BackTrack: Oblivious Routing to Reduce the Idle Power Consumption of Sparsely Utilized On-Chip Networks

Yao Wang and G. Edward Suh Computer Systems Laboratory
Cornell University
Ithaca, NY 14853, USA
{yao, suh}@csl.cornell.edu

Abstract

Technology scaling indicates that future microprocessors will be power limited. Since microprocessors will be power limited and only a portion of the silicon can be turned on, recent studies have proposed specialized and adaptive cores for future multi-core systems. However, the on-chip networks for these systems have been designed assuming that all nodes are simultaneously active. This paper investigates activating only a subset of routers in the network. Specifically, we propose an oblivious routing algorithm to maximize the number of unused routers which can then be turned off to save power. We show that this routing algorithm can reduce power usage with little to no performance degradation on the network.

1 Introduction

The semiconductor technology scaling trend indicates that the utilization of future microprocessors will likely be limited by the maximum power consumption. While the number of transistors still scales up exponentially, the energy consumption of each transistor is only decreasing slowly because the supply voltage cannot be scaled down as aggressively as before due to concerns for leakage current. As a result, a recent ITRS report [9] predicts that future microprocessors will only be able to actively use a small portion of its area at a given point of time. In essence, future microprocessors can contain a large number of processing cores in terms of its silicon area, but will only be able to use a small subset of them simultaneously.

Recent studies have investigated the implication of this "utilization wall" on the processing architecture and proposed heterogeneous or adaptive architectures [22, 9]. However, the on-chip interconnect network has still largely been designed assuming that all nodes are active simultaneously. This paper investigates the implication of having only a small number of active nodes on the on-chip network, and proposes an oblivious routing technique, named BackTrack, which is designed to reduce the idle power consumption of the sparsely utilized network with minimal to no impact on the network throughput and latency.

For sparsely utilized networks, the idle power consumption becomes a significant portion of the overall network power consumption because the network is lightly loaded on average. BackTrack tackles this problem by increasing the number of routers that can be completely turned off through intelligent oblivious routing that avoids using unnecessary routers. While fine-grained clock and power gating techniques are still possible without BackTrack, applying them to routers that can be used by active nodes may results in an increased delay or frequent power up and down. Instead, BackTrack reduces the number of routers that are used by active nodes so that they can simply be turned off without fine-grained clock and power management. Also, BackTrack focuses on oblivious routing techniques that can be implemented in simple hardware without run-time adaptation or knowledge of application traffic patterns.

The main intuition behind the BackTrack approach is to have flows in both directions between each pair of network nodes use the same set of routers. In today's routing schemes, these flows are often routed independently and use two different sets of routers. At a glance, BackTrack appears to reduce path diversity in routing and thus trade-off the network performance for the reduced idle power. However, an interesting observation is that the forward and backward flows between a pair of nodes do not share any network resources. As a result, sharing the same set of routers between these two flows does not necessarily incur more contention in network resources. In practice, we found that BackTrack routes perform as well as the baseline routing schemes that treat forward and backward flows independently, especially when network traffic is relatively low as in sparsely utilized networks.

This paper introduces how the general BackTrack approach can be applied to an oblivious routing scheme and studies two concrete instantiations based on an XY routing (BT-XY) and a randomized dimension order routing (BT-RDOR). Both routing schemes only require minimal hardware changes to the baseline router and can effectively reduce the number of active routers. We show that BT-XY is deadlock free while both RDOR and BT-RDOR can use a virtual channel allocation to avoid deadlocks. Simulation results shows that both BT-XY and BT-RDOR can significantly (up to 40%) reduce the number of active routers for sparsely utilized networks with minimal impact on performance. When about 20% of nodes are active for an 8-by-8 mesh network, BT-XY can reduce the number of active routers by about 20%, which results in about 10% total power savings. Also, BT-XY can match the performance of XY when a small number of nodes are active. BT-RDOR provides less power savings, but with a stronger performance guarantee.

This paper makes the following contributions. First, the paper points out future many-core platforms are likely to rely on sparsely utilized networks and discusses its implications. Second, the paper presents a simple yet novel routing approach to reduce the number of active routers with minimal overheads. Finally, the evaluation demonstrates that the BackTrack approach can significantly reduce the idle power consumption without sacrificing performance when the network utilization is low.

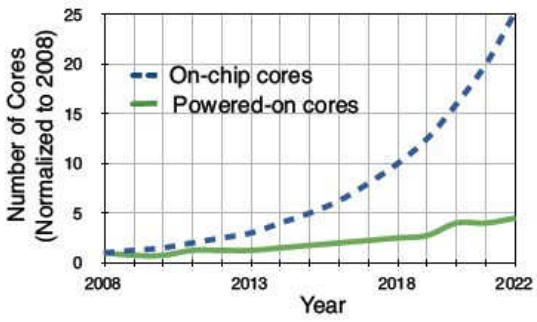


Figure 1: Prediction on the number of cores that can be simultaneously powered on. The graph is from [9].

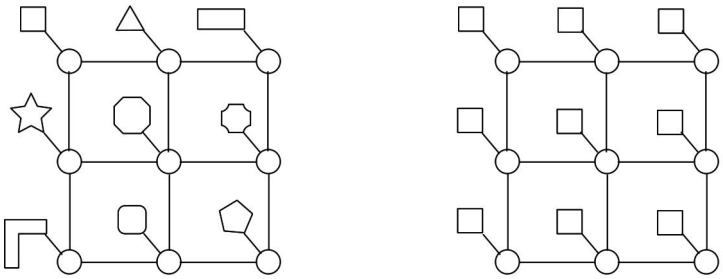
The rest of the paper is organized as follows. Section 2 provides a background context on the architectural implications of the technology scaling trend and discusses the significance of reducing idle power consumption on sparsely utilized networks. Section 3 introduces the proposed BackTrack approach in general as well as two concrete instantiations based on XY and RDOR, and discusses their properties such as deadlock freedom, network contention, and hardware implementations. Section 4 evaluates the BackTrack routing schemes through detailed simulations. Section 5 discusses related work and Section 6 concludes the paper.

2 Sparsely Utilized On-Chip Networks

2.1 Utilization Wall

Technology scaling enables more devices to be integrated on a silicon chip. However, unlike traditional scaling, the device power consumption no longer scales down enough to compensate for the increased number of transistors. The transistor threshold voltage (V_{th}) has stopped decreasing to control the leakage current at a reasonable level, which in turn keeps the supply voltage (V_{dd}) from scaling down. With an exponentially increasing number of transistors on the same size of chip and a non-scaling voltage, the chip power consumption increases rapidly and becomes one of the major limitations.

Figure 1 shows a recent study based on the 2008 ITRS report that estimated the number of cores that can be powered on simultaneously assuming Intel Core i7-like cores and a constant power budget of 100 Watts [9]. The study predicted that only a small portion of on-chip processing cores can be activated at the same time due to the limited power budget. This limitation of the number of devices that can be powered on simultaneously is often called the "utilization wall". This trend also suggests that on-chip networks should be optimized to minimize the power consumption exploiting the small number of active nodes.



(a) Heterogeneous Accelerators (b) Adaptive Cores

Figure 2: Architecture options for power-limited many-core platforms.

2.2 Power-Limited Many-Core Systems

The utilization wall indicates that future microprocessors cannot simply rely on increasing the number of cores to improve performance. Because the power consumption of each core cannot scale with the exponentially increasing number of cores, it is unlikely that all cores can be turned on at the same time. Using simple in-order cores may allow more cores to be powered on. However, simple cores only reduce the power consumption by a constant factor. Moreover, some applications may not have enough thread-level parallelism to utilize a large number of simple cores. To improve performance for a broad range of applications within a limited power budget, each core must become more energy efficient by more closely matching application characteristics.

In a high level, there are two major approaches to improve the energy efficiency of processing cores on a many-core platform. First, a processor may contain many types of heterogeneous cores, each of which is optimized for a different workload. Figure 2(a) illustrates such a massively heterogeneous many-core processor with a 2-D mesh network. The heterogeneous many-core platform can map an application to the most appropriate type of cores to improve performance as well as energy efficiency. An extreme example of such massively heterogeneous cores is the Conservation Core (C-Core) architecture [22] where an accelerator core is automatically generated from application code.

Another approach to improve the energy efficiency of processing cores is to rely on homogeneous, yet adaptive cores that can dynamically trade-off its single-thread performance and energy efficiency. The adaptivity can be achieved in many different ways. For example, a processing core can use dynamic voltage frequency scaling (DVFS), support dynamic reconfiguration of micro-architectural resources, or even contain multiple types of physical cores internally. Figure 2(b) shows such a homogeneous many-core.

In both types of architectures, the limited power budget in future processors implies that only a subset of processing cores that best match the characteristics of an application can be turned on. For on-chip networks, this trend indicates that only a portion of network nodes will be active at a time. For the massively

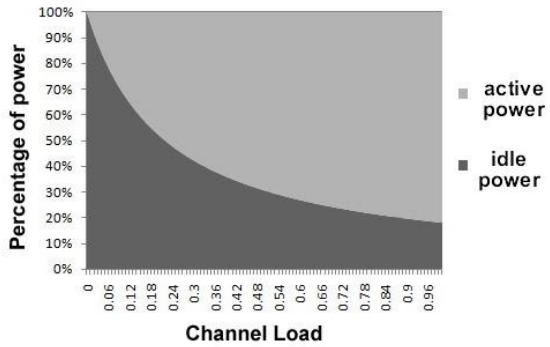


Figure 3: Idle and active power breakdowns for on-chip networks as a function of channel loads.

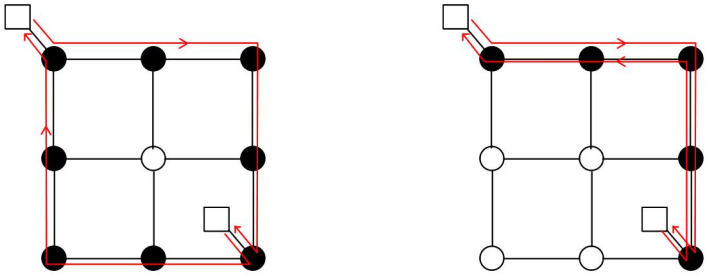
heterogeneous architecture, the active nodes will be determined by the physical locations of processing cores that match an active application characteristics. Given that there exist many diverse set of applications, from the network design perspective, we can consider that active nodes are randomly selected. The homogeneous architecture has more flexibility in choosing active nodes, but it is unclear what the best strategy would be. In this study, we assume active nodes are arbitrarily distributed, reflecting the heterogeneous architecture case.

2.3 Idle Power Consumption

Figure 3 shows the breakdown between the idle power and the active power consumption of a router as a function of channel load. Here, the idle power includes leakage and clock power whereas the active power includes the rest of dynamic power consumption. The power numbers are estimated using the Orion2 power models under 65nm technology [8]. The figure suggests that the idle power consumption can be a significant portion of the total power consumption, especially when the channel load is low.

Previous studies show that the typical traffic load on on-chip networks is relatively low [15, 5, 6, 17]. To ensure low latency communications, on-chip networks are often designed to operate well below their maximum capacity. Moreover, a computing node with a sequential program usually does not generate heavy network communications. Therefore, the idle power consumption has a significant contribution to the overall network power consumtion. Hiroki et al. [14] points out that the idle power is comparable to the active power when the network throughput is low, and notes that the idle power is likely to become more significant as the active power consumption decreases with the technology scaling. Banerjee et al. [1] also show that the idle power can dominate the active power consumption. For sparsely utilized networks, the idle power consumption is likely to be even more important because the network load will be even lower.

The main objective in this paper is to reduce the idle power consumption of on-chip networks by increasing the number of unused routers, which can be completely turned off. While fine-grained power management



(a) XY routing (b) BackTrack: BT-XY

Figure 4: Forward and backward routes for XY and BackTrack-XY on a 3x3 mesh network. The dark nodes indicate that they need to be active.

techniques are also available, we believe that increasing the number of unused routers is attractive because it enables simple power management without an impact on performance. For example, power gating is a common technique to reduce the power consumption. However, it suffers from a long wake-up time and wake-up energy. To minimize the performance degradation, the power gating needs to know the traffic ahead of time and start the wake-up operation in advance. Similarly, clock gating can reduce the clocking power without turning off the whole router. However, clock gating cannot reduce the leakage power, which is likely to become more important as technology scales, and still introduces delay and energy overheads. If a router is not used by an application, it can be simply turned off without worrying about delay and energy penalties for the duration of the application.

3 BackTrack Routing (BT)

3.1 Intuition

Traditional routing algorithms are designed with an assumption that all nodes are turned on and active. As a result, the number of active routers is not a major design concern, and often all routers and links are assumed to be available for routing. In this context, the traditional routing algorithms generally treat forward and backward flows between a pair of nodes, u to v and v to u, independently. Figure 4(a) illustrates how an XY dimension order routing algorithm routes flows between two nodes on a mesh network. As shown in the figure, the two flows in opposite directions between $(1, 1)$ and $(3, 3)$ use different sets of routers. Eight out of nine routers need to be active in order to support bi-directional communications between just two nodes.

The main intuition behind the BackTrack routing scheme is to use the same set of routers for communications between a pair of nodes in both directions. This approach reduces the number of active routers, especially when only a small number of nodes are active. Figure 4(b) shows how the BackTrack approach

can be applied to the XY routing (BT-XY). The left-to-right flow from $(1,1)$ to $(3,3)$ uses the traditional XY route. On the other hand, the right-to-left flow from $(3,3)$ to $(1,1)$ backtracks the corresponding flow in the opposite direction, effectively following a YX route. This simple approach reduces the number of active routers from eight to five.

At a glance, the BackTrack approach appears to create more network contention by having forward and backward routes use the same set of routers. However, a careful inspection shows that there is in fact no resource sharing between the corresponding forward and backward flows when a router uses bi-directional channels. There are separate links, buffers, and crossbars for flows in each direction. Therefore, the use of BackTrack does not necessarily degrade the network performance. In fact, we show that the performance of BackTrack in terms of channel loads will be comparable to the baseline routing scheme if the baseline scheme evenly distributes loads from all "forward" routes[1] across the network. Moreover, even when this condition is not satisfied, we show that the BackTrack routes perform as well as its baseline scheme in practice if a small number of nodes are active.

The goal of the BackTrack approach is to reduce the number of active routers in order to reduce the idle power consumption in the context of sparsely utilized networks. While there can be many different approaches to achieve this goal, we limit ourselves to oblivious routing techniques in this paper so that the approach does not significantly increase the complexity of routers or require additional information on applications. As a result, we do not consider adaptive approaches or application-aware optimizations.

In the rest of this section, we introduce more formal definitions of BackTrack and discuss two BackTrack routing schemes based on the XY routing (BT-XY) and a pseudo-random dimension order routing (BT-RDOR). In addition to introducing these schemes, we address the main challenges that are associated with applying the BackTrack approach: deadlock freedom and potential performance degradation due to restrictions on routing paths.

3.2 Routing Algorithm

In a high-level, BackTrack transforms a baseline routing algorithm so that communications between a pair of nodes use the same set of routers for both directions. Here, we discuss how the BackTrack routing technique can be applied to an oblivious routing scheme in general by first defining how a baseline routing scheme can be represented and then showing how the baseline scheme can be changed in BackTrack.

Definition 1 Network: *We represent a network as a graph $G(V,E)$ where vertices represent network nodes and edges represent channels. An edge (u,v) in E has capacity $c(u,v)$ that is the available bandwidth on the*

[1] We will provide a more formal definition later in this section.

BACKTRACK(G, K, F_{base}, $dir()$)
1. For a flow $K_i = (s_i, t_i, d_i) \in K$,
2. If $dir(s_i, t_i) = 1$ (forward flow), copy the path from F_{base} to F_{BT}: $f_{BT,(s_i,t_i)}(u, v) = f_{base,(s_i,t_i)}(u, v)$.
3. Otherwise (backward flow), backtrack the path from t_i to s_i in F_{base}: $f_{BT,(s_i,t_i)}(u, v) = f_{base,(t_i,s_i)}(v, u)$.
4. Repeat the Step 1 to 3 for all flows.

Figure 5: BackTrack oblivious routing framework.

corresponding channel.

Definition 2 Flow*: An application has a set of k data transfers or flows $K = \{K_1, K_2, \ldots, K_k\}$. $K_i = (s_i, t_i, d_i)$, where s_i and t_i are the source and sink, respectively, for flow i, and d_i is the bandwidth demand. We assume $s_i \neq t_i$ and $d_i > 0$.*

Definition 3 Route*: For a flow i, its route is a path p_i in the network graph $G(V, E)$ from its source s_i to the sink t_i. The route for each flow can be represented by flow variables $f_i(u, v)$ on edge (u, v); The edges along the routing path will have $f_i(u, v) = 1$, other edges will have $f_i(u, v) = 0$, assuming single-path routing. In the following discussion, we also use $f_{(s_i,t_i)}(u, v)$ to refer to flow variables for a flow from s_i to t_i. A routing scheme can be seen as a set F that includes flow variables for all possible source-sink pairs and edges in the network.*

Traditional routing schemes often determine paths for two flows in opposite directions between a pair of nodes independently. In other words, for a pair of nodes s and t, the path from s to t is independent from the path from t to s. BackTrack explicitly identifies the two flows between the same pair of nodes so that their routes can be determined together. For this purpose, we group flows and paths into two categories: *forward* and *backward*.

Definition 4 Forward and Backward Flows/Paths*: For each flow from s to t, a function $dir(s, t)$ determines its direction; the flow is a* forward *flow if $dir(s, t) = 1$ and a* backward *flow if $dir(s, t) = 0$. The direction function assigns one flow between a pair of nodes as a forward flow and the opposite flow as the backward flow: $dir(s, t) = 1 - dir(t, s)$.*

Based on the direction, the flow set K can be split into two subsets, K_f for forward flows and K_b for backward flows: $K_f = \{K_{f,1}, K_{f,2}, \ldots, K_{f,k_f}\}$, and $K_b = \{K_{b,1}, K_{b,2}, \ldots, K_{b,k_f}\}$.

Given the distinction between forward and backward flows, BackTrack generates a route by following a baseline oblivious routing scheme for a forward flow and assigning the opposite path along the same routers to the corresponding backward flow. More formally, the steps in Figure 5 show how new routes F_{BT} are obtained from baseline routes F_{base}.

The BackTrack method allows us to generate many routing schemes that use the same set of routers for both forward and backward paths between a pair of nodes by choosing different baseline routes and a

direction function. In practice, however, we found that a simple direction function is sufficient to obtain any BackTrack routes because a different direction function can be reflected by changing the baseline routes.

3.3 Deadlock Freedom

The BackTrack routing scheme may introduce paths that do not exist in the baseline routes. As a result, there is a potential for a deadlock in the BackTrack routes even if the baseline routes are deadlock-free. In general, the BackTrack routes need to be analyzed for each baseline scheme. For simple baseline routing schemes, the BackTrack routes can be deadlock free. Otherwise, complex routing schemes may require measures such as an escape channel or static virtual channel allocations for deadlock freedom. We discuss the details of deadlock freedom for two BackTrack routing schemes later in this section.

3.4 Impact on Network Contention

Another major concern for BackTrack is its implication for the network performance: throughput and latency. Intuitively, the BackTrack scheme restricts routes for backward flows and has less freedom compared to the baseline routes. Here, we discuss the implication of the BackTrack restrictions on network performance in terms of network channel loads.

Definition 5 Channel Load*: For edge (u, v), we define a channel load $l(u, v)$ as the aggregated demands of all flows that are routed through the edge.* $l(u, v) = \sum_{i=1}^{k} d_i \cdot f_i(u, v)$*. The maximum channel load is the maximum load across all channels E:* $\max_{\forall (u,v) \in E} l(u, v)$.

While not an accurate measure of the network throughput and latency, the channel loads indicate the level of congestion in the network and are often used as the optimization metric in routing algorithms [3]. In general, routes with high channel loads are likely to result in high latencies and low throughput.

If we split the channel loads into two components: one for forward flows $l_f(u, v)$ and one for backward flows $l_b(u, v)$, from the way that routes are constructed, $l_f(u, v)$ is equal to $l_b(v, u)$. Therefore, if routes for forward flows equally distribute loads across channels ($l_f(u, v) = constant$), the total channel loads will also be well distributed and the BackTrack routes can be as good as the baseline routes. On the other hand, if the forward routes stress only a small number of channels in both directions, the maximum channel load can be doubled in the worst case for BackTrack. However, for sparsely utilized networks, the BackTrack routes are likely to perform close to the baseline routes even when the forward routes are not well distributed. We will discuss the network contention issues for more for specific routing algorithms in the following subsections.

We note that the throughput and the latencies of the BackTrack routes depend on individual network traffic patterns. BackTrack routes may perform better than the baseline routes on certain traffic patterns

BT-XY(s, t)
1. If $dir(s,t) = 1$, use the XY routing.
2. Otherwise, use the YX routing.

Figure 6: BackTrack routing algorithm for XY (BT-XY).

while worse on other patterns.

3.5 BackTrack for Dimension Order Routing

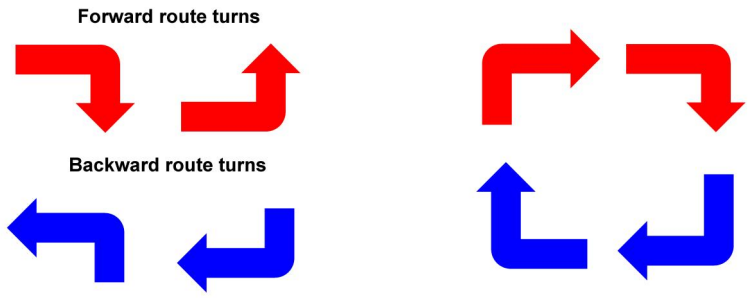
Given the general BackTrack scheme, let us consider how the scheme can be applied to specific network topologies and routing schemes. For the following discussions, we focus on an N-by-N 2-dimensional mesh network. A node u in the mesh network can be represented by a 2-dimensional coordinate (x, y) where $1 \leq x, y \leq N$. Except for the nodes on edges of the network, each node at (x, y) is connected to its four nearest neighbors $(x + 1, y)$, $(x - 1, y)$, $(x, y + 1)$, and $(x, y - 1)$ through bi-directional channels; the network graph has two edges between connected nodes, one in each direction. This subsection discusses how BackTrack can be applied to the dimension ordered routing such as XY or YX routing, which is arguably the most common oblivious routing scheme. The next subsection discusses BackTrack for a more complex routing scheme.

Routing Algorithm Dimension order routing is a simple oblivious routing scheme that routes a flow in one dimension at a time. For example, the XY routing from $s = (x_s, y_s)$ to $t = (x_t, y_t)$ first routes a flow along the x-axis to x_t and next along the y-axis to y_t. Similarly, the YX routing route a flow along the y-axis then the x-axis to the sink.

BackTrack can be applied to the XY or YX routing scheme by choosing a route based on the flow direction. Here, we will use a simple direction function that labels flows from left-to-right as forward flows: $dir(s,t) = 1$ if $x_s \leq x_t$, 0 otherwise. We refer to this BackTrack routing algorithm based on XY as BT-XY. Figure 6 shows the BT-XY routing algorithm.

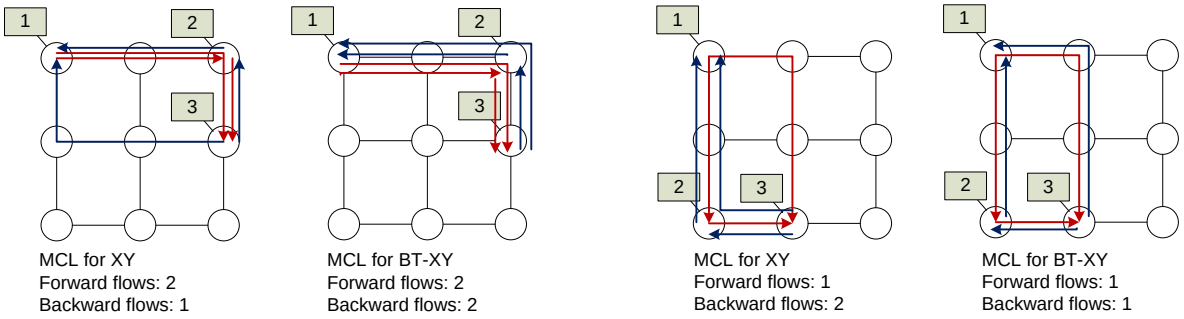
Note that the BackTrack routes for backward flows can be simply determined by the YX routing when the baseline routing scheme is XY. Figure 4 shows examples of XY and BT-XY. As shown in the figure, the baseline XY routing uses two different sets of routers for forward and backward flows while BT-XY uses the same set of routers.

Hardware Changes BT-XY only requires minor changes to the routing module of network hardware. As in the baseline XY routing, BT-XY still allows that a router in each node to determine the next hop only based on the sink (destination) location and the current location. Given that the router frequency is often limited by the virtual channel allocation stage, not the routing state, we believe that the additional routing



(a) Deadlock freedom of the XY routing (b) A deadlock example

Figure 7:



(a) An example where XY is better than BT-XY. (b) An example where BT-XY is better than XY.

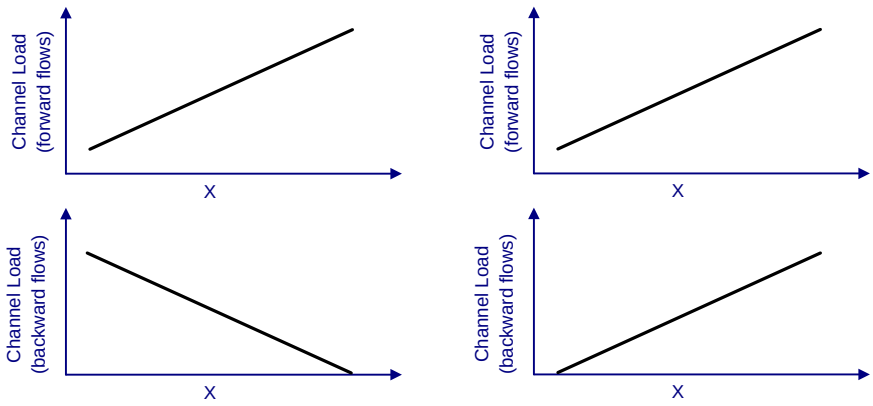
Figure 8: Example traffic patterns whose maximum channel loads are different between XY and BT-XY.

complexity will have no impact the performance of routers.

Deadlock Freedom Figure 7(a) shows the deadlock freedom of BackTrack. As shown in the top half of the figure, forward paths from left to right in BT-XY can only make two turns - east-to-north or east-to-south. Similarly, backward paths can only make two turns - north-to-west or south-to-west. Therefore, the forward and backward paths together cannot form a cycle and thus are deadlock free.

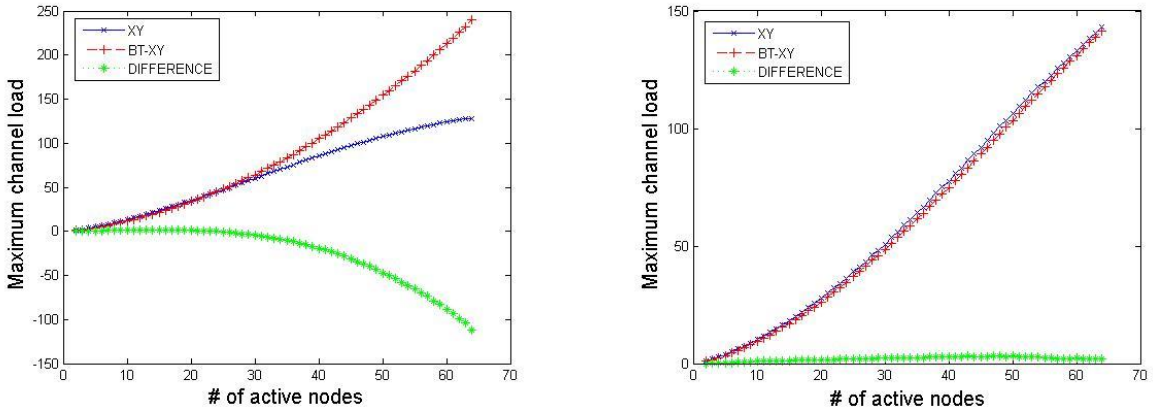
Network Contention As in virtually all oblivious routing schemes, the actual network latency and the throughput of XY and BT-XY depend largely on individual traffic patterns. For certain traffic patterns, XY may result in better network performance whereas BT-XY may be better for other patterns. For example, Figure 8(a) shows an example where XY has less link contention than BT-XY because XY for backward flows result in lower channel loads. Here, we consider a case where three nodes are active and each node sends messages to all other nodes. For XY-BT, flows from node 2 to 1 and from 3 to 1 share the same horizontal links on the top whereas these flows do not share a link for XY. On the other hand, Figure 8(b) illustrates an opposite case where BT-XY results in lower channel loads because YX routes perform better for backward flows. In this case, flows from 2 to 1 and from 3 to 1 share links for XY but not for BT-XY.

However, because the baseline XY routing scheme is more likely to use vertical channels on the right



(a) XY (b) BT-XY

Figure 9: Channel loads on a vertical link (x, y) to $(x, y+1)$ (or $(x, y-1)$).



(a) XY (b) RDOR

Figure 10: Comparisons of maximum channel loads between the baseline and the BackTrack routes.

side of the network than the left side for forward (left-to-right) flows, BT-XY results in higher maximum channel loads than XY when the network is heavily loaded. To understand the phenomenon, let us consider a channel load on the link between node (x_l, y_l) and $(x_l, y_l + 1)$. For both XY and BT-XY, a forward flow from (x_s, y_s) to (x_t, y_t) $(x_s \leq x_t)$ uses the link if $x_t = x_l$ and $y_s \leq y_l < y_t$. Now let us consider the case where all nodes are active and all nodes communicate with all other nodes. Because there are more forward flows $(x_s \leq x_t)$ for sink nodes on the right side of the network, the vertical links are more heavily loaded on the right side. Figure 9 illustrates this trend for both XY and BT-XY.

For the XY routing, the imbalance in the channel loads for forward flows is offset by the opposite imbalance for backward flows, which utilize channels on the left side more heavily as shown in Figure 9(a). On the other hand, BT-XY more heavily uses the right side of the network even for backward flows because it uses YX routes for right-to-left flows. As a result, BT-XY may perform worse than XY in general when a large number of active nodes are randomly distributed across the network.

Fortunately, in practice, we found that such an imbalance in BT-XY is rather insignificant when only a small number of nodes are active as in the sparsely utilized network. If active nodes are randomly distributed across the network, it is unlikely for a large number of flows to share a link. Figure 10(a) shows the average maximum channel load as a function of the number of active nodes. For a particular number of active nodes, we generated one thousand randomly selected placements of nodes and computed the average of the maximum channel loads. In this experiment, we assumed that all active nodes send messages to all other active nodes with a constant demand. As shown in the figure, the maximum channel load for BT-XY can be significantly higher than that of XY for heavily loaded networks, but is virtually identical to that of XY when less than 30 out of 64 nodes (about 45%) are active. Given that only a small portion of nodes (20%) are expected to be powered on simultaneously, in practice, the performance of BT-XY will be comparable to that of XY.

Moreover, as we will show in the following subsection, the imbalance is not inherent to the BackTrack approach. If a baseline routing scheme more evenly utilizes channels for forward flows, the corresponding BackTrack scheme does not degrade the network performance.

3.6 BackTrack for Randomized Dimension Ordered Routing

While the simple dimension order routing is widely used thanks to its simplicity and reasonable performance, it is also well known that the XY or YX routing can easily result in sub-optimal routes for non-uniform traffic patterns. To address this limitation, researchers have proposed randomized variants to the dimension order routing. For example, Valiant [21] routes a flow through a randomly chosen intermediate node to evenly distribute traffic. Similary, O1Turn [18] evenly chooses between XY and YX to distribute traffic between the two routes.

As discussed in the previous subsection, BackTrack for the simple dimension order routing (BT-XY) suffers from load imbalance for heavily loaded network. This load imbalance comes from the imbalance for forward flows and thus can be eliminated if we use a baseline routing scheme such as Valiant or O1Turn that evenly distributes forward flows across network. However, Valiant and O1Turn randomizes a route on each message and uses multiple paths for each flow, effectively removing the benefit of the BackTrack optimization.

To achieve better load balancing while keeping the number of active routers low, we use a baseline routing scheme that randomizes the route for each flow, but keeps the route for each flow fixed over time. In this new scheme, a route is pseudo-randomly chosen between XY and YX based on the source and sink locations for each flow. Effectively, this baseline scheme follows the idea in O1Turn [18] without dynamically switching

BT-RDOR(s, t)
1. If $dir(s,t) = 1$, use the baseline route based on $hash(s,t)$: XY if $hash(t,s)$ is even, YX otherwise.
2. Otherwise, use the backtrack route based on $hash(t,s)$: YX if $hash(t,s)$ is even, XY otherwise.

Figure 11: BackTrack routing algorithm for RDOR (BT-RDOR).

between XY and YX. More formally, a route from $s = (x_s, y_s)$ to $t = (x_t, y_t)$ is determined by a pseudo-random hash $H = hash(s,t)$: XY if H is even, YX otherwise. The scheme can use any hash function as long as it distributes the traffic evenly between XY and YX. We call this baseline scheme Randomized Dimension Order Routing (RDOR).

Routing Algorithm The BackTrack routes for the RDOR routing scheme can be obtained using the same direction function as BT-XY: $dir(s,t) = 1$ if $x_s \leq x_t$, 0 otherwise. However, in this case, the baseline routes themselves can use either an XY or YX route based on the source and destination locations. Figure 11 shows the XY-RDOR routing algorithm.

Hardware Changes Because the RDOR scheme simply chooses between XY and YX routes, the routing hardware complexity will be comparable to the dimension order routing except that both XY and YX need to be implemented. Also, because RDOR may arbitrarily choose a route between XY and YX, a packet needs to contain one additional "route" bit conveying this choice. The BackTrack version of RDOR (BT-RDOR) only changes the way a source node sets the route bit. As a result, the hardware changes for BT-RDOR compared to the baseline RDOR is minimal.

Deadlock Freedom Allowing both XY and YX routes on any flow makes a deadlock possible for both the baseline RDOR and BT-RDOR. As an example, Figure 7(b) shows how four flows can create a cycle in resource dependency. As a result, both RDOR and BT-RDOR need to deal with deadlocks. In fact, O1Turn [18] has the same problem and solves a problem by statically allocating virtual channels to XY and YX flows; a half of virtual channels are only used by XY flows and the rest is only used by YX flows. We adopt the same approach in this paper. Alternatively, an escape channel can be used as a fall-back route on a deadlock [4].

Network Contention Similar to XY and BT-XY, the amount of resource contention for RDOR and BT-RDOR depends largely on individual traffic patterns. However, unlike BT-XY, BT-RDOR does not suffer from more load imbalance even when the network is heavily loaded. Because the baseline RDOR evenly distributes flows between XY and YX, each channel will be used by roughly the same number of forward and backward flows if we consider a case where all nodes are active and communicate with all others. In this

case, the channel loads are comparable between RDOR and BT-RDOR as shown in Figure 10(b). Therefore, the restrictions on routing that BackTrack imposes do not necessarily degrade the network performance if the baseline routing scheme distributes loads evenly for forward routes.

4 Evaluation

4.1 Experimental Setup

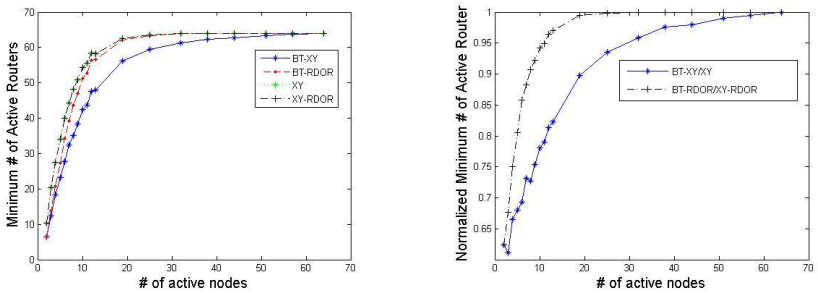
To evaluate the power consumption as well as the performance of BackTrack routing algorithm, we used the Orion2 power models [8] and a cycle-accurate network simulator named Darsim [13]. Darsim is a parallel, highly-configurable, cycle-level network-on-chip simulator based on an ingress-queued wormhole router architecture. We can get the accurate average latencies of individual flows after we feed network parameters such as geometry, bandwidth, crossbar dimensions, and pipeline depths into the Darsim simulator.

Our methodology to evaluate our routing algorithms with Orion2 and Darsim is as follows. First, we generate the input files which include the network size, virtual channel allocation, traffic flows, and the routing algorithms. These input files are fed into Darsim, which can produce average latency of each flow, average traffic load of each link and router, and the number of flits that are sent and received. Based on the output data, we can get the performance of the specific routing algorithm. At the same time, we feed the traffic loads from Darsim into Orion2 and estimate the power consumption. The Orion2 power model can output static power consumption (leakage), clock power, and dynamic power consumption of various components based on activity (load) information.

We use 2-D mesh networks with various sizes (8-by-8, 12-by-12, and 16-by-16). For simulations, we use uniform traffic among active nodes, i.e. each active node sends equal amount of traffic to all the other active nodes. The simulator is configured to have a per-hop latency of 1 cycle, 4 virtual channels (VC) per port, and the flit buffer size per VC of 8 flits. Packets are 8 flits long. We assume the system frequency of 2GHz and a 65nm process technology. Because the BackTrack routing scheme only requires minor changes in the router architecture, we assume an identical clock frequency for all routing algorithms. For each simulation, the network was warmed up for 20,000 cycles and then simulated for 100,000 cycles to collect statistics, which was enough for convergence.

4.2 Network Power Consumption

We first study the effectiveness of BackTrack routing on reducing the network power consumption. For this purpose, we ran a set of experiments on a 8-by-8 mesh network with the traffic injection rate of 0.2



(a) Number of Active Routers (b) Normalized Number of Active Routers

Figure 12: Number of Active Routers vs. Number of Active Nodes

flit/cycle/node while varying the number of active nodes from 2 to 64. The location of the active nodes are randomly selected. For each case, the result is computed as the average of 100 runs.

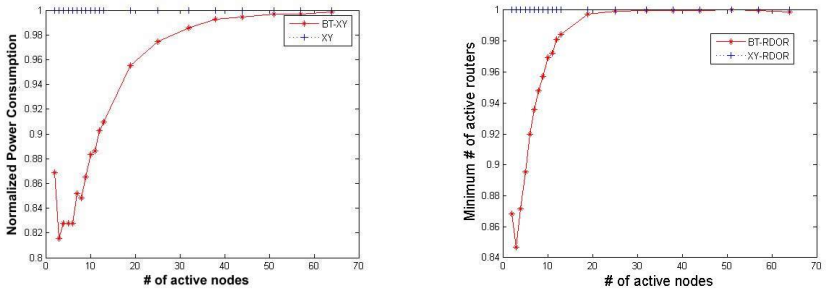
Figure 12 shows the number of routers that are used by the routing algorithm as a function of the number of active nodes. Essentially, this graph shows how effective BackTrack is in increasing the number of unused routers. As shown in the figure, BT-XY uses significantly fewer routers than XY, especially when the number of active nodes is small. If only a few nodes are active, BT-XY can remove almost 40% routers from XY. BT-RDOR also uses fewer routers than XY-RDOR. However, the savings from BT-RDOR is smaller than BT-XY because RDOR uses more diverse paths.

Figure 13 shows the total power consumption of BackTrack routing normalized to the baseline routing algorithm. Thanks to the idle power savings from the unused routers, BT-XY and BT-RDOR can both result in significant power savings (up to 18%) for a small number of active nodes. Even when 20% of nodes are active, BT-XY can still reduce the overall network power consumption by about 10%. However, BackTrack is ineffective when a large number of nodes are active. As the number of active nodes increases, the network flows are more likely to use more links and routers, which reduce the potential benefit of BackTrack routing.

To further understand the sources of the power savings, Figure 14 shows the breakdown between idle and active power consumption for each routing scheme when the number of active nodes is 6. As shown in the figure, the idle power consumption is significant when a small number of nodes are powered on, and BackTrack mainly reduces the idle power consumption.

4.3 Performance

To evaluate the impact of BackTrack routing on performance, we study the average latency of network flows while changing the injection rate and the number of active nodes. Figure 15 shows the average latency of flows with a fixed injection as a function of the number of active nodes. As expected from the analytical analysis, the performance BT-RDOR closely matches that of RDOR independent of the number of active



(a) BT-XY vs XY (b) BT-RDOR vs RDOR

Figure 13: Power Savings of BackTrack against Baseline Routing Algorithm

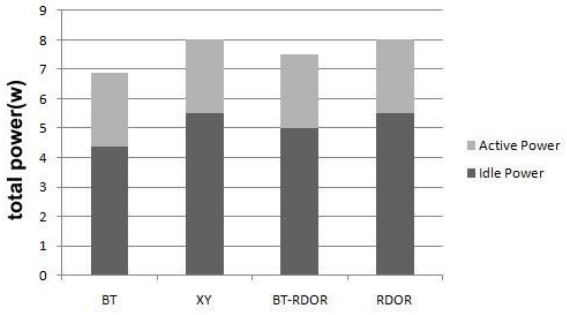
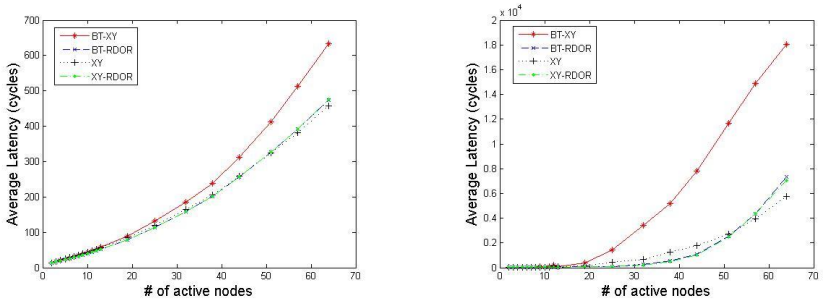


Figure 14: power breakdown: idle power and active power

nodes. BT-XY can also match the performance of XY for a small number of active nodes. However, as the number of active nodes becomes large, the average latency of BT-XY increases rapidly, resulting in worse performance than XY. The performance loss also relates to the corresponding injection rate. Comparing Figure 15(a) with Figure 15(b), we can see the performance drops faster when the injection rate is high. However, if less than 20% of nodes are active, the experiments show that BT-XY has almost the same performance as XY even for a high injection rate of 0.4 flit / cycle / node.

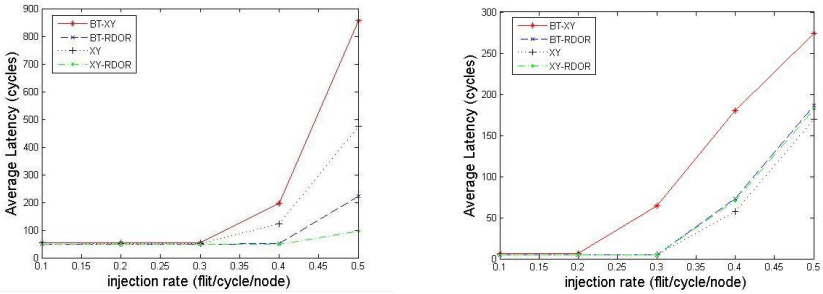
Figure 16 shows the average latency as a function of the injection rate when the number of active nodes is fixed. The figure shows that the network gets saturated at a lower injection rate with a larger number of active nodes. Again, the performance numbers for BT-RDOR and RDOR are almost identical across all injection rates. BT-XY matches the performance of XY for low injection rate but becomes worse as the injection rate increases.

The experimental results suggest that BT-XY can provide significant reduction in the idle power consumption without any performance loss when the network injection rate is low and only a small number of nodes are active. Given the technology scaling trend, we believe that future many-core processors will satisfy these conditions. In these cases, BT-XY provides power savings at virtually no costs. For the cases when the network is likely to experience high injection rates, BT-RDOR provides a way to reduce the idle power consumption without any impact on performance.



(a) 0.2 flit/cycle/node (b) 0.4 flit/cycle/node

Figure 15: Average Latency vs. number of active nodes with fixed injection rate



(a) 12 active nodes (b) 64 active nodes

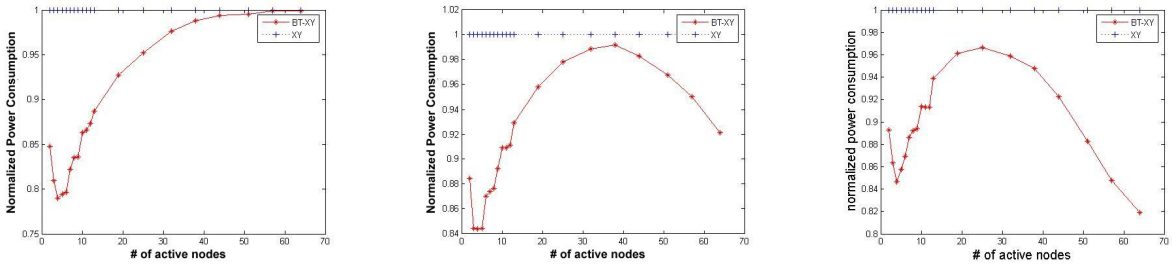
Figure 16: Average Latency vs. injection rate with fixed number of active nodes

4.4 Power Saving with Variable Injection Rate

Here, we study the relation between power savings and the injection rate. Due to the space limit, we only show the results comparing BT-XY and XY. The trend is similar for the comparison between BT-RDOR and RDOR. Figure 17 show the power savings as a function of the number of active nodes for three different injection rates. As expected, for all injection rates, the graphs show that BT-XY reduces power consumption significantly when the number of active nodes are small. On the other hand, surprisingly Figure 17(b) and 17(c) seem to indicate that BT-XY saves a lot of power for a large number of active nodes, which is counter intuitive. This drop in the normalized power consumption is an anomaly due to the performance difference between BT-XY and XY. During the simulations, BT-XY transfers fewer data flits than XY does, thus the active power of BT-XY becomes much smaller than XY. The drop of the curve results from active power reduction, not the idle power. Figure 18 shows the power breakdown between active power and idle power, and clearly shows less active power for BT-XY compared to XY.

4.5 Scalability of BackTrack

To test the scalability of BackTrack, we simulated the network sizes of 12-by-12 and 16-by-16. Figure 19 shows the power savings for the 12-by-12 mesh network with the injection rate of 0.2 flit/cycle/node. The



(a) 0.1 flit/cycle/node (b) 0.3 flit/cycle/node (c) 0.4 flit/cycle/node

Figure 17: Power Consumption under different injection rate

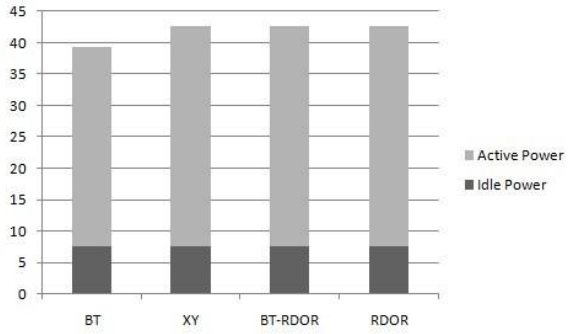
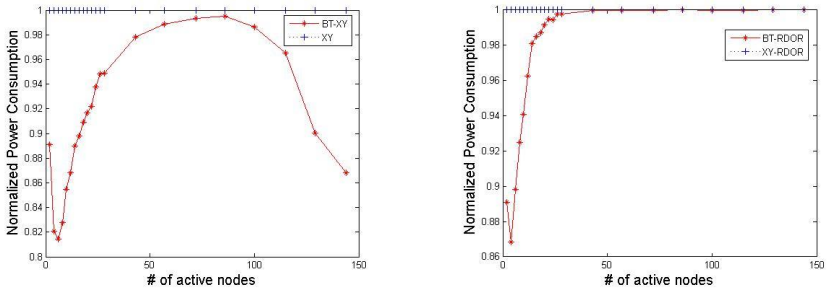


Figure 18: power breakdown: idle power and active power

overall shape of the graph is the same as that of 8-by-8 mesh, which validates the scalability of BackTrack scheme. The 16-by-16 experiments have almost the same results and not shown here due to space limit.

Figure 20 presents the average latency as we change the injection rate for a 12-by-12 mesh with 28 active nodes. Since 28 is a small protion (20%) of the total nodes, BackTrack works pretty well without any performance loss. Even BT-XY closely matches the performance of XY across all injection rates. This results suggest that BackTrack will become more attractive for future networks as the number of network nodes scale up and the percentage of active nodes decreases.



(a) BT-XY vs. XY under 12-by-12 mesh (b) BT-RDOR vs. RDOR under 12-by-12 mesh

Figure 19: Power Consumption for 12-by-12 mesh networks

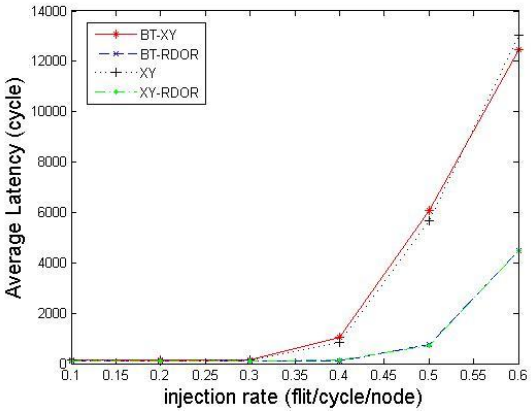


Figure 20: Latency vs. injection rate curve for 12-by-12 mesh size

5 Related Work

There are many techniques that target to reduce the power consumption of on-chip networks. A large number of network power saving techniques aims to reduce active power consumption and are complementary to BackTrack. Previous approach to reduce the idle power consumption relies on dynamically turning off components. On the hand, BackTrack exploits an observation that only a small number of nodes can be active in future on-chip networks and aims to reduce unused routers without dynamic adaptation. This section provides an overview of these techniques and discusses how they are related to BackTrack.

One commonly used technique to save the link power is dynamic voltage scaling (DVS). Shang et al.[19] developed a history-based DVS scheme to adjusts operating voltage and clock frequency of each link according to the utilization of the link and buffer. The main idea is to use past network utilization to predict future traffic in order to tune both link voltage and frequency. The DVS scheme aims to dynamically reduce network power consumption while maintaining the performance. However, current DVS techniques take thousands of cycles to shift between voltage levels and require additional hardware such as transmitter, receiver, PLL and adaptive power supply regulator for each link.

Shin and Kim [20] proposed an off-line link speed assignment algorithm to reduce the link power usage. Given the task graph of the real-time application, the algorithm can generate an appropriate speed assignment to each link according to the traffic pattern, thus saving link power consumption. However, this approach suffers from the static assignment and thus is only applicable to periodic real-time application.

Seung and Nader [11] presented a variable frequency link which adopts dynamic frequency scaling (DFS) based on the previous link utilization parameters. In this scheme, the link frequency is dynamically adjusted using a clock boosting mechanism, and the link utilization is measured by sampling a link during a pre-defined period. Unlike DVS, DFS uses the same voltage for all links and offers fast response time and

less hardware costs. The DVFS techniques target to reduce dynamic power consumption and complement BackTrack that reduces idle power consumption.

Besides DVFS links, another alternative to save the link power is to power down the under-utilized links. Kim et al. [10] proposed a dynamic link shutdown (DLS) algorithm which gate power to links when their utilizations are below a certain threshold level. An adaptive routing strategy is used to continue communication only with a subset of links.

Hale et al.[7] came up with a more aggressive scheme, named "segment gating", to further reduce the idle power consumption. Instead of just powering down under-utilized links, the scheme also powers down arbitration logic, switches, and buffers. This technique is a combination of link power gating and partially turning off routers. The technique needs to keep track of link utilization and uses dedicated control logic to turn off individual parts of the routers.

Some other link power saving technique makes use of the compilers. Li et al.[12] proposed a compiler-directed technique in order to turn off the link based on the compiling information. Even though it saves significant power during the idle periods, this technique heavily depends on the compiler and is only valid for loop-intensive applications.

In terms of reducing router power consumption, Rahmani et al.[16] presented a technique called dynamic virtual channels allocation (DVCA) to reduce the router power. This technique makes use of the traffic conditions and past buffer utilization to dynamically forecast the number of the virtual channels that should be active, thus reducing the power while maintaining the performance. The clock gating power management technique is used to activate/deactivate the VCs.

Concatto et al.[2] proposed a reconfigurable router where channels can lend or borrow buffers from neighboring channels based on buffer utilization. In this manner, the heavily used channel can have up to three times buffer size, improving performance without adding buffers to each channel. The router can also use smaller buffers to reduce power without sacrificing performance. However, this reconfigurable router needs extra hardware overhead such as multiplexer and wires and is helpless when all the channels are busy.

6 Conclusion

Although technology scaling will continue to allow more transistors to be placed on a chip, power limitations will restrict the amount of silicon that can be active simultaneously. For multi-core systems the implication is that only a small number of cores will be on at a given time. Given this situation, the BackTrack routing technique attempts to increase the number of unused routers by using the same forward and backward path for a flow. Applying this technique to XY and RDOR routing, we found that BackTrack was able to save

power while having little or no impact on performance.

References

[1] A. Banerjee, R. Mullins, and S. Moore. A power and energy exploration of network-on-chip architectures. In *Networks-on-Chip, 2007. NOCS 2007. First International Symposium on*, pages 163 –172, May 2007.

[2] C. Concatto, D. Matos, L. Carro, F. Kastensmidt, A. Susin, and M. Kreutz. Noc power optimization using a reconfigurable router. In *VLSI, 2009. ISVLSI '09. IEEE Computer Society Annual Symposium on*, pages 235 –240, May 2009.

[3] W. J. Dally and B. P. Towles. *Principles and Practices of Interconnetion Networks*. Morgan Kaufmann, 2004.

[4] J. Duato. A new theory of deadlock-free adaptive routing in wormhole networks. *IEEE Trans. Parallel Distrib. Syst.*, 4:1320–1331, December 1993.

[5] P. Gratz, B. Grot, and S. Keckler. Regional congestion awareness for load balance in networks-on-chip. In *High Performance Computer Architecture, 2008. HPCA 2008. IEEE 14th International Symposium on*, pages 203 –214, February 2008.

[6] P. Gratz and S. W. Keclker. Realistic Workload Characterization and Analysis for Networks-on-Chip Design. In *The 4th Workshop on Chip Multiprocessor Memory Systems and Interconnects (CMP-MSI)*, 2010.

[7] K. Hale, B. Grot, and S. Keckler. Segment gating for static energy reduction in networks-on-chip. In *Network on Chip Architectures, 2009. NoCArc 2009. 2nd International Workshop on*, pages 57 –62, December 2009.

[8] A. Kahng, B. Li, L.-S. Peh, and K. Samadi. Orion 2.0: A fast and accurate noc power and area model for early-stage design space exploration. In *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09.*, pages 423 –428, April 2009.

[9] U. Karpuzcu, B. Greskamp, and J. Torrellas. The bubblewrap many-core: Popping cores for sequential acceleration. In *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*, pages 447 –458, December 2009.

[10] E. Kim, K. Yum, G. Link, N. Vijaykrishnan, M. Kandemir, M. Irwin, M. Yousif, and C. Das. Energy optimization techniques in cluster interconnects. In *Low Power Electronics and Design, 2003. ISLPED '03. Proceedings of the 2003 International Symposium on*, pages 459 – 464, August 2003.

[11] S. E. Lee and N. Bagherzadeh. A variable frequency link for a power-aware network-on-chip (noc). *Integr. VLSI J.*, 42:479–485, September 2009.

[12] F. Li, G. Chen, M. Kandemir, and M. J. Irwin. Compiler-directed proactive power management for networks. In *Proceedings of the 2005 international conference on Compilers, architectures and synthesis for embedded systems*, CASES '05, pages 137–146, New York, NY, USA, 2005. ACM.

[13] M. Lis, K. S. Shim, M. H. Cho, P. Ren, O. Khan, and S. Devadas. Darsim: a parallel cycle-level noc simulator. In *MOBS-6: Sixth Annual Workshop on Modeling, Benchmarking and Simulation*, 2010.

[14] H. Matsutani, M. Koibuchi, D. Wang, and H. Amano. Run-time power gating of on-chip routers using look-ahead routing. In *Design Automation Conference, 2008. ASPDAC 2008. Asia and South Pacific*, pages 55 –60, March 2008.

[15] T. Moscibroda and O. Mutlu. A case for bufferless routing in on-chip networks. In *Proceedings of the 36th annual international symposium on Computer architecture*, ISCA '09, pages 196–207, New York, NY, USA, 2009. ACM.

[16] A.-M. Rahmani, M. Daneshtalab, A. Afzali-Kusha, S. Safari, and M. Pedram. Forecasting-based dynamic virtual channels allocation for power optimization of network-on-chips. In *VLSI Design, 2009 22nd International Conference on*, pages 151 –156, January 2009.

[17] E. Salminen, A. Kulmala, and T. Hamalainen. On the credibility of load-latency measurement of network-on-chips. In *System-on-Chip, 2008. SOC 2008. International Symposium on*, pages 1 –7, November 2008.

[18] D. Seo, A. Ali, W.-T. Lim, N. Rafique, and M. Thottethodi. Near-Optimal Worst-Case Throughput Routing for Two-Dimensional Mesh Networks. In *Proceedings of the 32nd Annual International Symposium on Computer Architecture (ISCA 2005)*, pages 432–443, 2005.

[19] L. Shang, L.-S. Peh, and N. Jha. Dynamic voltage scaling with links for power optimization of interconnection networks. In *High-Performance Computer Architecture, 2003. HPCA-9 2003. Proceedings. The Ninth International Symposium on*, pages 91 – 102, February 2003.

[20] D. Shin and J. Kim. Power-aware communication optimization for networks-on-chips with voltage scalable links. In *Hardware/Software Codesign and System Synthesis, 2004. CODES + ISSS 2004. International Conference on*, pages 170 – 175, September 2004.

[21] L. G. Valiant and G. J. Brebner. Universal schemes for parallel communication. In *STOC '81: Proceedings of the thirteenth annual ACM symposium on Theory of computing*, pages 263–277, 1981.

[22] G. Venkatesh, J. Sampson, N. Goulding, S. Garcia, V. Bryksin, J. Lugo-Martinez, S. Swanson, and M. B. Taylor. Conservation cores: Reducing the energy of mature computations. In *Proceedings of the 15th International Conference on Architectural Support for Programming Languages and Operation Systems*, ASPLOS '10, March 2010.