Evaluating Ruche Networks: Physically Scalable, Cost-Effective, Bandwidth-Flexible NoCs

Dai Cheol Jung
University of Washington
Seattle, USA
dcjung@uw.edu

Michael Taylor
University of Washington
Seattle, USA
prof.taylor@gmail.com

Abstract

2-D mesh has been widely used as an on-chip network topology, because of its low design complexity and physical scalability. However, its poor latency and throughput scaling have been well-noted in the past. Previous solutions to overcome its unscalability relied on outdated assumptions that no longer hold true in recent architectures. Concentrated routers make an assumption that low injection rate would cause low conflicts; however, recent manycore processors and accelerators require streaming bandwidth for their data-intensive workloads. Widening the channel width to recover the bisection bandwidth halved by concentration assumes that the underlying architecture can flexibly adapt to the wider channel width; however, it comes with an additional cost associated with wider datapaths, serialization, and additional buffering.

Ruche Networks retain all the desirable properties of 2-D mesh to remain physically scalable, yet provide an architecturally flexible and cost-effective mechanism to effortlessly scale up the network performance by adding uniform long-range physical links. While their feasibility in real silicon has been demonstrated, there has not been any detailed evaluation of its network performance, scalability, and energy efficiency. This paper aims to fill the gap in research by providing some insight on design tradeoffs. Using RTL-level implementations, we demonstrate that Ruche Networks are superior to 2-D mesh and torus in terms of power, area efficiency, cycle time and network performance.

Keywords

Network-on-Chip, VLSI, Parallel Architecture

ACM Reference Format:
Dai Cheol Jung and Michael Taylor. 2025. Evaluating Ruche Networks: Physically Scalable, Cost-Effective, Bandwidth-Flexible NoCs. In *Proceedings of the 52nd Annual International Symposium on Computer Architecture (ISCA '25), June 21–25, 2025, Tokyo, Japan.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3695053.3731010

1 Introduction

2-D mesh continues to be a predominant on-chip network topology used today. A primary reason for its popularity has been due to how well it plays out with tiled physical-design methodology [25]. Unlike high-radix topologies [1, 12, 17], 2-D mesh has local and regular wire routing between tiles. This allows every tile to have an identical shape that can be stamped out in an array of any size until it consumes the available silicon area – even non-power-of-2, as seen in 6×6 ET-SoC-1 [11] and 18×20 Tesla DOJO [30]. The complexity of 2-D mesh routers does not grow as the number of nodes increases, whereas additional inputs (router radix) must be added to the routers of high-radix topologies. This prevents scalability beyond 256 nodes for high-radix topologies, which only offer an ad-hoc solution of the increased network dimension or a hybrid topology without providing much evaluation [12, 17]. 2-D mesh can be implemented using a standard-cell-based, automated CAD flow, which makes it relatively easy to adapt an existing design to a more advanced process technology.

Despite its low design complexity, 2-D mesh's poor latency and throughput scaling have been pointed out by many research papers in the past [1, 3, 17, 21, 29]. Energy inefficiency also worsens with the increased number of routers and added hops [7]. As the network size grows, the bisection bandwidth becomes a critical bottleneck, which forces cores to inject less and less packets before completely saturating the bisection bandwidth [29].

Previous solutions to overcome the unscalability of 2-D mesh rely on a number of outdated assumptions that may no longer hold true in recent architectures. *Concentration* co-locates a number of cores (usually 2 to 4) within a single tile to share a network router with a multiplexed input [3]. This makes an assumption that the traffic injection rate of each core is low; therefore, the probability of conflict at the network input is also low. This may be true for a cache coherence network, where a processor fires a request and waits, but for recent manycore processors without cache coherence that are specialized for data-intensive kernels, word-level packets are sent and received every cycle in a stream [16].

Concentration reduces the total number of network nodes, and thus the network latency and overall network energy, but it also reduces the bisection bandwidth. In order to compensate for this lost bisection bandwidth, a typical remedy has been to double the network channel width. However, previous NoC studies incorrectly assume that the underlying core architecture can efficiently adapt to the wider channel width. In fact, a wider channel would require an additional logic for serialization and deserialization to match the intrinsic ingress and egress bandwidth of the endpoint (e.g. a number of words that can be read out of SRAM each cycle). The mismatch in the channel and endpoint bandwidth not only introduces a serialization latency, which negates the latency reduction benefit of concentration, but also creates more area overhead by adding more buffers to prevent stalls in the network. On the other hand, the channel bandwidth can be matched by widening the SRAM and the processor datapaths, but this leads to a larger tile size, and ironically, does not help increase the network bandwidth in an absolute physical term (e.g. Tera-bit/s/mm).

This work is licensed under a Creative Commons Attribution 4.0 International License.
ISCA '25, Tokyo, Japan
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1261-6/25/06
https://doi.org/10.1145/3695053.3731010

Even without the use of concentration, increasing the channel width is not a scalable way to increase bandwidth for 2-D mesh, as it linearly increases the crossbar and buffer area. Furthermore, as observed in previous parallel processor designs, 2-D mesh still struggles to fully utilize the available wiring tracks between the tiles (only about 24%) [27]. Also, a concentrated router and a wide channel opt for using a larger and slower crossbar, which may increase the number of cycles per network hop or slow down the clock frequency to meet the timing. This ironically negates the latency benefit gained by concentration. Putting NoCs in a slower clock domain suffers the same problems related to bandwidth mismatch.

Ruche Networks [15, 25] have been proposed to overcome the scalability challenges of 2-D mesh by augmenting it with regular, long-range links. While retaining all the desirable properties of 2-D mesh to remain physically scalable, Ruche Networks provide an architecturally flexible and cost-effective mechanism to effortlessly scale up the bisection bandwidth without overhauling the underlying processor architecture. Its feasibility in real silicon has already been demonstrated [16], but there has not been any detailed evaluation of its network performance and energy efficiency, compared to other NoC topologies. This paper aims to fill the gap in research by providing some important insights on design tradeoffs.

2 Contributions & Insights

This paper makes the following contributions:

- This paper addresses the major deficiencies in two prior papers on Ruche networks [15, 25]. Both papers purely rely on analytical modeling without any cycle-accurate simulation using synthetic or application traffic. Neither paper has any analysis on power or the implications of Ruche Factors and depopulated crossbars on energy efficiency. There is neither qualitative nor quantitative comparison against other types of NoCs other than 2-D mesh.
- Using RTL-level implementations, we compare Full Ruche against 2-D torus and mesh using synthetic traffic patterns (Figure 6). We demonstrate that Full Ruche is capable of achieving higher bandwidth (Figure 6) at lower area cost, power and cycle time than 2-D torus (Figure 7, Table 3).
- Using scalable, throughput-oriented *cellular manycore* [16], we compare Half Ruche against torus and evaluate the impact of network size, aspect ratio, and Ruche Factor, using data-intensive parallel workloads (Figure 10, 11, 12, 13).

This paper makes the following insights:

- A key advantage of Ruche routers over folded torus routers is that they are deadlock-free without the use of virtual channels. Ruche routers make better use of hardware resources to provide more crossbar bandwidth than virtual channel routers (Figure 3). Due to their complex *allocation* logic, virtual channel routers cannot compete against Ruche routers in terms of cycle time without pipelining, which increases hop latency and buffer overheads (Figure 7).
- Compared to 2-D mesh, 2-D torus halves the network diameter and doubles the bisection bandwidth; however, 2-D torus using virtual-channel routers struggles to provide the promised bandwidth, because of the reason above. In uniform random traffic, its throughput saturates at injection rates lower than Ruche-One (Figure 1), a Ruche network without any skip links.
- Depopulating crossbars (Figure 5) provides an excellent way to lower area and power overhead without greatly sacrificing network performance (Table 2, 3). Using higher Ruche Factor is generally a more cost-effective way to gain more performance than using fully-populated crossbars (Table 6).
- Our detailed energy modeling based on 12 nm process shows that Ruche NoCs are able to reduce total NoC energy with long-range links; folded torus, on the other hand, ends up spending more total NoC energy than 2-D mesh due to its higher router energy (Figure 13).

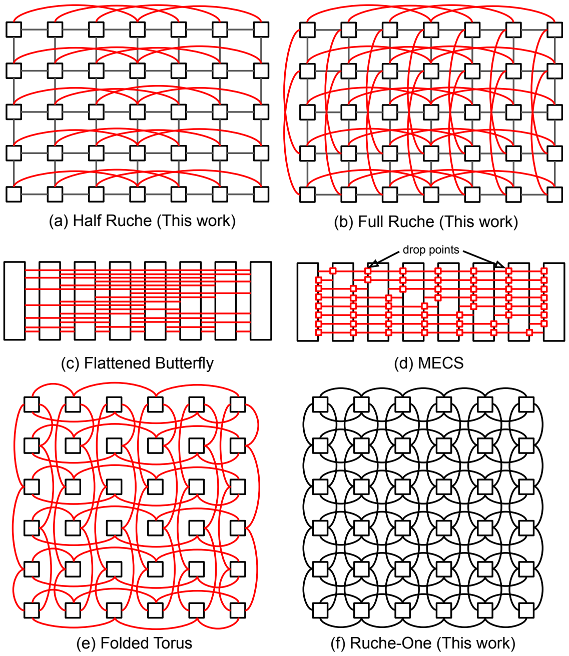

(a) Half Ruche (This work)
(b) Full Ruche (This work)
(c) Flattened Butterfly
drop points
(d) MECS
(e) Folded Torus
(f) Ruche-One (This work)

Figure 1: Low-diameter NoC topologies with non-local links. A regular tile shape and constant router complexity of Ruche Networks make them easy to physically scale with tiled design methodology.

The rest of the paper is organized as follows: Section 3 provides background information on Ruche router architecture. Section 4 presents various results on Full Ruche and Half Ruche evaluation. Section 5 surveys the related work.

3 Architecture

This section summarizes necessary background information on Ruche networks – its tileable, physically scalable network topology, routing algorithm, and router architecture.

3.1 Physically Scalable Ruche Topology

Ruche networks [15] augment 2-D mesh with equidistant, long-range, physical channels (a.k.a *Ruche channels*) between remote tiles in the same row or column. As these channels pass through the tiles, they consume the available VLSI wiring tracks. As *Ruche Factor* (RF), the skip distance of Ruche channels, increases, more wires are consumed, and the network diameter decreases. By adjusting the Ruche Factor, architects can scale up the bisection bandwidth without widening the channel width. Figure 1a and 1b show the Half and Full Ruche topology with Ruche Factor of 3. Full Ruche adds Ruche channels in both vertical and horizontal axes, whereas Half Ruche adds in only one axis. Ruche-One (Figure 1f) is a special case, when the Ruche Factor is one, and is topologically equivalent to having two mesh networks in parallel (Figure 3a). In this paper, we consider networks that provide in-order delivery of packets as is commonly required for streams and ordered memory traffic.

Figure 2 illustrates a bitwise pattern of mapping a Half Ruche network (RF = 3) using the tile-based method. Long-range wires cross a tile in straight lines using less resistive, upper-mid metal layers. Repeaters are placed in between tiles to drive these long wires. Note that all tiles have identical VLSI layouts. Local and Ruche channels are bit-interleaved, with the *i*-th bit of all links routed together, so that swizzling moves signals just a few wiring tracks.

Figure 1 shows a number of other low-diameter topologies with non-local links (a.k.a. express links) proposed in the past. A key differentiator for Ruche Networks is that its router radix remains constant, as the network size grows. In contrast, for high-radix topologies (e.g. MECS, flattened butterfly) (Figure 1c, 1d), router radix grows linearly with the network size in each dimension. As network size increases, MECS must continually reduce the channel width to not exhaust wire tracks and to keep cycle time and area under control. A narrower channel width, however, limits the injection bandwidth coming out of each node and adds serialization latency. Another differentiator is that the physical distance of Ruche channels remains constant, as the network size grows. In MECS, the longest channel distance (and its wire delay) grows with the network size. At some point, the wire delay starts to dominate, so either the cycle time must be increased or the wires need to be pipelined, increasing hop latency. These properties make it difficult to efficiently tile the architecture. Another issue with MECS is the difficulty in closing timing. Normally, the input and output ports of a tile are constrained with estimated external delays for setup and hold timing closure. Unlike the Ruche channels, where all channels have a single sender and a single receiver, multi-drop channels have multiple receivers each with different external delays. This makes it difficult to create a consistent set of constraints that can be uniformly applied to all tiles. Table 1 compares the low-diameter NoC topologies based on the physical scalability criteria.

Folded 2-D torus (Figure 1e) is an interesting alternative, because it maintains the same router radix as 2-D mesh, yet reduces the network diameter by half and doubles the bisection bandwidth. It meets the physical scalability criteria, as 2-D mesh and Ruche networks do (Table 1). It uses a similar tiling technique in Figure 2, as implemented in recent ML accelerators by Tenstorrent [14, 32], so the physical-design complexity is at a similar level with Ruche networks. Ruche routers, on the other hand, have higher router radix, so 2-D torus appears to have less area cost. However, a key disadvantage of 2-D torus is that it is deadlock-prone due to its cyclic channel dependency. A standard way to achieve deadlock freedom is by adding virtual channels (VC) and dateline logic to break the channel dependency cycle [8]. In this light, the area overhead of 2-D torus is now comparable to the that of Ruche networks. However, VC routers require complex *allocation* logic [6], which adds significant cycle time, area, and power overhead, as opposed to the simple logic that Ruche routers use. Fairness and matching quality that affect utilization are additional concerns for VC routers [6].

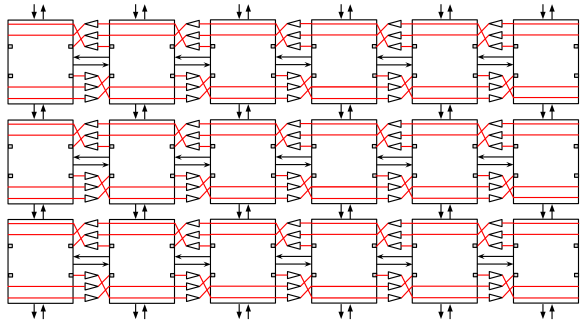


Figure 2: A bitwise pattern of mapping Half Ruche (RF = 3) on a tile-based design. Long-range wires pass through the tiles in a straight line using less resistive, global routing layers. The wires criss-cross between tiles, where repeater cells are placed.

Table 1: Comparing various NoC topologies based on physical scalability criteria. Both Ruche Networks and 2-D torus retain the desirable properties of 2-D mesh and augment it with long-range links.

Topology	**Regular Tile Shape**	**Regular Wire Routing**	**Constant Router Radix**	**Standard-Cell Based**	**Non-power-of-2 Tiling**	**Long-range Links**	**Constant Link Distance**
Ruche (This paper)	✓	✓	✓	✓	✓	✓	✓
2-D Folded Torus	✓	✓	✓	✓	✓	✓	✓
2-D Mesh	✓	✓	✓	✓	✓		✓
Multi-mesh	✓	✓	✓	✓	✓		✓
Flattened Butterfly [17]				✓		✓	
MECS [12]				✓	✓	✓	
Swizzle-Switch [1]					✓	~	

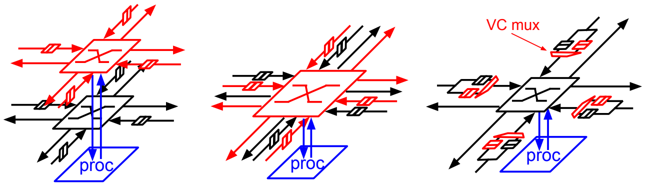
Figure 3a shows a 2x multi-mesh, where a second parallel mesh router is introduced to double the NoC bandwidth. Figure 3b and 3c illustrate two approaches to combine two parallel routers in a 2x multi-mesh. In terms of input FIFOs, both VC and Full Ruche routers have the same capacity. Full Ruche routers (Figure 3b) combines two mesh crossbars into one larger crossbar, so the peak router bandwidth is kept the same. VC routers, on the other hand, (Figure 3c) discard one of the mesh crossbars, so the peak bandwidth is halved. VC routers spend extra area to implement the virtual channels (VC mux) and get rid of one crossbar, but doing this does not improve the router bandwidth. In this paper, we compare 2-D torus and Ruche networks in terms of power, cycle time, area, and network throughput and latency.

3.2 Ruche Router Architecture

Ruche Networks augment 2-D mesh with extra directions (Ruche East/West/South/North or RE/RW/RS/RN for short), whose channels can be elongated to make connections with distant tiles. Ruche routers provide two variants of routing algorithms to tradeoff between crossbar area overhead and network latency: *fully-populated* and *depopulated* (Figure 4). In both variants, packets are routed using a modified dimension-ordered routing (DOR), which is deadlock-free without the use of virtual channels. Generally, the routing in the first dimension use "Ruche-first", where a packet starts off on a Ruche link, as if getting on a highway to travel the majority of distance, and then getting off the highway to a local road for the remaining distance. The routing in the second dimension uses "local-first", where a packet uses local links until the remaining distance modulo Ruche Factor becomes zero, and then uses Ruche links to get to the destination. In fully-populated, packets can make a direct turn from the lower to higher dimension using the Ruche links (e.g. from RE/RW to S/N/P/RS/RN). Depopulated routing is non-minimal in the sense that packets must get off Ruche links to local links before making a turn. Figure 5 shows the additional crossbar connectivities added to the minimal 2-D mesh router using DOR, employed in Celerity RISC-V manycore [9]. It shows that the depopulated router reduces the total connectivities by 16. The output port with the highest number of inputs (e.g. P output) has its inputs reduced from 9 to 7. Ruche-One is a special case (Figure 1f), where both local and Ruche links make connection to the nearest neighbors. In this case, packets take Ruche links for the entire path if the total distance is even (take local, if odd) to balance the traffic. Ruche-One works only on fully-populated routers.

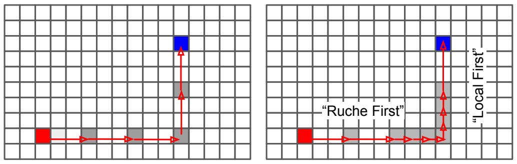
By default, the input ports of Ruche routers are minimally buffered by two-element FIFOs. Packets are arbitrated by a simple round-robin policy at each output direction. In high compute-density manycore processors [9, 16], the tile size is small, so the wire delay to cross a tile is relatively low, while the crossbar gate delay dominates the critical path delay between tiles. In this case, packets are able to traverse the network at the rate of single-cycle per hop. As the tile size or Ruche Factor increases, the wire delay starts to dominate, in which case the router and the physical link need to be pipelined using credit-based flow control. The capacity of input FIFOs needs to be increased accordingly to hide the credit-return latency. This is not a problem unique to Ruche Networks, as the same solution would be equally required by other router architectures. However, this problem is exacerbated by the routers that use virtual channels. In Ruche routers, the generation of request signals going to arbiter is independent of the ready signal from the output side in terms of combinational logic (e.g. ready-valid-and [31]). In VC routers, the generation of request signals going to the allocator must depend on the credit availability signal of the destination VC (e.g. ready-then-valid [31]), otherwise the allocator could grant access to VCs that could not have moved forward and block the progress of other VCs.

For these reasons, VC routers generally have longer critical delay paths and need to be pipelined. The canonical VC routers are assumed to have four sequential pipeline stages (e.g. routing, VC allocation, SW allocation, SW traversal). Speculative VC routers [26] reduce the number of stages down to three by allowing VC and SW allocation to take place in parallel. This relies on speculating that VC allocation will be successful but giving priority to non-speculative SW access over the speculative ones. This comes at the expense of replicating SW allocators for speculative and non-speculative requests. Mullins et al. [23] proposes a single-cycle router architecture, where switch traversal and speculation on the allocation decisions for the next cycle are overlapped. The key idea is that, by using the least-recently-granted priority scheme of matrix arbiters, it can speculate on the next grant signals, knowing that the current input that has been granted will have the lowest priority on the next cycle. However, due to its complex control logic, speculative VC routers add significant power overheads over wormhole and non-speculative VC routers [4].



(a) 2X multi-mesh (b) Full Ruche (c) Virtual Channel Router

Figure 3: Two approaches to combine 2x multi-mesh routers. Full Ruche combines two mesh crossbars into a larger one to maintain the router bandwidth. Virtual channel routers discard one of the crossbar, so the router bandwidth gets halved. The area overhead (VC mux) to implement virtual channels saves the crossbar area, but does not help improve the router bandwidth.



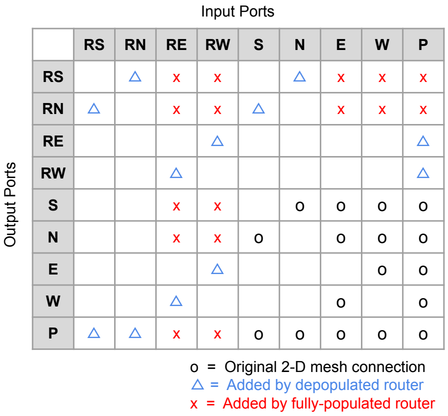
(a) fully-populated (b) depopulated

Figure 4: Two variants of a deadlock-free dimension-ordered routing algorithm for Full Ruche (RF = 3, X-Y order). Fully-populated optimizes the X-Y turn at the expense of crossbar area, but it may not be the most common case. Depopulated offloads more traffic on the local links, which tend to be less utilized in large networks.

Another important consideration is that network throughput is highly sensitive to the router latency as a side-effect of credit-return latency, which reduces the buffer utilization [26]. Thus, Ruche routers that can achieve lower cycle time at lower area than VC routers under the same conditions (e.g. tile size, channel width) without any form of speculation are more advantageous.

4 Evaluation

In this section, we evaluate Ruche NoCs against 2-D mesh and torus under two common scenarios. First, we evaluate generic, all-to-all communication patterns in square network sizes (8×8 and 16×16). In this case, both vertical and horizontal bisection could become a bottleneck. Therefore, we compare Full Ruche, which adds Ruche channels in both dimensions, against 2-D torus. Second, using the *cellular manycore* simulator [16], we evaluate *all-to-edge* communication patterns, where most network traffic go to memory ports on the northern and southern edge of the network. In this arrangement, using X-Y DOR for request traffic and Y-X DOR for response provide the best network throughput [2]. For all-to-edge, bisection bottleneck appears in horizontal directions, as packets try to first



Input Ports

Output Ports

	RS	RN	RE	RW	S	N	E	W	P
RS		△	x	x		△	x	x	x
RN	△		x	x	△		x	x	x
RE				△					△
RW			△						△
S			x	x		o	o	o	o
N			x	x	o		o	o	o
E				△				o	o
W			△				o		o
P	△	△	x	x	o	o	o	o	o

o = Original 2-D mesh connection
△ = Added by depopulated router
x = Added by fully-populated router

Figure 5: Full Ruche crossbar connectivity matrix (X-Y DOR). By restricting some crossbar paths, depopulated routers significantly reduce the crossbar area. Fully-populated connectivities (red x) are added on top of depopulated (blue triangle).

reach their destination column. We evaluate Half Ruche and half-torus, which add long-range channels in horizontal directions to relieve this bottleneck. We exclude adding vertical long-range links in this scenario, since vertical channel bandwidth already matches the memory port bandwidth 1:1, so the benefit is small when the network load is high. Although two scenarios are not mutually exclusive, we demonstrate that Ruche NoCs can be specialized for common traffic patterns found in target architectures.

4.1 Full Ruche - Synthetic Traffic

Figure 6 shows the synthetic traffic analysis on 2-D mesh, 2-D torus, multi-mesh, and various Full Ruche topologies. These results are based on a cycle-accurate simulation of RTL-level implementations. 2-D mesh and Full Ruche are without virtual channels, whereas 2-D torus has two VCs per input. We assume using a single-flit packet, and each FIFO/VC holds up to two packets. Packets move through at the rate of one cycle per hop in all networks. 2-D torus uses dateline VC partitioning to achieve deadlock freedom [8]. The routing algorithm used is DOR, which produces one output VC direction. SW allocation in 2-D torus is done by an acyclic implementation of wavefront allocator for maximal matching quality [5]. Evaluation is done on 8×8 and 16×16 networks. Four synthetic traffic patterns were used: uniform random, bit complement, transpose, and tornado. Five configurations of Full Ruche topologies are evaluated. Ruche Factors are varied from one to three (ruche1-3), and both fully-populated (pop) and depopulated (depop) crossbars are considered.

In uniform random 8×8, 2-D mesh has saturation throughput around 28%. Although 2-D torus doubles the bisection bandwidth, its saturation throughput peaks around 42%. ruche1-pop, which does not reduce network diameter yet provides the same bisection bandwidth as 2-D torus, outperforms 2-D torus in terms of throughput (~48%). As explained in Figure 3, VC routers halve the peak crossbar bandwidth, while Ruche routers retain the bandwidth of 2x multi-mesh.

This difference becomes even more prominent in the case of uniform random 16×16. 2-D mesh throughput saturates around 15%, whereas 2-D torus throughput only reaches up to 19%, far less than the throughput expected by doubling the bisection bandwidth. By maintaining the crossbar bandwidth, ruche1-pop is able to reach the saturation bandwidth of 28%, which is much closer to what is expected. 2x multi-mesh traces very close to the ruche1-pop curve.

As we increase the Ruche Factor, the zero-load latency generally decreases and the saturation throughput increases. In case of a relatively smaller network (8×8), if the Ruche Factor becomes too large (ruche3-depop), it reduces the chance that the Ruche links will be used, so the performance drops. In a smaller network, fully-populated routers (ruche2-pop and ruche3-pop) help by providing more routing flexibility. The benefit of greater Ruche Factors is more visible in larger networks (16×16).

In adversarial traffic patterns (bit complement, transpose, tornado), Ruche-one performs just as well as 2-D mesh with 2-D torus performing better. With Ruche Factor of 2 or 3, Full Ruche generally performs better than 2-D torus with some exception. ruche3-pop in bit complement 8×8 is an exceptional case, where there is zero conflict between network traffics, so the latency does not degrade. 2-D mesh, multi-mesh, and ruche1-pop perform very similarly, except in the transpose pattern where ruche1-pop performance better.

4.2 Full Ruche - Area and Cycle Time

We evaluate the area and cycle time of 2X multi-mesh, 2-D torus and Full Ruche routers, following the same methodology used in [5]. Figure 7 reports the cell area, as we sweep the target cycle time used for synthesis. Synthesis is done using Synopsys Design Compiler with 12 nm regular-Vt standard-cell library. Cycle time is normalized in terms of the fanout-of-four (FO4) delay of our target library. For each router, we decrease the cycle time with a fixed decrement until a timing violation is detected. For all routers, we use 128-bit channel width and X-Y DOR crossbar.

Figure 7 shows that Full Ruche routers can achieve much lower cycle time than 2-D torus routers, due to the fact that the complexity of the wavefront allocator is significantly higher than the decentralized round-robin logic that Ruche routers use. The depopulated Full Ruche has much lower area than the fully-populated and multi-mesh, due to the significant saving in crossbar area. When the cycle time is relaxed (~100 FO4), the fully-populated has slightly higher area than 2-D torus. However, the fully-populated is able to reach much lower cycle time than 2-D torus without timing violation. The result shows that both fully-populated and depopulated reach about the same minimum cycle time, slightly higher than that of 2-D mesh. Multi-mesh has a comparable minimum cycle time with Ruche, because of the higher fanout at the P-input port and the additional route compute logic to decide between two meshes when injecting packets (it uses mesh0, if the Manhattan distance is even; mesh1, otherwise). The maximum number of crossbar mux input is 7 and 9 for depopulated and fully-populated (Figure 5), respectively, which is only a few gate delay differences. A key advantage of Ruche routers is that it can achieve competitive cycle time without pipelining. On the other hand, VC routers will need to be pipelined, which will either increase pipeline latency, or require complex speculation schemes that increase power [26].

Table 2 shows the router area breakdown when the target cycle time is most relaxed (~ 98 FO4). The depopulated router significantly reduces the crossbar area by 40%, which is considerably

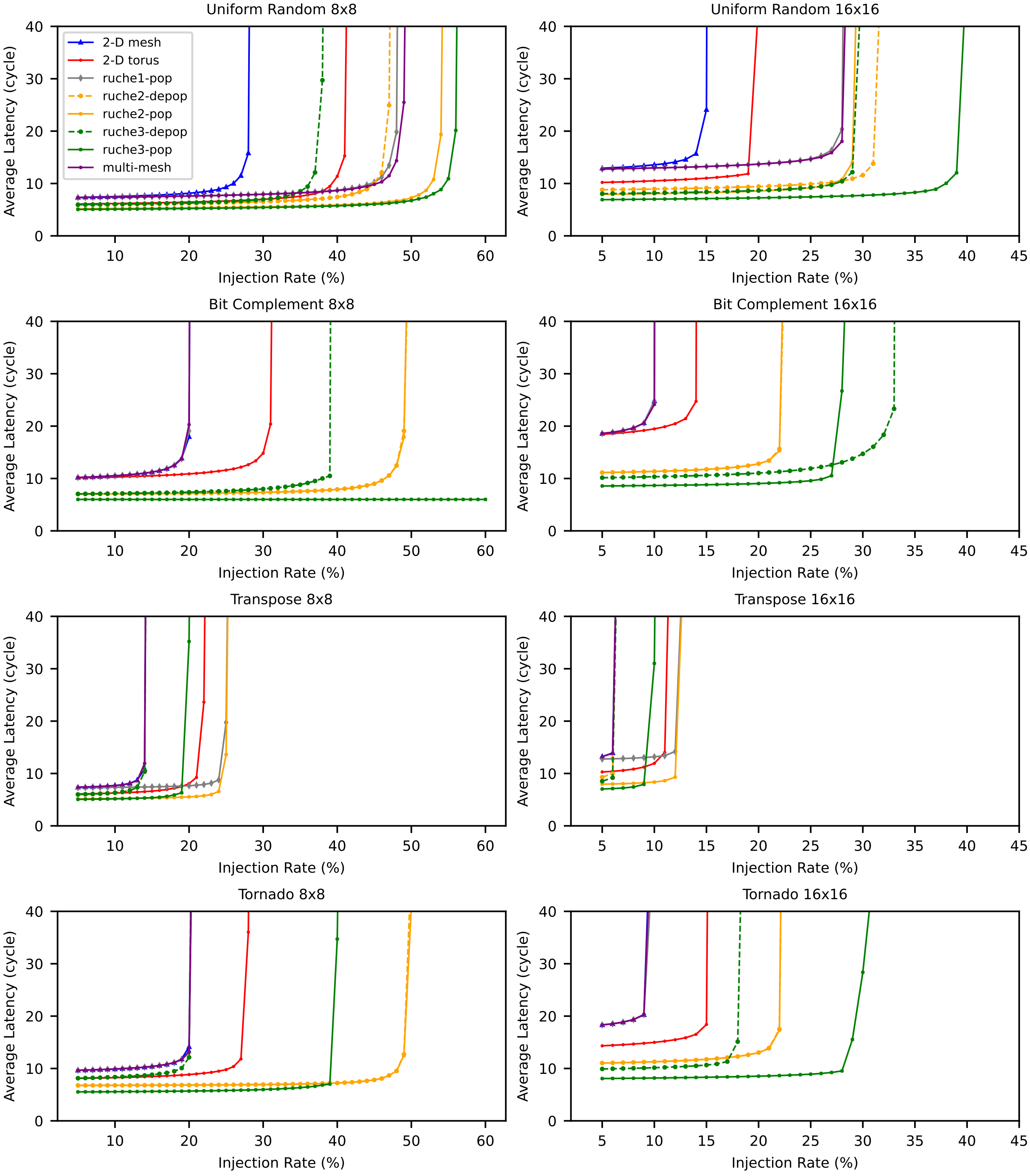


Figure 6: Synthetic traffic analysis on 2-D mesh, 2-D torus, and various Full Ruche topologies. In uniform random, Ruche-one, which has no express links, performs better than 2-D torus in terms of throughput, even though both topologies are normalized on the bisection bandwidth, because its crossbar provides higher bandwidth than the VC router crossbar. With Ruche Factor of 2 or 3, Full Ruche generally performs better than 2-D torus for both uniform random and adversarial traffic patterns with a few exceptions.

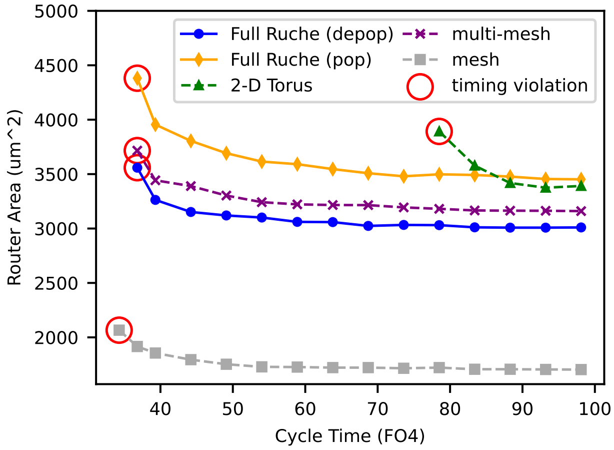


Figure 7: Area vs. Cycle Time comparison of Mesh, Multi-mesh, Full Ruche and 2-D torus routers. Full Ruche routers can achieve much lower cycle time without pipelining, because of their simplified router architecture. 2-D torus would need to be pipelined in order to achieve competitive cycle time at the expense of increased hop latency. Full Ruche (depop) achieves lower area than multi-mesh, while retaining the peak crossbar bandwidth and cycle time, by efficiently combining two mesh crossbars (Table. 2)

Table 2: Multi-mesh, Full Ruche and 2-D torus Router Area Breakdown.

	Multi-mesh (um^2)	**Full Ruche (depop)** (um^2)	**Full Ruche (pop)** (um^2)		**2-D Torus** (um^2)
Crossbar	791	599	986	Crossbar	410
Decode	96	99	100	Decode	349
FIFO	2250	2250	2250	VC	2435
Arbiter	53	42	74	Allocator	194
TOTAL	**3190**	**2991**	**3411**	**TOTAL**	**3388**

less than the double 2-D mesh crossbars used in multi-mesh. Route computation (decode), allocation logic and virtual channels of 2-D torus contribute to more area overhead. Overall, depopulated Full Ruche has 12% less area than 2-D torus.

4.3 Full Ruche - Energy

Table 3 reports the amount of energy required to transmit one packet across the router. We place and route the routers with 128-bit wide channel using Synopsys IC Compiler 2. The tile region is 187 um x 187 um, which is roughly the 1.3× size of a dense RISC-V core [16]. After place and route, wire parasitics are extracted by StarRC. In simulations using the gate-level netlist, for each output direction, we collected switching activities when a stream of packets are sent from corresponding valid inputs. We assume that packet payloads have activity factor of 0.25 (i.e. half of bits switching every cycle). With wire parasitics and switching activities, we use Synopsys PrimeTime to accurately calculate average energy per packet. This result does not include the energy dissipated by long-range links outside the tile area.

The result shows that Full Ruche routers are generally more energy efficient than 2-D torus. The depopulated crossbar lowers the energy even further, especially for the Ruche directions, which have less crossbar inputs. As shown in Figure 5, the depopulation reduces the number of mux inputs for RS and RN by 5.

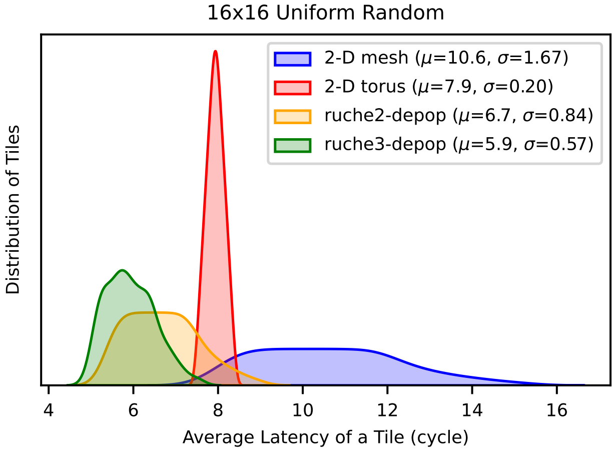


Figure 8: Ruche links mitigate the network unfairness found in 2-D mesh. Although Ruche never reaches the perfect fairness of 2-D torus, it significantly reduces the variance, compared to 2-D mesh, and reduces the average below that of 2-D torus.

Table 3: Full Ruche and 2-D torus Router Energy per Packet

Direction	**Full Ruche (depop)** (pJ)	**Full Ruche (pop)** (pJ)	**2-D Torus** (pJ)
Horizontal	1.66	1.95	2.41
Vertical	1.82	2.01	3.35
Ruche Horizontal	1.40	1.81	–
Ruche Vertical	1.49	2.00	–

4.4 Full Ruche - Fairness

In 2-D mesh, an average latency of a tile highly depends on its location. Tiles that are near the edge of the network have higher average latencies than the ones near the center, and this unfairness increases with the network size. 2-D torus topology is theoretically the network with most fairness, since the network is symmetric from every node.

Figure 8 shows the distribution of average latency experienced by each individual tile, assuming 16×16 uniform random and low network load. As expected, 2-D mesh has much higher stdev ($\sigma = 1.67$) and mean ($\mu = 10.6$) than 2-D torus. Adding Full Ruche reduces both stdev and the mean. Although Full Ruche never reaches the ideal fairness of 2-D torus, Ruche2 and Ruche3 reduce the stdev by 2.0× and 2.93× respectively, compared to 2-D mesh,. Ruche2 and Ruche3 reduce the mean average latencies, compared to 2-D torus, by 1.18× and 1.34×, respectively.

4.5 Half Ruche - Synthetic Traffic

We first evaluate Half Ruche networks using synthetic traffic patterns to establish how network size, aspect ratio, and Ruche Factor affect the network performance. We evaluate 16×8 as a baseline, and scale up the network by 4× to evaluate scalability. Wide, rectangular aspect ratios are motivated by the desire to have more bandwidth at the edge of the network. 32×16 maintains the aspect ratio of 16×8, but the compute-to-memory tile ratio is reduced by half. In 64×8, the compute-to-memory tile ratio is preserved; however, wider aspect ratio stresses the bisection bandwidth bottleneck even more. We consider two types of uniform random traffic that are common in cellular manycore [16]. *tile-to-tile* represents all-to-all communication among all tiles. *tile-to-memory* represents a traffic pattern going to the top and bottom edges of the network for memory access.

Figure 9 shows the analyses of two traffic patterns and three network sizes. In general, Half Ruche topologies perform consistently better than mesh in all cases. Saturation throughput of half-torus always falls in between 2-D mesh and Ruche2, as we hypothesized earlier that torus networks would lose on the peak crossbar bandwidth. There is no significant difference between the depopulated and fully-populated routers of the same Ruche Factor. In 16×8, the network size is not large enough to distinguish the performance benefit between Ruche2 and Ruche3. Even in 32×16, Ruche3 has only a slightly lower zero-load latency than Ruche2, although the throughput saturates near the similar injection rate.

In tile-to-tile, adding Half Ruche channels almost doubles the saturation throughput; however, the benefit of Half Ruche is less noticeable in tile-to-memory. Ideally, the saturation throughput of tile-to-memory should be similar to the compute-to-memory tile ratio. For example, in 16×8, the ratio of compute to memory tiles is 4:1, so each compute tile can ideally inject a packet every 4 cycles before completely saturating the memory bandwidth. However, because of the X-Y DOR, the horizontal bisection bandwidth, which has the capacity to cross 16 packets per cycle in both direction (in case of 2-D mesh), is likely to saturate before saturating the memory tile bandwidth (32 packets per cycle). In 16×8, 2-D mesh saturates around 16~17%, but the Half Ruche bring it closer (~21%) to the theoretical limit by breaking the bisection bottleneck. In 32×16 tile-to-memory, Half Ruche seems to perform a lot worse in terms of the saturation throughput; however, the compute-to-memory ratio is increased to 8:1 (12.5%), so the saturation throughput of 11% is actually pretty close to this limit.

64×8 is somewhat extreme in terms of the aspect ratio. Normally, in 2-D mesh or Full Ruche, the aspect ratio of 1:1 would be preferred to minimize the network diameter. Nevertheless, 64×8 is an interesting option to consider, since it maintains the 4:1 compute-to-memory ratio. By adding more hops horizontally rather than vertically, it creates more opportunities to utilize horizontal Ruche channels to reduce latency. However, because of its wide aspect ratio, the bisection bottleneck becomes a big problem. For 2-D mesh, the zero-load latency almost goes off the chart even at the lowest injection ratio we measured (5%). The performance difference between Ruche2 and Ruche3 is now more distinguished. Although Ruche3 did not come close to the theoretical maximum saturation throughput we hoped to gain (~25%) by maintaining the 4:1 compute-to-memory tile ratio. We also explore Ruche4 for 64×8. Saturation throughput continues to improve with Ruche4, significantly higher than the maximum throughput measured in 32×16.

Table 4 summarizes these trends. For 16×8, the bisection bandwidth can easily exceed the memory-tile bandwidth by adding Ruche channels (the bisection bandwidth is twice as much as the memory-tile bandwidth for Ruche). Maintaining the same 2:1 aspect ratio, 32×16 can do the same with Ruche channels; however, the compute-memory ratio gets halved, which limits the per-core bandwidth. 64×8 preserves this compute-memory ratio at the expense of having a more disproportionate aspect ratio (8:1); however, despite using Ruche channels, it becomes more difficult to match the bandwidth (in fact, it would require as high as Ruche7 to match). Although not evaluated in this paper, 32×8 with Ruche3 appears to be an interesting design point, since it can match the bisection and memory-tile bandwidth 1:1.

Table 4: Comparison of Bandwidth Ratio. Highlighted are where Bisection BW ≥ Memory Tile BW.

Network Size	Aspect Ratio	NoC	Bisection BW	Memory Tile BW	Compute-Memory Ratio
16×8	2:1	mesh	16	32	4:1
		ruche2	**48**	32	
		ruche3	**64**	32	
32×16	2:1	mesh	32	64	8:1
		ruche2	**96**	64	
		ruche3	**128**	64	
64×8	8:1	mesh	16	128	4:1
		ruche2	48	128	
		ruche3	64	128	
32×8	4:1	mesh	16	64	4:1
		ruche2	48	64	
		ruche3	**64**	**64**	

Table 5: Benchmarks and Input Datasets – Graphs from [10].

Benchmarks	Input Dataset
Jacobi	512×512×64 FP32
SGEMM	512×512×512 FP32
2-D FFT	16K/32K FP32
Barnes-Hut (BH)	16K/32K/64K bodies
BFS	See Graphs
PageRank (PR)	See Graphs
SpGEMM	See Graphs

Graph Name	Type	Edges / Nodes
offshore (OS)	Scientific	4.2M / 260K
roadNet-CA (CA)	Road	5.5M / 1.9M
road-central (RC)	Road	33.8M / 14.1M
road-usa (US)	Road	57.7M / 23.9M
ljournal-2008 (LJ)	Social	79.0M / 5.3M
hollywood-2009 (HW)	Social	113.9M / 1.1M
soc-Pokec (PK)	Social	30.6M / 1.6M

To summarize, first, the compute-memory tile ratio should be adjusted based on the application characteristics (4:1 may be too high). Then, array aspect ratio and Ruche Factor can be picked to adjust the bisection and memory-tile bandwidths. Ideally, the bisection bandwidth should be greater than or equal to the memory-tile bandwidth. This analysis is based purely on bandwidth without taking the effects of latency, area, and power into account. The rest of this section covers these other metrics in greater details.

4.6 Half Ruche - Benchmark Speedup

For Section 4.6–4.9, we modified a recent cellular manycore simulator [16] to evaluate various networks using parallel workloads. The evaluation is based on a full system, RTL-level, cycle-accurate simulation of RISC-V core execution and various NoCs; unlike *trace-driven*, our *execution-driven* method preserves the feedback effect of network backpressure and remote load latency on core execution, which is more realistic. Parallel workloads and datasets used are described in Table 5. Our evaluation includes the compute phases of the program when the working set of data is cached in the LLCs and excludes the data transfer phases in between, where memory system dominates the overall latency and does not produce any meaningful results for the NoC evaluation.

Figure 10 shows the parallel benchmark speedup over 2-D mesh and half-torus in both 16×8 and 32×16. Overall, Half Ruche NoCs provide consistent performance improvements over half-torus and 2-D mesh across all benchmarks. Generally, fully-populated routers (pop) performs better than depopulated (depop), and Ruche3 performs better than Ruche2. This result contradicts the synthetic

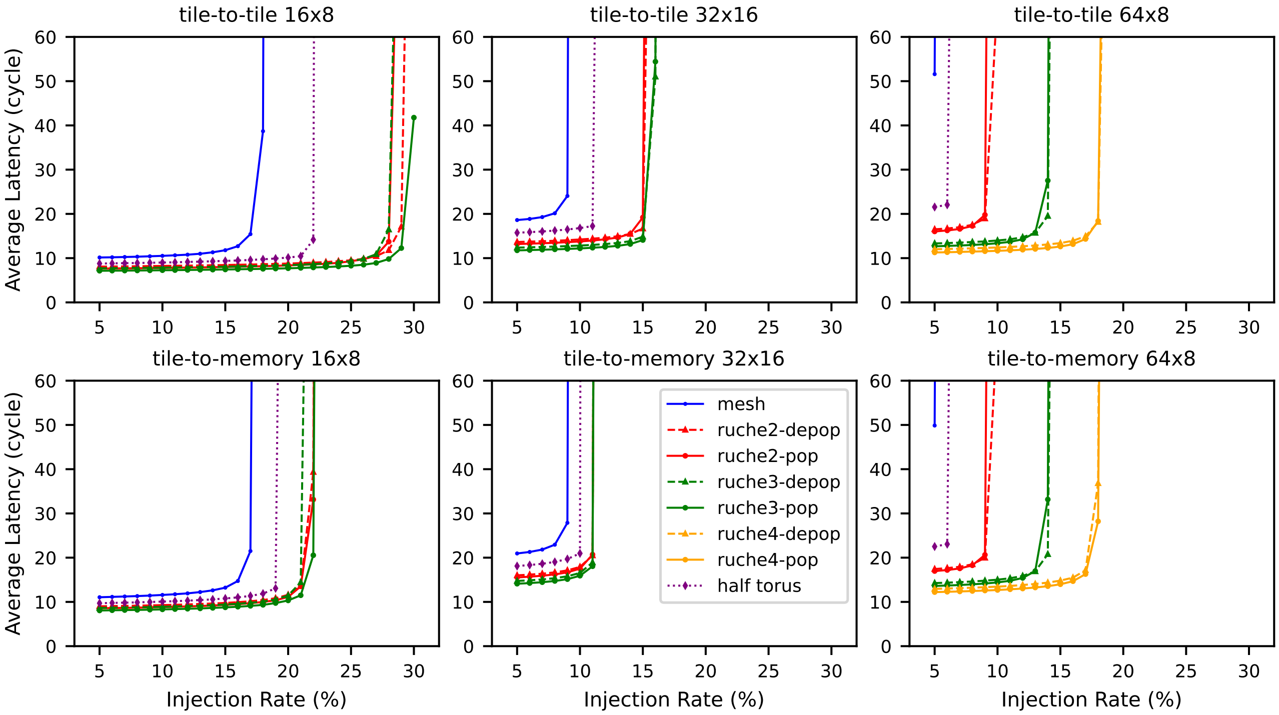


Figure 9: Synthetic traffic analysis on 16×8, 32×16, and 64×8. Ruche Networks consistently improve zero-load latency and saturation throughput. Depopulated and fully-populated routers have similar performance. The benefit of higher Ruche Factors is more noticeable in 64×8. For tile-to-memory, Ruche Networks help reach the maximum saturation throughput bounded by compute-to-memory tile ratio. Despite having the same skip distance and FIFO capacity, half-torus has lower saturation throughput than Ruche2 due to its crossbar.

traffic results (Figure 9), which suggest there should not be much difference in performance. This discrepancy can be explained by the fact that the traffic patterns in real benchmarks tend to be more bursty and not as random as they are in synthetic simulations. Moreover, since our simulation methodology is execution-driven, the timing of subsequent packet injections is affected by the changes in network congestion and latency. In the synthetic traffic testbench, packets are randomly injected based on a fixed probability. The benefits of Ruche networks are greater in 32×16 than in 16×8. For 32×16, SpGEMM (US,RC) did not show much improvement, because of its heavy use of an atomic add variable for dynamic linked list node allocation, which creates a hotspot in the network. Larger core array exacerbates this hotspot problem. With some algorithmic changes, SpGEMM performance can be improved as well. One exceptional case is Jacobi in 32×16 half-torus, where performance drops by 20%. Jacobi kernel accesses the scratchpads of the nearest neighbors, but since folded torus topology skips every other tile, packets must take the longest route around the network to reach the nearest tiles. This problem exacerbates as the network size increases.

4.7 Half Ruche - Benchmark Scalability

Here we define scalability as how much application speedup we can gain by adding more compute nodes. Since we are quadrupling the number of cores, the most speedup we can expect is 4×. In Figure 11, we measure the speedup of 32×16 and 64×8, compared to 16×8 mesh for different network configurations. In all cases, Ruche Networks help with better scalability. Half torus always scales worse than Ruche and sometimes worse than mesh. 64×8 mesh really struggles to make use of additional cores, because of the massive pressure on the horizontal bisection bottleneck. At Ruche2, 32×16 scales better than 64×8. Once Ruche3 is reached, 64×8 performs better than 32×16 by taking advantage of higher compute-to-memory tile ratio. For benchmarks with sequential and block-sized memory access patterns (e.g. Jacobi, FFT, SGEMM), Ruche3 with fully-populated could almost reach the ideal 4× scalability. BFS with social graphs (HW, LJ), which is more prone to load imbalance, has shown limited scalability with 4× cores. 32×16 can be more favorable than 64×8, since it can obtain good return on a low investment in area and routing. As noted in Table 4, 64×8 requires more Ruche channels to fully utilize the extra memory-tile bandwidth.

4.8 Half Ruche - Remote Load Latency

The average remote load latency can be an interesting metric, because lowering it implies less effort in software and less hardware resources for latency hiding. We split the total latency into two components: (1) *intrinsic* latency is the minimum amount of latency when there is no congestion (i.e. zero-load latency). (2) *congestion* latency is any extra latency induced by network stalls. Figure 12 shows the average remote load latency of 32×16 array. The average intrinsic delay is almost identical for all benchmarks, suggesting

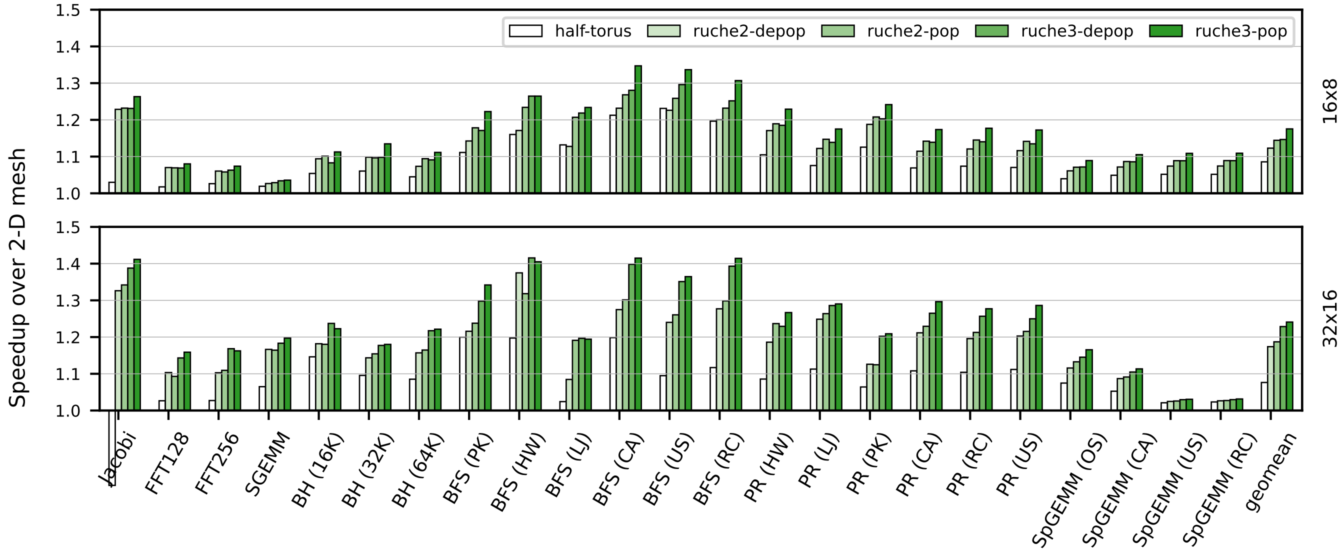


Figure 10: Speedup over 2-D mesh on 32×16 and 16×8. Half Ruche provides consistent speedup across a wide range of parallel benchmarks. The most gain comes with the Half Ruche in its simplest form (ruche2-depop), and the rest provides incremental gains. Half Ruche provides more speedup in a larger network size. Folded torus underperforms 2-D mesh in Jacobi due to its inability to reach the nearest tiles using local links.

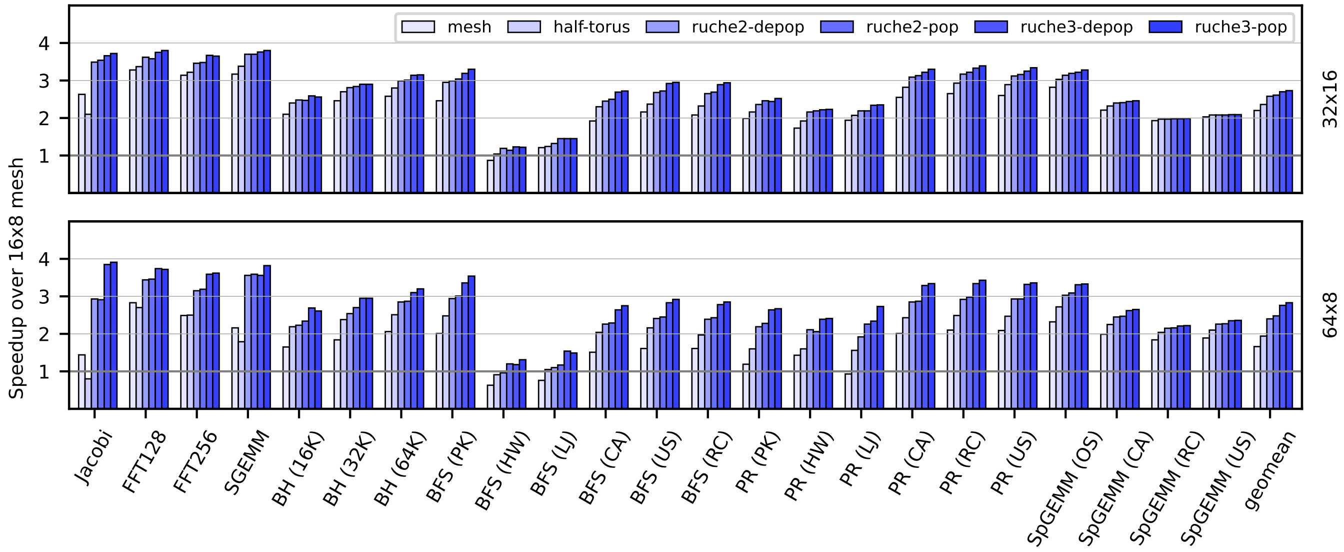


Figure 11: Here we define "scalability" as how much speedup we gain by adding more cores. In ideal scaling, 4× speedup would be the upper bound. Half Ruche helps reach this ceiling across all benchmarks, especially for the ones with good parallelism. Despite its skewed aspect ratio, 64×8 performs marginally better than 32×16 in terms of geomean speedup, when the Ruche Factor is 3.

that the IPOLY hashing [28] that is used to hash the address space to interleave among the LLC banks effectively balances the traffics. In all benchmarks, both half-torus and Half Ruche reduce the intrinsic delay (except for half-torus in Jacobi, as explained earlier). ruche2-depop reduces intrinsic latency by ~27%, and more expensive routers provide only incremental benefits. Congestion-induced latency is especially high for regular workloads and PageRank with social networks, where injection rates are very high. SpGEMM (US, RC) are characterized as being latency-bound and pointer-chasing, because road networks in general have low degrees, and the SpGEMM kernels are implemented using linked lists. Due to latency reduction by Ruche links, cores stall less often and consequently inject packets more frequently; as a result, congestion latency increased slightly. Half Ruche effectively reduce congestion by providing more network links that cross the bisection. For some

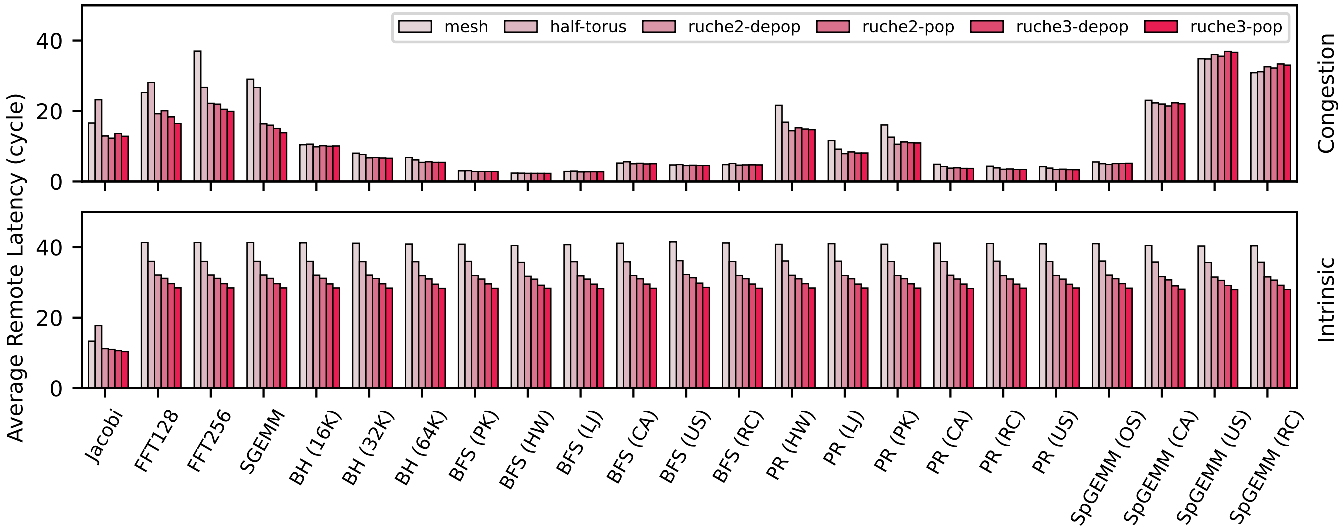


Figure 12: Average remote load latency for 32×16. Ruche Networks effectively reduce both 'intrinsic' and 'congestion-induced' latencies. In 2-D mesh, workloads with a stream of sequential accesses (e.g. FFT, SGEMM) suffer the most from congestion. Ruche Networks help by relieving the bisection bottleneck.

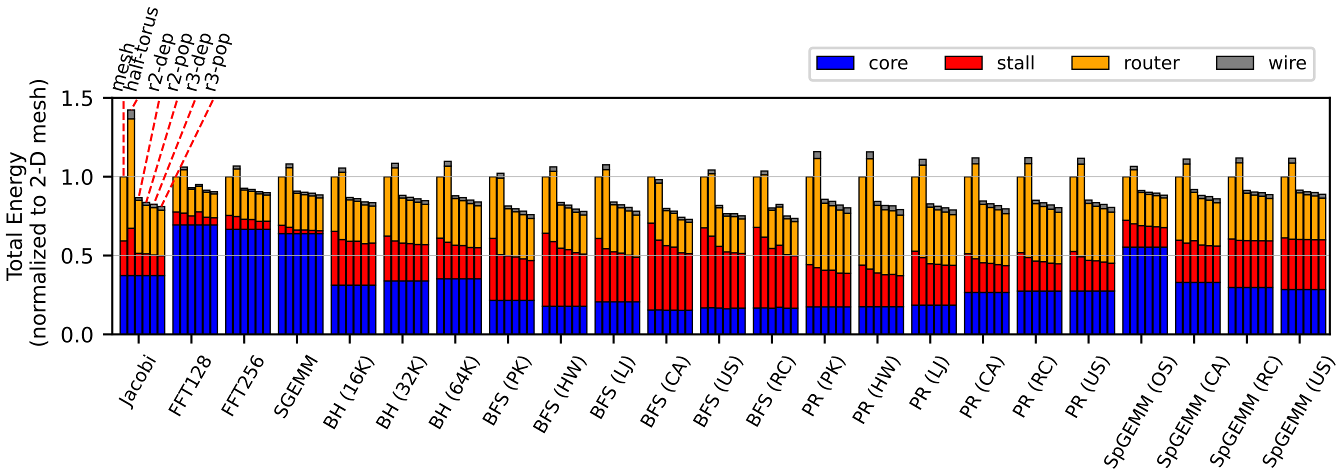


Figure 13: Total energy breakdown for 32×16 (normalized to 2-D mesh). Half Ruche effectively reduces router energy by reducing the network hops with the long-range Ruche channels. Sending packets over those long-range links are much more energy efficient than the local links. By reducing the number of cycles that the cores stall (e.g. remote load latency), it also reduces the static stall energy wasted by the idle cores. Half-torus increases the total energy due to its higher router energy.

workloads, higher Ruche Factor or populated routers does not significantly reduce congestion-induced latency. However, congestion latency is never made worse by adding Ruche channels, so when the reduction of intrinsic latency is taken into account, the total latency is almost likely to be reduced.

4.9 Half Ruche - Benchmark Energy

We divide the total energy spent by the whole system into four categories: (1) *core energy* represents dynamic energy dissipated by cores when executing instructions. Per-instruction energy comes from the measurement taken in [16]. (2) *stall energy* represents a leakage component of both core and router and some ungated dynamic clock tree energy, when the core and NoC are idle. (3) *router energy* represents the dynamic energy dissipated by the NoC routers when transmitting packets. Router energy is calculated by using the similar method to generate Table 3. (4) *wire energy* represents the dynamic energy dissipated by the long-range Ruche links. Wire energy of Ruche links is estimated by using the first-order repeater model [13] and the process-independent, per-length wire capacitance (0.2 pF/mm). The diffusion and gate capacitance of the repeaters driving the Ruche links was taken from the 12 nm standard-cell library. We made the same assumption about the activity factor on each bit and the length of the Ruche wires, as done in Table 3.

Figure 13 shows the total energy breakdown for 32×8 (normalized to energy spent by 2-D mesh). As expected, the core energy remains constant, since the instruction execution count is not affected by changes in network configurations. However, by reducing the remote load latency, the stall energy is reduced. For memory-bound workloads like BFS, stall energy can become a large portion of the total, even though stall energy per cycle is relatively small compared to energy per instruction. Except for a few compute-intensive workloads (e.g. FFT and SGEMM), routers can dissipate power as much as cores do. In half-torus, we observe that energy saving with long wire is not enough to compensate for the increased router energy. In almost all benchmarks, half-torus ends up spending more total energy than 2-D mesh. Again, while ruche2-depop results in the sharpest reduction in energy, the fully-populated routers and higher Ruche Factors provide marginal benefits. Even for Ruche Factor of 3, wire energy is a very small percentage of the total energy.

4.10 Half Ruche - Geomean Summary

Table 6 summarizes the Half Ruche evaluation with geomean scores. Across the board, ruche3-pop outperforms all other network configurations. As mentioned earlier, the general trend is that most gains come from ruche2-depop initially. This is a promising result, since most designs that already use 2-D mesh or torus can easily reap most of the benefits with a Ruche router with the least complexity. Looking at the area-normalized speedup, depopulated routers perform better than the fully-populated. Increasing Ruche Factor is a cost-effective means to increase performance. On the other hand, there is hardly any area-normalized speedup for half-torus (1.01×). We can summarize the results from Half Ruche evaluation with a following guideline: *save area with a depopulated router, and use longer wires for more cost-effective performance gains.*

Although Ruche NoC reduces the total NoC energy, the question still remains whether *NoC power* also decreases. If we define 'NoC power' as NoC energy divided by total runtime, as long as the NoC energy efficiency does not fall behind the speedup, NoC power should remain constant. Table 6 shows that NoC energy efficiency for Ruche NoCs is greater than speedup vs mesh for 32×16 by at least 0.07×; therefore, NoC power does indeed decrease. Similarly, *total power*, defined as total energy divided by total runtime, does not increase significantly for Ruche NoCs, since the difference between total energy efficiency and speedup vs mesh is very small (0.01-0.03×).

5 Related Work

This paper discusses in detail comparisons of Ruche networks with a wide variety of other proposed NoCs [1, 3, 6, 12, 17, 26, 32].

Generalized Express Cube (GEC) framework [12] proposes to express existing NoC topologies using the 6-tuple $\langle n, k, c, o, d, x \rangle$. However, these parameters are not general enough to express Ruche networks. The parameters $\langle n, k, c, x \rangle$ are orthogonal to describing the connectivity between nodes in each dimension. The parameters $\langle o, d \rangle$, which describe the router radix per dimension and sinks per channel, would be $\langle 4, 1 \rangle$ for Ruche networks. However, it lacks a parameter to describe the Ruche Factor (the skip distance of Ruche channels), so even the most broad generalization of express link topologies does not include Ruche networks.

Table 6: Summary of Half Ruche eval using geomean scores.

Metric	mesh	ruche2 depop	ruche2 pop	ruche3 depop	ruche3 pop	half torus
16×8 Speedup vs mesh	1.00×	1.12×	1.14×	1.15×	1.18×	1.09×
32×16 Speedup vs mesh	1.00×	1.17×	1.19×	1.23×	**1.24**×	1.08×
32×16 Scalability (vs 16×8 mesh)	2.20×	2.58×	2.61×	2.70×	**2.73**×	2.36×
64×8 Scalability (vs 16×8 mesh)	1.66×	2.40×	2.48×	2.76×	**2.83**×	1.94×
Load Latency Reduction (32×16, Intrinsic)	1.00×	1.28×	1.32×	1.38×	**1.44**×	1.12×
Load Latency Reduction (32×16, Congestion)	1.00×	1.21×	1.19×	1.20×	**1.22**×	1.05×
Load Latency Reduction (32×16, Total)	1.00×	1.27×	1.30×	1.35×	**1.40**×	1.11×
Energy Efficiency (32×16, Compute)	1.00×	1.12×	1.12×	1.15×	**1.16**×	1.06×
Energy Efficiency (32×16, NoC)	1.00×	1.28×	1.28×	1.30×	**1.35**×	0.75×
Energy Efficiency (32×16, Total)	1.00×	1.18×	1.18×	1.20×	**1.22**×	0.91×
Tile Area Increase	1.000×	**1.058**×	1.085×	1.063×	1.090×	1.071×
32×16 Speedup vs mesh (area normalized)	1.00×	1.11 ×	1.10×	**1.16** ×	1.14×	1.01×

Express Virtual Channel (EVC) [20] is a *flow control technique* that allows packets to bypass some of the pipeline stages in intermediate routers to reduce latency. EVC uses the same 2-D mesh topology without adding any new physical links; hence, there is no improvement in bisection bandwidth, and the throughput will converge to that of 2-D mesh at higher load or larger network. Instead, the mesh links are *virtualized* by adding express VCs that are prioritized over local ones. This prioritization enables EVC flits to skip some of the pipeline stages (e.g. buffer write, allocation) in intermediate routers, thereby reducing their latencies. However, EVC flits still need to be latched, and go through each crossbar, in every router on their way. In contrast, Ruche packets use the long-range physical links that skip the intermediate routers entirely. Fundamentally, while previous NoC techniques, such as speculative VC routers and EVC, sought to improve network performance by spending more area on *control logic*, usually with a tradeoff between performance and energy, Ruche NoC finds a way to efficiently utilize unused interconnects to improve *both* performance and energy.

EVC has two major flaws that limit its scalability for larger networks [18, 19]. First, EVC must conservatively guarantee that there is enough buffer space at the destination before it can safely inject EVC flits, and this leads to overprovisioning and underutilization of buffer spaces in average case. EVC uses *on-off signaling*, where the destination node sends a token to the remote source to stop injecting more flits when the buffer space goes below a certain threshold. Since it takes multiple cycles for the token to reach the source, the total buffer space must account for the *worst-case*, which is calculated based on the number of cycles for this signal to reach the source and the number of flits that could already be in flight during that time. In other words, the minimum buffer space has to grow with the maximum EVC distance. In contrast, Ruche routers are minimally buffered with two-element FIFOs, which remains constant with the network sizes and Ruche Factor. Second, EVC is restricted to assign one express VC for each k-hop express paths, because control latency overhead required for dynamic assigning makes it impractical. Static assigning creates a problem, where the source node may only send a limited number of packets for a particular distance, even though there may be other free VCs available. Furthermore, the source nodes that are further away take longer time to learn from the destination that the *k*-hop VC became free, thereby lowering the VC utilization. For these reasons, the maximum EVC distance is practically limited to only 3 to 4 hops.

Flit bubble flow control (FBFC) [22] proposes an alternative scheme for deadlock freedom in torus networks without using virtual channels. The main insight is that, even if cyclic channel dependency exists, as long as there is one bubble per each ring, packets can make forward progress without deadlock. FBFC imposes a restriction on when packets can be injected; the receiving FIFO must have at least one more free buffer slot than the packet length.

Dalorex [24], a manycore architecture that accelerates workloads with irregular memory access patterns by remote task invocation, purports to have done a comparison between Ruche and torus networks using C++ simulation. While no exact detail was given about which Ruche Factor or crossbar schemes were used, it reports that Ruche has a larger area overhead and lower performance than torus. Figure 7 shows that 2-D torus has a larger area with virtual channels and depopulated Ruche routers in consideration. Dalorex distributes its data array using low-order index bits so all-to-all uniform random best reflects its traffic pattern. Figure 6 shows that Ruche networks achieve much higher throughput than 2-D torus in uniform random.

6 Conclusion

Ruche networks share the same qualities with 2-D mesh that make them widely usable in real chip designs. Although express (non-local) links themselves are not novel, and their benefits have been studied in the past, they have not been adopted in mainstream architecture because of the physical-design limitations. Our analysis reveals that the simplest form of Ruche networks (RF = 2, depopulated) is able to deliver the most benefit upfront. This suggests that existing architectures that already use 2-D mesh or torus can leverage Ruche networks at low cost, while unlocking most of the benefits from underutilized VLSI wiring resources. Once the initial overhead is added to the router, extending the range of Ruche channels provides a cost-effective mechanism for additional performance gains.

We evaluate Half and Full Ruche networks in two common scenarios. We demonstrate that Ruche networks can improve performance and scalability across a wide range of parallel workloads. They can reduce the average remote load latency so that less software effort or hardware resources are required to hide latency. Finally, sending packets over the long Ruche channels is significantly more energy efficient than hopping through routers. This paper provides insights on important tradeoffs when designing with Ruche networks.

Acknowledgments

This work was supported in part by ACE and CHIMES, two of the seven centers in JUMP 2.0, a Semiconductor Research Corporation (SRC) program sponsored by DARPA, and by NSF PPoSS Award 2118628.

References

[1] Nilmini Abeyratne, Reetuparna Das, Qingkun Li, Korey Sewell, Bharan Giridhar, Ronald G. Dreslinski, David Blaauw, and Trevor Mudge. 2013. Scaling towards kilo-core processors with asymmetric high-radix topologies. In *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*. 496–507.

[2] Dennis Abts, Natalie D. Enright Jerger, John Kim, Dan Gibson, and Mikko H. Lipasti. 2009. Achieving predictable performance through better memory controller placement in many-core CMP. In *Proceedings of the 36th Annual International Symposium on Computer Architecture* (Austin, TX, USA) *(ISCA '09)*. Association for Computing Machinery, New York, NY, USA, 451–461.

[3] James Balfour and William J. Dally. 2006. Design tradeoffs for tiled CMP on-chip networks. In *ACM International Conference on Supercomputing 25th Anniversary Volume* (Munich, Germany). Association for Computing Machinery, New York, NY, USA, 390–401.

[4] Arnab Banerjee, Pascal T. Wolkotte, Robert D. Mullins, Simon W. Moore, and Gerard J. M. Smit. 2009. An Energy and Performance Exploration of Network-on-Chip Architectures. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 17, 3 (2009), 319–329.

[5] Daniel U Becker. 2012. *Efficient microarchitecture for network-on-chip routers*. Ph. D. Dissertation. Stanford University.

[6] Daniel U. Becker and William J. Dally. 2009. Allocator Implementations for Network-on-Chip Routers. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis* (Portland, Oregon) *(SC '09)*. Association for Computing Machinery, New York, NY, USA, Article 52, 12 pages.

[7] Shekhar Borkar. 2007. Thousand core chips: a technology perspective. In *Proceedings of the 44th Annual Design Automation Conference* (San Diego, California) *(DAC '07)*. Association for Computing Machinery, New York, NY, USA, 746–749.

[8] Dally and Seitz. 1987. Deadlock-Free Message Routing in Multiprocessor Interconnection Networks. *IEEE Trans. Comput.* C-36, 5 (1987), 547–553.

[9] Scott Davidson, Shaolin Xie, Christopher Torng, Khalid Al-Hawai, Austin Rovinski, Tutu Ajayi, Luis Vega, Chun Zhao, Ritchie Zhao, Steve Dai, Aporva Amarnath, Bandhav Veluri, Paul Gao, Anuj Rao, Gai Liu, Rajesh K. Gupta, Zhiru Zhang, Ronald Dreslinski, Christopher Batten, and Michael Bedford Taylor. 2018. The Celerity Open-Source 511-Core RISC-V Tiered Accelerator Fabric: Fast Architectures and Design Methodologies for Fast Chips. *IEEE Micro* 38, 2 (2018), 30–41.

[10] Timothy A. Davis and Yifan Hu. 2011. The University of Florida Sparse Matrix Collection. *ACM Trans. Math. Softw.* 38, 1, Article 1 (dec 2011), 25 pages.

[11] David R. Ditzel and the Esperanto team. 2022. Accelerating ML Recommendation With Over 1,000 RISC-V/Tensor Processors on Esperanto's ET-SoC-1 Chip. *IEEE Micro* 42, 3 (2022), 31–38.

[12] Boris Grot, Joel Hestness, Stephen W. Keckler, and Onur Mutlu. 2009. Express Cube Topologies for on-Chip Interconnects. In *2009 IEEE 15th International Symposium on High Performance Computer Architecture*. 163–174.

[13] R. Ho, K.W. Mai, and M.A. Horowitz. 2001. The future of wires. *Proc. IEEE* 89, 4 (2001), 490–504.

[14] Drago Ignjatović, Daniel W. Bailey, and Ljubisa Bajić. 2022. The Wormhole AI Training Processor. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, Vol. 65. 356–358.

[15] Dai Cheol Jung, Scott Davidson, Chun Zhao, Dustin Richmond, and Michael Bedford Taylor. 2020. Ruche Networks: Wire-Maximal, No-Fuss NoCs. In *2020 14th IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*. 1–8.

[16] Dai Cheol Jung, Max Ruttenberg, Paul Gao, Scott Davidson, Daniel Petrisko, Kangli Li, Aditya K Kamath, Lin Cheng, Shaolin Xie, Peitian Pan, Zhongyuan Zhao, Zichao Yue, Bandhav Veluri, Sripathi Muralitharan, Adrian Sampson, Andrew Lumsdaine, Zhiru Zhang, Christopher Batten, Mark Oskin, Dustin Richmond, and Michael Bedford Taylor. 2024. Scalable, Programmable and Dense: The HammerBlade Open-Source RISC-V Manycore. In *International Symposium on Computer Architecture (ISCA)*.

[17] John Kim, James Balfour, and William Dally. 2007. Flattened Butterfly Topology for On-Chip Networks. In *40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 2007)*. 172–182.

[18] Tushar Krishna, Amit Kumar, Patrick Chiang, Mattan Erez, and Li-Shiuan Peh. 2008. NoC with Near-Ideal Express Virtual Channels Using Global-Line Communication. In *2008 16th IEEE Symposium on High Performance Interconnects*. 11–20.

[19] Tushar Krishna, Amit Kumar, Li-Shiuan Peh, Jacob Postman, Patrick Chiang, and Mattan Erez. 2009. Express Virtual Channels with Capacitively Driven Global Links. *IEEE Micro* 29, 4 (2009), 48–61.

[20] Amit Kumar, Li-Shiuan Peh, Partha Kundu, and Niraj K. Jha. 2007. Express virtual channels: towards the ideal interconnection fabric. In *Proceedings of the 34th Annual International Symposium on Computer Architecture* (San Diego, California, USA) *(ISCA '07)*. Association for Computing Machinery, New York, NY, USA, 150–161.

[21] Pejman Lotfi-Kamran, Boris Grot, and Babak Falsafi. 2012. NOC-Out: Microarchitecting a Scale-Out Processor. In *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*. 177–187.

[22] Sheng Ma, Zhiying Wang, Zonglin Liu, and Natalie Enright Jerger. 2015. Leaving One Slot Empty: Flit Bubble Flow Control for Torus Cache-Coherent NoCs. *IEEE Trans. Comput.* 64, 3 (2015), 763–777.

[23] Robert Mullins, Andrew West, and Simon Moore. 2006. The design and implementation of a low-latency on-chip network. In *Proceedings of the 2006 Asia and South Pacific Design Automation Conference* (Yokohama, Japan) *(ASP-DAC '06)*. IEEE Press, 164–169.

[24] Marcelo Orenes-Vera, Esin Tureci, David Wentzlaff, and Margaret Martonosi. 2023. Dalorex: A Data-Local Program Execution and Architecture for Memory-bound Applications. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. 718–730.

[25] Yanghui Ou, Shady Agwa, and Christopher Batten. 2020. Implementing Low-Diameter On-Chip Networks for Manycore Processors Using a Tiled Physical Design Methodology. In *2020 14th IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*. 1–8.

[26] L.-S. Peh and W.J. Dally. 2001. A delay model and speculative architecture for pipelined routers. In *Proceedings HPCA Seventh International Symposium on High-Performance Computer Architecture*. 255–266.

[27] Daniel Petrisko, Chun Zhao, Scott Davidson, Paul Gao, Dustin Richmond, and Michael Bedford Taylor. 2020. NoC Symbiosis (Special Session Paper). In *2020 14th IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*. 1–8.

[28] B Ramakrishna Rau. 1991. Pseudo-randomly interleaved memory. In *Proceedings of the 18th Annual International Symposium on Computer Architecture*. 74–83.

[29] Daeho Seo, Akif Ali, Won-Taek Lim, and N. Rafique. 2005. Near-optimal worst-case throughput routing for two-dimensional mesh networks. In *32nd International Symposium on Computer Architecture (ISCA'05)*. 432–443.

[30] Emil Talpes, Debjit Das Sarma, Doug Williams, Sahil Arora, Thomas Kunjan, Benjamin Floering, Ankit Jalote, Christopher Hsiong, Chandrasekhar Poorna, Vaidehi Samant, John Sicilia, Anantha Kumar Nivarti, Raghuvir Ramachandran, Tim Fischer, Ben Herzberg, Bill McGee, Ganesh Venkataramanan, and Pete Banon. 2023. The Microarchitecture of DOJO, Tesla's Exa-Scale Computer. *IEEE Micro* 43, 3 (2023), 31–39.

[31] Michael Bedford Taylor. 2018. BaseJump STL: SystemVerilog needs a Standard Template Library for Hardware Design. In *Proceedings of the 55th Annual Design Automation Conference* (San Francisco, California) *(DAC '18)*. Association for Computing Machinery, New York, NY, USA, Article 73, 6 pages.

[32] Jasmina Vasiljevic and Davor Capalija. 2024. Blackhole & TT-Metalium: The Standalone AI Computer and its Programming Model . In *2024 IEEE Hot Chips 36 Symposium (HCS)*. IEEE Computer Society, Los Alamitos, CA, USA, 1–30.