Flattened Butterfly Topology for On-Chip Networks

John Kim, James Balfour, and William J. Dally
Computer Systems Laboratory
Stanford University, Stanford, CA 94305
{*jjk12, jbalfour, dally*}*@cva.stanford.edu*

Abstract

With the trend towards increasing number of cores in chip multiprocessors, the on-chip interconnect that connects the cores needs to scale efficiently. In this work, we propose the use of high-radix networks in on-chip interconnection networks and describe how the flattened butterfly topology can be mapped to on-chip networks. By using high-radix routers to reduce the diameter of the network, the flattened butterfly offers lower latency and energy consumption than conventional on-chip topologies. In addition, by exploiting the two dimensional planar VLSI layout, the on-chip flattened butterfly can exploit the bypass channels such that non-minimal routing can be used with minimal impact on latency and energy consumption. We evaluate the flattened butterfly and compare it to alternate on-chip topologies using synthetic traffic patterns and traces and show that the flattened butterfly can increase throughput by up to 50% compared to a concentrated mesh and reduce latency by 28% while reducing the power consumption by 38% compared to a mesh network.

1 Introduction

Chip multiprocessors are becoming more widely utilized to efficiently use the increasing number of transistors available in a modern VLSI technology. As the number of cores increases in such architectures, an on-chip network is used to connect the cores to provide a communication substrate. These on-chip networks should provide both low latency and high bandwidth communication fabrics that efficiently support a diverse variety of workloads.

On-chip interconnection networks have recently attracted considerable research attention [5], much of which has focused on microarchitecture improvements [1,19] and routing algorithms [22]. However, selecting an appropriate topology is one of the most critical decisions in the design of an interconnection network because it bounds critical performance metrics such as the network's zero-load latency and its capacity [11], and directly influences the implementation costs, both in terms of on-chip resources and implementation complexity. In this paper, we explore the topology aspect of on-chip interconnection network and argue that high-radix topologies such as the flattened butterfly are more cost-efficient.

Most on-chip networks that have been proposed are low-radix, mostly utilizing a 2-D mesh such as the networks found in the RAW processor [24], the TRIPS processor [12], the 80-node Intel's Teraflops research chip [25], and the 64-node chip multiprocessor from Tilera [2]. Although such low-radix networks provide a very simple network and lead to very short wires in the architecture, these networks have several disadvantages which include long network diameter as well as energy inefficiency because of the extra hops. By reducing the diameter of the network, high-radix networks are advantageous both for latency and power since delay due to intermediate routers is greatly reduced as well as the power consumed by intermediate routers. In this paper, we describe how on-chip networks can take advantage of high-radix routers and describe how the flattened butterfly topology can be used for on-chip networks [14]. A 64-node flattened butterfly network is able to provide lower latency than a concentrated mesh [3] with an equivalent bisection bandwidth. In addition, we describe how the flattened butterfly topology for on-chip networks can exploit the planar VLSI layout and be augmented to allow non-minimal routing without traversing non-minimal physical distances – further reducing latency as well as energy consumption. This can be achieved by adding bypass muxes to the network to improve the efficiency of utilizing *bypass* channels.

This paper is organized as follows. We describe some of the advantages of using high-radix on-chip networks in Section 2. In Section 3, we present the flattened butterfly topology for on-chip networks and describe routing algorithms. The router microarchitecture and bypass channels are described in Section 4. We present evaluation results in Section 5, and then elaborate on the flattened butterfly topology and its on-chip scalability in Section 6. Section 7 discusses related work; we conclude in Section 8.

1072-4451/07 $25.00 © 2007 IEEE
DOI 10.1109/MICRO.2007.29

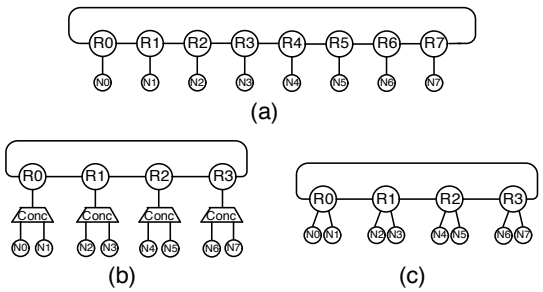

Figure 1. The use of concentration in interconnection networks -- (a) 8 node (N0 - N7) ring with 8 routers (R0 - R7) without concentration, (b) 4 node ring with 2-way concentrator and (c) the same topology as (b) with the 2-way concentrator integrated into the router.

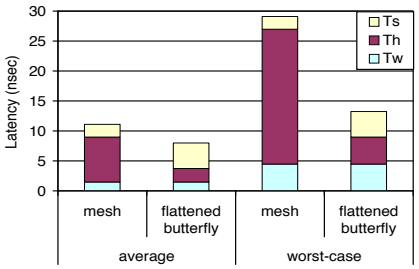

Figure 2. Latency of a packet in on-chip networks.

2 Background

This section describes how the cost structures of on-chip networks differ from system interconnection networks and presents an argument for high-radix networks on chip.

2.1 Cost of On-Chip Networks

The cost of an off-chip interconnection network typically increases with the channel count. As the channel count increases with the hop count, reducing the diameter of the network reduces the cost of the network [16]. On-chip networks differ because bandwidth is plentiful because of inexpensive wires, while buffers are comparatively expensive. However, reducing the diameter of on-chip networks remains beneficial for several reasons. The power consumed by the channels is a significant part of aggregate on-chip network power consumption [3]; thus, reducing the number of channels can reduce the overall power consumption. Furthermore, since buffered flow control is often used in on-chip networks, the aggregate area allocated to the input buffers in the routers decreases as the number of channels is reduced. In this work, we show how the use of high-radix routers and the flattened butterfly topology leads to lower diameter and as a result, leads to lower cost by reducing the number of channels and the amount of buffers required in the network.

In addition to reducing the diameter, the use of *concentration*, where network resources are shared among different processor nodes, can improve the efficiency of the network. An example of concentration is shown in Figure 1. Using a ring topology, 8 nodes can be connected with 8 routers as shown in Figure 1(a). By using a concentration factor of two, the 8 nodes can be connected in a 4 *node* ring where each *node* consists of two terminal nodes and a router, as shown in Figure 1(b). The use of concentration aggregates traffic from different nodes into a single network interface. This reduces both the number of resources allocated to the network routers and the average hop count, which can improve latency. Thus, while providing the same bisection bandwidth, concentration can reduce the cost of the network by reducing the network size. The concentrator can be integrated into the router by increasing the radix of the router, as shown in Figure 1(c), to allow all of the terminal nodes to access the network concurrently, instead of allowing only one terminal node associated with a router to access the network during any one cycle. The use of concentration in on-chip networks also reduces the wiring complexity as shown in Section 6.1.

Concentration is practical for on-chip networks because the probability that more than one of the processors attached to a single router will attempt to access the network on a given cycle is relatively low. For example, in a chip multiprocessor (CMP) architecture with a shared L2 cache distributed across the chip, the L1 miss rate is often under 10% [7], which results in relatively low traffic injection rates at the processors. Consequently, using concentration to share the network resources is an effective technique for CMP traffic. The advantages of using concentration in on-chip network were previously described for 2-D mesh networks [3]. In this paper, we describe how the flattened butterfly topology [15] for on-chip networks use concentration as well as high-radix routers to reduce the diameter of the network to improve cost-efficiency.

2.2 Latency in On-Chip Networks

Latency is a critical performance metric for on-chip networks. The latency of a packet through an interconnection network can be expressed as the sum of the header latency (T_h), the serialization latency (T_s), and the time of flight on the wires (T_w),

$$\begin{aligned} T &= T_h + T_s + T_w \\ &= Ht_r + L/b + T_w \end{aligned}$$

where t_r is the router delay, H is the hop count, L is the packet size, and b is the channel bandwidth.

Minimizing latency requires establishing a careful balance between T_h and T_s. For on-chip networks, wires are

abundant, on-chip bandwidth plentiful, and consequently T_s can be reduced significantly by providing very wide channels. However, traditional 2-D mesh networks tend to establish T_s and T_h values that are unbalanced, with wide channels providing a low T_s while T_h remains high due to the high hop-count. Consequently, these networks fail to minimize latency. For example, in the 2-D mesh network used in the Intel TeraFlop [25], with uniform random traffic, T_h is approximately 3 times T_s and for worst case traffic, there is approximately $10\times$ difference. [1] However, by using high-radix routers and the flattened butterfly topology, the hop count can be reduced at the expense of increasing the serialization latency, assuming the bisection bandwidth is held constant, as is shown in Figure 2. As a result, the flattened butterfly achieves a lower overall latency. Note that the wire delay (T_w) associated with the Manhattan distance between the source and the destination nodes generally corresponds to the minimum packet latency in an on-chip network [18]. This paper describes how high-radix routers and the flattened butterfly topology can be used the build on-chip networks that try to approach this ideal latency by significantly reducing the number of intermediate routers.

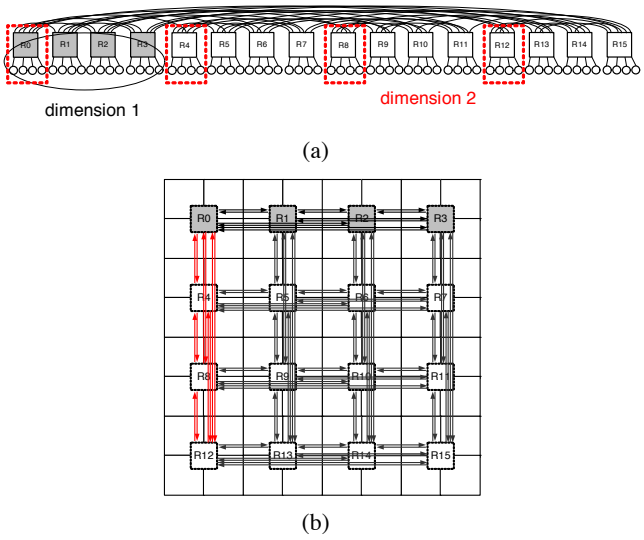

Figure 3. (a) Block diagram of a 2-dimension flattened butterfly consisting of 64 nodes and (b) the corresponding layout of the flattened butterfly where dimension1 routers are horizontally placed and dimension2 routers are vertically placed.

3 On-Chip Flattened Butterfly

3.1 Topology Description

The flattened butterfly topology [15] is a cost-efficient topology for use with high-radix routers. The flattened butterfly is derived by combining (or *flattening*) the routers in each row of a conventional butterfly topology while preserving the inter-router connections. The flattened butterfly is similar to a generalized hypercube [4]; however, by providing concentration in the routers, the flattened butterfly significantly reduces the wiring complexity of the topology, allowing it to scale more efficiently.

To map a 64-node on-chip network onto the flattened butterfly topology, we collapse a 3-stage radix-4 butterfly network (4-ary 3-fly) to produce the flattened butterfly shown in Figure 3(a). The resulting flattened butterfly has 2 dimensions and uses radix-10 routers. With four processor nodes attached to each router, the routers have a concentration factor of 4. The remaining 6 router ports are used for inter-router connections: 3 ports are used for the dimension 1 connections, and 3 ports are used for the dimension 2 connections. Routers are placed as shown in Figure 3(b) to embed the topology in a planar VLSI layout with each router placed in the middle of the 4 processing nodes. Routers connected in dimension 1 are aligned horizontally, while routers connected in dimension 2 are aligned vertically; thus, the routers within a row are fully connected, as are the routers within a column.

[1] The latency calculations were based on Intel TeraFlop [25] parameters (t_r = 1.25ns, b = 16GB/s, L = 320bits) and estimated value of wire delay for 65nm (t_w = 250ps per mm).

The wire delay associated with the Manhattan distance between a packet's source and its destination provides a lower bound on latency required to traverse an on-chip network. When minimal routing is used, processors in this flattened butterfly network are separated by only 2 hops, which is a significant improvement over the hop count of a 2-D mesh. The flattened butterfly attempts to approach the wire delay bound by reducing the number of intermediate routers – resulting in not only lower latency but also lower energy consumption. However, the wires connecting distant routers in the flattened butterfly network are necessarily longer than those found in the mesh. The adverse impact of long wires on performance is readily reduced by optimally inserting repeaters and pipeline register to preserve the channel bandwidth while tolerating channel traversal times that may be several cycles. The longer channels also require deeper buffer sizes to cover the credit round trip latency in order to maintain full throughput.

3.2 Routing and Deadlock

Both minimal and non-minimal routing algorithms can be implemented on the flattened butterfly topology. A limited number of virtual channels (VCs) [8] may be needed to prevent deadlock within the network when certain routing algorithms are used. Additional VCs may be required for purposes such as separating traffic into different classes or avoiding deadlock in the client protocols. Dimension-ordered routing (DOR) can be used as a minimal routing algorithm for the flattened butterfly (e.g. route in dimension1,

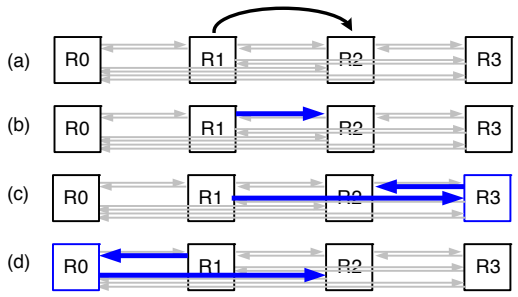

Figure 4. Routing paths in a 2-D on-chip flattened butterfly. (a) All of the traffic from nodes attached to R1 is sent to nodes attached to R2. The minimal path route is shown in (b) and the two non-minimal paths are shown in (c) and (d). For simplicity, the processing nodes attached to these routers are not shown and only the first row of routers is shown.

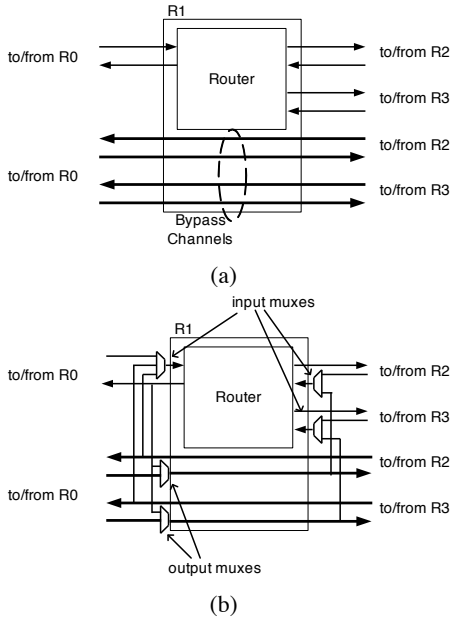

Figure 5. Flattened butterfly router diagram with bypass channels in (a) a conventional flattened butterfly and (b) a flattened butterfly with muxes to efficiently utilize the bypass channels. The router diagram is illustrated for router R1 in Figure 4 with the connections shown only for a single dimension of the flattened butterfly.

then route in dimension 2); in this case, the routing algorithm itself is restrictive enough to prevent deadlock. Non-minimal routing allows the path diversity available in the flattened butterfly network to be used to improve load balance and performance. For these reasons, we use the non-minimal global adaptive routing (UGAL) [23] algorithm when evaluating the flattened butterfly topology in this work. UGAL load balances by determining whether it is beneficial to route minimally or nonminimally. If it selects nonminimal routing, UGAL routes minimally to an intermediate node in the first phase, and then routing minimally to the destination in the second phase. To reduce the number of VCs that are needed, we use DOR within each phase of UGAL routing – thus, only 2 VCs are needed.

4 Bypass Channels and Microarchitecture

As shown in Figure 3, the routers are fully connected in each dimension. As a result, the channels that pass over other routers in the same row or column can be referred to as *bypass channels* – i.e. channels that bypass local routers to connect directly to its destination router. The use of these bypass channels with non-minimal routing in on-chip flattened butterfly topology can result in non-minimal physical distance being traversed. An example is shown in Figure 4 with a traffic pattern shown in Figure 4(a). The minimal path for this traffic pattern is shown in Figure 4(b) and the non-minimal paths are shown in Figure 4(c,d). The layout of an on-chip flattened butterfly can result in the non-minimal routes *overshooting* the destination on the way to the intermediate node selected for load-balancing (Figure 4(c)). A non-minimal route may also route a packet away from its destination before it is routed back to its destination on a bypass channel that passes over the source (Figure 4(d)). To avoid the inefficiencies of routing packets on paths of non-minimal physical lengths, the bypass channels can be connected to those routers they pass over. These additional connections allow packets to enter or leave the bypass channels early when doing so is beneficial. In this section, we explain how the router microarchitecture and the flow control mechanisms can be modified to connect the bypass channels directly to the router switch in order to reduce latency and improve energy efficiency.

4.1 Router Bypass Architecture

A high-level diagram of a router in an on-chip flattened butterfly is shown in Figure 5(a). It consists of the switch and bypass channels that connect the neighbouring routers. One method to connect the bypass channels to the local router is to add additional inputs to the switch. However, doing so would significantly increase the complexity of the switch. For example, in the flattened butterfly shown in Figure 3, the switch would increase from 10×10 to 18×18 in the worst case, nearly quadrupling the area consumed by the switch. In addition, the use of bypass channels is not intended to increase the bandwidth of the topology, but rather to reduce latency and energy – thus, the larger switch, which would provide additional bandwidth, is not needed. Instead, we break the bypass channels as they pass over the router and insert muxes as shown in Figure 5(b).

As illustrated in Figure 5(b), two types of muxes are added to connect the bypass channels to the local router: input muxes, and output muxes.[2] The inputs to the muxes can be classified as either bypass inputs (e.g. inputs from the bypass channels) or direct inputs (e.g. inputs/outputs to/from the local router). The input muxes receive packets destined for the local router that would otherwise bypass the local router enroute to the intermediate node selected by the routing algorithm, as illustrated in Figure 4(c). Thus, each input mux receives both the direct inputs that the packet would have used if the non-minimal path was taken and the inputs from the bypass channels. The output muxes are used by packets that would have initially been routed away from their destinations before being routed back over the local router, as illustrated in Figure 4(d). The inputs to the output muxes are the direct outputs from the local router and the bypass channel inputs – the path the packet would have taken if non-minimal routing path was taken. The addition of these muxes does not eliminate the need for non-minimal routing for load-balancing purpose. Instead, the muxes reduce the distance traveled by packets to improve energy efficiency and reduce latency.

4.2 Mux Arbiter

The arbiters that control the bypass muxes are critical to the proper utilization of the bypass channels. A simple round-robin arbiter could be implemented at the muxes. While this type of arbiter leads to a locally fair arbitration at the mux, the arbiter does not guarantee global fairness [11]. To provide global fairness, we implement an *yield* arbiter that yields to the *primary* input – i.e. the input that would have used the channel bandwidth at the output of the mux if the bypass channels were not connected to the local router. Accordingly, the direct input is given priority at an input mux, while the bypass channel is given priority at an output mux. Thus, if the primary input is idle, the arbiter grants access to the non-primary input.

To prevent starvation of the non-primary inputs, a control packet is sent along the non-minimal path originally selected by the routing algorithm. This control packet contains only routing information and a marker bit to identify the packet as a control packet. The control packet is routed at the intermediate node as though it were a regular packet, which results in it eventually arriving at the primary input of the muxes at the destination router. When the control packet arrives, the non-primary input is granted access to the mux output bandwidth. This policy guarantees that a packet waiting at the non-primary input of a bypass mux will eventually be granted access to the bypass mux. In the worst-case

[2]Using the analogy of cars and highways, the long wires introduced correspond to adding highways. The input muxes correspond to adding additional exit ramps to the highways while the outputs muxes correspond to adding entrance ramps to get on the highway.

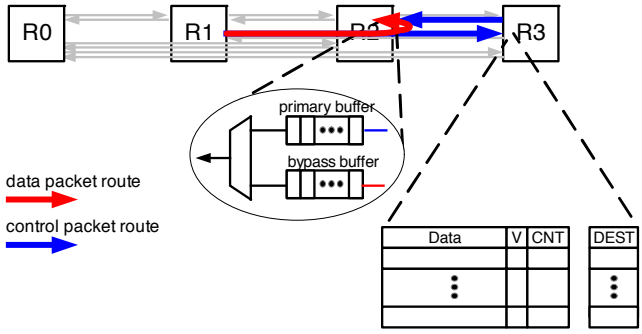

Figure 6. Modification to the buffers introduced into the flow control with the utilization of bypass channels. The additional bits of the buffers correspond to V : valid bit, CNT : count of control packet, and DEST corresponds to control packet content which contains a destination. Once a flit in the input buffer is processed, if the V bit is set, CNT number of control packets are processed in the router before the next flit in the input buffer is processed.

(i.e. highly congested) environment, the latency of the non-minimal routed packets will be identical to the flattened butterfly that does not directly utilize the bypass channels. However, there will still be energy savings because the flit does not traverse the non-minimal physical distance; instead, only the control flit, which is much smaller than a data packet, travels the full physical distance of the non-minimal route.

4.3 Switch Architecture

With minimal routing, the crossbar switch can be simplified because it need not be fully connected. Non-minimal routing increases the complexity of the switch because some packets might need to be routed twice within a dimension, which requires more connections within the crossbar. However, by using the bypass channels efficiently, non-minimal routing can be implementing using a switch of lesser complexity – one that approaches the complexity of a flattened butterfly that only supports minimal routing. If the bypass channels are utilized, non-minimal routing does not require sending full packets through intermediate routers, and as a result, the connections within the switch themselves approaches that of the switch that only supports minimal routing.

4.4 Flow Control and Routing

Buffers are required at the non-primary inputs of the bypass muxes for flow control as illustrated in Figure 6. Thus with non-minimal routing, credits for the bypass buffers are needed before packet can depart a router. The control packets that are generated can be buffered in the input buffers.

However, since buffers are an expensive resource in on-chip networks, instead of having the short control packets occupying the input buffers, minor modifications can be made to the input buffers to handle the control packets. Once a control packet arrives, the destination of the control packets is stored in a separate buffer (shown with DEST field in Figure 6) and the control bits of the input buffer need to be updated properly such that the control packet can be properly processed in the router. These modifications should introduce little overhead because the datapaths of on-chip networks are typically much wider than required for the control packet.

The bypass channels are exploited when non-minimal routing is utilized and to properly load balance the channels, we modify the UGAL [23] routing algorithm to support bypass channels. When non-minimal routing is selected, instead of routing to an intermediate node and then routing to the destination, UGAL is broken into multiple phases such that UGAL is applied in dimension 1 followed by UGAL in dimension 2. These ensure that load balancing is achieved while using the bypass channel to provide the minimum physical path from source to destination.

5 Evaluation

We compare the performance of the following topologies in this section:

1. conventional 2-D mesh (MESH)
2. concentrated mesh with express channels [3] (CMESH)
3. flattened butterfly (FBFLY)
   (a) flattened butterfly with minimal routing only (FBFLY-MIN)
   (b) flattened butterfly with non-minimal routing (FBFLY-NONMIN)
   (c) flattened butterfly with non-minimal routing and use of bypass channels (FBFLY-BYP)

The topologies were evaluated using a cycle accurate network simulator. We compare the networks' performance and power consumption. The power consumption is based on the model described in [3] for a 65nm technology. We accurately model the additional pipeline delay required for the high-radix routers as well as the additional serialization latency through the narrower channels. The bisection bandwidth is held constant across all topologies for the purpose of comparing the performance of the different networks.

5.1 Performance

We compare the performance of the different topologies for a 64-node on-chip network by evaluating their throughput using open-loop simulation and also compare them using closed-loop simulation, using both synthetic traffic pattern and traces from simulations. The routing algorithms used for the different topologies are described in Table 1. For flow control, virtual channel flow control is used with 2 VCs to break routing deadlock and another 2 VCs needed to break protocol deadlock for the closed-loop simulations.

Topology	Routing
FBLY-MIN	randomized-dimension
FBLY-NONMIN	UGAL [23]
FBLY-BYPASS	UGAL [23]
CMESH	O1Turn with express channels [3]
MESH	O1Turn [22]

Table 1. Routing algorithms used in simulation comparison.

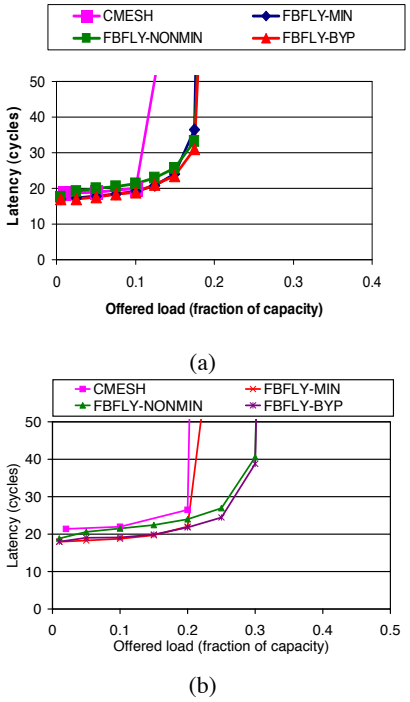

Figure 7. Throughput comparison of CMESH and FBFLY for (a) tornado and (b) bit complement traffic pattern.

5.1.1 Throughput Comparison

To evaluate the throughput, the simulator is warmed up under load without taking measurements until steady-state is reached. Then a sample of injected packets is labeled during a measurement interval. The simulation is run until all labeled packets exit the system. For the throughput analysis, packets are assume to be single-flit packets.

In Figure 7, we compare the latency vs. offered load on two adversarial traffic patterns for CMESH and FBFLY – two topologies that utilize concentration to reduce the cost

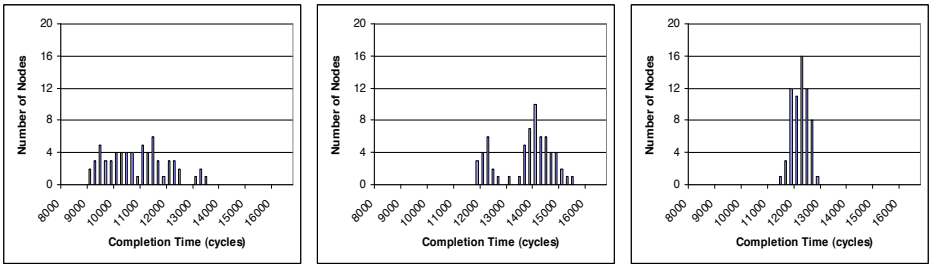

Figure 9. Node completion time variance for the different topologies (a) mesh (b) CMESH and (c) flattened butterfly.

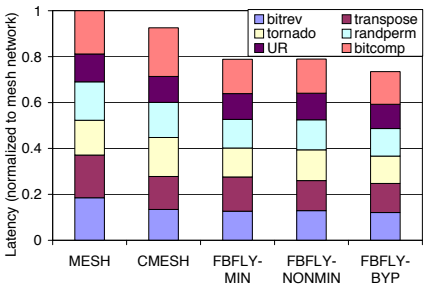

Figure 8. Latency comparison of alternative topologies across different synthetic traffic pattern.

of the network. By effectively utilizing non-minimal routing and smaller diameter of the topology, FBFLY can provide up to 50% increase in throughput compared to CMESH while provide lower zero-load latency. Although the MESH can provide higher throughput for some traffic pattern, it has been previously shown that CMESH results in a more cost- and energy-efficient topology compared to the MESH [3].

5.1.2 Synthetic Batch Traffic

In addition to the throughput measurement, we use a batch experiment to model the memory coherence traffic of a shared memory multiprocessor. Each processor executes a fixed number of remote memory operations (e.g. remote cache line read/write requests) during the simulation and we record the time required for all operations to complete. Read requests and write acknowledgements are mapped into 64-bit messages, while read replies and write requests are mapped into 576-bit messages. Each processor may have up to four outstanding memory operations. The synthetic traffic pattern used are uniform random (UR), bit complement, transpose, tornado, a random permutation, and bit reverse [11].

Figure 8 shows the performance comparison for the batch experiment and we normalize the latency to the mesh network. CMESH reduces latency, compared to the MESH, by 10% but the flattened butterfly reduces the latency further. By using FBFLY-NONMIN, the latency can actually *increase* because of the extra latency incurred with the non-minimal routing. However, the FBFLY-BYP provides the benefit of non-minimal routing but reducing the latency as all packets take minimal physical path and results in approximately 28% latency reduction, compared to the MESH network.

In addition to the latency required to complete the batch job, we also plot the variance of the completion time of the 64 nodes. A histogram is shown in Figure 9 that collects the completion time for *each* processing node. With the flattened butterfly, because of the lower diameter, the completion time has much more tighter distribution and smaller variance in the completion time across all of the nodes. Less variance can reduce the global synchronization time in chip multiprocessor systems. The CMESH, because it is not a symmetric topology, leads to an unbalanced distribution of completion time.

5.1.3 Multiprocessor Traces

Network traces were collected from an instrumented version of a 64-processor directory based Transactional Coherence and Consistency (TCC) multiprocessor simulator [6]. In addition to capturing the traffic patterns and message distributions, we record detailed protocol information so that we can infer dependencies between messages and identify sequences of interdependent communication and computation phases at each processor node. This improves the accuracy of the performance measured when the traces are replayed through a cycle-accurate network simulator. When capturing the traces, we use an idealized interconnection network model which provides instantaneous message delivery to avoid adversely biasing the traces. The traffic injection process uses the recorded protocol information to reconstruct the state of each processor node. Essentially, each processor is modelled as being in one of two states: an active computing state, during which it progresses towards its next communication event; or, an idle communicating state, in which it is stalled waiting for an outstanding communication re-

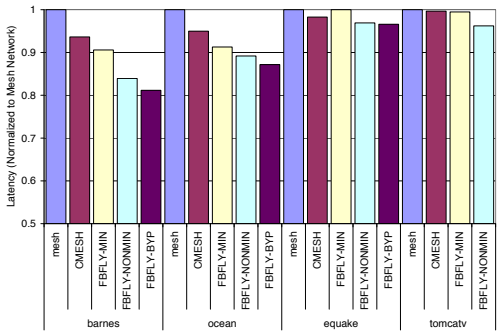

Figure 10. Performance comparison from SPLASH benchmark traces generated from a distributed TCC simulator.

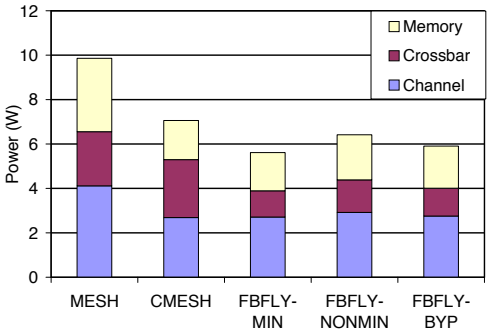

Figure 11. Power consumption comparison of alternative topologies on UR traffic.

quest to complete. Incoming communication events which do not require processor intervention, such as a remote read request, are assumed to be handled by the memory controller and therefore do not interrupt the receiving processor.

Four benchmarks (barnes, ocean, equake, and tomcatv) from the SPLASH benchmarks [26] were used to evaluate the alternative topologies and the results are shown in Figure 10. For two of the benchmarks (equake, tomcatv), the flattened butterfly on-chip network provides less than 5% reduction in latency. However, for the other two benchmarks (barnes, ocean), the flattened butterfly can provide up to 20% reduction in latency.

5.2 Power Comparison

The power consumption comparison is shown in Figure 11. The flattened butterfly provides additional power saving, compared to the CMESH. With the reduction in the width of the datapath, the power consumption of the crossbar is also reduced – thus, achieving approximately 38% power reduction compared to the mesh network.

The area of a router tends to increase with its radix. Control logic, such as the allocators, consumes area proportional to the radix of the router; however, it represents a small fraction of the aggregate area. Consequently, the buffers and switch dominate the router area. The buffer area can be kept constant as the radix increases by reducing the buffer space allocated per input port. The total switch area can be approximated as $n(bk)^2$ where n is the number of routers, b is the bandwidth per port, and k is the router radix. As k increases, b decreases because the bisection bandwidth is held constant, and n also decreases, because each router services more processors. Consequently, we expect high-radix on-chip network will consume less area. We estimate that the flattened butterfly provides an area reduction of approximately 4x compared to the conventional mesh network and a reduction of 2.5x compared to the concentrated mesh. [3] Although the introduction of the bypass muxes can increase the area as well as the power, the impact is negligible compare to the area and power consumption of the buffers and the channels.

6 Discussion

In this section we describe how the flattened butterfly topology can be scaled as more processors are integrated on chip. We also describe how the long direct links used in the flattened butterfly topology are likely to benefit from advances in on-chip signalling techniques, and how these direct links provide some of the performance benefits traditionally provided by virtual channels.

6.1 Comparison to Generalized Hypercube

The flattened butterfly topology is similar to the generalized hypercube [4] but the main difference is the use of concentration of in the flattened butterfly. The use of concentration significantly reduces the wiring complexity because the resulting network requires fewer channels to connect the routers. Furthermore, it is often possible to provide wider channels when concentration is used, because there are fewer channels competing for limited wire resources, which improves the typical serialization latency. With the embedding of the topology into a planar VLSI layout constraint for on-chip networks, as the number of channels crossing a particular bisection increases, the total amount of bandwidth needs to be divided among a larger number of channels – thus, decreasing the amount of bandwidth per channel.

A layout of a conventional mesh network and the 2-D flattened butterfly is shown in Figure 12(a,b). Although the flattened butterfly increases the number of channels crossing neighboring routers in the middle by a factor of 4, the use of

[3] Although there is a radix increase from radix-8 to radix-10 comparing CMESH to the flattened butterfly, since b is reduced in half, there is an overall decrease in total area.

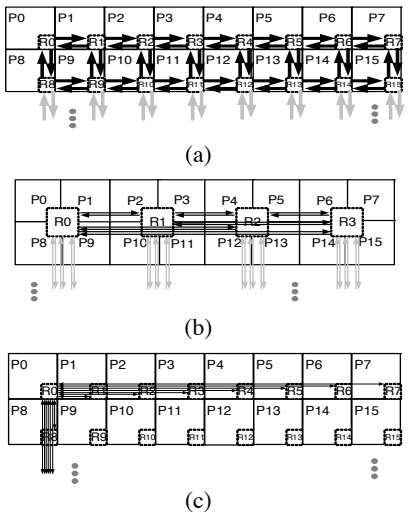

Figure 12. Layout of 64-node on-chip networks, illustrating the connections for the top two rows of nodes and routers for (a) a conventional 2-D mesh network, (b) 2-D flattened butterfly, and (c) a generalized hypercube. Because of the complexity, the channels connected to only R0 are shown for the generalized hypercube.

concentration allows the two rows of wire bandwidth to be combined – thus, resulting in a reduction of bandwidth per channel by only a factor of 2. However, the generalized hypercube (Figure 12(c)) topology would increase the number of channels in the bisection of the network by a factor of 16, which would adversely impact the serialization latency and the overall latency of the network.

6.2 Scaling On-Chip Flattened Butterfly

The flattened butterfly in on-chip networks can be scaled to accommodate more nodes in different ways. One method of scaling the topology is to increase the concentration factor. For example, the concentration factor can be increased from 4 to 8 to increase the number of nodes in the network from 64 to 128 as shown in Figure 13(a). This further increases the radix of the router to radix-14. With this approach, the bandwidth of the inter-router channels needs to be properly adjusted such that there is sufficient bandwidth to support the terminal bandwidth.

Another scaling methodology is to increase the dimension of the flattened butterfly. For example, the dimension can be increased from a 2-D flattened butterfly to a 3-D flattened butterfly and provide an on-chip network with up to 256 nodes as shown in Figure 13(b). To scale a larger number of nodes, both the concentration factor as well as the number of dimensions can be increased as well.

However, as mentioned in Section 6.1, as the number of channels crossing the bisection increase, reduction in bandwidth per channel and increased serialization latency be-

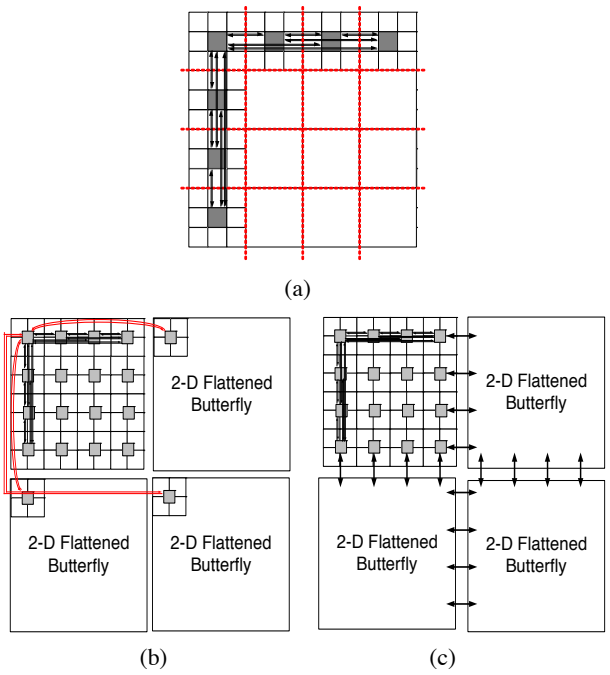

Figure 13. Different methods to scale the on-chip flattened butterfly by (a) increasing the concentration factor, (b) increasing the dimension of the flattened butterfly, and (c) using a hybrid approach to scaling.

come problematic. To overcome this, an hybrid approach can be used to scale the on-chip flattened butterfly. One such possibility is shown in Figure 13(c) where the 2-D flattened butterfly is used locally and the cluster of 2-D flattened butterfly is connected with a mesh network. This reduces the number channels crossing the bisection and minimizes the impact of narrower channels at the expense of slightly increase in the average hop count (compared to a pure flattened butterfly).

6.3 Future Technologies

The use of high-radix routers in on-chip networks introduces long wires. The analysis in this work assumed optimally repeated wires to mitigate the impact of long wires and utilized pipelined buffers for multicycle wire delays. However, many evolving technology will impact communication in future on-chip networks and the longer wires in the on-chip flattened butterfly topology are suitable to exploit these technologies. For example, on-chip optical signalling [17] and on-chip high-speed signalling [13] attempt to provide signal propagation velocities that are close to the speed of light while providing higher bandwidth for on-chip communications. With a conventional 2D mesh topology, all of the wires are relatively short and because of the overhead involved in these technologies, low-radix topologies can not exploit their benefits. However, for the on-chip flattened

butterfly topology which contains both short wires as well as long wires, the topology can take advantage of the cheap electrical wires for the short channels while exploiting these new technologies for the long channels.

6.4 Use of Virtual Channels

Virtual channels (VCs) were originally used to break deadlocks [9] and were also proposed to increase network performance by provide multiple virtual lanes for a single physical channel [8]. When multiple output VCs can be assigned to an input VC, a VC *allocation* is required which can significantly impact the latency and the area of a router. For example, in the TRIPS on-chip network, the VC allocation consumes 27% of the cycle time [12] and an area analysis show that VC allocation can occupy up to 35% of the total area for an on-chip network router [20].

In Figure 14(a), an example of how blocking can occur in conventional network with wormhole flow control is shown. By utilizing virtual channel flow control (Figure 14(b)), buffers are partitioned into multiple VCs, which allow packets to pass blocked packets. However, with the use of high-radix routers and the flattened butterfly topology, the additional wire resources available can be used to overcome the blocking that can occur as shown in Figure 14(c). As a result, blocking is reduced to only packets originating from the same source router and destined to the routers in the same column of the network. Thus, the topology reduces the need for the buffers to be partitioned into multiple VCs and takes advantage of the abundant wiring available in an on-chip network. VCs for other usage such as separating traffic into different classes might still be needed but such usage does not require VC allocation.

Figure 15 shows how the performance of the flattened butterfly is changed by increasing the number of VCs. In the simulation comparison, the total amount of storage per physical channel is held constant – thus, as the number of VCs is increased, the amount of buffering per VC is decreased. As a result, increasing the number of VCs for an on-chip flattened butterfly can slightly degrade the performance of the network since the amount of buffering per VC is reduced.

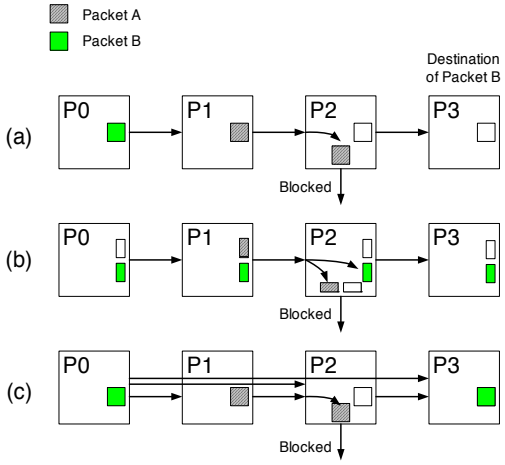

Figure 14. Block diagram of packet blocking in (a) wormhole flow control (b) virtual channel flow control and (c) flattened butterfly.

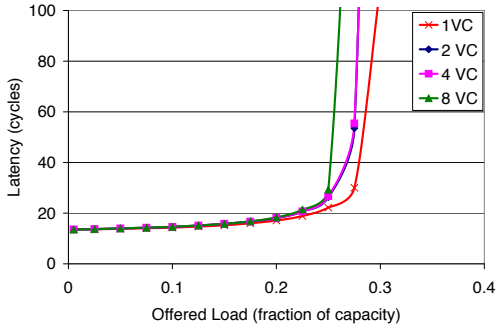

Figure 15. Performance comparison as the number of virtual channels is increased in the flattened butterfly.

7 Related Work

Most on-chip networks that have been proposed are low-radix, mostly utilizing a 2D mesh or a torus network [10]. Balfour and Dally proposed using concentrated mesh and express channels in on-chip networks to reduce the diameter and energy of the network [3]. This work expands on their idea and provides a symmetric topology that further reduces latency and energy. In addition, the benefits of creating parallel subnetworks [3] can also be applied to the flattened butterfly topology in on-chip networks.

Kim et al. [16] showed that the increasing pin bandwidth can be exploited with high-radix routers to achieve lower cost and lower latency in off-chip interconnection networks. Although on-chip networks have very different constraints compared to off-chip networks, we showed how the flattened butterfly topology [15] proposed for high-radix off-chip interconnection networks can also be applied to on-chip networks. Kumar et al. [18] proposed the use of express virtual channels (EVC) to reduce the latency of 2-D mesh on-chip network by bypassing intermediate routers. However, EVC requires sharing the bandwidth between EVC and non-EVC packets in a 2-D mesh network. Both the EVC and the flattened butterfly topology share similar objective of trying to achieve *ideal* latency in on-chip networks but they differ in the approach – EVC is a flow control while the flattened butterfly is a topology.

The yield arbiter described in Section 4.2 is similar in concept to the flit-reservation flow control (FRFC) [21]. In FRFC, a control flit is sent ahead of the data flits and reserves

the buffers and channels for the ensuing data flits. However, the scheme described in this paper uses the control flit to ensure bandwidth for the data flits in the worst-case scenario (i.e. very congested network) – otherwise, the bypass channel is used regardless of the control flit.

The scaling of the topology with an hybrid (mesh) approach at the top level has been proposed for off-chip interconnection networks to reduce the length of the global cables [27]. Similar scaling can also be applied for on-chip networks as shown in Section 6.2 with the benefits being not just shorter wires but also reduced on-chip wiring complexity.

8 Conclusion

In this paper, we described how high-radix routers and the flattened butterfly topology can be utilized in on-chip networks to realize reduction in latency and power. By reducing the number of routers and channels in the network, it results in a more efficient network with lower latency and lower energy consumption. In addition, we describe the utilization of bypass channels to utilize non-minimal routing with minimal increase in power while further reducing latency in the on-chip network. We show that the flattened butterfly can increase throughput by up to 50% compared to the concentrated mesh and reduce latency by 28% while reducing the power consumption by 38% compared to a mesh network.

Acknowledgments

The authors would like to thank the anonymous reviewers for their insightful comments. This work has been supported in part by the National Science Foundation under Contract CCF070234, in part by Cray, Inc. under Contract CR03-C-0002, in part by the Semiconductor Research Corporation under Contract SRC2007-HJ-1591, and in part by a Cadence Design Systems Stanford Graduate Fellowship.

References

[1] P. Abad, V. Puente, J. A. Gregorio, and P. Prieto. Rotary Router: An Efficient Architecture for CMP Interconnection Networks. In *Proc. of the International Symposium on Computer Architecture (ISCA)*, pages 116–125, San Diego, CA, June 2007.

[2] A. Agarwal et al. Tile Processor: Embedded Multicore for Networking and Multimedia. In *Hot Chips 19*, Stanford, CA, Aug. 2007.

[3] J. Balfour and W. J. Dally. Design tradeoffs for tiled cmp on-chip networks. In *Proc. of the International Conference on Supercomputing (ICS)*, pages 187–198, 2006.

[4] L. N. Bhuyan and D. P. Agrawal. Generalized hypercube and hyperbus structures for a computer network. *IEEE Trans. Computers*, 33(4):323–333, 1984.

[5] T. Bjerregaard and S. Mahadevan. A survey of research and practices of network-on-chip. *ACM Comput. Surv.*, 38(1):1, 2006.

[6] H. Chafi, J. Casper, B. D. Carlstrom, A. McDonald, C. Cao Minh, W. Baek, C. Kozyrakis, and K. Olukotun. A scalable, non-blocking approach to transactional memory. In *13th International Symposium on High Performance Computer Architecture (HPCA)*. Feb 2007.

[7] S. Cho and L. Jin. Managing Distributed, Shared L2 Caches through OS-Level Page Allocation. In *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 455–468, Orlando, FL, 2006.

[8] W. J. Dally. Virtual-channel Flow Control. *IEEE Transactions on Parallel and Distributed Systems*, 3(2):194–205, 1992.

[9] W. J. Dally and C. L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers*, 36(5):547–553, 1987.

[10] W. J. Dally and B. Towles. Route Packets, Not Wires: On-Chip Inteconnection Networks. In *Proc. of the 38th conference on Design Automation (DAC)*, pages 684–689, 2001.

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

[12] P. Gratz, C. Kim, R. McDonald, S. Keckler, and D. Burger. Implementation and Evaluation of On-Chip Network Architectures. In *International Conference on Computer Design (ICCD)*, 2006.

[13] A. Jose, G. Patounakis, and K. Shepard. Near speed-of-light on-chip interconnects using pulsed current-mode signalling. In *Digest of Technical Papers. 2005 Symposium on VLSI Circuits*, pages 108–111, 2005.

[14] J. Kim, J. Balfour, and W. J. Dally. Flattened butterfly for on-chip networks. *IEEE Computer Architecture Letters*, July 2007.

[15] J. Kim, W. J. Dally, and D. Abts. Flattened Butterfly : A Cost-Efficient Topology for High-Radix Networks. In *Proc. of the International Symposium on Computer Architecture (ISCA)*, pages 126–137, San Diego, CA, June 2007.

[16] J. Kim, W. J. Dally, B. Towles, and A. K. Gupta. Microarchitecture of a High-Radix Router. In *Proc. of the International Symposium on Computer Architecture (ISCA)*, pages 420–431, Madison, WI, 2005.

[17] N. Kirman et al. Leveraging optical technology in future bus-based chip multiprocessors. In *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 492–503, Orlando, FL, 2006.

[18] A. Kumar, L.-S. Peh, P. Kundu, and N. K. Jhay. Express Virtual Channels: Towards the Ideal Interconnection Fabric. In *Proc. of the International Symposium on Computer Architecture (ISCA)*, pages 150–161, San Diego, CA, June 2007.

[19] R. D. Mullins, A. West, and S. W. Moore. Low-Latency Virtual-Channel Routers for On-Chip Networks. In *Proc. of the International Symposium on Computer Architecture (ISCA)*, pages 188–197, Munich, Germany, 2004.

[20] C. A. Nicopoulos, D. Park, J. Kim, N. Vijaykrishnan, M. S. Yousif, and C. R. Das. ViChaR: A Dynamic Virtual Channel Regulator for Network-on-Chip Routers. In *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Orlando, FL, 2006.

[21] L.-S. Peh and W. J. Dally. Flit-reservation flow control. In *International Symposium on High-Performance Computer Architecture (HPCA)*, pages 73–84, 2000.

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

[23] A. Singh. *Load-Balanced Routing in Interconnection Networks*. PhD thesis, Stanford University, 2005.

[24] M. B. Taylor, W. Lee, S. Amarasinghe, and A. Agarwal. Scalar Operand Networks: On-Chip Interconnect for ILP in Partitioned Architectures. In *International Symposium on High-Performance Computer Architecture (HPCA)*, pages 341–353, Anaheim, California, 2003.

[25] S. Vangal et al. An 80-Tile 1.28TFLOPS Network-on-Chip in 65nm CMOS. In *IEEE Int'l Solid-State Circuits Conf., Digest of Tech. Papers (ISSCC)*, 2007.

[26] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. The SPLASH-2 programs: Characterization and methodological considerations. In *Proc. of the International Symposium on Computer Architecture (ISCA)*, pages 24–36, Santa Margherita Ligure, Italy, 1995.

[27] M. Woodacre. Towards Multi-Paradigm Computing at SGI. Talk at Stanford University EE380 seminar, Jan. 2006.