Exploring Concentration and Channel Slicing in On-chip Network Router

Prabhat Kumar, Yan Pan, John Kim[†], Gokhan Memik, Alok Choudhary
Northwestern University
2145 Sheridan Road, Evanston, IL
{prabhat-kumar, panyan,memik,alok}@northwestern.edu

[†]KAIST
Daejeon, Korea
jjk12@cs.kaist.ac.kr

Abstract

Sharing on-chip network resources efficiently is critical in the design of a cost-efficient network on-chip (NoC). Concentration has been proposed for on-chip networks but the trade-off in concentration implementation and performance has not been well understood. In this paper, we describe cost-efficient implementations of concentration and show how external *concentration provides a significant reduction in complexity (*47% *and* 36% *reduction in area and energy, respectively) compared to previous assumed integrated (high-radix) concentration while degrading overall performance by only* 10%*. Hybrid implementations of concentration is also presented which provide additional trade-off between complexity and performance. To further reduce the cost of NoC, we describe how channel slicing can be used together with concentration. We propose* virtual *concentration which further reduces the complexity – saving area and energy by 69% and 32% compared to baseline mesh and 88% and 35% over baseline concentrated mesh.*

1 Introduction

Increasing number of transistors in modern VLSI technology has increased the number of cores on a chip and is making chip multiprocessors (CMP) widely available. As a result, on-chip communication is becoming critical and requires efficient Network-on-chip (NoC) designs where data are routed in packets on shared channels instead of dedicated buses [16, 19]. Designing cost-efficient on-chip networks will require efficient sharing of on-chip network resources such as buffers and wire bandwidth. In this paper, we explore two techniques, concentration and channel slicing [8], in on-chip networks to create a cost-efficient on-chip network architecture. Concentration and slicing are well-known techniques in interconnection networks that have been adopted in different off-chip networks such as the Cray X1 [1] and the Cray BlackWidow network [24]. However, their use in on-chip networks have not been well explored.

Concentration in on-chip networks was first proposed by Balfour and Dally [4]. Recently proposed on-chip network topologies such as the hybrid topology [10], flattened butterfly [14], hierarchical firefly [22], and the multidrop express channels topology [12] have also used concentration. However, the trade-off in implementing concentration is not well understood and little work has been done to evaluate the optimal degree of resource sharing. Most prior work in on-chip network concentration have also assumed an increase in router radix to support concentration which increases the complexity and the cost of the on-chip network routers. In this work, we present an alternative *external* concentration which significantly reduces concentration complexity and evaluate performance impact of alternative concentration.

However, concentration can lead to performance degradation because of the channel sharing between multiple terminal nodes. As a result, we also explore the use of channel slicing to increase utilization of the wiring resources and increase performance. We propose how *virtual concentration* – where concentration and channel slicing are combined to further reduce the cost of the network – can be implemented. We hold the on-chip resources (buffers and bisection bandwidth) constant in our evaluation to provide a better understanding of the optimal resource utilization scheme. Our results show that using channel slicing factor of 4 can provide a concentrated mesh which matches the throughput of a conventional mesh while providing 69% and 32% reduction in area and energy cost, respectively.

In this work, we focus on evaluating concentration and channel slicing on a 2D mesh topology since it is a commonly used topology for NoC [23, 26, 25] and maps well to a 2D VLSI planar layout. However, the techniques presented can be easily extended to other topologies which use concentration [12, 10, 14, 22]. In addition, other techniques to increase the performance of on-chip networks such as express virtual-channel flow control [20], token flow control [6], on-chip data compression [11], micro-architectural techniques [15, 9], and load-balanced routing algorithm [5],

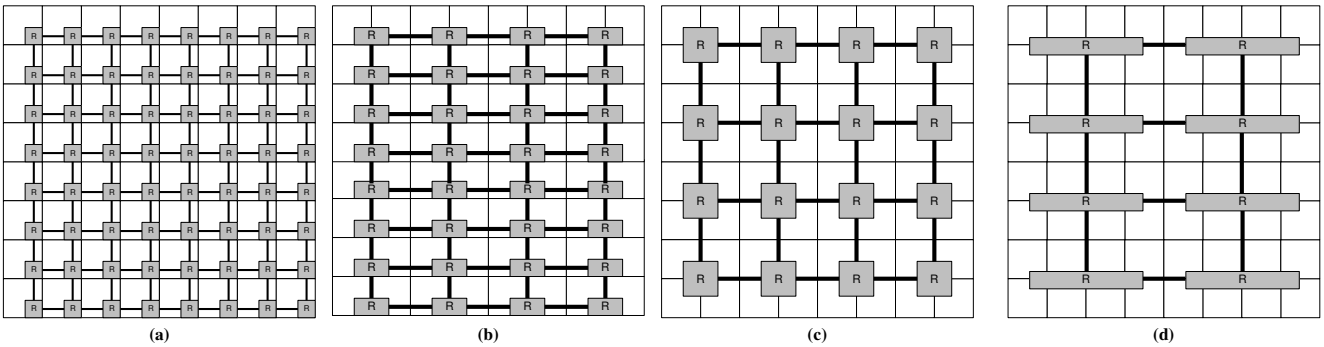

Figure 1. 64-node on-chip network using (a) conventional 2D mesh, (b) concentrated mesh with $C = 2$ ($C_x = 2, C_y = 1$), (c) $C = 4$ ($C_x = C_y = 2$) and (d) $C = 8$ ($C_x = 4, C_y = 2$)

can all be used in conjunction with the proposed techniques of concentration and channel slicing to reduce network cost.

In summary, the contribution of the work includes:

- Exploration of the performance implication of varied degrees of concentration.
- Exploration of the performance and efficiency implication of *integrated* and *external* concentration.
- Virtual concentration techniques, combining concentration with channel slicing, for efficient resource sharing.

The rest of the paper is organized as follows. In Section 2, we provide an overview of using concentration in on-chip networks and describe alternative concentration implementation. In Section 3, we describe how channel slicing can be implemented on a concentrated mesh topology. We describe the simulation setup in Section 4 and present the results and discussion in Section 5. Section 6 presents related work and we conclude in Section 7.

2 Concentration

2.1 Overview

Conventional 2D mesh network does not implement any concentration as each terminal node is connected to a single router (Figure 1(a)). However, the routers and the channels can be shared to create a concentrated mesh (CMESH). The degree of concentration in a 2D mesh network can be characterized with the following parameters:

- C_x : concentration in the x-dimension
- C_y : concentration in the y-dimension
- C : topology concentration ($= C_x \times C_y$)

The total degree of concentration in the topology is the product of the concentration in the two dimensions. With these parameters, a conventional $2D$ mesh topology can be described as $C = C_x = C_y = 1$. As C increases and approaches $\sqrt{N}$ where N is the network size, the resulting topology is a fully-connected or a crossbar network. Different values of C are shown in Figure 1. For example, with a concentration factor of 4 with $C_x = 2$ and $C_y = 2$ (Figure 1(c)), 2 terminal nodes in the x-dimension and 2 in the y-dimension share a single router resulting in $C = 4$ – four terminals sharing a single router and the inter-router channels connected to the router. The benefits of using concentration include:

- *Sharing of resource (routers and channels)*: By combining two rows of channels into a single channel (e.g. $C_y = 2$), the x-dimension channels of CMESH can have $2\times$ bandwidth compared to the channels of a mesh.
- *Reduction of network diameter*: Concentration reduces the number of intermediate routers and thus, reduces the hop count and zero-load latency.
- *Reduction in the cost of local communication*: All communication in a mesh network requires traversing at least two routers but communication with $(C - 1)$ terminal nodes require accessing only a single router in a CMESH.

However, concentration presents some disadvantages which include:

- *Router Complexity*: Additional router ports increase router complexity such as switch and virtual channel allocators and leads to higher per-hop router latency.
- *Area*: Crossbar area is quadratically proportional to data path width and wider channels adversely impact the crossbar area.
- *Performance*: Increase in the number of ports creates higher probability of contention which can reduce performance.

In this work, we explore alternative concentration implementations to create an efficient on-chip network architecture that provides the benefits of concentration yet achieves while minimizing the cost of implementing concentration.

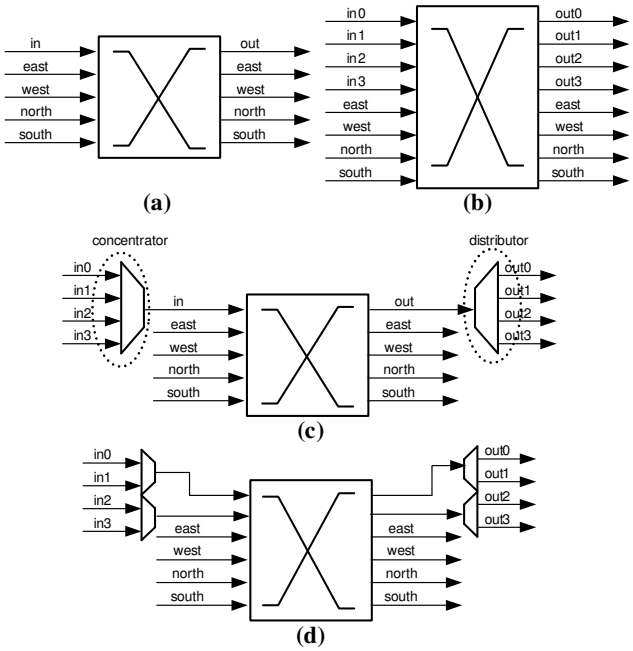

Figure 2. Router block diagram of (a) conventional 2D mesh topology and $C = 4$ CMESH router micro-architecture using (b) integrated concentration, (c) external concentration and (d) hybrid implementation using $M2D2$.

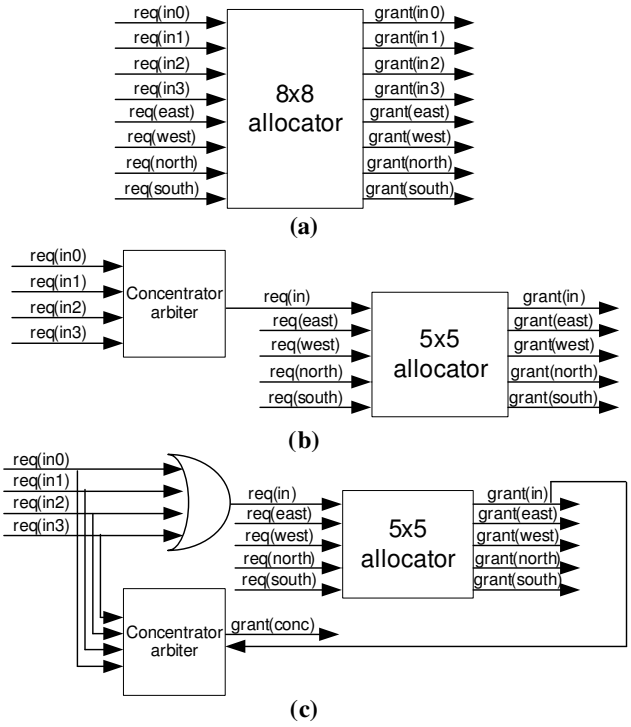

Figure 3. Allocator diagram of router with an (a) integrated concentrator and (b) external concentrator using serial allocator and (c) parallel allocator

2.2 Concentration Implementation

Prior work on concentration in on-chip networks have implemented concentration by increasing the router radix [4, 12, 14]. We refer to this approach as *integrated* concentration as the concentrator is integrated into the router by increasing the router radix. Integrated concentration is shown in Figure 2(b) for $C = 4$ and results in a radix-8 router, compared to a radix-5 router for a conventional 2D mesh router (Figure 2(a)). In this work, we evaluate an alternative concentration implementation using *external* concentration (Figure 2(c)). The radix of the router is identical to MESH as terminal nodes connected to the router share a single input to the router with a concentrator (or a multiplexer). Similarly, there is only a single output of the switch and a distributor is added to the router output to route the packets to the appropriate terminal node. Thus, the distributor is the inverse of a concentrator.

The main difference between integrated and external concentration is the amount of switch bandwidth – resulting in a trade-off between router micro-architecture complexity and performance. Integrated concentrator requires a high-radix implementation of on-chip network routers and increases router micro-architecture complexity such as the switch and virtual channel allocators and the crossbar. External concentration reduces the router complexity but can create additional contention with the external concentrator. In addition, it requires an external input arbitration and if not done properly, can degrade the performance of an on-chip network router.

In addition to these two implementations, we also evaluate hybrid concentration implementations by varying the number of concentrators and distributors used. For example, in Figure 2(d), two concentrators and two distributors are used to create a radix-6 router with two terminal nodes sharing a single concentrator instead of 4 terminals as in Figure 2(c). To describe the hybrid implementation, we use $MxDy$ notation where x describes the number of multiplexers (mux) or additional input ports added and y describes the number of demultiplexers (demux) or output ports added. The number of mux inputs and the demux outputs will be C/x and C/y respectively. When x or y equal C, the number of mux inputs or demux outputs will be one – thus, no additional mux or demux logic is needed and only the number of ports on the router is increased. Using this notation, integrated concentration shown in Figure 2(b) is $M4D4$ while the external concentration is $M1D1$. The x and y parameters do not have to be equal. For example, $M1D4$ micro-architecture results in a single concentrator but provides output speedup while a $M4D1$ micro-architecture provides input speedup.

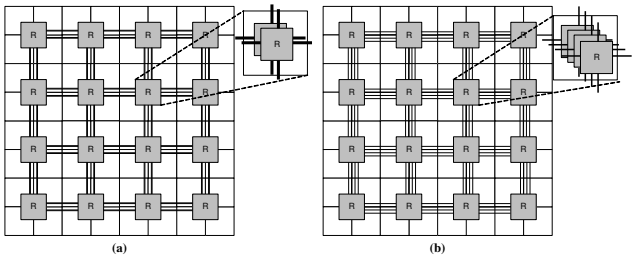

Figure 4. Concentrated mesh topology with channel slicing (a) $S = 2$ and (b) $S = 4$. Each router is composed of S parallel router.

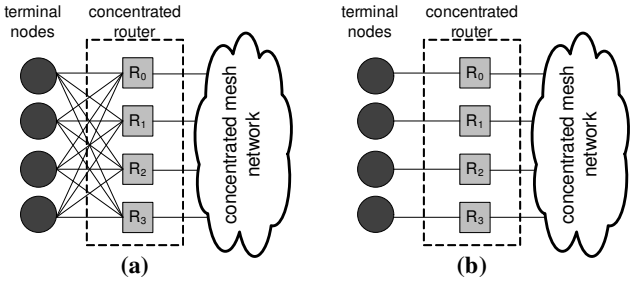

Figure 5. Channel-sliced network using (a) full connection and (b) virtual concentration.

2.3 Parallel Arbitration

A critical aspect of external concentration is the additional arbitration required at the input. Integrated concentration requires a single allocator for all the inputs (Figure 3(a)). However, two-stage allocation is needed for external concentration – an input arbitration among the inputs to the concentrator mux and the switch allocation. If these two arbitrations are done sequentially (Figure 3(b)), this not only adds router latency but also create unnecessary head-of-line (HoL) blocking since the state of the router switch information is not used in the input MUX arbitration. For example, if *in0* win the concentrator mux arbitration, it can cause HoL blocking as it waits for the output resource to become available. In order to avoid these problems, we implement a parallel arbitration scheme that parallelizes the input (concentrator) arbitration with the router switch arbitration (Figure 3(c)). If there is one or more packet among the injection port, a request is sent to the router switch arbitration while the input arbitration is done in parallel. Prior to asserting this request, the output resource is checked to determine the availability of the output virtual channel. This technique cannot be applied to the serial allocation in Figure 3(b) since it requires multiple pipeline stages. To ensure that there is no starvation, the priority pointer in the concentrator arbitration is not updated unless the output of the mux is also granted from the switch allocation similar to how pointers are updated in islip allocation algorithm [19].

3 Channel Slicing

Concentration results in inter-router channels being shared and results in inter-router channels that are wider than the conventional 2D mesh topology. However, wider channels result in poor channel utilization since some packets in on-chip network are small [10]. For example, while cache lines can exceed 512 – 1024 bits in width, request packets, control packets, or coherency messages are much narrower. In addition, recent work has shown that frequent data patterns exist in on-chip network traffic which can be compressed into a narrower data-path [11].

To provide an efficient usage of wire resource, we evaluate channel slicing [8] in on-chip networks. Channel slicing divides the wide data-path channel into multiple, narrower channels. Examples of a channel-sliced concentrated mesh is shown in Figure 4 for $S = 2$ and $S = 4$, where S is the slicing factor. The sliced network consists of S parallel concentrated mesh with each network consisting of channels that are reduced in width by $1/S$ compared to an unsliced ($S = 1$) network. Thus, each router in Figure 4 is composed of S parallel routers. Channel slicing results in higher serialization latency with narrower channels and results in increased zero-load latency for long packets. However, for short packets, there is no impact on zero-load latency and providing multiple channels between neighboring routers increases the utilization of the wiring resources.

In implementing channel slicing on top of concentration, the C terminal nodes can be connected to S parallel routers as shown in Figure 5(a) with $C = 4$ and $S = 4$. Using external concentration described in Section 2 the router complexity can be reduced but wiring complexity becomes problematic with the wiring to the S routers from the C terminal nodes. This approach also requires careful load-balancing in selecting the slice as if not done properly, one or more of the slices can become the bottleneck. To overcome these problems, we introduce *virtual* concentration (Figure 5(b)). Each physical router in the network ($R0$ to $R3$) is only connected to a single terminal node to minimize wiring. Thus, each terminal node in the sliced network has access to its own dedicated router without sharing the input bandwidth. Although the individual routers no longer implement concentration, the collection of routers create virtual concentration. The outputs of the routers still need to be routed to all C terminal nodes to ensure that regardless of which slice the packet is injected into, the packet can be routed to its destination. This requires either an integrated distributor or an external distributor as described in Section 2. The different channel-sliced architecture can be described as $SaRb$, where a is the number of slices and b is the number of injecting nodes connected to each router. For example, the

Table 1. Architectural configuration

Code Name	# Slices	Inj Node	Channel Width	Buffer Depth	Router Latency for MxDy		
MESH (C=1)	1	1	0.5w	0.6x	M1D1	M1D2	M1D4
CMESH (C=2)	1	2	0.5w	1.0x	2	2	3
CMESH (C=8)	1	8	1w	1.0x			
S1R4MxDy	1	4	1w	0.75x	M2D1	M2D2	M2D4
S2R4MxDy	2	4	0.5w	0.75x	2	2	3
S2R2MxDy	2	2	0.5w	1.0x			
S4R4MxDy	4	4	0.25w	0.75x	M4D1	M4D2	M4D4
S4R2MxDy	4	2	0.25w	1x	3	3	3
S4R1MxDy	4	1	0.25w	1.2x			

architecture shown in Figure 5(a) is $S4R4$ and virtual concentration in Figure 5(b) is described as $S4R1$.

4 Evaluation Setup

4.1 Simulation Environment

A cycle accurate network simulator is developed based on the booksim simulator [8, 4] and modified to represent the alternative architectures evaluated. Channel traversal time is modeled as 1 cycle between neighboring routers for baseline mesh topology. For CMESH topologies, the channel traversal time is modeled as C_x and C_y cycles in the respective dimensions. [1]

For example, with $C = 8$ ($C_x = 4, C_y = 2$), the x-dimension link latency is 4 cycles, while the y-dimension link latency is 2 cycles. Table 1 summarizes the architectural configuration parameters. The alternative architectures are describe as $SaRbMxDy$ with $SaRb$ describing the channel slicing (Section 3) and $MxDy$ describing the concentration (Section 2.2). With this notation, $x \leq b$ since the number of additional injections ports x can not exceed b.

To provide a fair comparison, we hold the on-chip resources (wires and buffers) constant in comparing alternative architectures. We assume constant bisection bandwidth across the different architectures and hence, the channel width and flit size vary based on number of bisection channels. We use w as the width of the flit and channel for $C = 8$ architecture and the other architectures' channel width are described in terms of w. However, to keep the bandwidth of all the channels constant for a given architecture, the bisection of $C = 2$ and $C = 8$ are different. This would results in an asymmetric router design. We also keep the total amount of storage constant. Thus, buffer depth varies depending on the number of ports/VCs needed as shown in Table 1. The router latency of high-radix switches is increased to account for additional complexity.

We compare the alternative architectures using both latency/throughput and synthetic workload. For the latency/throughput comparisons, we plot the injection rate in terms of w bits/cycle to provide a fair comparison of the alternative architectures. The synthetic workload models the memory coherence traffic of a shared memory with each processor generating 10K remote memory requests. Once the requests are received, responses are generated and the total execution time is measured. We allow 8 outstanding requests per router to model the effect of MSHRs – thus, when 8 outstanding requests are injected into the network, new requests are blocked from entering the network until response packets are received. Traces from SPLASH2 [27] and MineBench [21] applications are also used in the evaluation. Traces are generated using GEMS [17] simulator with the Garnet [3] on-chip network model using a mesh topology, 4-stage pipeline router, and single cycle channel latency. Limited by space, selected representative results are presented in the following section.

[1] The performance of $C = 8$ will be worse than presented here since we do not consider the additional wire delay from the terminal nodes to the router which can not be physically adjacent to all 8 terminal nodes.

Table 2. Network Traffic Pattern Evaluated

Traces	
MineBench	apriori, hop, kmeans, scalparc_flex
Splash	barnes, cholesky, lu, radix, water_spatial
Synthetic Load Details	
Closed Loop Batch Job	10K requests/node, request and reply dependence
Synthetic Traffic Patterns	
Traffic Name	**Details**
Bitcomp	dest = bitwise-not (src)
Uniform	Uniform Random Traffic
NonUR	75% 1-Hop Neighbor Traffic, 25% Uniform Random

5 Results and Discussion

5.1 Impact of Concentration

Latency and saturation throughput are compared as concentration is increased from $C = 1$ (MESH) to $C = 8$ in Figure 6 for different traffic patterns. With concentration, the per hop router latency increases but the overall zero-load latency decreases due to the reduction of average hop count. The decrease in the hop count is proportional to concentration and can be described with the following equation

$$H_{avg} = \frac{k_x/C_x + k_y/C_y}{3}$$

where k_x, k_y represent the dimension of the unconcentrated mesh in the x and the y dimensions. For uniform random traffic, increasing concentration reduces zero-load latency by up to 10% when $C = 4$ and by up to 23% when $C = 8$. There is no change in the zero-load latency as C increases from 2 to 4 because the decrease in hop count is offset by the increase in per-hop latency.

The saturation throughput of $C = 4$ matches that of $C = 1$ and exceeds the throughput of $C = 2$ and $C = 8$ by 79% (Figure 6). The throughput for $C = 2$ and $C = 8$ is degraded by the asymmetry of the network as described earlier in Section 4. For permutation traffic such as bitcomp, $C = 1$ exceeds the throughput of $C = 4$ by 9.5% but for nonUR traffic, $C = 4$ exceeds $C = 1$ by 30% as concentration exploits traffic locality to increase the network throughput.

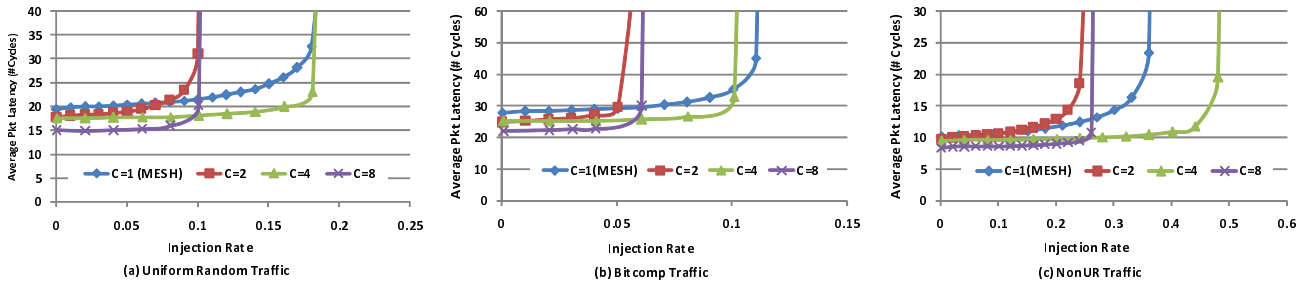

Figure 6. Performance comparison of different concentration

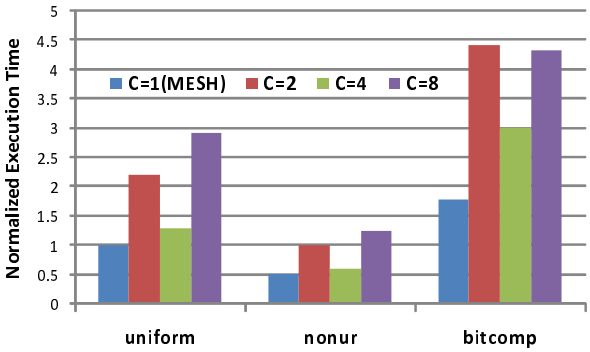

Figure 7. Synthetic workload completion time comparison for different concentrations normalized to that of MESH with UR traffic.

For synthetic workload, MESH with no concentration leads to higher performance, exceeding the performance of $C = 4$ on UR traffic by 22% and by 66% compared to $C = 8$ (Figure 7). Because of the bottleneck in y-dimension, $C = 2$ and $C = 8$ results in poor performance. The performance of $C = 1$ also exceed $C = 4$ by 13% and 41% on nonUR and bitcomp traffic, respectively. With the reduced number of channels as C increases, more contention occurs in the network and results in lower performance. Although $C = 1$ (MESH) provides high performance, as we will show in Section 5, the area/power cost of a MESH is much higher and results in poor efficiency. For the rest of this paper, we focus on $C = 4$ architecture and alternative implementations to improve performance and efficiency. [2]

5.2 Hybrid Implementation of Concentration

The results presented in the previous section assumed an integrated concentration implementation. In this section, we compare alternative concentration implementations including external concentration ($M1D1$), integrated concentration ($M4D4$), and hybrid implementations ($MxDy$). Figure 8 shows the latency vs. load curves for the different implementations and for clarity, only selected curves are shown. As expected, the reduced switch bandwidth of $M1D1$ implementation increases contention in the network and results in approximately 11% reduction in throughput as compared to the integrated implementation ($M4D4$) for UR traffic and approximately 58% with nonUR traffic. As for hybrid implementations, any architecture with a single distributor ($MxD1$) achieve throughput similar to $M1D1$ because of the reduced bandwidth at the ejection port. However, any hybrid implementation with $MxDy$ where $y \geq 2$ achieves saturation throughput similar to $M4D4$ with significant reduction in complexity. For example, $M1D2$ achieve identical saturation throughput to $M4D4$ but results in 21% reduction in zero-load latency.

The saturation throughput for bitcomp traffic does not change with alternative concentration implementation while the zero-load latency is dependent on the router latency (Figure 8(b)). For permutation traffic such as bitcomp, the routers are not the bottleneck and results in the different implementation achieving identical throughput. However, $M1D1$ results in a reduction of zero-load latency by an average of 21% compared to $M4D4$ on the three traffic patterns because of its reduced router complexity.

For nonUR traffic pattern, configurations with a single concentrator or distributor, i.e., $M1Dy$ or $MxD1$, significantly reduces the saturation throughput, by up to 58% compared to $M4D4$. The performance of $M1Dy$ and $MxD1$ achieve very similar throughput and for clarity, only $M1D1$ and $M4D1$ are shown in Figure 8(c). On traffic such as nonUR with significant locality, the throughput of $MxDy$ hybrid implementation is proportional to $min\{x, y\}$ as the concentrated router implementation determines the overall throughput of the network. Thus, the throughput increases by 79% as the distributor is increased from $M4D1$ to $M4D2$ and by 33% as the distributor is increased further from $M4D2$ to $M4D4$.

Synthetic workload execution time is compared in Figure 9. Results show that output speedup (i.e. increase y) can increase performance while providing only input speedup (i.e. increase x) can actually *reduce* performance. For a fixed number of concentrators (x), as the number of distributors (y) increases, the performance improves or at least remains the same for all the three traffic patterns. However,

[2]For 3D, stacked architectures, higher values of C might be suitable but we focus on conventional, 2D architecture.

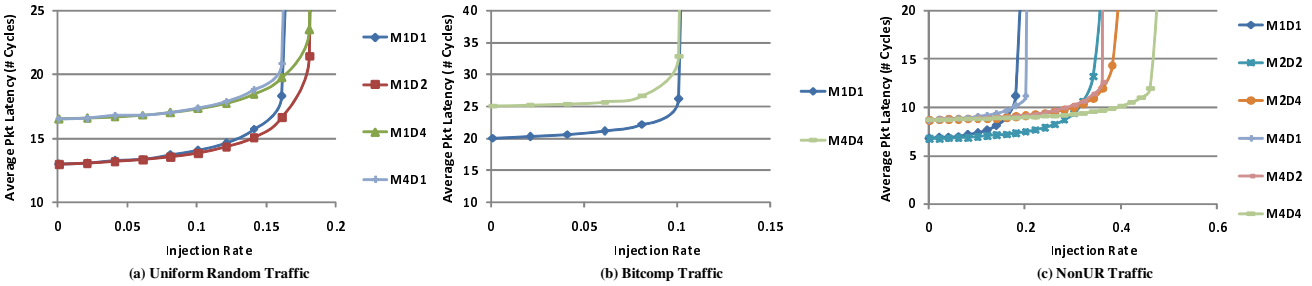

Figure 8. Performance comparison of concentration implementations

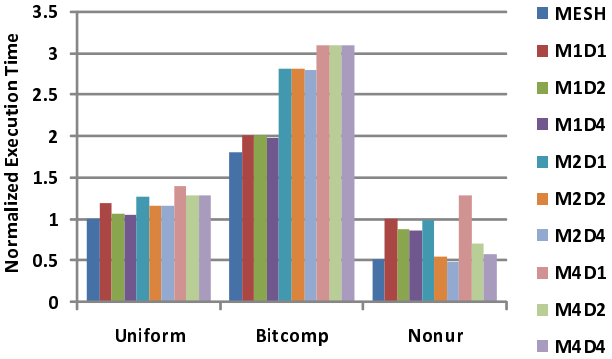

Figure 9. Synthetic workload comparison for alternative $MxDy$ concentration implementations normalized to that of MESH on uniform traffic.

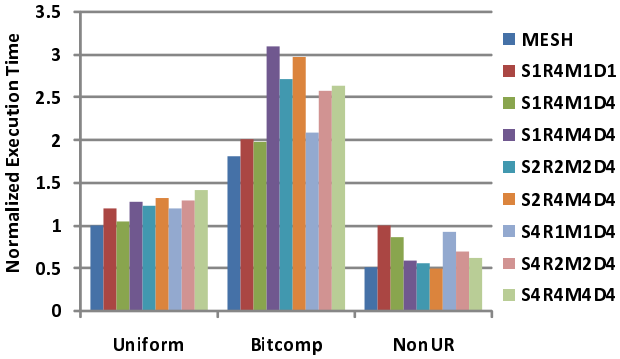

Figure 10. Synthetic workload comparison for alternative channel sliced architectures normalized to MESH under UR.

with fixed y, as x increases, the performance is reduced because of additional input traffic is injected into the network and causes more contention. The performance is reduced by as much as 20% for UR and 53% for bitcomp compared to MESH. Similar to the latency vs. load curve comparison, nonUR traffic follows a different trend because of the traffic locality and increasing x and y improves the performance as the switch bandwidth is increased to handle local traffic.

5.3 Network Slicing

Comparison of alternative configurations using concentration and channel slicing is shown in Figure 11. As we increase the slicing factor (S) to 2 and 4, the zero-load latency is increased by 6% (16%) for $S = 2$ ($S = 4$) under UR. However, sliced configuration ($S4R4MxDy$) achieves similar throughput as $S1R4MxDy$ without any slicing on UR while providing 99% higher throughput on nonUR traffic. As described earlier in Section 4, the use of slicing in addition to concentration creates wiring complexity but virtual concentration ($S4R1$) reduces the complexity. On UR traffic and bitcomp traffic, $S4R1$ exceeds the throughput of $S4R4$ by 6% and 18%, respectively. However, for nonUR, the throughput of $S4R1$ is significantly lower (by 46%) and achieves similar throughput as $S = 1$ configuration since the performance of nonUR is dominated by the router switch bandwidth. [3] For synthetic workloads (Figure 10), $S4R1M1D4$ is within 20% and 15% of MESH on UR and bitcomp traffic but suffers 79% increase in execution time on nonUR traffic.

5.4 Impact of Router Micro-architecture

Since the router micro-architecture can impact the overall performance, we evaluate the impact of virtual channels [7] (VCs) and router speedup on the alternative concentration implementation. As shown in Figure 12, $C = 1$ has very little benefit from increasing the number of VCs. The external concentration ($M1D1$) benefits from increasing VCs by reducing execution time by up to 13% but high-radix router implementation of concentration ($M4D4$) achieve higher benefit as execution time is reduced by up to 24.5%. With larger port count, head-of-line (HoL) blocking [13] limits the router throughput and VCs helps to reduce the impact of HoL. We also vary the router speedup from 1.0× (no speedup) to 1.7× and for UR, benefits are similar to increasing VCs while for bitcomp, there is no benefit from providing router speedup. However, for nonUR traffic (Figure 13), providing router speedup provides significant reduction of execution time – for $M1D1$,

[3] $S4R1M4D4$ configuration is not possible since with virtual concentration, each router is only connected to a single router.

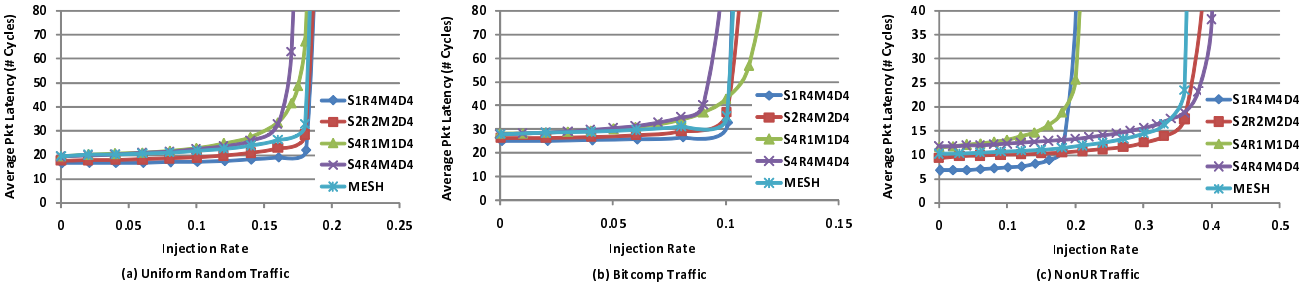

Figure 11. Performance comparison of channel slicing

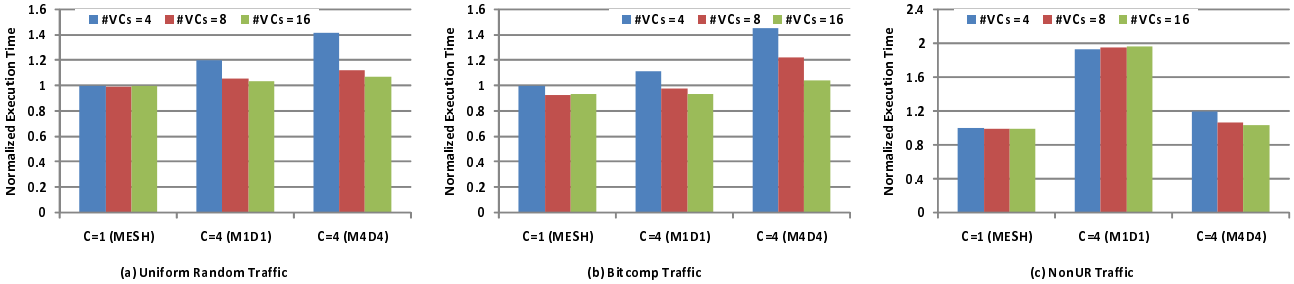

Figure 12. Synthetic workload comparison for the impact of virtual channels

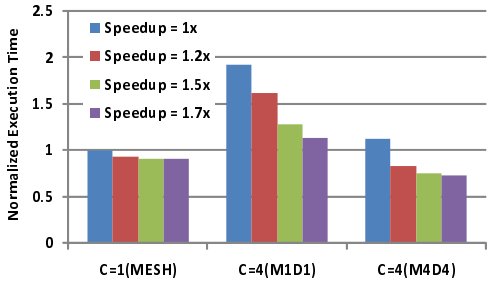

Figure 13. Impact of Router Speedup on nonUR traffic.

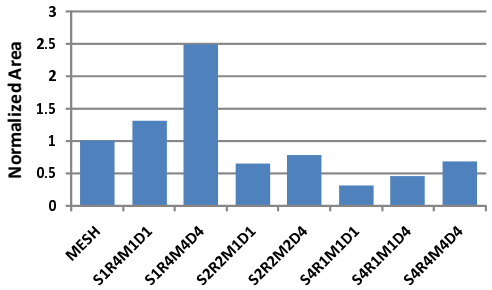

Figure 14. Area comparison of alternative architectures.

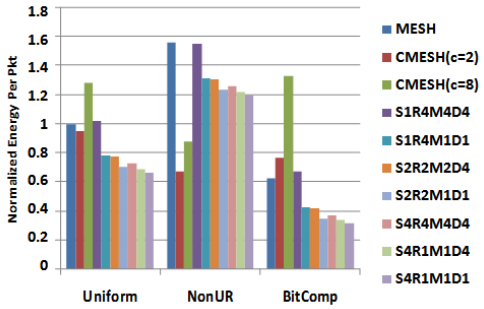

Figure 15. Average packet energy consumption during synthetic workload normalized to MESH under UR traffic.

router speedup of 1.7× results in 41% reduction of execution time and achieving performance that is within 14% of MESH.

5.5 Area / Energy Analysis

The on-chip network area and energy consumption of the different architectures discussed in Section 2 and 3 are compared in this section. We estimate the router area using the model in [4] for the router components. As shown in Figure 14, external concentration ($S1R4M1D1$) results in reduction of 47% area compared to integrated concentration ($S1R4M4D4$). The area analysis includes the additional area for the muxes needed in $S1R4M1D1$. Channel slicing further reduces the area with the reduction of crossbar complexity and the virtual concentration implementation ($S4R1M1D1$) achieves 69% reduction in area compared to the baseline MESH topology.

We compare the average packet energy consumption in Figure 15. The total energy consumed by a packet consists of crossbar energy, buffer energy and link traversal energy [4, 12]. Under UR traffic, the internal concentration scheme ($S1R4M4D4$) consumes 6.6% more energy per packet than the baseline MESH, while the external ($S1R4M1D1$) reduces energy by 22%. Further energy reduction can be achieved by adopting channel slicing, where

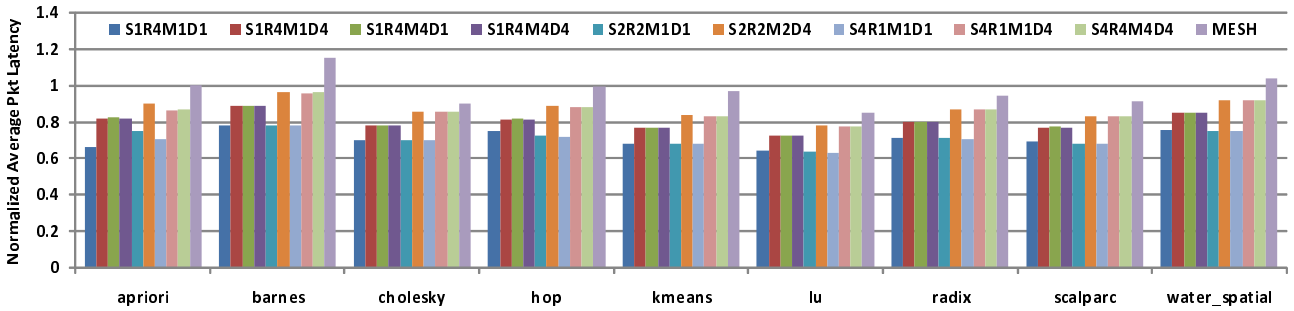

Figure 16. Average packet latency from 64-core CMP SPLASH2 and Minebench traces.

the proposed virtual concentration routers further reduces energy by 32% under both UR and bitcomp traffic.

5.6 Trace Based Evaluation

Since the total execution time for a trace is dependent on the timing information in the trace and does not capture any inter-dependencies between packets, we focus on the average latency of the packets from the trace. From Figure 16, because the network is lightly loaded for these application, the average packet latency matches closely with the zero-load latency comparison of the different architectures as virtual concentration ($S1R4M1D1$) reduces average packet latency by up to 17% compared to the baseline MESH.

6 Related Work

As discussed earlier, concentrated mesh with express channels was proposed by Balfour and Dally [4]. Our CMESH architecture differs as we do not provide any express channels but still provide the same bisection bandwidth. They also proposed CMESH2x with two parallel CMESH networks. However, their CMESH and CMESH2x comparison is not the same as our use of channel slicing as their CMESH2x doubled the bisection bandwidth. The external concentration approach in on-chip networks was described by Kim et al. [14] but they did not provide a quantitative comparison of alternative concentration implementations. In the hybrid topology proposed by Das et al. [10], an 8-way concentration is implemented through the use of a shared bus. Although a bus can provide a high bandwidth medium, the wire delay becomes problematic and proper bus arbitration becomes critical to fully utilize the network bandwidth.

The MIT RAW [25] and Tilera Tile64 [2] processors use a similar approach to channel slicing in their on-chip networks as they provide 5 parallel 2D mesh networks. However, their networks use dedicated traffic class for each network and they do not incorporate concentration to reduce network cost. The MECS topology [12] describes their topology using $S = 2$ sliced topology as they evaluate MECSx2. Similar to our study, they hold the constant bisection bandwidth constant in their comparison. However, they do not explore slicing beyond $S = 2$ and do not exploit the ability to implement virtual concentration. The XShare router micro-architecture [10] provides a micro-architecture that provides the ability to share the wide data path with multiple short packets. Our proposed channel sliced techniques provides similar benefits by using a sliced network without adding any complexity to the router micro-architecture.

Concentrated or bristled networks and channel slicing have been used in off-chip networks and large-scale systems. For example the Cray X1 [1] network uses a dual-bristled torus topology by connecting two nodes to a single router. The Cray BlackWidow network [24] uses a folded-Clos topology and implements a channel slicing factor of 4 – creating four parallel folded-Clos network. Martinez et al. [18] showed the benefits of using virtual channels in bristled hypercube topology. Our work shows similar trend for on-chip network concentrated mesh topology. However, the constraints of on-chip networks and off-chip networks are very different and this work describes the benefits and trade-off in implementing concentration and channel slicing in on-chip networks.

7 Conclusion

In this paper, we explored cost-efficient techniques for reducing complexity of on-chip network through the use of concentration and channel slicing. Instead of the commonly assumed, high-radix (integrated) implementation of concentration, we show how external implementation of concentration significantly reduces complexity with small loss in performance. Hybrid implementations of concentration is also presented which approaches the performance of an integrated concentration implementation while still reducing complexity. We further demonstrated that adopting both concentration and channel slicing can yield a highly efficient *virtual concentration* architecture, which saves 69% in area and 32% in energy as compared to a conventional 2D mesh topology, while matching its throughput.

Acknowledgements

This work is supported by NSF grants CNS-0551639, IIS-0536994, CCF-0747201, and CCF-0541337. We would

like to thank all the anonymous referees for their helpful suggestions and comments.

References

[1] Cray X1. `http://www.cray.com/products/x1/`.

[2] A. Agarwal, L. Bao, J. Brown, B. Edwards, M. Mattina, C.-C. Miao, C. Ramey, and D. Wentzlaff. Tile Processor: Embedded Multicore for Networking and Multimedia. In *Hot Chips 19*, Stanford, CA, Aug. 2007.

[3] N. Agarwal, L. S. Peh, and N. Jha. Garnet: A detailed interconnection network model inside a full-system simulation framework. Technical report, Dept. of Electrical Engineering, Princeton University, Feb. 2008.

[4] 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, Carns, Queensland, Australia, 2006.

[5] L. Benini and G. De Micheli. Networks on chips: A new soc paradigm. In *IEEE Computer*, volume 35.

[6] G. Chen, H. Chen, M. Haurylau, N. Nelson, P. M. Fauchet, E. Friedman, and D. Albonesi. Predictions of cmos compatible on-chip optical interconnect. In *7th International Workshop on System-Level Interconnect Prediction (SLIP)*, pages 13–20, San Francisco, CA, 2005.

[7] W. J. Dally. Virtual-channel flow control. In *IEEE Transactions on Parallel and Distributed Systems*, volume 3, pages 194–205, 1992.

[8] W. J. Dally and T. B. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishing Inc., 2004.

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

[10] R. Das, S. Eachempati, A. Mishra, V. Narayanan, and C. Das. Design and evaluation of a hierarchical on-chip interconnect for next-generation cmps. In *International Symposium on High-Performance Computer Architecture (HPCA)*, pages 175–186, Raleigh, NC, USA, Feb. 2009.

[11] R. Das, A. K. Mishra, C. Nicopolous, D. Park, V. Narayanan, R. Iyer, and C. R. Das. Performance and power optimization through data compression in network-on-chip architectures. In *International Symposium on High-Performance Computer Architecture (HPCA)*, pages 215–225, Salt Lake City, UT, 2008.

[12] B. Grot, J. Hestness, S. W. Keckler, and O. Mutlu. Express cube topologies for on-chip interconnects. In *International Symposium on High-Performance Computer Architecture (HPCA)*, pages 163–174, Raleigh, NC, 2009.

[13] M. J. Karol, M. G. Hluchyj, and S. P. Morgan. Input versus output queuing on a space-division packet switch. In *IEEE Transaction on Communications (COM-35)*, pages 1347–1356, 1987.

[14] J. Kim, J. Balfour, and W. J. Dally. Flattened butterfly topology for on-chip networks. In *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Chicago, Illinois, Dec. 2007.

[15] A. Kumar, L. S. Peh, and N. K. Jha. Token flow control. In *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 342–353, Lake Como, Italy, 2008.

[16] A. Kumar, L. S. Peh, P. Kundu, and N. K. Jha. 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.

[17] M. M. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, and D. A. Wood. Multifacet's general execution-driven multiprocessor simulator (gems) toolset. *ACM SIGARCH Computer Architecture News*, 33(4):92–99, Sep. 2005.

[18] J. F. Martinez, J. Torrellas, and J. Duato. Improving the performance of bristled cc-numa systems using virtual channels and adaptivity. In *Proc. of the International Conference on Supercomputing (ICS)*, pages 202–209, Rhodes, Greece, June 1999.

[19] N. McKeown. The islip scheduling algorithm for input-queued switches. In *IEEE/ACM Transactions on Networking*, volume 7, pages 188–201, Apr. 1999.

[20] R. Mullins, A. West, and S. Moore. Low-latency virtual-channel routers for on-chip networks. In *Proc. of the International Symposium on Computer Architecture (ISCA)*, page 188, München, Germany, June 2004.

[21] R. Narayanan, B. Ozisikyilmaz, J. Zambreno, G. Memik, and A. Choudhary. Minebench: A benchmark suite for data mining workloads. In *IEEE International Symposium on Workload Characterization (IISCW)*, pages 182–188, San Jose, CA, 2006.

[22] Y. Pan, P. Kumar, J. Kim, G. Memik, Y. Zhang, and A. Choudhary. Firefly: Illuminating future network-on-chip with nanophotonics. In *To appear in International Symposium on Computer Architecture (ISCA)*, Austin, TX, 2009.

[23] K. Sankaralingam and e. al. Distributed microarchitectural protocols in the trips prototype processor. In *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 480–491, Orlando, FL, 2006.

[24] S. Scott, D. Abts, J. Kim, and W. J. Dally. The black-widow high-radix clos network. In *Proc. of the International Symposium on Computer Architecture (ISCA)*, pages 16–28, Boston, MA, June 2006.

[25] M. B. Taylor, W. Lee, S. Amarashinghe, 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, CA, 2003.

[26] S. R. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, V. Erraguntla, C. Roberts, Y. Hoskote, N. Borkar, and S. Borkar. An 80-tile sub-100-w teraflops processor in 65-nm cmos. 43(1):29–41, 2008.

[27] S. Woo, M. Ohara, E. Torrie, J. 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, Jun. 1995.