Power-Aware NoCs through Routing and Topology Reconfiguration

Ritesh Parikh, Reetuparna Das and Valeria Bertacco

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

ABSTRACT

With the advent of multicore processors and system-on-chip designs, intra-chip communication demands have exacerbated, leading to a growing adoption of scalable networks-on-chip (NoCs) as the interconnect fabric. Today, conventional NoC designs may consume up to 30% of the entire chip's power budget, in large part due to leakage power. In this work, we address this issue by proposing Panthre: our solution deploys power-gating to provide long intervals of uninterrupted sleep to selected units. Packets that would normally use power-gated components are steered away via topology and routing reconfiguration, while Panthre provides low-latency alternate paths to their destinations. The routing reconfiguration operates in a distributed fashion and guarantees that deadlock-free routes are available at all times. At runtime, Panthre adapts to the application's communication patterns by updating its power-gating decisions. It employs a feedback-based distributed mechanism to control the amount of sleeping components and of packets detours, so that performance degradation is kept at a minimum. Our design is flexible, providing a mechanism that designers can use to tradeoff power savings with performance, based on application's requirements.

Our experiments on multi-programmed communication-light workloads from the SPEC CPU2006 suite show that Panthre reduces total network power consumption by 14.5% on average, with only a 1.8% degradation in performance, when all processor nodes are active. At times when 15-25% of the processor cores are communication-idle, Panthre enables leakage power savings of 36.9% on average, while still providing connected and deadlock-free routes for all other nodes.

Categories and Subject Descriptors

C.1.2 [**PROCESSOR ARCHITECTURES**]: Multiprocessors—*Interconnection Architectures*

General Terms

Design,Algorithms

Keywords

network-on-chip, power-gating, routing-reconfiguration

1. INTRODUCTION

Networks-on-chip (NoCs) have become increasingly widespread in recent years due to the extensive integration of many components in modern multicore processors and SoC designs. NoCs are scalable and flexible, however, they are crippled by excessive power consumption [6]. Particularly problematic for NoC structures is leakage power, which is dissipated regardless of communication activity or lack thereof. At high network utilization, static power may comprise more than 74% of the total NoC power at a 22nm technology node [16], and this figure is expected to increase in future technology generations. At low network utilization, leakage power is an even higher fraction of the total power budget for the NoC. With growing system integration, larger and larger portions of the NoC will be only lightly used at any point in time, with the lightly used set varying with each application and even within a single application over time.

As the NoC is a distributed and shared resource, conventional power-gating schemes [7] that opportunistically put components to sleep during periods of no activity are ineffective. The problem is two-fold: i) even when lightly utilized, NoC components often do not observe long idle-periods, failing even to compensate for the energy spent in the power-gating event itself, and ii) packets that encounter sleeping components in their paths accrue latencies due to wake-up delays. Power-gating at a finer granularity than entire routers [11] provides more sleeping opportunities. However, it further worsens the problem of accumulated wakeup latencies, as it puts components to sleep more aggressively. Early wakeup with lookahead routing was proposed to compensate for wakeup latency [10]. However, for a typical 2-stage pipeline router, lookahead can only hide a small fraction of the wakeup latency, which is typically many cycles. In our evaluations with multi-programmed workloads, we have identified that such conventional schemes often lead to significant application slowdown.

A workload stressing only a portion of the network creates opportunities for power-gating the remaining, lightly-used portions of the network. However, deterministic routing algorithms provide fixed routes among source-destination pairs, and in practice do not allow for isolation of any network component. A possible solution is to use an adaptive routing algorithm and deflect packets toward active units when they encounter a sleeping component on their regular path. However, this approach requires additional resources to maintain deadlock freedom, which must be kept active at all times. Moreover, accruing multiple deflections leads to increased packet latency.

Our solution, called Panthre (for Power-aware NoC through Routing and Topology Reconfiguration), overcomes these issues by modifying routing paths periodically so to *exclude* lightly used portions of the topology. When Panthre determines that the set of power-gated components must be updated, it executes route reconfiguration to avoid the new set of power-gated components, while providing deadlock-free routes for all packets. This step eliminates deflections and the need for dedicated resources to support deadlock-freedom. Panthre leverages the rich set of alternate paths that are available in NoC fabrics to keep traffic away from sleeping components. It also proactively adapts to application demands by power-gating only those network components that are under-utilized. Based on our analysis, Panthre may disable multiple router resources, adding up to 99% of the router's static power (Section 3.1).

Naturally, Panthre leads to an increase in traffic on the links kept active, by channeling traffic away from sleeping components. It is therefore essential for Panthre that substantial low-usage links exist in the NoC. To this end, we conducted a study, whose findings are plotted in Figure 1. The plot shows the contribution of network links to total network activity. Our testbed consisted of an 8x8 mesh CMP running a network-light multiprogrammed mix of applications from the SPEC CPU2006 suite. Links are sorted by increasing utilization during the execution, and the plot on the right indicates what fraction of network traffic (Y axis) was carried out by a given fraction of sorted links. The plot on the left is an enlargement of the contribution by the bottom 30% of active links: only 10% of the traffic travels through the bottom 30% of used links. Beyond the 30 percentile of utilization, this disparity is no longer obvious, thus, Panthre's goal is to identify and leverage the 30% least used links, so to maximize power savings without a significant load increase on active links.

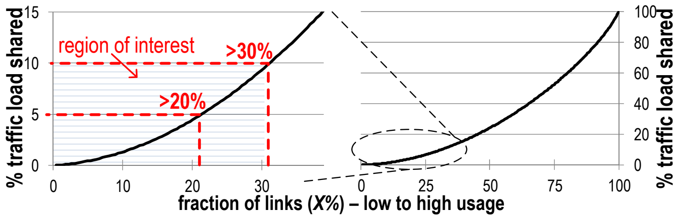
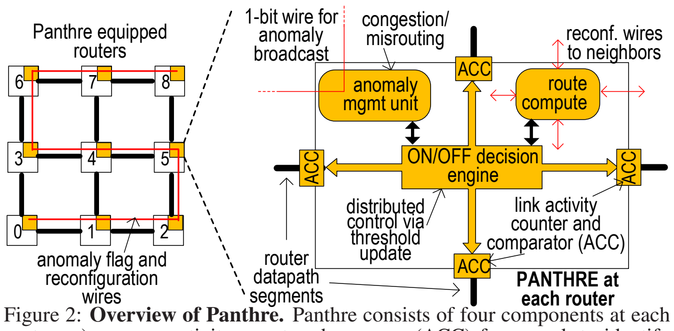


Figure 1: **Fraction of traffic load shared by the least utilized links.** The fraction of traffic transferred by the bottom 30% of used links is small, thus Panthre targets this pool for power-gating without burdening other links.



Panthre equipped routers
1-bit wire for anomaly broadcast
congestion/ misrouting
reconf. wires to neighbors
anomaly mgmt unit
route compute
ACC
ON/OFF decision engine
distributed control via threshold update
link activity counter and comparator (ACC)
anomaly flag and reconfiguration wires
router datapath segments
PANTHRE at each router

Figure 2: **Overview of Panthre.** Panthre consists of four components at each router: a) a usage activity count and compare (ACC) framework to identify lightly used components, b) an ON/OFF decision engine that determines the set of components to be power-gated, c) a Panthre-enabled route computation unit that can execute in the background without interrupting regular NoC operation, and d) an anomaly-based feedback algorithm that tracks application's needs dynamically and send updates to the ON/OFF engine.

Panthre deploys a simple and distributed framework for activity collection and subsequent exclusion of low-usage components. Even though frequent decisions to power-gate components are made locally at each router, Panthre's novel reconfiguration solution ensures uninterrupted full connectivity and deadlock-freedom of the NoC topology globally at runtime. Panthre, by construction, is also free of reconfiguration-induced routing deadlock [9], thus eliminating the need for costly deadlock-recovery protocols [2, 14]. Finally, it can reconfigure frequently without ever interrupting normal network operation.

In our evaluation with multiprogrammed benchmarks running on a 8x8 mesh network, Panthre was able to reduce NoC power consumption by 14.5% on average for communication-light workloads, while causing less than a 2% application slowdown. In contrast, power-gating with lookahead [10, 11], leads to 9-11% application slowdown if implemented at a router level, while causing as much as 20% performance loss if fine-grained power-gating is applied. Finally, Panthre leakage power savings can be as much as 36.9% on average, under the fair assumption that 10-16 nodes of a 64-node CMP are communication-idle at any given time.

2. RELATED WORK

A number of power-gating schemes have been been proposed for NoC components, operating at different levels of granularity: routers [10], ports [11], VCs [12], buffers [8]. However, all such schemes suffer from accumulated wakeup times and excess energy spent in each power-gating event. NoRD [2] elongates periods of router sleep by steering light, sleep-interrupting traffic to a low-power ring network that is always kept active. However, NoRD requires additional VC resources for deadlock-freedom. In addition, NoRD requires substantial dedicated hardware design and verification effort. Router Parking [14] proposes to completely switch off routers associated with idle cores by leveraging route-reconfiguration at a central node. Such a process requires dedicated channels to communicate with the central entity and typically takes a long time, often requiring to suspend network operation. As a result, such centralized schemes provide little adaptivity and can only be applied at a coarse-granularity (entire routers), and only when communication patterns are known well in advance. Catnap [4] proposes the use of multiple lightweight networks in CMPs, while applying power-gating at the network granularity to keep the performance overhead low. However, Catnap is only applicable to CMPs with high-bandwidth requirements, and unlike Panthre, it cannot be utilized for fine-grained power-gating.

Finally, route-reconfiguration has been well studied in the literature in the context of fault-tolerant routing. However, faults are rare occurrences, and therefore the majority of the algorithms designed for fault tolerance are centralized and take significant time and hardware resources. A notable exception is Ariadne [1], which describes a very quick (4K cycles for a 64 node network) and lightweight route reconfiguration scheme (<2% area overhead). Ariadne, however, also requires suspending the NoC operation during reconfiguration and can lead to reconfiguration-induced deadlocks. In our context, since we frequently reconfigure the NoC routing, network suspension would be detrimental. In contrast, our route reconfiguration algorithm, although inspired by Ariadne, provides deadlock-freedom throughout the reconfiguration, with no interruption of the mainstream NoC activity.

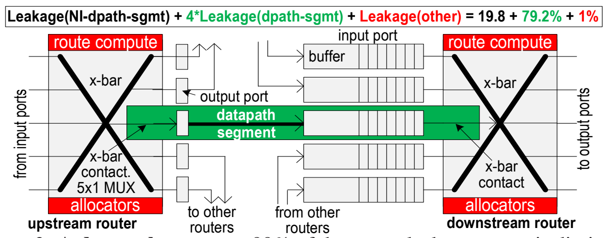


Figure 3: **A datapath segment.** 99% of the router leakage power is dissipated in the 5 datapath segments of the baseline mesh router. Results from DSENT at 22nm node.

3. PANTHRE DESIGN

Panthre consists of four components at each router: i) a component usage collection framework (ACC), ii) a lightweight ON/OFF decision engine that determines the set of links to power-gate based on local usage data, iii) a route compute module that updates the routing tables using broadcasts after each decision event, without interrupting normal NoC operation, and iv) a feedback-based anomaly-detection and management unit that communicates updates to the ON/OFF decision engine so that Panthre can adapt dynamically to changing communication patterns in the application over time. The four components of Panthre are highlighted in Figure 2. These four components are implemented using fast and lightweight distributed hardware, with minimal information communicated globally via a few single-bit wires. The lightweight distributed hardware allows Panthre to adjust to application communication needs very quickly and without ever interrupting normal network operation.

3.1 Fine-Grained Power Gating

Panthre provides fine-grained power-gating by allowing components to be excluded at the granularity of a single unidirectional link. Upon careful examination of routers' datapath, we identified that powering-down a unidirectional link between two routers is equivalent to powering down the corresponding crossbar contact and the output port at the upstream router, the unidirectional link itself, and the input port, input buffer and crossbar contacts at the downstream router. We call this combined set of components, a *datapath segment*: it represents the smallest granularity at which routing-based reconfiguration can be applied for the purpose of power-gating. The concept of a datapath segment is illustrated in Figure 3. Fortunately, crossbar, links and buffers consume most of the leakage power in a router, and they can all be powered off by our approach. DSENT [16] reports that 99% of the leakage power consumption of our baseline mesh router synthesized at 22nm can be attributed to its 5 datapath segments. The remaining 1% is attributed to shared units such as route computation and allocators. Note that Panthre is unable to exclude the local datapath segment (the one connecting to the local core) in absence of alternate paths for them to connect to the NoC. Therefore, in the rest of this paper, we exclude these local datapath segments from our computations, and assume that orthogonal power-saving schemes are being deployed for them.

3.2 Execution Flow

Panthre operates in epoch-based execution, with power-gating decisions made at the beginning of each epoch for the entire epoch. Thus, Panthre can ensure that power-gated components can be off for at least one epoch-long interval. The distributed activity counter units (ACC, Figure 2) periodically collect datapath segment usage statistics by means of simple counters: the data is then used to guide the decision process for the next epoch. Panthre's decision process is simple: all datapath-segments that experience activity below a threshold (A_{TH}) are put to sleep. Thereafter, a route update process updates routing tables to operate the network in the new configuration. Note, however, that different communication loads require different A_{TH} for Panthre to be effective. A fixed value could lead to an excessive or insufficient number of power-gated segments. Therefore, we propose a threshold update algorithm that leverages feedback from the distributed *anomaly detection units*. When the number of segments power-gated is excessive, two types of anomalies may arise: i) a large fraction of packets suffer long detours to their destinations, and ii) congestion due to the increased load on active links.

Thus, we detect these anomalies locally at each router and broadcast them globally using single-bit wires. Each ON/OFF decision engine is equipped with logic to update the threshold value based on

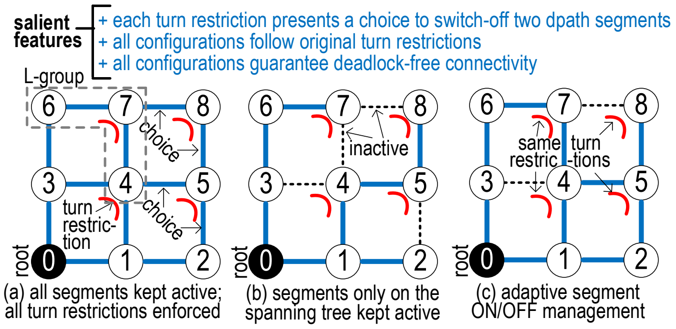

salient features
+ each turn restriction presents a choice to switch-off two dpath segments
+ all configurations follow original turn restrictions
+ all configurations guarantee deadlock-free connectivity
L-group
choice
choice
turn restric-tion
root
inactive
same turn restric-tions
(a) all segments kept active; all turn restrictions enforced
(b) segments only on the spanning tree kept active
(c) adaptive segment ON/OFF management

Figure 4: **Panthre's reconfiguration algorithm** allows power-gating decisions to be made independently at each router, without causing disconnection or deadlock. a) Breadth-first construction of the up*/down* spanning tree and corresponding turn restrictions. Each turn restriction node (L-group), presents a power-gating choice between datapath-segments. b) A minimally-connected network configuration degenerates into a spanning tree. c) A dynamically-adapted NoC where low-usage datapath-segments are power-gated.

this information. Note that the global anomaly broadcast ensures that A_{TH} values are kept consistent throughout the NoC. This aspect, in turn, guarantees that power-gating decisions are fair, tackling the least used segments in the NoC. In addition, Panthre naturally provides the ability to systematically trade-off performance for power savings by adjusting criterion for the detection of these anomalies. With Panthre, stable and power-efficient configurations are typically attained 10-15 epochs (1 epoch is 10K cycles in our design) after a program phase change, which is quick considering that application phases are up to 10s of millions of cycles.

3.3 Reconfiguration Algorithm

A hallmark of Panthre is that all power-gating decisions can be made independently at each router, while deadlock-freedom and connectivity among all nodes is still guaranteed throughout execution. This allows for frequent reconfigurations, in the order of one reconfiguration event per tens of thousands of cycles. In addition, Panthre eliminates the need of any additional hardware to recover from pathological scenarios such as deadlock. This is a great advantage in terms of silicon cost (and power), and it also limits the impact of reconfiguration on performance. Panthre's reconfiguration is based on the up*/down* routing algorithm, which breaks deadlocks by forbidding certain through-router connections between non-local links ('turns'). Up*/down* routing works by organizing all network nodes on a spanning tree, starting from a root node of choice. Each node receives a unique order based on its distance from the root, equidistant nodes are ordered arbitrarily. Thereafter, all routes involving going first away from the root node (down-traversal) and then towards it (up-traversal) are marked invalid. This ensures deadlock-freedom, as all dependency cycles involve at least one 'down$\rightarrow$up turn'(down-traversal followed by up-traversal). A breadth-first construction of the spanning tree rooted at node 0 is shown in Figure 4a. As an example, 1$\rightarrow$4 is a down-traversal, while 4$\rightarrow$3 is an up-traversal. Therefore, turn 1$\rightarrow$4$\rightarrow$3 must be marked invalid.

Note that a spanning tree, by definition, connects all the nodes in the graph and it is acyclic. In this context, Panthre's route-reconfiguration algorithm leverages the fact that turn restrictions are placed between two links when only one of them can be part of the spanning tree. Because of this, it is possible to power-gate one of the two datapath segments connected to a disabled turn and still maintain full network connectivity. Figure 5 illustrates the property just outlined: the left portion of the figure shows a spanning tree construction, such that nodes 0,1 and 3 are already on the spanning tree rooted at node R, while node 2 is being added. It can be noted that either link 0-2 or link 3-2 are sufficient to connect to node 2. Since both are available, there will be a turn restriction 0-2-3. A similar situation is shown on the right side of the figure, where links 1-3 and 2-3 are sufficient to reach the to-be-added node 3, and the turn restriction is 1-2-3. The middle part of the figure shows a more general case, where both node 2 and 3 are being added to the spanning tree, using links 0-2 and 1-3, respectively. In this case, either the turn 0-2-3 or 1-3-2 must be disabled to break the cycle. Depending on the turn restriction placement, this situation degenerates into the one shown on the left or the right.

In order to organize Panthre's reconfiguration process, we call any two links connected by a disabled turn an *L-group*, as shown in Figure 4a. Therefore, a decision can be taken locally at each L-group, to power-gate one of the two bi-directional datapath links, while the network would still be connected globally. Note that each bi-directional links comprises two opposite unidirectional datapath segments (see Section 3.1). In the extreme case when all L-groups decide to power-down two datapath segments each, the topology will degenerate into a spanning tree, as shown in Figure 4b. Panthre leverages a distributed and adaptive datapath-segment ON/OFF decision engine that determines the power-gating decisions locally at each L-group, according to application communication characteristics. An example network configuration produced using Panthre is shown in Figure 4c.

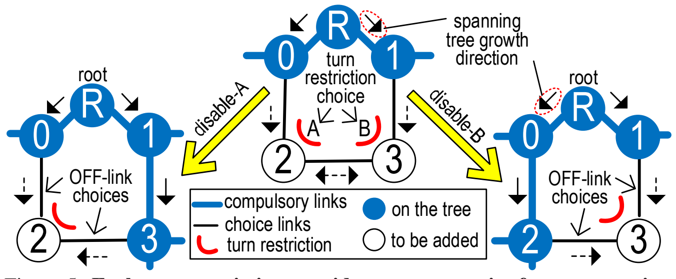


Figure 5: **Each turn restriction provides an opportunity for power-gating one bidirectional link.** Panthre leverages this property to put low-utilization links to sleep. Note how the property holds for any topology.

Even though many reconfiguration algorithms [1, 14] replace deadlock free routing paths with another set of deadlock-free routing paths after reconfiguration, packets in-transit following the old routing paths can cause deadlocks by interacting with packets following the new routing paths. This is because paths valid in the old routing function might be disabled in the new routing function, or vice-versa. To circumvent this issue, Panthre ensures that any newly developed routing configuration complies with the turn-restrictions that were determined for an all-powered-ON NoC configuration (e.g., Figure 4a), eliminating the possibility of reconfiguration-induced deadlocks. Since Panthre only disables a link if it is part of a turn-restriction (L-group), the corresponding turn would not be exercised, whether the corresponding link is enabled or disabled. Intuitively, if all L-groups maximally power-gate, the network topology will degenerate into a spanning tree (Figure 4b), and none of the restricted turns would be exercised. Note that Panthre's reconfiguration, though transparent and deadlock-free, may lead to reordering of packets between a source-destination pair. For systems where point-to-point ordering is essential, such as certain cache coherence protocols, we suggest the use of tag matching and reordering of packets at network interfaces (*e.g.*, Tilera [17]).

Table 1 summarizes Panthre's leakage power saving potential when applied to different topologies. The table reports the total number of non-local datapath-segments in each topology and the number of datapath-segments required to construct the spanning tree (#span-seg). During periods of low activity, all the non-spanning datapath-segments could potentially be power-gated without sacrificing connectivity. We observe that Panthre has a great potential for reducing power consumption in popular topologies, such as meshes and tori, up to a 51% static power saving in a 8x8 torus.

topology	#seg	#span-seg	%off	topology	#seg	#span-seg	%off
mesh-4x4	48	30	38	mesh-8x8	224	126	44
torus-4x4	64	30	53	torus-8x8	256	126	51

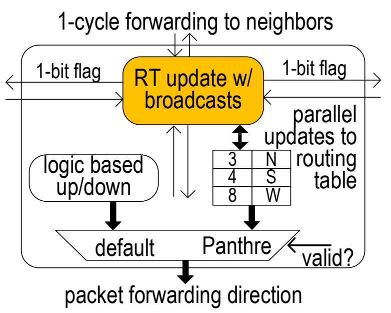
Table 1: **Panthre's leakage power saving potential** for various topologies.

3.4 Panthre Implementation

In this section we discuss the detailed functionality and the hardware requirements of each of Panthre's four components as shown in Figure 2. We then overview the application-adaptive algorithm.

Activity Counters and Comparator (ACC). An ACC unit is associated with each datapath-segment that belongs to an L-group. The activity counters are 10-bit counters, incremented upon each flit traversing the corresponding datapath-segment. In addition, a 6-bit comparator (compares high order bits only) is required to compare against the A_{TH} value provided by the ON/OFF decision engine to determine the power-gating status of the segment in the next epoch. An 8x8 mesh has 49 L-groups, with 4 datapath-segments each.

We monitor usage activity on an epoch-to-epoch basis. The smaller the epoch size, the more frequently Panthre initiates reconfiguration


1-cycle forwarding to neighbors
1-bit flag
RT update w/ broadcasts
1-bit flag
parallel updates to routing table
3 N
4 S
8 W
logic based up/down
default
Panthre
valid?
packet forwarding direction

RT update [1] features:	
hardware	: 1-bit wire/port
actions	: flag broadcast
	flag forwarding
	RT update
one brcast	: 64 cycles
#brcasts	: 64
reconf.	: 4096 cycles

Figure 6: **Panthre's route computation unit** consists of an up-to-date routing table reflecting the power-gating status of the NoC, and a default logic-based routing unit to provide a backup route, if the routing table is unavailable due to ongoing reconfiguration.

to quickly adapt to application characteristics. However, the lower-bound on epoch size is constrained by the latency of our: i) reconfiguration (∼4K cycles), and ii) statistics collection (few thousand cycles for capturing patterns). We determined that an epoch size of 10K cycles provides a good tradeoff between reconfiguration overhead and the amount of time Panthre takes to adapt to changing application characteristics.We further determined that power-gating a datapath segment that is used for more than 2^{10} cycles within an interval of 10K cycles, is always detrimental to performance. Therefore, our activity counters are only 10-bits wide, and all datapath-segments with even higher activity are always kept active. In addition, the A_{TH} values are incremented or decremented in quanta of at least 2^4, and thus comparing the 6 high order bits of the counters is sufficient.

The **ON/OFF Decision Engine** is deployed for each L-group and maintains and updates the A_{TH} value. It interfaces with the anomaly management unit to decide when the A_{TH} value should be incremented or decremented. To this end, a 6-bit adder/subtracter circuit is required at each L-group. In addition, the ON/OFF decision engine instructs the route computation unit to initiate a route-update once new power-gating decisions are made. For simplicity of implementation, decision engines associated with all L-groups operate in a synchronized manner. This is achieved by simply ensuring that the A_{TH} updates, the ON/OFF decisions and the route-updates are all applied only at epoch ends. After a datapath-segment has been power-gated, a few packets may still require to go through old routes to make forward progress. In this situation, the datapath-segment behaves as a conventional power-gating state machine: waking up on packet arrival and sleeping again upon its departure. Note that this scenario is extremely rare, and does not offset the benefits of Panthre: our experiments incorporate the delays and power costs due to these situations.

The **Panthre-enabled Route Computation Unit** is shown in Figure 6. It consists of two sub-components: i) a logic-based distributed routing (LBDR) implementation [5] that provides routes corresponding to an all-segments-ON configuration, and ii) a routing table that stores the most up-to-date routes, reflecting the power-gating status of the network. Having a backup LBDR implementation has three advantages: i) upon detection of an anomaly it allows the network to instantly switch to an all-segments-ON mode and limit performance impact, ii) it allows the routing table to be updated bit-by-bit in the background by providing a default path if no valid option yet exists in the table, and iii) it can be implemented cheaply. LBDR is a critical unit for Panthre as it allows uninterrupted operation even during reconfiguration. Note that all dynamic NoC route configurations follow the minimal set of turn restrictions, and hence packets are never stalled in router buffers due to unavailability of valid routing paths.

To reduce the natural congestion around the root node that is typical of up*/down* routing, we select root nodes in low-congestion topology locations, and implement popular optimizations such as depth-first construction of the spanning tree and load-balanced path selection [15]. We are therefore able to extract at-par performance compared to XY routing, as is evident from the results in Section 5, where the baseline uses XY routing. Also note that Panthre disables all its functionalities at heavy loads, as at that operating condition its power savings are minimal anyway. This keeps Panthre free of any additional congestion or power dissipation at high loads.

We leverage a distributed route-update algorithm inspired by Ariadne [1] to update the routing table on each reconfiguration event. Ariadne utilizes time-synchronized broadcasts from all destination routers in turn, communicated to all routers through simple forwarding operations. Each broadcast takes 64 cycles in an 8x8 mesh, and the entire reconfiguration process is completed in ∼4K cycles. Figure 6 summarizes the features of Ariadne's route-update algorithm. If Ariadne-style functionality is already available in fault-tolerant NoCs, it can be leveraged by Panthre with only minor modifications.

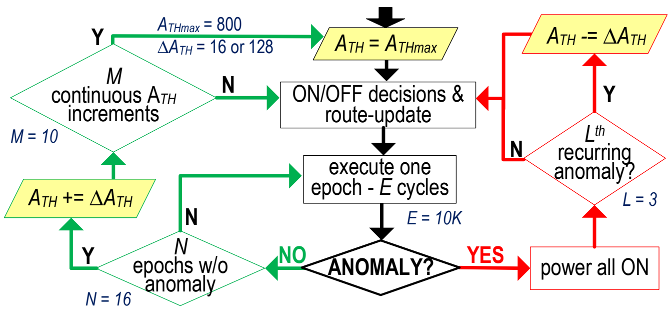

A_{THmax} = 800
ΔA_{TH} = 16 or 128
Y
$A_{TH} = A_{THmax}$
M continuous A_{TH} increments
M = 10
N
ON/OFF decisions & route-update
A_{TH} -= ΔA_{TH}
Y
N
L^{th} recurring anomaly?
L = 3
execute one epoch - E cycles
E = 10K
A_{TH} += ΔA_{TH}
N
Y
N epochs w/o anomaly
N = 16
NO
ANOMALY?
YES
power all ON

Figure 7: **Panthre's application-adaptive algorithm - flow chart.**

The **Anomaly Management Unit** monitors two types of adverse behaviors due to power-gated datapath segments: i) excessive detours, and ii) network congestion. For the purpose of detecting excessive detours, a special 'misroute' bit is reserved in the header flit of every packet. We set this bit if, at any router, a packet is routed through a port that takes it further away from the destination. For a mesh topology, this is as simple as calculating the relative X and Y coordinates of current and destination nodes: this information is already available in logic-based routing algorithms [5]. Each destination counts the fraction of packets that suffered a detour over those that did not. If the ratio is >1, the destination node will broadcast a misrouting flag on a single-bit wired-OR ring (Figure 2). We deploy one wired-OR ring for each of 4 8x2 regions in our 8x8 mesh. These 4 rings drive a separate wired-AND connection, and the root node is designated to monitor its anomaly status. If the wired-AND connection is set, the root node broadcasts an anomaly code on a 1-bit global wire that all routers can snoop. In other words, our detour detection scheme raises a flag if at least one node in each of the 4 regions observes more than 50% misroutes. We use a similar, but simpler scheme for congestion detection, inspired by the maximum buffer occupancy metric of [4]. If the total buffer occupancy, at any time and at any router in the NoC, is more than a certain threshold (29 in our implementation), the router noting the congestion broadcasts the anomaly code on the same 1-bit global wire used for reporting excessive detours.

Panthre's Application-adaptive Algorithm is shown in Figure 7. At the start of execution, A_{TH} is initialized to its maximum value, A_{THmax} (= 800 in our setup). All datapath segments with utilization above this value are never considered for power-gating. Among the others, the ones with activity below A_{TH} are switched-off. At the end of the execution epoch, the application-adaptive algorithm takes different actions based on whether or not an anomaly is flagged. If an anomaly is flagged (right part of the figure), suggesting that too many links are powered-off, then all components are instantly powered back on. This anomaly indicates that the current A_{TH} value is causing too aggressive power-gating. Therefore, if anomalies are detected in the last L consecutive epochs (L=3 in our design), A_{TH} is lowered. Power-gating decisions are then reassessed in light of the new A_{TH} value. Note that decreasing A_{TH} reduces the amount of switched-off segments and, in turn, decreases the likelihood of anomalies in the near future. In addition, the threshold values are lowered in two phases, with first a coarse-grain (ΔA_{TH} = 128 in our setup), and then a fine-grained tuning (ΔA_{TH} = 16).

If an entire epoch is executed without any anomaly being flagged (left portion of the figure), indicating that our current configuration is performance-friendly, the next epoch is executed without updating power-gating decisions. However, if no violations are observed in the last N consecutive epochs (N=16 in our design), suggesting that our power-gating selection is too conservative and there is room for greater power savings, A_{TH} is increased and ON/OFF decisions are redone. After M (=10) successive A_{TH} increments, we determine that the application load has considerably increased and the current A_{TH} value is far from optimal, thus we update A_{TH} to A_{THmax}, and re-execute the algorithm from the start. The state machine for our algorithm is

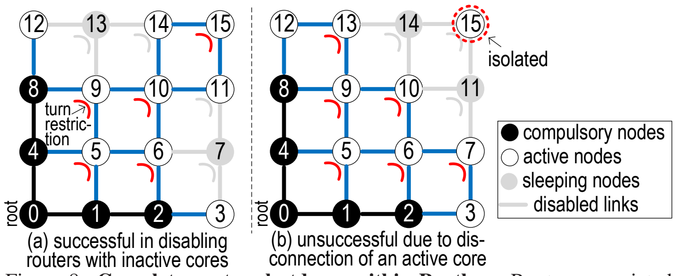


Figure 8: **Complete router shutdown within Panthre**. Routers associated with idle cores can be put to sleep while maintaining system connectivity, as long as they correspond to leaf nodes in the routing's spanning tree . Routers that can never be considered for shutdown are called 'compulsory'. a) Successful shutdown of routers 7 and 13. b) Routers 11 and 14 cannot be simultaneously shutdown because this would lead to isolating node 15.

very simple and can be implemented in hardware at low cost. It is replicated in each ON/OFF decision engine at every L-group, requiring a 6-bit adder/subtracter for A_{TH} updates, and small counters and comparators for the other parameters (*L,M,N*).

Implementation overhead. The unidirectional ring wire for anomaly broadcast, combined with wired-OR and wired-AND wires for detection of excess detours, leads to an wiring area overhead of only 0.39% [16], in comparison to the channels of our baseline router. We also assume that Ariadne-style route-updating functionality is already available for fault-tolerance. All other Panthre components are extremely lightweight, with small counters, comparators and adders added to each L-group. Therefore, compared to the deep buffers and many virtual channels of modern routers, Panthre-specific logic is trivial both in terms of power and area. Finally, the additional hardware is primarily for monitoring, and thus does not add to the critical path delay.

4. COMPLETE ROUTER SHUTDOWN

If some cores in a CMP system are idle, neither sending nor receiving traffic, the routers corresponding to these cores could be completely power-gated, as long as doing so does not isolate any active node. A complete router shutdown is equivalent to shutting down 8 datapath segments (4 incoming and 4 outgoing), and hence is a very lucrative power-saving option. Panthre's deadlock-free and connectivity-preserving reconfiguration algorithm can be easily extended to shut down entire routers, and still provide all its valuable properties.

A router can be considered a candidate for shutdown only if it is a leaf node in at least one of Panthre's spanning tree constructions. The intuition behind this observation is that a leaf node is connected to the rest of the tree only via a single link. In addition, that link is only used for transferring packets originating or destined for that leaf router. Therefore, if the leaf node is communication-idle, that single link can be switched-off without affecting the remaining topology. To provide an example, in Figure 8, the 'compulsory' routers that can never be completely shut down are shaded in black. In the setup of the Figure, the root is node 0 and the spanning tree was generated breadth-first. Note that the compulsory routers are set once a root node and a turn-restriction configuration have been selected. Note also that in an 8x8 mesh, only 13 out of 64 routers are compulsory. We equipped Panthre to distinguish between compulsory nodes and those that can be shut down, and to apply shutdowns whenever possible for communication-idle nodes. If all compulsory routers are kept active, and if connectivity is at all possible after shutting down all routers associated with sleeping cores, Panthre too is successfully able to provide connectivity and deadlock-freedom.

For applications that lead to 25% idle cores in the network, entire router shutdowns can be successfully applied to 48% of routing configurations. The integration of this scheme with Panthre is also simple: upon a group of cores notifying their idle state, the routers associated with them (if they are not compulsory for connectivity) are put to sleep. A route-update procedure is then executed and, if all active cores are still connected to the root core (this can be easily detected by analyzing the routing table), Panthre continues in router shutdown mode. If, however, connectivity is lost, Panthre defaults back to its baseline routing.

5. EXPERIMENTAL RESULTS

We evaluated Panthre on a cycle-level trace-driven multi-core simulator [3], modeling a 64-core CMP system as described in Table 2. We used a front-end functional simulator based on Pin [13] to collect instruction traces from applications, which are then transfered to the trace-driven cycle-level simulator. We also integrated a detailed on-chip network model, simulating a state-of-the-art two-stage router, described in Table 2b. In addition to the baseline design with no power-gating and the Panthre design, we also implemented router-level conventional power-gating with lookahead wakeup (*PG_conv*) [10] and a fine-grained port-level power-gating scheme (*PG_fg*) [11], for comparison. For both PG_conv and PG_fg, we used an idle-detection time of 4 cycles of inactivity [10]. The Panthre design parameters described in Section 3.4 (e.g., A_{THmax}), are calibrated after detailed design space exploration to provide a suitable trade-off between performance and power: Figure 7 indicates our balanced values.

We analyzed our framework with two types of workloads: synthetic uniform random traffic, as well as 35 applications from the SPEC CPU2006 and commercial ($sap, tpcw, sjbb, sjas$) benchmark suites. We experiment across 40 randomly generated multi-programmed workload mixes, with each mix containing 10 copies each of 6 applications randomly picked from our suite of 35 applications. For brevity of results, we categorize the 40 workload mixes into four categories of 10 workloads each, based on the amount of cache misses per kilo instructions (MPKI). Most network transactions originate because of misses in the caches, and hence MPKI correlates well with application communication load. The *light* category has benchmarks within MPKI of 200, while *light-med* spans the MPKI range 200-500. The *med* group includes relatively network heavy benchmarks, with MPKI between 500-1500, while the *heavy* category covers the MPKI range 1500-2500.

A considerable amount of energy is spent in putting components to sleep and bringing them back up. The amount of sleeping time required to compensate for this energy loss is called the 'breakeven time'. In our evaluation, we assume a breakeven time of 10 cycles and a wakeup delay of 4ns, in agreement with previous research [2, 10]. The effective sleeping time after accounting for breakeven energy, is called compensated sleep cycles (CSC) [10]. The CSC over total execution is a direct measure of leakage power savings. In our results, we report CSC values, in addition to latency increase and application slowdown. Finally, we use DSENT [16] to estimate total network power, accounting for both dynamic and static power at the 22nm technology node. For dynamic energy, DSENT is used to report energy spent per event (for e.g., buffer write), which is then tracked accurately in our cycle-level simulator.

5.1 Synthetic Traffic

We first compare Panthre with other power-gating schemes using synthetic random traffic. Random traffic is the worst case scenario for Panthre, as it distributes traffic uniformly across the network, reducing the number of low-usage links. Panthre's primary goal is to extract maximum power-savings by only turning-off low value datapath-segments, keeping latency degradation in check. Figure 9a plots the average packet latency for each of the solutions evaluated. It is clear from the figure that PG_conv and PG_fg both lead to a high latency increase (>2x at low load). This is due to the fact that, at low load, network components observe packet traversals infrequently, and spend most of their time sleeping. Therefore, packets accumulate wakeup latency at each hop. At this injection rate, both the conventional power-gating schemes spend more than 75% of the time asleep. However, this level of latency degradation leads to unacceptable (>10%) application slowdown, as we will note in Section 5.2. Note that the latency and CSC for PG_conv, both decrease with increasing network load. However, we observe that at the injection rate where the latency degradation becomes acceptable for PG_conv (0.16 flits/cycle/node),

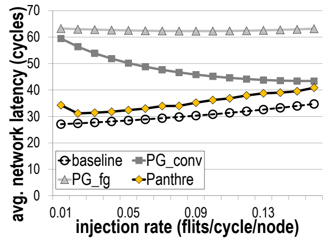
(a) Processor @2GHz

Cores	2-wide fetch/commit 64-entry ROB
coherence	4-hop MESI, 64B block
L1 cache	Private: 32KB/node ways:4 latency:2
L2 cache	Shared: 256KB/node ways:16 latency:6
Memory	Distributed: 1GB/bank banks:4 latency:160

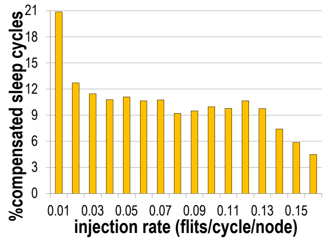
(b) Network @2GHz

Topology	8x8 mesh, 128 bit links
Pipeline	2-stage VC flow ctrl
VCs	4 VCs/port, 8 flits/VC
Routing	XY for baseline up*/down* for Panthre
Workload	synthetic: uniform multi-prog: SPEC CPU06
Simulation (cycles)	synthetic: 5M multi-prog: 10M

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



(a) **Average network latency**. Conventional power-gating schemes suffer up to 2x increase in latency and, hence, are detrimental to performance. In contrast, Panthre keeps latency degradation under check.



(b) **Compensated sleep cycles** under Panthre. Panthre's leakage power savings decrease with increasing load, the average ranges between 9.8% and 20.8% for injection rates varying from 0.01 to 0.16.

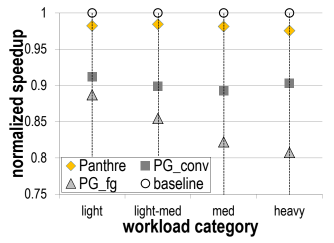
Figure 9: **Network latency and CSC under uniform random traffic.**

the CSC value drops down to only 2.7%.

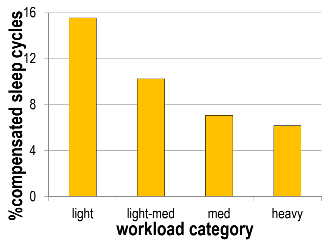
In contrast, Panthre's latency degradation is only 16.5% on average for the range of injection rates shown in the graph. At an injection rate beyond 0.16 flits/cycle/node, Panthre leads to negligible latency degradation, as it tries to keep all links active at such high load. Naturally, the power-savings are also little at that point, as almost all components are considered vital and not switched off. At low traffic loads, however, Panthre can lead to more than 20% leakage power savings (CSC), as can be noted from Figure 9b. With increasing load, CSC values decrease, but Panthre still save 10.3% leakage power on average for the range shown in the graph.

5.2 Multiprogrammed Workloads

Panthre is designed to save leakage power only when it is possible to do so without degrading performance. Figure 10a shows the speedup values for all power-gating schemes, normalized to the baseline CMP with no power-gating capability. The average slowdown with Panthre is only 1.9% across all four benchmark categories. In contrast, PG_conv and PG_fg lead to 9.8% and 15.7% slowdown averaged over all benchmark categories, respectively. Slowing down the application by such an amount is detrimental to system energy consumption, and therefore these conventional power-gating schemes cannot be deployed in modern CMPs. In Figure 10b, we show that Panthre saves 15.6% leakage power for communication-light workloads on average, while 9.8% leakage power is saved on average across all benchmark categories. The total network power is reduced by 14.5%, 9.3%, 6.1% and 5.1% for light, light-med, med and heavy workloads, respectively. With substantial power savings at very little performance degradation, Panthre provides a good design trade-off for power-aware systems. Additionally, if the designers are willing to sacrifice performance, the Panthre algorithm can be easily tuned for aggressive power-gating.



(a) **Normalized application speedup**. Panthre limits the performance degradation to within 2% in most cases, while conventional power-gating causes more than 10% application slowdown on average.

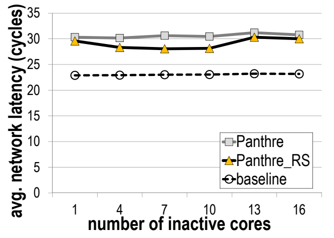


(b) **Compensated sleep cycles (CSC)** saved by Panthre. For light workloads, the average leakage power savings are 15.6%, while Panthre saves less for heavy workloads (6.7%).

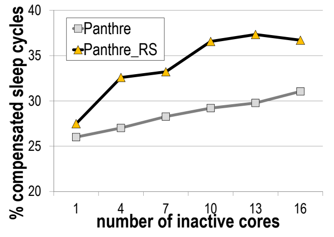
Figure 10: **Network latency and CSC with multi-programmed workloads.**

5.3 Complete Router Shutdown

As discussed in Section 4, Panthre can shutdown completely, a significant number of routers associated with idle cores, and still ensure connectivity and deadlock-freedom among the active cores. This property is useful in scenarios where certain cores are communication-idle for substantial periods of time. The gains are reflected in the power-savings with complete-router-shutdown (*Panthre_RS*) under low injection rate (0.01 flits/cycle/node) and uniform traffic, as shown in Figure 11b. It can be noted that 36.9% of leakage power can be saved on average for 10-16 idle cores. Note that, if Panthre_RS is unsuccessful at maintaining connectivity for active cores after complete router shutdown, it reverts to its baseline approach of isolating only datapath



(a) The **average network latency** increase in both Panthre and Panthre_RS is kept under check. Panthre_RS delivers a lower latency compared to basic Panthre.



(b) **Percentage compensated sleep cycles (CSC)** for Panthre_RS is 37% for 10-16 idle cores. Panthre_RS improves power savings further over Panthre.

Figure 11: **Panthre's latency and CSC when complete router shutdown is enabled** at a low injection rate and for uniform random traffic.

segments. As shown in Figure 11a, Panthre_RS keeps latency increase under 30% on average, providing both better power saving and latency profile than our baseline Panthre solution.

6. CONCLUSION

Panthre maximizes leakage power savings by guaranteeing long periods of uninterrupted power-gating for NoC components. It leverages topology and routing reconfiguration to steer traffic away from sleeping components to minimize the latency impact. Panthre's application-adaptive reconfiguration algorithm is implemented in a lightweight distributed manner, and guarantees a globally-connected and deadlock-free network at all times. In addition, Panthre monitors events indicating network performance degradation and updates its power-gating decisions to provide a more suitable power-performance trade-off. Our experiments with light multi-programmed workloads show that Panthre reduces total network power by 14.5% on average, with only a 1.8% degradation in performance. Panthre's network power savings can be as much as 36.9% on average if 10-16 nodes are idle in a 64-node CMP.

Acknowledgements

This work was partially supported by NSF grant #0746425 and C-FAR, within STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

7. REFERENCES

[1] K. Aisopos, A. DeOrio, L.-S. Peh, and V. Bertacco. ARIADNE: Agnostic reconfiguration in a disconnected network environment. In *Proc. PACT*, 2011.

[2] L. Chen and T. M. Pinkston. NoRD: node-router decoupling for effective power-gating of on-chip routers. In *Proc. MICRO*, 2012.

[3] R. Das, O. Mutlu, T. Moscibroda, and C. Das. Application-aware prioritization mechanisms for on-chip networks. In *Proc. MICRO*, 2009.

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

[5] J. Flich and J. Duato. Logic-based distributed routing for NoCs. *Computer Architecture Letters*, 7(1), 2008.

[6] J. Howard et al. A 48-core ia-32 message-passing processor with dvfs in 45nm cmos. In *Proc. ISSCC*, 2010.

[7] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson, and P. Bose. Microarchitectural techniques for power gating of execution units. In *Proc. ISLPED*, 2004.

[8] G. Kim, J. Kim, and S. Yoo. Flexibuffer: reducing leakage power in on-chip network routers. In *Proc. DAC*, 2011.

[9] O. Lysne, J. M. Montañana, J. Flich, J. Duato, T. M. Pinkston, and T. Skeie. An efficient and deadlock-free network reconfiguration protocol. *IEEE Trans. Computers*, 57(6), 2008.

[10] H. Matsutani, M. Koibuchi, H. Amano, and D. Wang. Run-time power gating of on-chip routers using look-ahead routing. In *Proc. ASPDAC*, 2008.

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

[12] H. Matsutani, M. Koibuchi, D. Wang, and H. Amano. Adding slow-silent virtual channels for low-power on-chip networks. In *Proc. NoCs*, 2008.

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

[14] A. Samih, R. Wang, A. Krishna, C. Maciocco, C. Tai, and Y. Solihin. Energy-efficient interconnect via router parking. In *Proc. HPCA*, 2013.

[15] J. Sancho, A. Robles, and J. Duato. An effective methodology to improve the performance of the up*/down* routing algorithm. *IEEE Trans. Parallel and Distributed Systems*, 15(8), 2004.

[16] C. Sun, C.-H. 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 *Proc. NoCs*, 2012.

[17] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C.-C. Miao, J. Brown, and A. Agarwal. On-chip interconnection architecture of the tile processor. *Micro, IEEE*, 27(5), 2007.