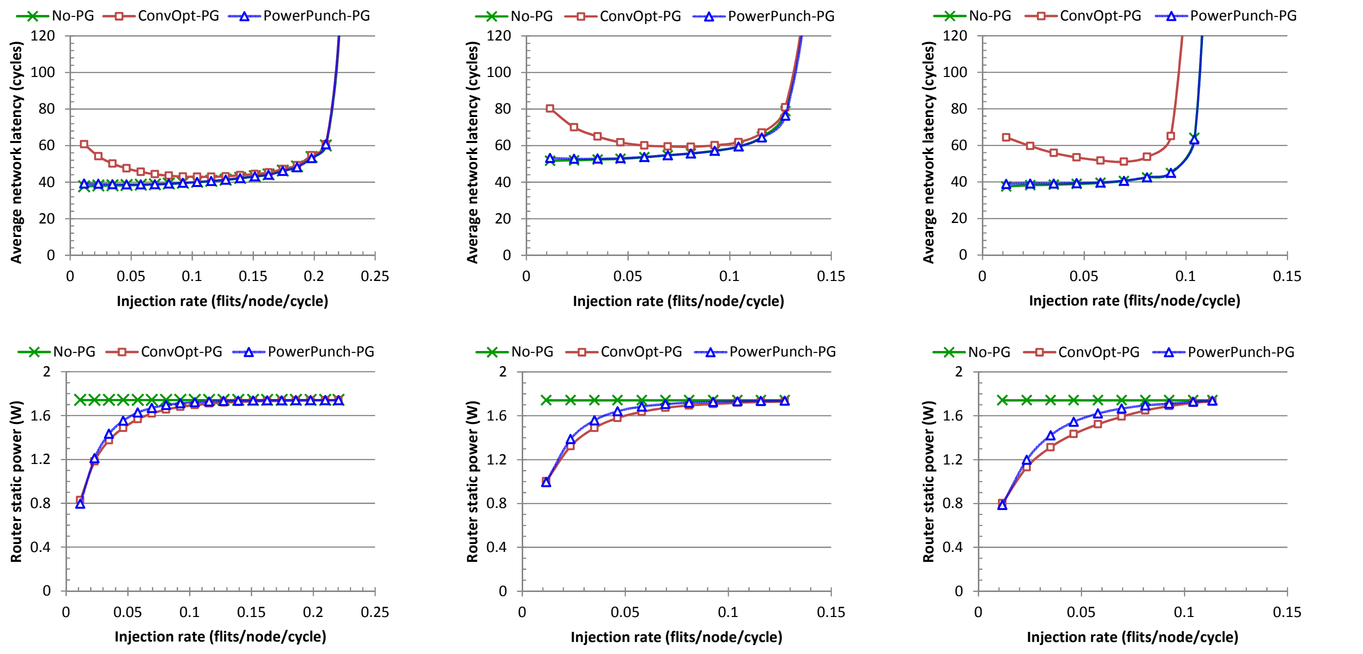
6.2 Effect on Reducing Blocking

To gain more insight on the effectiveness of Power Punch in mitigating blocking due to power-gating of routers, Figure 9 compares the average number of powered-off routers (i.e., blocked routers) that a packet encounters when transported from source to destination. As can be seen, the average number of blocked routers is dramatically reduced from 4.21 in ConvOpt-PG to 1.09 in PowerPunch-Signal; PowerPunch-PG further reduces that number to 0.96 due to the use of slack at the injection node (11.8% improvement over PowerPunch-Signal). However, this metric cannot fully reveal the advantage of exploiting NI slack since a blocked router is always counted as one even if the majority of its wakeup latency is hidden by the slack. To reflect this difference, Figure 10 plots the actual number of cycles that packets spend in waiting for routers to become fully awoken. While both PowerPunch-Signal and PowerPunch-PG significantly decrease the number of waiting cycles compared with ConvOpt-PG, the improvement of PowerPunch-PG over PowerPunch-Signal is actually 36.2%, revealing the true advantage of exploiting NI slack. Figures 9 and 10 illustrate the substantial reduction in performance penalty evidenced in the previous subsection and clearly demonstrate the impact of Power Punch.

6.3 Effect on Router Energy

The performance advantage of Power Punch does not come at a sacrifice in energy savings. Figure 11 shows the breakdown of router energy across the benchmarks, normalized to No-PG. The router energy is decomposed into dynamic energy, static energy and power-gating energy overhead. The power-gating overhead includes all the energy wasted owing to power-gating and its control, such as the energy consumed in powering on/off routers, in distributing sleep signals, and in generating and propagating punch signals.

We first compare router static energy. For fair comparison with No-PG, the power-gating overhead is added to the router static energy as the total router static energy for the three power-gating schemes (i.e., the total height of the bottom two bars) to reflect the net static energy savings. On average, the three power-gating schemes have similar static



(a) Uniform random (b) Bit-complement (c) Transpose

Figure 12: Packet latency and router static power across full range of network loads.

energy savings, with an improvement of around 83% (relative to the static energy in No-PG).

As for the total router energy, compared with No-PG, ConvOpt-PG saves 50.3%, with PowerPunch-Signal saving 52.9% and PowerPunch-PG saving 54.1%. Hence, Power Punch has slightly more energy savings. This is mainly due to two reasons: 1) Power Punch provides better filtering for short idle periods, and 2) Power Punch has a shorter execution time, which leads to less total router energy. Considering Figure 7 to Figure 10 together, it can be seen that, compared with the optimized conventional power-gating (ConvOpt-PG), Power Punch is better in both performance and energy.

6.4 Comparison across Full Network Load Range

To understand the behavior of different schemes more fully, we conduct simulations with synthetic traffic and vary the network load from zero to saturation. Figure 12 presents the performance and power results for three common traffic patterns: uniform random, transpose and bit-complement. Statistics are collected after sufficiently long NoC warm up.

Regarding performance, the typical "power-gating curve" is observed for ConvOpt-PG. That is, for low loads, the average packet latency is very high as many routers are powered off and packets are likely to be blocked multiple times. As load increases, the packet latency starts to decrease as more routers are powered on, and then rises again as load approaches saturation. This creates a massive performance gap compared with No-PG. In contrast, the average packet latency of PowerPunch-PG is almost identical to that of No-PG across the entire load range, essentially achieving non-block power-gating. Note that at high load for the transpose traffic pattern, due to the uneven load distribution among routers, it is possible that some routers are still powered off while other routers are congested. ConvOpt-PG performs poorly in this case, whereas PowerPunch-PG is able to reach the same maximum throughput as the no-power-gating case. This highlights the advantage of Power Punch in supporting high traffic load phases of application execution.

With regard to static power savings, both ConvOpt-PG and PowerPunch-PG save considerable static power as expected. ConvOpt-PG is slightly better for some medium loads but is achieved at the cost of significant performance penalty.

6.5 Sensitivity on Wakeup Latency and Router Pipeline

To illustrate that Power Punch is applicable to a variety of designs, Figure 13 compares the average packet latency of No-PG, ConvOpt-PG and PowerPunch-PG with varying values of wakeup latency and router pipeline stages. In this sensitivity study, uniform random traffic is used with load rate set to the average load rate of PARSEC benchmarks. A 3-hop punch signal is used in Power Punch. As can be seen, compared with No-PG, ConvOpt-PG has a large penalty of average packet latency in all cases, varying from 1.5X to more than 2X. In contrast, PowerPunch-PG has only 2.4% to 9.2% increase in the average packet latency. The highest increase of 9.2% occurs in the case of $T_{wakeup} = 10$ and $T_{router} = 3$, where the 3-hop punch signal does not cover the entire wakeup latency. We intentionally include this case to show that most of the penalty reduction of Power Punch indeed comes from hiding wakeup latency; otherwise performance penalty may occur. For this case, the performance penalty of

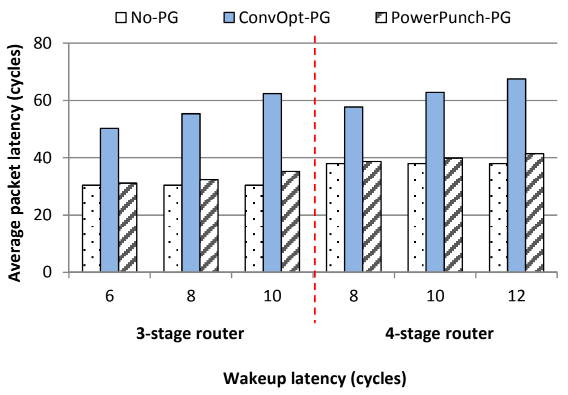


Figure 13: Wakeup latency sensitivity.

Power Punch becomes negligible when a 4-hop punch signal is used.

6.6 Discussion

(1) Hardware implementation and cost

While the prior description in Section 4 on how and why Power Punch works is detailed, the final implementation is actually very simple, without the need of any table or complex hardware. In the example of Figure 5, each bit in the 5-bit punch signal in the right X+ direction is a direct combinational logic function of the 5-bit punch signal from the left (no need to monitor Y directions due to routing restrictions). Similarly, each bit in the 2-bit Y+ direction punch signal is also a direct and simple logic function of the punch signals in the X and Y- directions. The overall area overhead of these logic gates, plus the 5-bit or 2-bit signal lines and other minor control logic, consumes only 2.4% of additional NoC area as compared to conventional power-gating.

(2) Scalability

Power Punch provides very good scalability and is suitable for larger network sizes. Power Punch does not have any particular central element that limits its scalability. The key parameter – the width of the punch signals – depends on the number of targeted router hops, not network size. This is unlike reconfiguration approaches that have complexity which depends greatly on the size of the network. In addition, conventional power-gating suffers from cumulative wakeup latency, which increases linearly with network size. In contrast, Power Punch does not have any of these issues, thus achieves relatively higher improvement for larger networks. For example, at an injection rate of 0.01 flits/node/cycle, compared with ConvOpt-PG, PowerPunch-PG reduces the average packet latency by 43.4%, 54.9% and 69.1% for 4x4, 8x8, and 16x16 networks, respectively.

(3) Comparison to other recent power-gating schemes

Previous work has proposed to send wakeup signals early using information already provided in look-ahead routing [24]. However, wakeup signals in this method can be sent at most 2 hops ahead, and sending beyond 2 hops requires monitoring a large number of dedicated signals. This paper follows a similar intuition, but proposes schemes that solve the critical problems of minimizing and merging wakeup signals with minimal hardware cost and no contention. This enables wakeup signals to be sent multiple hops ahead to hide router wakeup latency completely. In [5], NR-Mesh is proposed to improve NoC topology for power-gating by connecting a core to multiple routers. This scheme does not hide any wakeup latency at the injection if all the connected routers are powered off, and latency penalty due to detours is significant. It also requires high-radix routers. NoRD [6] is a recently proposed scheme that uses bypass paths to circumvent powered-off routers. It falls under the category of fast reconfiguration-based schemes as mentioned in Section 2.3. As NoRD relies on packet detours, its performance overhead is about 5 times that of Power Punch (9.3 cycles of packet latency penalty in NoRD versus 1.8 cycles in Power Punch for the 64-node system). NoRD also requires extra VCs for deadlock avoidance whereas Power Punch works with any number of VCs. Router Parking [28] and Panthre [27] are two other reconfiguration-based NoC power saving schemes with reasonably efficient algorithms. However, these schemes similarly suffer from issues associated with reconfiguration such as detour, long epoch and limitations on core-to-core communication. Compared with these two schemes, Power Punch has more router static energy savings capability and less packet latency penalty. MP3 [7] is a recently proposed NoC power-gating scheme that is very effective at achieving near-zero performance penalty. However, it is applicable to Clos and other indirect networks, not meshes (a direct network) targeted in this work. Another recent power-gating scheme, Catnap [9], uses multiple narrow networks to increase the efficiency of power-gating, but it is proposed mainly for CMPs with high-bandwidth.

7. Related Work

A number of closely related works on power-gating of NoC routers have already been discussed in detail [6, 7, 9, 24, 25, 26, 27, 28]. In addition, topology-aware power-gating has also been proposed recently [31] that specifically targets Flattened bufferfly [19] and MECS [14] networks. Besides on-chip network routers, power-gating techniques have been successfully applied to cores and execution units in CMPs and GPGPUs [18, 21, 23]. These applications of power-gating highlight the potential of this approach to save static power. The notion of NoC slack has been used before [8] in the context of the criticality of packet delivery to an application's execution, which is quite different from the slack concept exploited in this paper. Another related approach is bufferless routing [12, 15] which saves router power by eliminating buffers, but it may introduce potential livelock, misrouting and packet reassembly issues that must be handled appropriately. Moreover, besides buffers, there are other key components in NoC routers that also consume a considerable amount of static power which is reduced with Power Punch.

8. Conclusion

Current and future many-core chips require on-chip networks to be designed with both low power and high performance. While conventional power-gating of on-chip routers incurs significant performance penalty due to the blocking problems, this paper investigates the challenges and viability of achieving non-blocking power-gating. Power Punch, a novel and effective power-gating scheme, is proposed in this work. Power Punch exploits the slack in hop count as well the slack at source nodes to send power control signals ahead of packets to "punch through" any blocked routers along the imminent path of packets, turning them on. With Power Punch, packets do not suffer router wakeup latency or detour latency. Simulation results verify that significant router static energy savings with little performance penalty can be had, demonstrating the viability of achieving near non-blocking power-gating.

Acknowledgement

We sincerely thank the anonymous reviewers for their helpful comments and suggestions. This research was supported, in part, by the National Science Foundation (NSF), grant CCF-1321131 and the Software and Hardware Foundations program of the NSF.

References

[1] Adapteva. http://www.adapteva.com/epiphanyiv/, 2014.

[2] N. Agarwal, T. Krishna, L.-S. Peh, and N. K. Jha, "GARNET: A detailed on-chip network model inside a full-system simulator," in *International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 33-42, 2009.

[3] C. Bienia and K. Li, "Parsec 2.0: A new benchmark suite for chip-multiprocessors," in *5th Annual Workshop on Modeling, Benchmarking and Simulation*, 2009.

[4] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basil, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 Simulator," *Computer Architecture News,* vol. 39, pp. 1-7, 2011.

[5] J. Camacho, J. Flich, J. Duato, H. Eberle, and W. Olesinski, "Towards an efficient NoC topology through multiple injection ports," in *14th Euromicro Conference on Digital System Design: Architectures, Methods and Tools*, pp. 165-172, 2011.

[6] L. Chen and T. M. Pinkston, "NoRD: Node-Router Decoupling for Effective Power-gating of On-Chip Routers," in *45th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 270-281, 2012.

[7] L. Chen, L. Zhao, R. Wang, and T. M. Pinkston, "MP3: Minimizing Performance Penalty for Power-gating of Clos Network-on-Chip," in *20th IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2014.

[8] R. Das, O. Mutlu, T. Moscibroda, and C. R. Das, "Aergia: Exploiting packet latency slack in on-chip networks," in *37th International Symposium on Computer Architecture (ISCA)*, pp. 106-116, 2010.

[9] R. Das, S. Narayanasamy, S. K. Satpathy, and R. G. Dreslinski, "Catnap: Energy Proportional Multiple Network-on-Chip," in *40th International Symposium on Computer Architecture (ISCA)*, 2013.

[10] B. K. Daya, C. H. O. Chen, S. Subramanian, K. Woo-Cheol, P. Sunghyun, T. Krishna, J. Holt, A. P. Chandrakasan, and P. Li-Shiuan, "SCORPIO: a 36-core research chip demonstrating snoopy coherence on a scalable mesh NoC with in-network ordering," in *International Symposium on Computer Architecture (ISCA)*, 2014.

[11] J. Duato, O. Lysne, R. Pang, and T. M. Pinkston, "Part I: A theory for deadlock-free dynamic network reconfiguration," *IEEE Transactions on Parallel and Distributed Systems (TPDS),* vol. 16, pp. 412-427, 2005.

[12] C. Fallin, C. Craik, and O. Mutlu, "CHIPPER: A low-complexity bufferless deflection router," in *International Symposium on High Performance Computer Architecture (HPCA)*, pp. 144-55, 2011.

[13] P. Gratz, K. Changkyu, K. Sankaralingam, H. Hanson, P. Shivakumar, S. W. Keckler, and D. Burger, "On-chip interconnection networks of the TRIPS chip," *IEEE Micro,* vol. 27, pp. 41-50, 2007.

[14] B. Grot, J. Hestness, S. W. Keckler, and O. Mutlu, "Express cube topologies for on-chip interconnects," in *15th International Symposium on High Performance Computer Architecture (HPCA)*, pp. 163-74, 2009.

[15] M. Hayenga, N. E. Jerger, and M. Lipasti, "SCARAB: A single cycle adaptive routing and bufferless network," in *42nd IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2009.

[16] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, "A 5-GHz mesh interconnect for a Teraflops processor," *IEEE Micro,* vol. 27, pp. 51-61, 2007.

[17] J. Howard, *et al.*, "A 48-core IA-32 message-passing processor with DVFS in 45nm CMOS," in *IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 108-109, Feb. 2010.

[18] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson, and P. Bose, "Microarchitectural techniques for power gating of execution units," in *International Symposium on Lower Power Electronics and Design (ISLPED)*, pp. 32-37, 2004.

[19] J. Kim, J. Balfour, and W. J. Dally, "Flattened butterfly topology for on-chip networks," in *40th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 172-182, 2007.

[20] A. Kumar, L.-S. Peh, P. Kundu, and N. K. Jha, "Express virtual channels: Towards the ideal interconnection fabric," in *34th Annual International Symposium on Computer Architecture (ISCA)*, pp. 150-161, 2007.

[21] A. Lungu, P. Bose, A. Buyuktosunoglu, and D. J. Sorin, "Dynamic power gating with quality guarantees," in *International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 377-382, 2009.

[22] O. Lysne, T. M. Pinkston, and J. Duato, "Part II: A methodology for developing deadlock-free dynamic network reconfiguration processes," *IEEE Transactions on Parallel and Distributed Systems (TPDS),* vol. 16, pp. 428-443, 2005.

[23] N. Madan, A. Buyuktosunoglu, P. Bose, and M. Annavaram, "A case for guarded power gating for multi-core processors," in *17th International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 291-300, 2011.

[24] H. Matsutani, M. Koibuchi, W. Daihan, and H. Amano, "Run-time power gating of on-chip routers using look-ahead routing," in *13th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 55-60, 2008.

[25] H. Matsutani, M. Koibuchi, D. Wang, and H. Amano, "Adding slow-silent virtual channels for low-power on-chip networks," in *2nd International Symposium on Networks-on-Chip (NOCS)*, 2008.

[26] H. Matsutani, M. Koibuchi, D. Ikebuchi, K. Usami, H. Nakamura, and H. Amano, "Ultra fine-grained run-time power gating of on-chip routers for CMPs," in *4th International Symposium on Networks on Chip (NOCS)*, 2010.

[27] R. Parikh, R. Das, and V. Bertacco, "Power-aware NoCs through routing and topology reconfiguration," in *51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, p. 6, 2014.

[28] A. Samih, W. Ren, A. Krishna, C. Maciocco, C. Tai, and Y. Solihin, "Energy-efficient interconnect via router parking," in *IEEE 19th International Symposium on High Performance Computer Architecture (HPCA), 23-27 Feb. 2013*, pp. 508-19, 2013.

[29] C. Sun, C.-H. O. Chen, G. Kurian, L. Wei, J. Miller, A. Agarwal, L.-S. Peh, and V. Stojanovic, "DSENT - A tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling," in *6th IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, pp. 201-210, 2012.

[30] Tilera. http://www.tilera.com/products/processors, 2014.

[31] S. Yue, L. Chen, D. Zhu, T. M. Pinkston, and M. Pedram, "Smart butterfly: reducing static power dissipation of network-on-chip with core-state-awareness," in *IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 311-314, 2014.