Appears in the Proceedings of the Int'l Symp. on Networks-on-Chips (NOCS-14), September 2020

Implementing Low-Diameter On-Chip Networks for Manycore Processors Using a Tiled Physical Design Methodology

Special Session Paper

Yanghui Ou, Shady Agwa, Christopher Batten
School of Electrical and Computer Engineering, Cornell University, Ithaca, NY
{ yo96, sr972, cbatten }@cornell.edu

***Abstract*—Manycore processors are now integrating up to 1000 simple cores into a single die, yet these processors still rely on high-diameter mesh on-chip networks (OCNs) without complex flow-control nor custom circuits due to three reasons: (1) manycores require simple, low-area routers; (2) manycores usually use standard-cell-based design; and (3) manycores use a tiled physical design methodology. In this paper, we explore mesh and torus topologies with internal concentration and/or ruche channels that require low area overhead and can be implemented using a traditional standard-cell-based tiled physical design methodology. We use a combination of analytical and RTL modeling along with layout-level results for both hard macros and a 3×3 mm 256-terminal OCN in a 14-nm technology for twelve topologies. Critically, the networks we study use a tiled physical design methodology meaning they: (1) tile a homogeneous hard macro across the chip; (2) implement chip top-level routing between hard macros via short wires to neighboring macros; and (3) use timing closure for the hard macro to quickly close timing at the chip top-level. Our results suggest that a concentration factor of four and a ruche factor of two in a 2D-mesh topology can reduce latency by over 2× at similar area and bisection bandwidth for both small and large messages compared to a 2D-mesh baseline.**

I. Introduction

Today's network, embedded, and server processors already integrate tens of processor cores on a single chip, and there is growing interest in using a *manycore approach* to integrate an even larger number of relatively simple cores within a single die. Early manycore research prototypes included 16–110 cores [13,14,22,23,32], complemented by manycore processors in industry with 64–128 cores [3,12,30,33,34]. Recent research prototypes have scaled core counts by an order-of-magnitude including the 496-core Celerity [28], 1000-core KiloCore [5], and 1024-core Epiphany-V [26]. The manycore approach has demonstrated significant improvements in energy efficiency and throughput per unit area for highly parallel workloads.

Almost all manycore processors use a simple 2D-mesh on-chip-network (OCN) topology [3, 5, 12, 22, 23, 28, 33, 34] (possibly with limited external concentration [13, 30]), scaling from a 4×4 mesh in the RAW processor [32] up to a 32×32 mesh in the Epiphany-V processor [26]. It is well known that the high diameter of 2D-mesh topologies can significantly increase packet latency and thus reduce system-level performance [8]. Indeed, there is a rich body of literature proposing numerous techniques to reduce packet latency in on-chip networks. Novel OCN flow-control schemes [20,25,27] and/or OCN custom circuits [6, 18] can be used to reduce router and channel latencies. Alternatively, novel OCN topologies can reduce the network diameter including concentrated mesh [2], fat-tree [2], flattened butterfly [19], multi-drop express channels [10, 11], Clos [17], Slim NoC [4], and asymmetric high-radix topologies [1]. However, this raises the question: **Why do manycore processor silicon implementations continue to use simple high-diameter on-chip networks given the potential benefit reported in the literature for adopting novel on-chip network flow-control schemes, custom circuits, and/or topologies?**

Based on our experiences contributing to the Celerity manycore processor [9,28,29] and building an open-source OCN generator [31], we argue there are three primary reasons for this gap between principle and practice.

Manycores Require Simple, Low-Area Routers – Manycore processors by definition use simple cores leaving modest area for the OCN routers (e.g., 10% of chip area in [26, 28]). Therefore, manycore processors usually use single-stage routers [5, 13, 26, 28], and protocol deadlock is often through multiple physical networks [22, 23, 32, 33] as opposed to using virtual channels. These simple single-stage OCN routers mitigate the need for complex flow-control schemes.

Manycores Use Standard-Cell-Based Design – Manycore processor design teams (and indeed chip design in general) have been steadily moving towards highly automated standard-cell-based design methodologies [22, 23, 26, 28]. Unfortunately, this complicates using more advanced circuit techniques in the literature to reduce router and/or channel latency.

Manycores Use a Tiled Physical Design Methodology – Physical design is a critical challenge in implementing manycore processors. A tiled physical design methodology is the key to overcoming this challenge and has been used in multiple manycore implementations [22, 23, 26, 28]. A *tiled physical design methodology* adheres to the following constraints: (1) the design is based on tiling a homogeneous hard macro across the chip; (2) all chip top-level routing between hard macros must use short wires to neighboring macros; and (3) timing closure for the hard macro must imply timing closure at the chip

This work was supported in part by NSF CRI Award #1512937, DARPA POSH Award #FA8650-18-2-7852, DARPA SDH Award #FA8650-18-2-7863, the Center for Applications Driving Architectures (ADA), one of six centers of JUMP, a Semiconductor Research Corporation program co-sponsored by DARPA, and equipment, tool, and/or physical IP donations from Intel, Synopsys, Cadence, and ARM. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of AFRL, DARPA, or the U.S. Government.

978-1-4673-9030-9/20/$31.00 ©2020 IEEE

top-level. Unfortunately, a tiled physical design methodology precludes using many low-diameter, high-radix topologies proposed in the literature which require long global channels routed at the chip top-level and/or heterogeneous hard macros.

In this paper, we seek to close this gap between principle and practice by exploring techniques for implementing low-diameter on-chip networks for manycore processors based on low-area routers, standard-cell-based design, and a tiled physical design methodology. Section II describes how mesh and torus topologies with concentration and/or ruche channels can use a tiled physical design methodology. Ruche channels[1] are a novel technique concurrently proposed in this work and by Jung et. al in [16] which provide dedicated channels for packets to skip past routers for efficient long distance communication. Ruche channels are better suited to on-chip networks using a traditional standard-cell-based physical design methodology compared to prior work on physical and virtual express channels [7, 10, 20]. Section III compares 12 topologies using an analytical model based on router and channel RTL implementations and a standard-cell-based flow. Section IV uses PyOCN (an open-source OCN generator [31]) to generate both hard-macro and full-chip layout for each topology suitable for use in a 3×3 mm 256-core manycore processor implemented in a 14-nm technology. Our results suggest that by leveraging a concentration factor of four and a ruche factor of one in a 2D-mesh topology, our approach can reduce latency by over 2× at similar area and bisection bandwidth for both small and large messages compared to a 2D-mesh baseline.

II. Manycore OCN Topologies

Our target system is a manycore with 256 cores arranged in a 16×16 grid (see Figure 1(a)). Figure 1(b–m) illustrates the 12 topologies explored in this work. Figure 1(b) illustrates our baseline 2D-mesh topology as implemented in most state-of-the-art manycore processors [3, 5, 12, 22, 23, 28, 33, 34]. We use elastic-buffer flow-control [24] and dimension-ordered routing on all mesh topologies.

We explore internal concentration where multiple cores share a single router [21]. Figure 1(c–d) illustrates a concentration factor of 4–8. Unlike external concentration, internal concentration reduces latency while maintaining per-terminal throughput and homogeneous channel bandwidths. Concentration reduces the number of routers, increases router radix, decreases the bisection channel count, and reduces network diameter.

Ruche channels are a novel technique which add dedicated channels to skip past some number of routers. A *ruche factor* of two means each ruche channel skips to a router two hops away, while a ruche factor of three means each ruche channel skips to a router three hops away. A ruche factor of zero means there are no ruche channels, and a ruche factor of one means the ruche channel directly connects nearest neighbors. Figure 1(e,h) illustrates a ruche factor of two and three. Ruche channels maintain the number of routers, increase router radix, increase the bisection channel count, and reduce network diameter. Ruche channels are related to but distinct from express channels [7, 10]. Ruche channels do not use separate interchanges and ensure all routers are homogeneous (i.e., all routers are a source and destination for exactly one ruche channel, ruche channels overlap). We use oblivious minimal routing on the ruche channels.

Figure 1(f–g,i–j) illustrates topologies that combine concentration and ruche channels. Finally, we explore 2D-torus topologies with similar concentration factors (see Figure 1(k–m)). We use minimal routing, credit-based flow-control, two virtual channels, and a dateline to avoid deadlock in the torus topologies. These 12 topologies provide a broad range of design points with different: topology styles (mesh/torus), numbers of routers, router radix, bisection channel count, channel lengths, and diameter. Figure 1 shows the naming convention we will use in the rest of the paper. For example, *mesh-c4r3* refers to a topology with a concentration factor of four and ruche factor of three. We will also use suffix such as *mesh-c4r3-b128* to refer to a topology with a channel bandwidth of 128 b/cycle.

Figure 1(n–q) illustrates how to map these 12 topologies to a tiled physical design methodology. Mapping *mesh-c1r0* simply requires careful placement of the pins for north, west, south, and east channels at the macro level to ensure short chip top-level routes (see Figure 1(n)). If $l \times l$ are the dimensions of each macro, then the channels in *mesh-c1r0* are approximately l long. Since all macros must be homogeneous, macros on the edge and corners require a few gates at the chip top-level to ensure input channels are never enabled and output channels are never ready. Mapping *mesh-c1r2* requires an additional set of north, west, south, and east channels, along with a set of feed-through channels (see Figure 1(o)). Again, careful placement of pins ensure short routes with a possible cross-over at the chip top-level. Ruche channels are approximately $2l$ long. Mapping *mesh-c1r3* requires an additional set of feed-through channels (see Figure 1(p)). Ruche channels are now approximately $3l$ long. Finally, mapping *torus-c1r0* requires just one set of north, west, south, and east channels along with one set of feed-through channels. Most channels are approximately $2l$ long, although the channels at the edges may be slightly shorter or longer due to the chip top-level wrap-around routing. While adding ruche channels to torus topologies is possible, it can be challenging to map these ruche channels into a tiled design methodology and/or to route on these topologies. Figures 1(r–t) illustrates floorplans which enable using the tiled physical designs in Figures 1(n–q) at higher concentration factors.

Our approach meets the three constraints of a tiled physical design methodology. First, all topologies can be implemented using a homogeneous hard macro which can then be tiled across the chip. Second, all chip top-level routing between hard macros is either short straight routing, short cross-over routing, or short wrap-around routing. Third, assuming careful consideration of timing constraints on register-to-output, input-to-register, and input-to-output paths, timing closure for all hard macros can imply timing closure at the chip top-level.

III. Manycore OCN Analyatical Modeling

In this section, we explore tradeoffs across different topologies using analytical modeling before presenting more realistic layout-level results in Section IV. To choose an appropriate core area, we implemented a RISC-V RV32IMAF in-order

[1] Ruching involves gathering fabric in a repeating pattern to make a pleat or ruffle. The logical topology diagram for mesh networks with ruche channels (see Figure 1(e)) resembles a ruched garment.

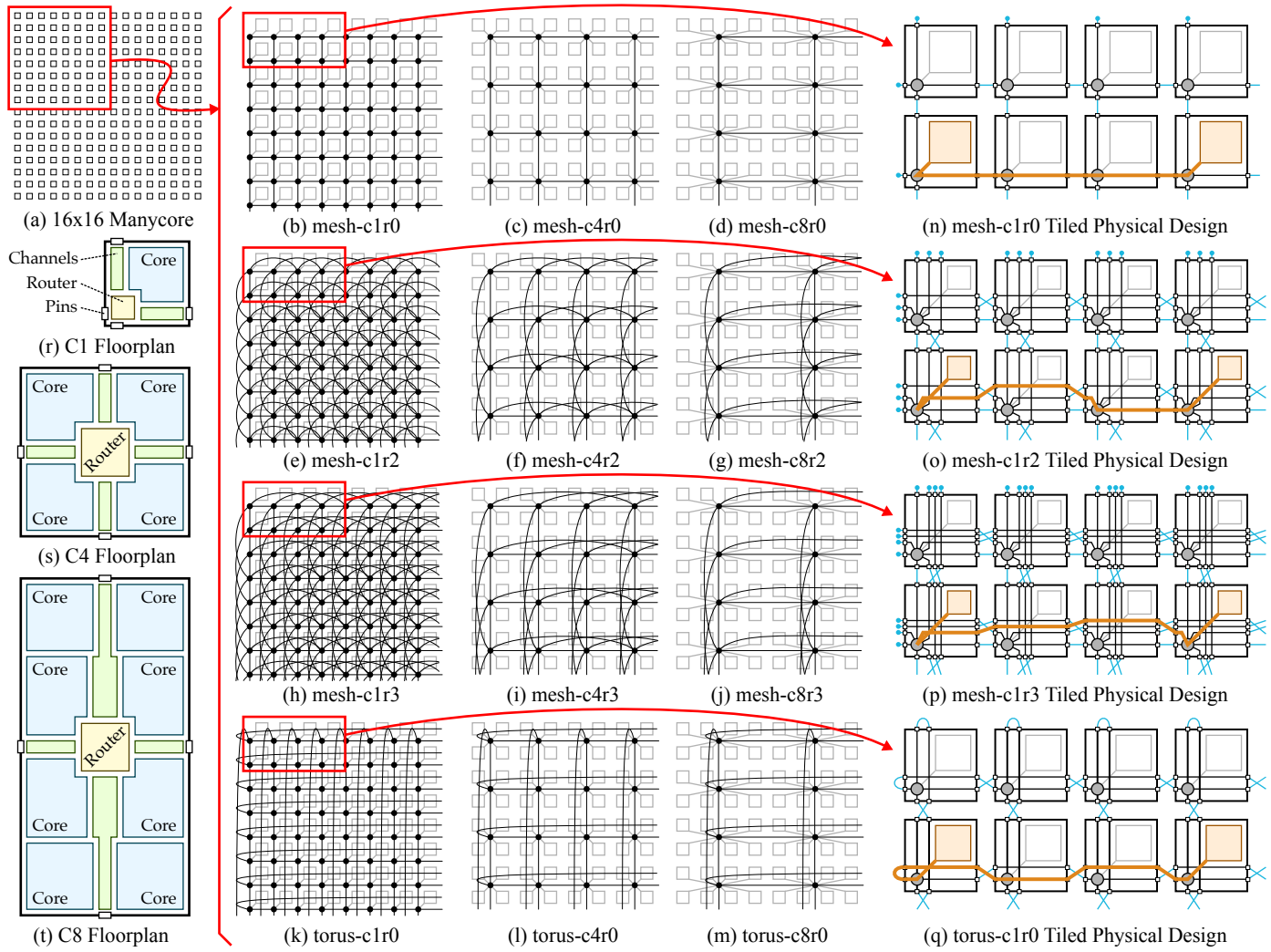

Figure 1. Twelve Topologies Implemented Using a Tiled Physical Design Methodology – (a) 16×16 manycore; (b–d) mesh with increasing concentration; (e–g) mesh w/ ruche factor of 2, increasing concentration; (h–j) mesh w/ ruche factor of 3, increasing concentration; (k–m) torus w/ increasing concentration; (n) = *mesh-c1r0* pin placement enables short chip top-level routing, unused channels terminated at top level; (o) = *mesh-c1r2* tile w/ feed-through channel, short cross-over chip top-level routing; (p) = *mesh-c1r3* tile w/ two feed-through channels, short cross-over chip top-level routing; (q) = *torus-c1r0* tile w/ folded torus, one feed-through channel, short cross-over chip top-level routing; (r–t) = macro floorplans for increasing concentration.

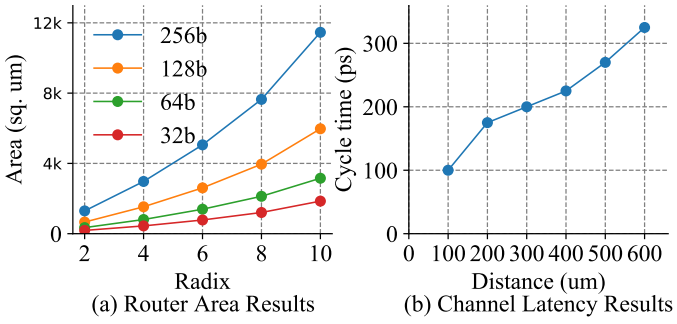

Figure 2. OCN Component-Level Results – (a) router area with different radices and port bitwidths under a 950 ps clock constraint; (b) minimum cycle time that can be achieved for queues manually placed at various distances with an auto-routed channel between them.

TABLE I. ANALYTICAL MODELING RESULTS

						B_B (Kb/cycle)				Area (%)			
Topology	N_R	r	N_{BC}	H_D	H_{avg}	32	64	128	256	32	64	128	256
mesh-c1r0	256	5	32	60	21.3	1	2	4	8	4.4	8.9	15.2	27.6
mesh-c4r0	64	8	16	28	10.5	0.5	1	2	4	2.0	3.9	7.3	13.6
mesh-c8r0	32	12	8	20	7.8	0.3	0.5	1	2	2.0	3.6	6.8	12.8
mesh-c1r2	256	9	96	32	11.6	3	6	12	24	11.6	20.6	36.5	61.2
mesh-c4r2	64	12	48	16	6.3	1.5	3	6	12	5.4	9.8	18.3	32.9
mesh-c8r2	32	16	24	12	4.9	0.8	1.5	3	6	4.6	8.1	15.5	28.1
mesh-c1r3	256	9	128	24	9.7	4	8	16	32	13.8	24.7	43.4	71.5
mesh-c4r3	64	12	64	12	6.0	2	4	8	16	6.7	12.2	22.6	40.5
mesh-c8r3	32	16	32	12	5.7	1	2	4	8	5.2	9.6	18.0	32.8
torus-c1r0	256	5	64	32	16.0	2	4	8	16	6.9	12.9	23.6	42.2
torus-c4r0	256	5	32	16	8.0	1	2	4	8	3.3	6.2	11.7	21.8
torus-c8r0	256	5	16	12	6.0	0.5	1	2	4	2.9	5.4	10.3	19.2

N_R = number of routers; r = router radix (i.e., number of ports per router); N_{BC} = number of bisection channels; H_D = diameter of the network; H_{avg} = average hop latency over all source/destination pairs; B_B = bisection bandwidth; Area = OCN area as a percentage of the full chip.

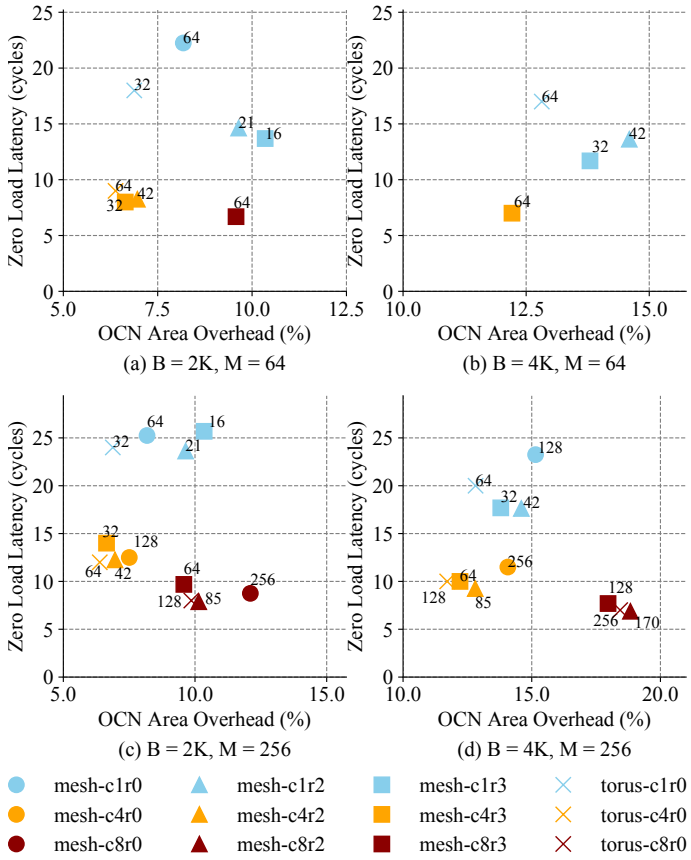

Figure 3. Latency and Area Tradeoffs – The zero-load latency and area overhead are compared for both 64-bit message and 256-bit message, with the bisection bandwidth normalized to 2 Kb/cycle and 4 Kb/cycle. Each topology is labeled with the channel bandwidth that corresponds to the normalized bisection bandwidth. Topologies that cannot reach the given bisection bandwidth even with channel bandwidth equal to the message size are not shown in the plot. B = normalized bisection bandwidth in bits per cycle; M = message size in bits.

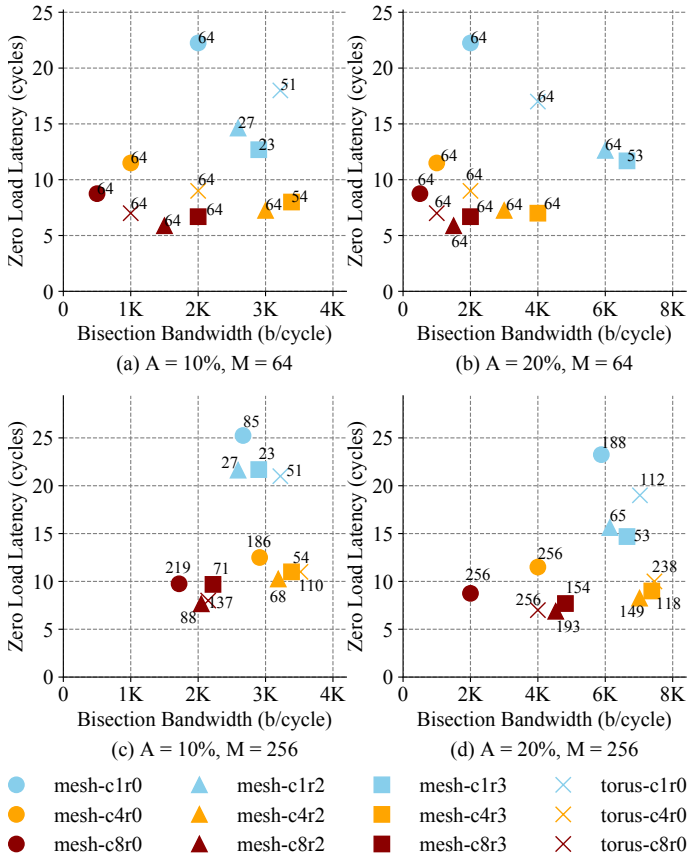

Figure 4. Latency and Bandwidth Tradeoffs – Zero-load latency and bisection bandwidth are compared for both 64-bit messages and 256-bit messages, with maximum area overhead for the OCN constrained at 10% and 20% of the total chip area. Each topology is labeled with the corresponding channel bandwidth that either reaches the maximum overhead or reaches the message size. A = maximum OCN area overhead; M = message size in bits.

single-issue processor with 4KB instruction and data caches in RTL using PyMTL3 [15] and then used a commercial standard-cell-based toolflow in a 14-nm technology. The resulting area is 37,029 μm^2 which roughly corresponds to a 3×3 mm chip area for 256 cores. This per-core area is roughly $1.5\times$ larger than the per-core area in Celerity [28], but this is expected since Celerity does not support floating point and uses scratchpad memories instead of caches. Our per-core area is roughly $3.3\times$ smaller than the per-core area in Epiphany-V [26], but again this is expected since Epiphany-V implements a 64-bit instruction set, supports dual-issue, and includes 64 KB of SRAM per core. Ultimately, we chose a tile size of 185×185 µm which is a reasonable target in between prior manycore implementations. We target a 1 GHz clock frequency which is comparable to the Celerity clock frequency when running at nominal voltage [9, 28].

We construct an analytical model for area, zero-load latency, and bisection bandwidth based on the OCN component-level results shown in Figure 2. We model the channel latency as a function of distance between routers. We measured the minimum delay using static-timing analysis that can be achieved for two queues that are manually placed at various distances with an auto-routed channel between them. We use this data to estimate the number of channel queues that need to be inserted in each channel to meet the target 1 GHz clock frequency. We model the area of the OCNs as a function of router radix and channel bandwidth. We pushed a number of OCN routers from PyOCN [31] with different radices and port bitwidths through the ASIC toolflow using a 950 ps timing constraint. We use the post-place-and-route area information as an estimate of the buffering and switching logic in the router. With the pitch and minimum width information of the metal layers that are used for local routing, we can calculate the linear wire density and estimate the area that needs to be reserved for channel pins on each edge of the hard macro. Based on the floorplans shown in Figure 1(r–t), we can calculate the area that is used by the OCN (both for the router and channels). We also interpolate and extrapolate component-level results to estimate the area of a router with any given radix and bitwidth. We calculate the zero-load latency under uniform random traffic and bisection bandwidth as described in [8]. In the analytical model, we assume one-cycle router latency and at least one-cycle channel latency. Topologies with short channels and low radix can potentially achieve a zero-cycle channel latency (i.e., router buffering/arbi-

tration and channel traversal can be completed in a single cycle). However, this will also push the ASIC tool to use larger and faster cells which can increase area. We will explore the impact of zero-cycle channel latency in Section IV. The analytical modeling results are shown in Table I. Figure 3 shows the zero-load latency vs. area overhead for a fixed bisection bandwidth. Figure 4 shows the zero-load latency vs. bisection bandwidth for a fixed area overhead. Results are shown for both small messages (64b) suitable for scratchpad-based manycores with word accesses and large messages (256b) suitable for cache-based manycores with cacheline accesses.

Impact of Concentration – Concentration reduces the number of routers, increases router radix, decrease the number of bisection channels, and increases the channel length. The router area model in Figure 2(a) suggests higher-radix single-cycle routers are still very feasible for concentration factors of 4–8. Similarly, the channel latency model in Figure 2(b) also suggests that longer single-cycle channels are still very feasible for these concentration factors. Thus concentration reduces network diameter by reducing the number of hops while maintaining router and channel latency (see Table I). If we focus only on the mesh topology without ruche channels, then across all scenarios in Figures 3 and 4, increasing concentration reduces the zero-load latency. Concentration also reduces the number of bisection channels. This can reduce the bisection bandwidth for small messages since increasing the channel bandwidth beyond the message size has no benefit (see Figure 4(a–b)). However, for large messages, concentration can compensate by increasing the channel bandwidth (see Figure 4(c–d)). In terms of reducing latency, a concentration factor of four is more area efficient than a concentration factor of eight. The benefit from *c4* to *c8* is less significant compared to from *c1* to *c4*, which indicates that the area benefit from the reduction in the number of routers is outweighed by the increase in router area due to increased radix.

Impact of Ruche Channels – Ruche channels are long physical channels that aggressively bypass routers. Ruche channels maintain the number of routers, increase router radix, and increase the number of bisection channels. As with concentration, the router area and channel latency models suggest higher-radix single-cycle routers and longer single-cycle channels are still very feasible for ruche factors of 2–3 (with the possible exception of *mesh-c8r3* which requires ruche channels that are over 2 mm long). Even so, ruche channels do increase router area and each hard macro needs to reserve additional area to accommodate the feed-through channels. When area is constrained, adding ruche channels may require narrower channels which increase serialization latency. Compared to concentration, ruche channels are less area efficient in terms of reducing latency, but given a sufficient area budget, ruche channels can be effective. For example, in Figure 4(b), the channel bandwidth of all topologies are limited by the small message size and the area budget is relatively large, and thus ruche channels improve both latency and bisection bandwidth.

Combining Concentration and Ruche Channels – Concentration significantly reduces latency and area but decreases the number of bisection channels, while ruche channels increase the number of bisection channels but add area overhead. Thus combining concentration and ruche channels can provide additional benefits. With concentration, a ruche factor of two is a better tradeoff; increasing the ruche factor to three adds more area overhead with marginal benefit or even a negative impact on latency. We consider *mesh-c4r2* as a promising design point. It is always on or close to the Pareto-optimal frontier across all scenarios in Figures 3 and 4, except for Figure 3(b) where it cannot achieve a bisection bandwidth of 4096 b/cycle because the channel bandwidth is limited by the small message size.

TABLE II. POST-PLACE-AND-ROUTE MACRO RESULTS

			Positive Slack (ps)				Area Overhead (%)			
Topology	H_D	H_{avg}	32	64	128	256	32	64	128	256
mesh-c1r0	60	21.3	175	66	19	33	5.9	9.6	16.3	35.3
mesh-c1r0q0	30	10.6	0.5	46	–	–	7.9	13.8	–	–
mesh-c4r0	28	10.5	94	73	51	24	2.7	4.6	10.2	18.2
mesh-c4r2	16	6.3	39	49	18	–	6.4	10.7	22.5	–
torus-c1r0	32	16.0	174	19	139	–	8.6	17.7	37.6	–

H_D = diameter of the network; H_{avg} = average hop latency; Positive Slack = worst-case positive slack for all constrained paths given 1 ns chip-level target cycle time; Area = OCN area overhead as a percentage of the full chip. Designs that do not meet timing or have prohibitively high area overhead are not shown.

Torus Topologies – Practical on-chip torus topologies always use a folded torus to ensure all channels are similar in length. Compared to mesh topologies, torus topologies do not increase the router radix, but can still take advantage of longer single-cycle channels. In Figure 3, *torus-c1r0* consumes less area for the same bisection bandwidth compared to *mesh-c1r2* and *mesh-c1r3*. This trend is less obvious for topologies with concentration because the radix of the router is already relatively high for these topologies. While torus topologies are certainly competitive, concentrated mesh topologies with ruche channels provide higher bandwidth on short messages. Perhaps just as importantly, mesh topologies are simpler than the corresponding torus topologies which require multiple virtual channels to avoid deadlock and use more complicated routing logic.

Summary – Concentration is very effective in reducing area overhead and zero-load latency but may reduce the bisection bandwidth at high concentration factors and thus limit overall throughput. Ruche channels, on the other hand, reduce the average hop count and increase the number of bisection channels but may require narrower channels due to the area overhead that comes from more feed-through channels and higher radix routers. Combining concentration and ruche channels provides an elegant hybrid solution. We find that *mesh-c4r2* is a promising topology. According to our analysis, *mesh-c4r2* dominates the baseline *mesh-c1r0* in zero-load latency, area, and bandwidth under different area constraints or bisection bandwidth constraints for both small and large message sizes.

IV. MANYCORE OCN PHYSICAL DESIGN

Based on the results from analytical modeling, we selected a set of promising topologies with different channel bandwidths for macro-level physical analysis. We used PyOCN to generate the hard macro as well as the full-chip layout. PyOCN is a unified Python-based framework for modeling, testing, and evaluating on-chip networks [31]. We pushed each design through the ASIC toolflow multiple times and recorded the minimum area that meets all timing constraints. We also experimented

with zero-cycle channel latencies for each topology (i.e., removing the channel queue so router buffering/arbitration and channel traversal take one cycle). We found that *mesh-c1r0* is the only topology that can achieve zero-cycle channel latency without introducing substantial area overhead. We will use *mesh-c1r0q0* to indicate a *mesh-c1r0* topology with zero-cycle channel latency. We carefully floorplan the macro and place the pins to enable short chip top-level routing (see Figure 1(r–t) and Figure 6). We use "dummy cores" to connect to the injection and ejection ports of the router to queues to prevent the ASIC toolflow from optimizing away any logic and to accurately model terminal channel latencies. We create a hard fence for each dummy core so that the router cannot place any cells into the area that is reserved for the actual processing cores. We also place routing blockages on top of the fences to prevent the router from using any routing resources that are reserved for use by the processing cores. Our 14-nm technology has a total of 13 metal layers. We use three metal layers for horizontal routing (M2, M4, M6) and three for vertical routing (M3, M5, M7). We reserve M8 and M9 for the local power grid, M10 and M11 for global routing (e.g., clock, reset, chip-level I/O), and M12 and M13 for the global power grid. The results of our macro-level analysis can be found in Table II.

To ensure that timing closure for the hard macro can imply timing closure at the chip top-level, we carefully constrain the maximum delay of each register-to-output, input-to-output, and input-to-register path such that the sum of the path delays which form a register-to-register path at the chip top-level is less than a clock cycle (T_c). For example, for *mesh-c4r2* we constrain the maximum delay of register-to-output paths that end at the east ruche output port to be $0.4T_c$, west to east feed-through paths to be $0.3T_c$, and input-to-register paths that start at the west ruche input port to be $0.25T_c$. Our timing constraints are a sufficient but not necessary condition for meeting timing at the chip top-level. Ideally, we only need to constrain the sum of the delays for these paths rather than constrain each of the three paths separately. Unfortunately, such complex constraints are not currently supported by the ASIC toolflow.

Figure 5(a) illustrates tradeoffs between bisection bandwidth and area for several topologies. As predicted in our analytical analysis, all topologies provide similar bisection bandwidth for a given area (ignoring message size limitations). By adding ruche channels to *mesh-c4r0*, *mesh-c4r2* achieves comparable bisection bandwidth at similar area overhead compared to *mesh-c1r0*. This supports our hypothesis that ruche channels can complement the reduced bisection channel count brought by concentration. Figure 5(b–c) illustrates tradeoffs between zero-load latency and area for both small (64b) and large (256b) messages. For both cases, *mesh-c4r2* achieves the lowest latency at similar area. Compared to *mesh-c4r0*, adding ruche channels further reduces the zero-load latency. Although ruche channels lead to narrower channels at the same area, the benefit of reduced average hop count still outweighs the increase in serialization latency. For example, *mesh-c4r2-b64* has similar area as *mesh-c4r0-b128*; it increases serialization latency by two cycles but reduces average hop latency by four cycles (see Table II).

One key observation is that packets can travel long distances in a single cycle. Thus topologies with long channels are critical to reducing the diameter of the network. In the baseline *mesh-c1r0*, a single-cycle channel is of length 185 µm. In *mesh-c4r0*, a single-cycle channel is of length 370 µm, and in *mesh-c1r2*, a single-cycle ruche channel is also of length 370 µm. Combining concentration and ruche channels results in even longer single-cycle channels. In *mesh-c4r2*, a single-cycle ruche channel is of length 740 µm which starts to approach the single-cycle limit.

We also observed that *mesh-c1r0q0* significantly reduces the diameter of the network compared to *mesh-c1r0*. However, it brings area overhead as it pushes the ASIC toolflow to use larger and faster standard cells. It is hard to meet timing with zero-cycle channels when these channels are wide. In our experiments, *mesh-c1r0q0* fails to meet timing at channel bandwidths larger than 64 bits, which limits the maximum bisection bandwidth it can achieve. Overall, even though *mesh-c1r0q0* reduces the average hop latency by 2×, a combination of concentration and ruche channels still achieves lower latency and higher bisection bandwidth at similar area compared to *mesh-c1r0q0*.

In this work, we assume all hard macros are implemented internally as a single flat module. This allows the cores in a hard macro to have different shapes and/or orientations. This may make a macro with concentration harder to implement as it is now four times or even eight times larger compared to a *mesh-c1r0* macro. We leave exploring multi-level hierarchical design methodologies which might compose core macros within a larger concentrated macro for future work.

To verify that our hard macro can indeed meet the 1 ns chip top-level timing constraint, we pushed full-chip layouts through the ASIC toolflow using each of the hard macros. The *mesh-c1r0-b64* chip has a positive slack of 47.7 ps, *mesh-c4r0-b128* chip has a positive slack of 240.1 ps, *mesh-c4r2-b128* chip has a positive slack of 292.6 ps, *torus-c1r0-b32* chip has a positive slack of 455.7 ps, and *torus-c1r0-b64* has a positive slack of 283.4 ps. The positive slack is significantly better than the worst case positive slack shown in Table II because our macro-level timing constraints are rather conservative. Figure 7 shows the full-chip layout for *torus-c1r0* and *mesh-c4r2*. By integrating feed-through channels into the macro, we enable short chip top-level routing for topologies that would otherwise require long and complicated chip top-level routing.

V. Conclusions

Practical manycore processor implementations usually avoid novel on-chip network flow-control schemes, custom circuits, and/or topologies due to various physical design issues. This paper makes the case that it is possible to implement low-diameter on-chip networks in manycore processors by creatively adapting mesh/torus topologies with concentration and ruche channels for a tiled physical design methodology. Through a combination of analytical modeling and rigorous layout-level evaluation in a traditional standard-cell-based flow, this paper demonstrated that 2D-mesh topologies with modest concentration factors (concentration factor of four) and modest length ruche channels (ruche factor of two) can reduce latency by over 2× at similar area and bisection bandwidth for both small and large messages compared to a 2D-mesh baseline.

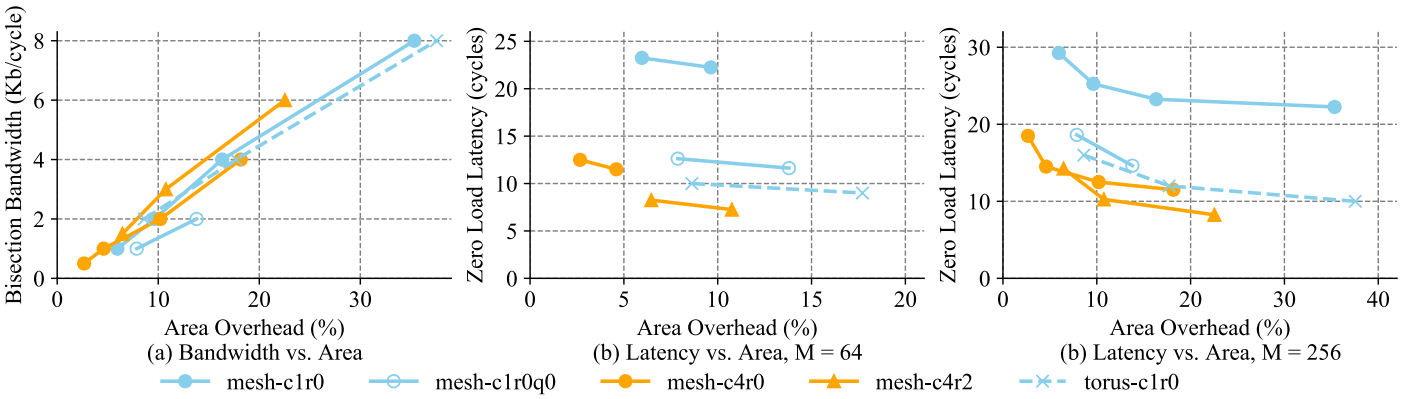

Figure 5. Bandwidth, Latency, and Area Tradeoffs for Post-Place-and-Route Results – M = message size in bits; (a) = bandwidth and area tradeoffs; (b) = latency and area tradeoffs for small messages (64b); (c) = latency and area tradeoffs for large messages (256b).

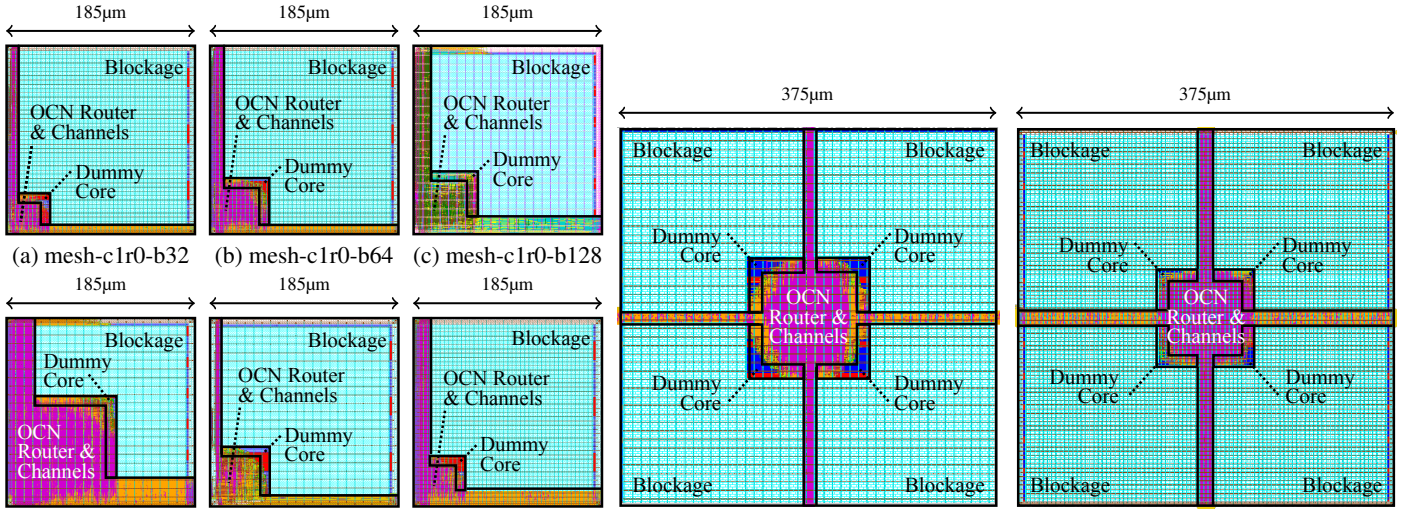

Figure 6. Example Macro-Level Post-Place-and-Route Layouts – All layouts are to scale and include 1–4 blockages, 1–4 dummy cores, and fences to constrain placement of the router and channels. (a–d) layouts for *mesh-c1r0* at four different channel bandwidths; (e) layout for *mesh-c1r0q0-b32* (i.e., no channel queues), OCN requires more area than *mesh-c1r0-b32*; (f) layout for *torus-c1r0-b32*, OCN requires comparable area to *mesh-c1r0-b32*; (g–h) layout for *mesh-c4r0-b128* and *mesh-c4r2-b64* both of which require comparable area (smaller OCN router "square" in *mesh-c4r2-b64* is outweighed by longer and wider channel "rectangles").

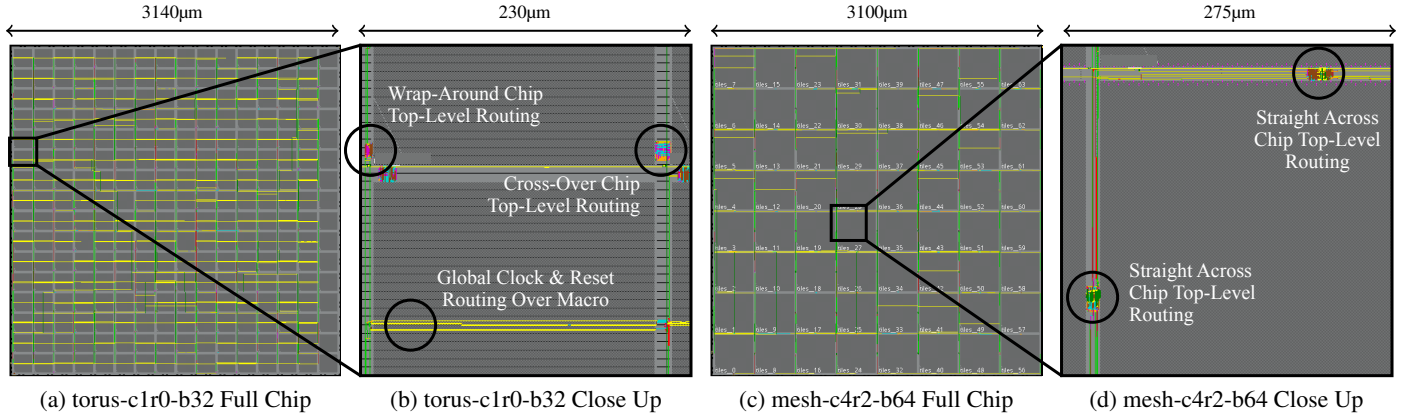

Figure 7. Example Chip-Level Post-Place-and-Route Layouts – (a) full-chip layout with 256 instances of the *torus-c1r0-b32* hard macro which is shown in Figure 6(f); (b) close-up of required chip top-level routing including cross-over routing to neighboring hard macro, wrap-around routing, and global clock and reset routing over the hard macro; (c) full-chip layout for 64 instances of the *mesh-c4r2-b64* hard macro which is shown in Figure 1(h); (d) = close-up of the required chip top-level routing including straight-across routing at the middle of each macro side.

ACKNOWLEDGMENTS

The authors thank Shunning Jiang and Peitian Pan for supporting PyMTL3, Cheng Tan for leading the initial development of PyOCN, Christopher Torng for supporting the ASIC toolflow, and Michael Taylor and David Wentzlaff for useful discussion on manycore on-chip networks.

REFERENCES

[1] N. Abeyratne et al. Scaling Towards Kilo-Core Processors with Asymmetric High-Radix Topologies. *HPCA*, Feb 2013.

[2] J. Balfour and W. Dally. Design Tradeoffs for Tiled CMP On-Chip Networks. *ICS*, Jun 2006.

[3] S. Bell et al. TILE64 Processor: A 64-Core SoC with Mesh Interconnect. *ISSCC*, Feb 2008.

[4] M. Besta et al. Slim NoC: A Low-Diameter On-Chip Network Topology for High Energy Efficiency and Scalability. *ASPLOS*, Mar 2018.

[5] B. Bohnenstiehl et al. KiloCore: A 32-nm 1000-Processor Computational Array. *IEEE JSSC*, 52(4):891–902, Apr 2017.

[6] C.-H. O. Chen et al. SMART: A Single-Cycle Reconfigurable NoC for SoC Applications. *DATE*, Mar 2013.

[7] W. J. Dally. Express Cubes: Improving the Performance of k-ary n-cube Interconnection Networks. *IEEE Trans. on Computers*, 40(9):1016–1023, Sep 1991.

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

[9] S. Davidson et al. The Celerity Open-Source 511-Core RISC-V Tiered Accelerator Fabric: Fast Architectures and Design Methodologies for Fast Chips. *IEEE Micro*, 38(2):30–41, Mar/Apr 2018.

[10] B. Grot et al. Express Cube Topologies for On-Chip Interconnects. *HPCA*, Feb 2009.

[11] B. Grot et al. Kilo-NOC: A Heterogeneous Network-on-Chip Architecture for Scalability and Service Guarantees. *ISCA*, Jun 2011.

[12] T. R. Halfhill. ThunderX3's Cloudburst of Threads: Marvell Previews 96-core 384-thread Arm Server Processor. *Microprocessor Report, The Linley Group*, Apr 2020.

[13] Y. Hoskote et al. A 5-GHz Mesh Interconnect for a Teraflops Processor. *IEEE Micro*, 27(5):51–61, Sep/Oct 2007.

[14] J. Howard et al. A 48-Core IA-32 Message-Passing Processor with DVFS in 45nm CMOS. *ISSCC*, Feb 2010.

[15] S. Jiang et al. PyMTL3: A Python Framework for Open-Source Hardware Modeling, Generation, Simulation, and Verification. *IEEE Micro*, 40(4):58–66, Jul/Aug 2020.

[16] T. Jung et al. Ruche Networks: Wire-Maximal No-Fuss NoCs. *Int'l Symp. on Networks-on-Chip*, Sep 2020.

[17] Y.-H. Kao et al. CNoC: High-Radix Clos Network-on-Chip. *IEEE TCAS*, 30(12):1897–1910, Dec 2011.

[18] B. Kim and V. Stojanović. Characterization of Equalized and Repeated Interconnects for NoC Applications. *IEEE Design and Test of Computers*, 25(5):430–439, Sep 2008.

[19] J. Kim et al. Flattened Butterfly Topology for On-Chip Networks. *MICRO*, Aug 2007.

[20] A. Kumar et al. Express Virtual Channels: Towards the Ideal Interconnection Fabric. *ISCA*, Jun 2007.

[21] P. Kumar et al. Exploring Concentration and Channel Slicing in On-Chip Network Router. *Int'l Symp. on Networks-on-Chip*, May 2009.

[22] M. Lis et al. Hardware-Level Thread Migration in a 110-Core Shared-Memory Multiprocessor. MIT CSAIL CSG, Technical Report 512, Nov 2013.

[23] M. McKeown et al. Piton: A Manycore Processor for Multitenant Clouds. *IEEE Micro*, 37(2):70–80, Mar/Apr 2017.

[24] G. Michelogiannakis et al. Elastic-Buffer Flow Control for On-Chip Networks. *HPCA*, Feb 2009.

[25] R. Mullins et al. Low-Latency Router Microarchitecture for Intra-Chip Networks. *ISCA*, Jun 2004.

[26] A. Olofsson. Epiphany-V: A 1024-processor 64-bit RISC System-On-Chip. *CoRR arXiv:1610.01832*, Aug 2016.

[27] L.-S. Peh and W. J. Dally. A Delay Model and Speculative Architecture for Pipelined Routers. *HPCA*, Jan 2001.

[28] A. Rovinski et al. Evaluating Celerity: A 16nm 695 Giga-RISC-V Instructions/s Manycore Processor with Synthesizable PLL. *IEEE Solid-State Circuits Letters*, 2(12):289–292, Dec 2019.

[29] A. Rovinski et al. A 1.4 GHz 695 Giga RISC-V Inst/s 496-core Manycore Processor with Mesh On-Chip Network and an All-Digital Synthesized PLL in 16nm CMOS. *VLSI*, Jun 2019.

[30] A. Sodani et al. Knights Landing: Second-Generation Intel Xeon Phi Product. *IEEE Micro*, 36:34–46, Mar/Apr 2016.

[31] C. Tan et al. PyOCN: A Unified Framework for Modeling, Testing, and Evaluating On-Chip Networks. *ICCD*, Nov 2019.

[32] M. B. Taylor et al. Evaluation of the Raw Microprocessor: An Exposed-Wire-Delay Architecture for ILP and Streams. *ISCA*, Jun 2004.

[33] D. Wentzlaff et al. On-Chip Interconnection Architecture of the Tile Processor. *IEEE Micro*, 27:15–31, Sep/Oct 2007.

[34] B. Wheeler. Ampere Maxes Out at 128 Cores. *Microprocessor Report, The Linley Group*, Jul 2020.