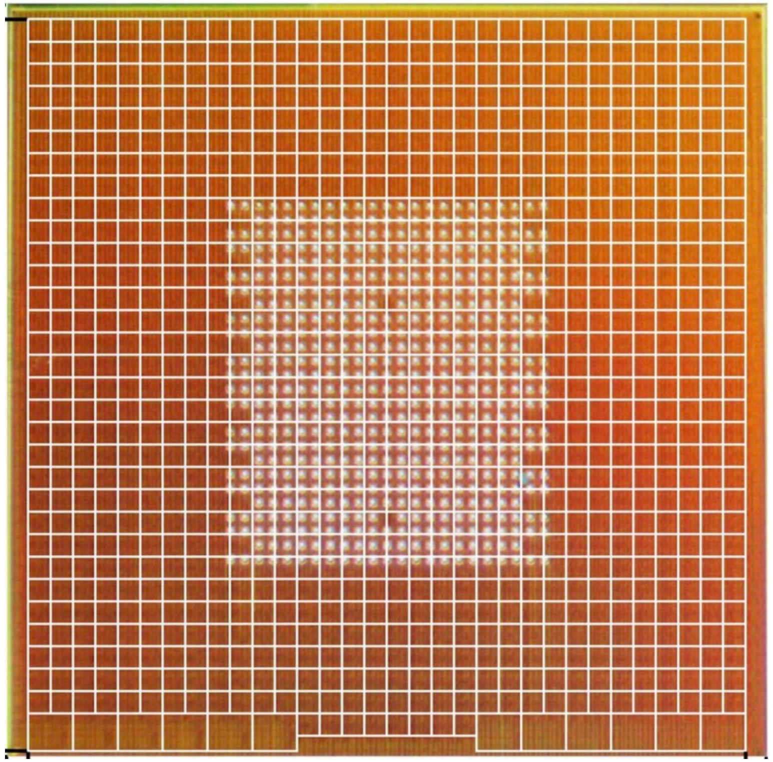
# Implementing Low-Diameter On-Chip Networks for Manycore Processors Using A Tiled Physical Design Methodology

Yanghui Ou, Shady Agwa, Christopher Batten

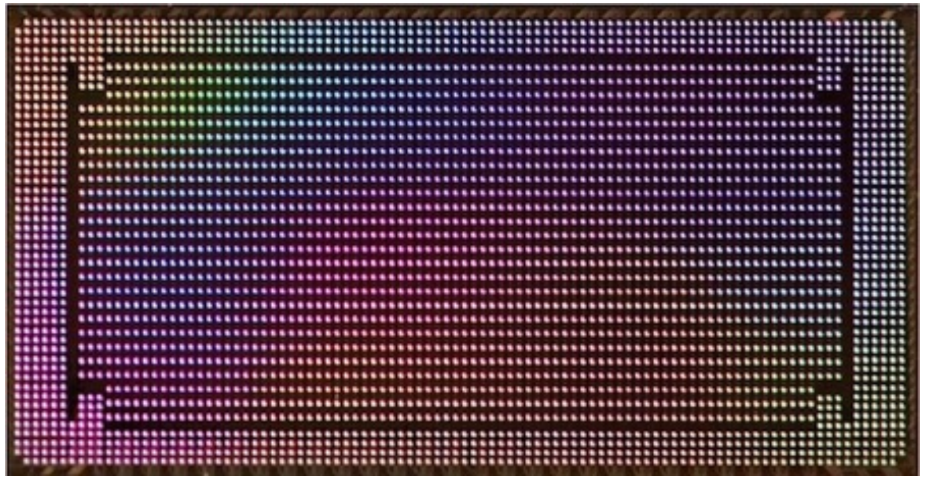
Computer Systems Laboratory
Cornell University

# Real Manycore Implementations Use Simple Mesh OCNs



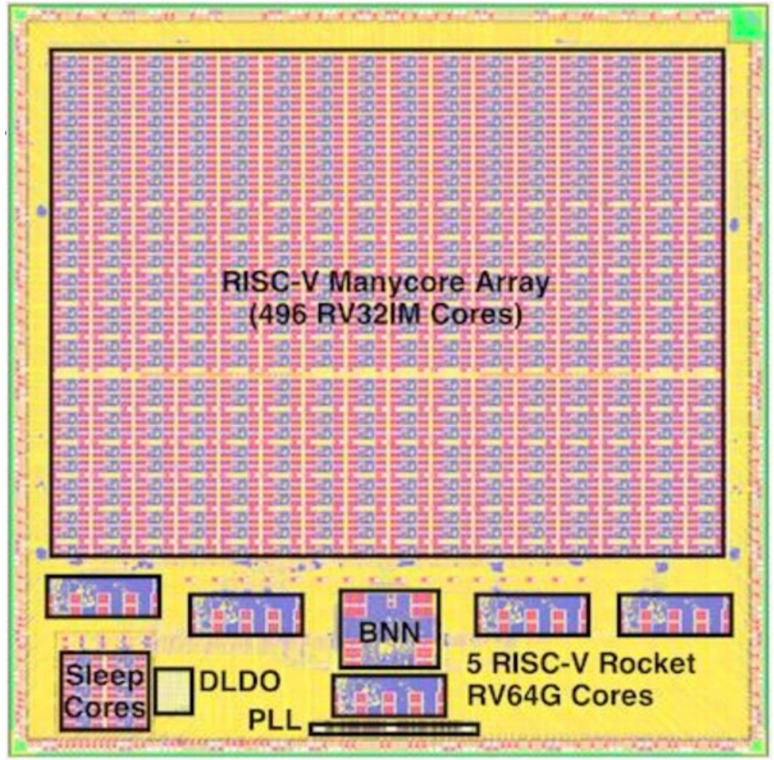
KiloCore, 1000 cores, 32x32 mesh

UC Davis



Epiphany-V, 1024 cores, 32x32 mesh

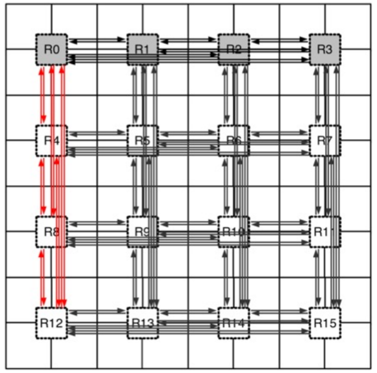
Adapteva, Inc



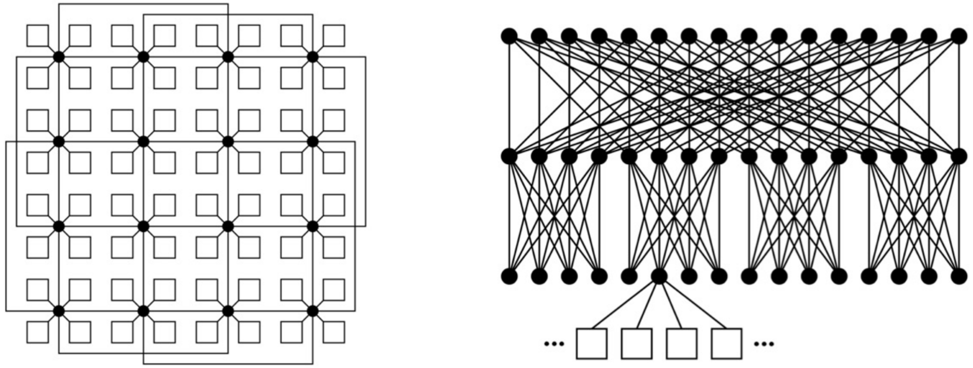
Celerity, 496 cores, 16x31 mesh

University of Washington,
University of Michigan,
Cornell University,
UC San Diego

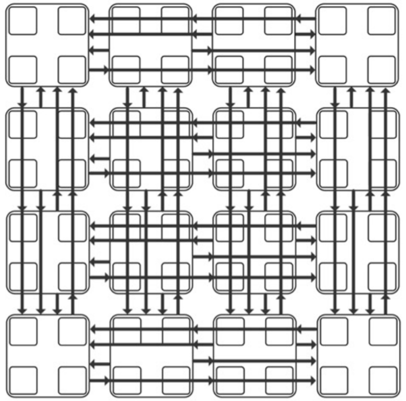
# Plenty of Novel OCN Topologies Proposed in the Academic Area



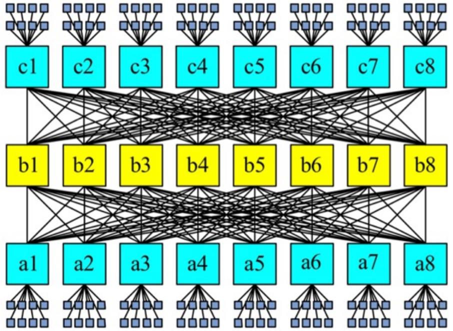
Flattened Butterfly
Kim+, MICRO’07



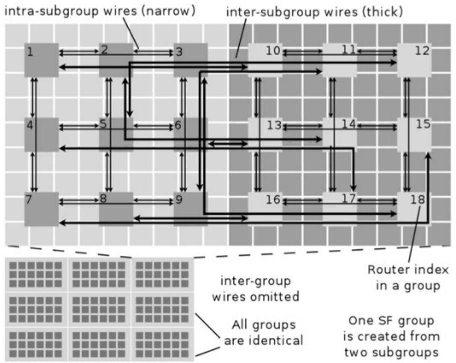
Concentrated Mesh, Fat-Tree
Balfour+, ICS’06



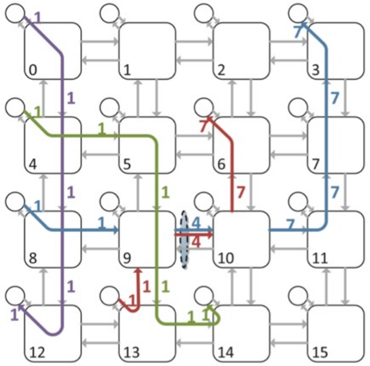
Multi-drop Express Channels
Grot+, HPCA’06/ISCA’11



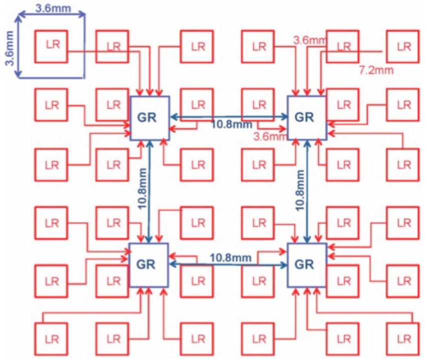
Clos Network
Kao+, TCAS’11


intra-subgroup wires (narrow)
inter-subgroup wires (thick)
1
2
3
10
11
12
4
5
6
13
14
15
7
8
9
16
17
18
inter-group wires omitted
Router index in a group
All groups are identical
One SF group is created from two subgroups

Slim NoC
Besta+, ASPLOS’18



SMART NoC,
Chen +, HPCA’13



Asymmetric High-Radix
Abeyratne+, HPCA’13

# Gap Between Principle and Practice

- **Why do manycore processor implementations with 500-1000 cores continue to use simple high-diameter on-chip networks?**

- Manycores require simple, low-area routers

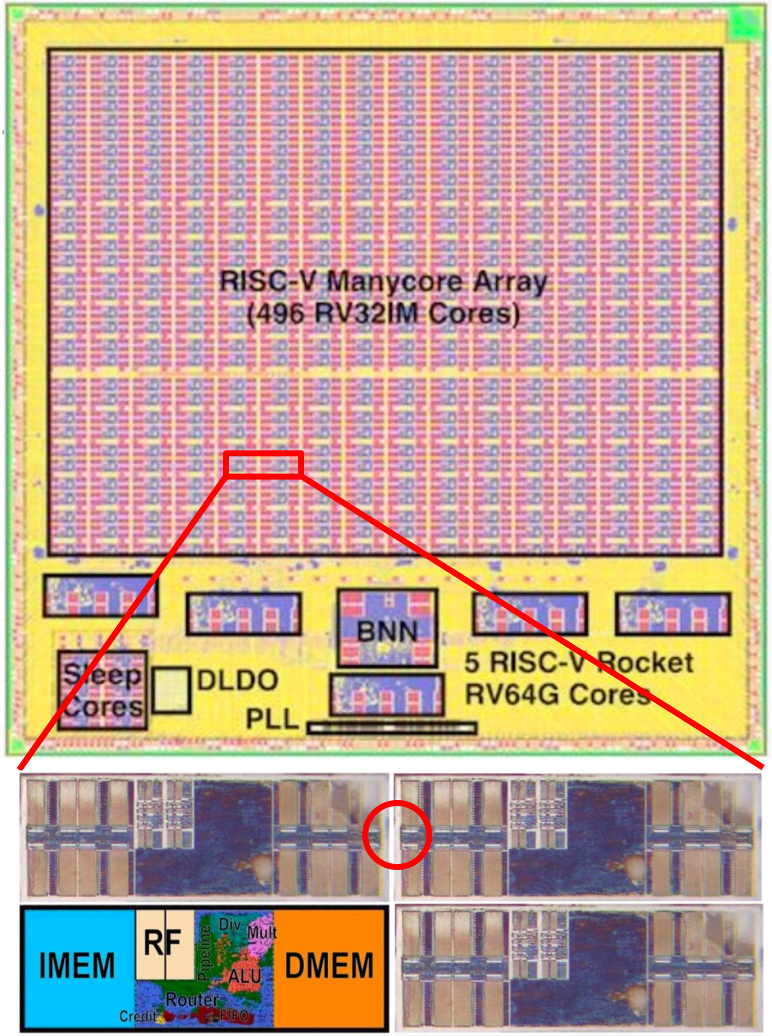
- Manycores use standard-cell-based design

- Manycores use a **tiled physical design methodology** with three key constraints:

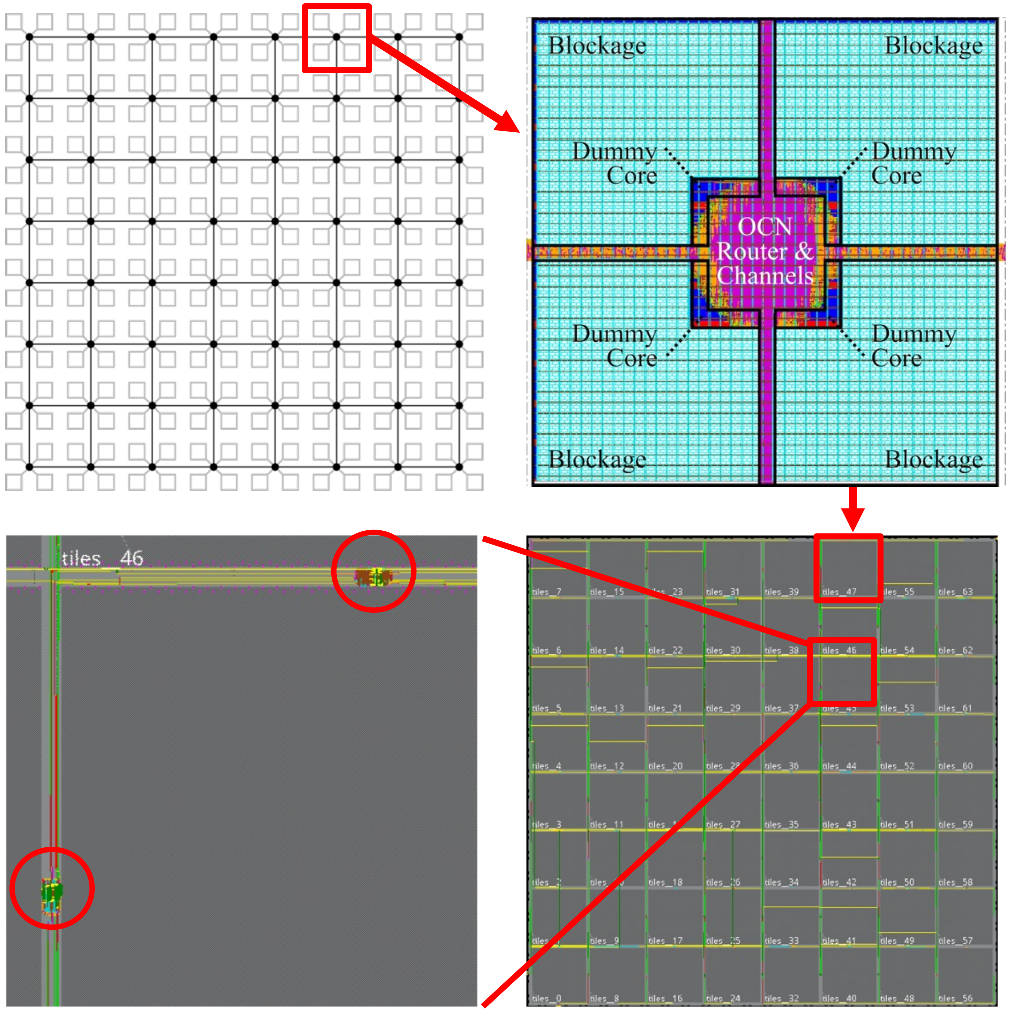
  1. Design is based on tiling a **homogeneous** hard macro across the chip

  2. All chip top-level routing between hard macros must use **short wires** to neighboring macros

  3. Timing closure for the hard macro must imply timing closure at the chip level



Hard Macros in Celerity



# Implementing Low-Diameter OCN for Manycore Processors Using A Tiled Physical Design Methodology

Motivation

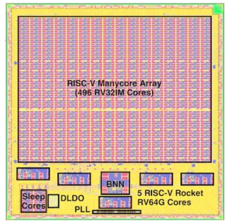
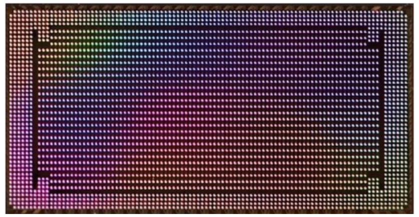
**Manycore OCN Topologies**

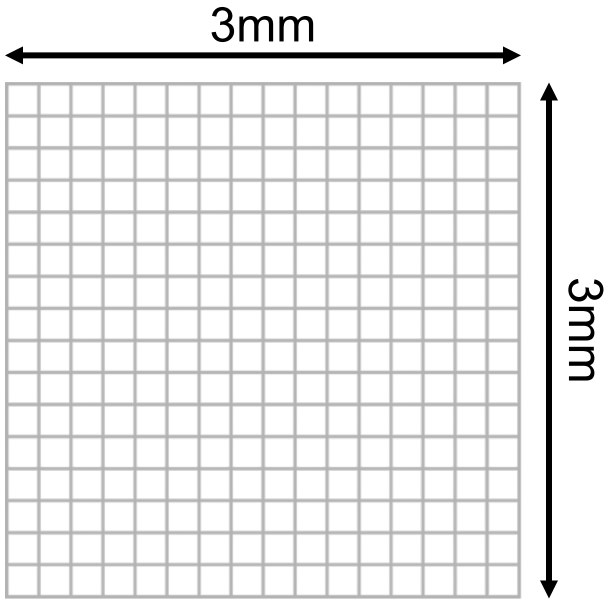
Manycore OCN Analytical Modeling

Manycore OCN Physical Design

PyOCN Framework

# Target Chip: 16 x 16 Manycore

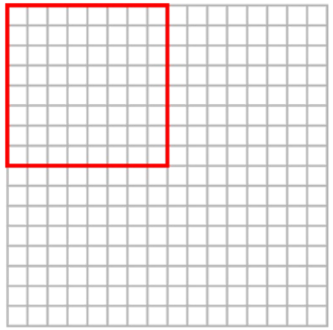
| Manycore |  |  |
|---|---|---|
| Per-core Area | 24,250μm$^2$ | 103,500μm$^2$ |
| Process | 16nm | 16nm |
| Frequency | ~1GHz | 500MHz |
| ISA | RV32IM | RV64G |
| Issue Width | Single | Dual |
| L1 Memory | 8KB | 64KB |



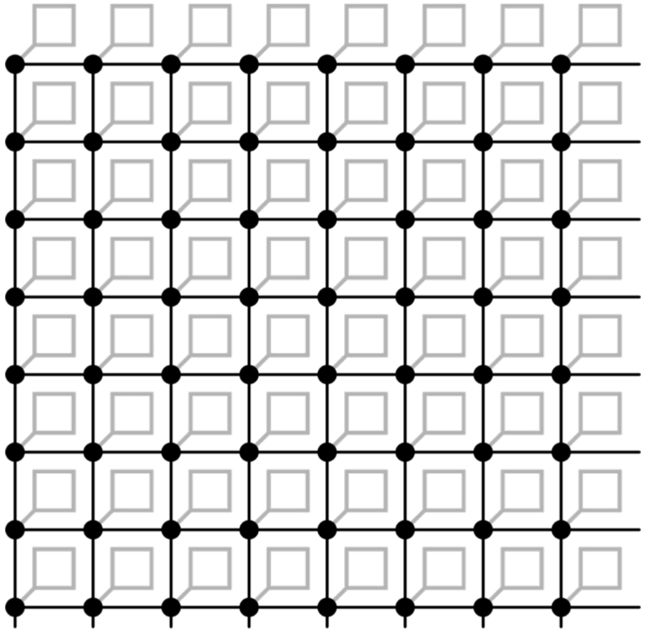
- 16x16 manycore at 1GHz using 14nm technology
- 3mm x 3mm, 185μm x 185μm per core
- Per-core area roughly corresponds to an in-order RV32IMAF processor with 4KB data cache and 4KB instruction cache

| Component | Area (μm$^2$) |
|---|---|
| RV32IMAF-IO | 15983 |
| 4KB data cache | 9407 |
| 4KB inst. cache | 9347 |
| Total | 34737 |

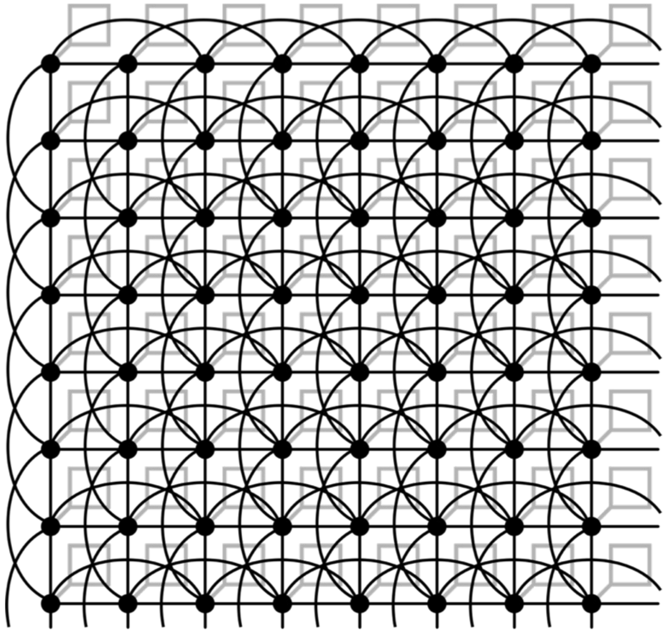
# Ruche Channels to Reduce the OCN Diameter



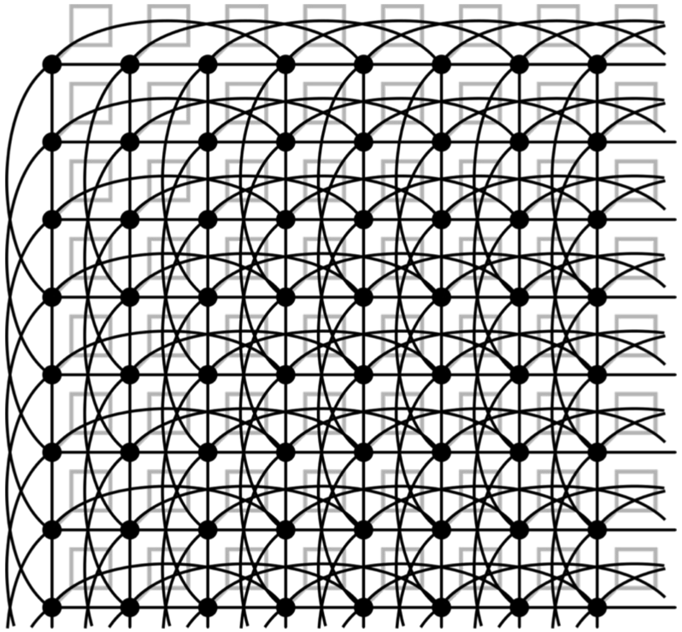
16x16 manycore



No ruche channels



Ruche factor of 2

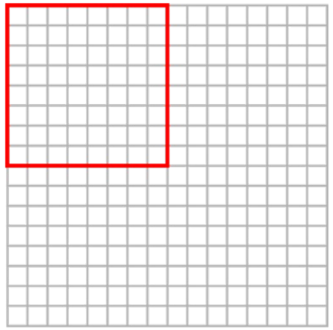


Ruche factor of 3

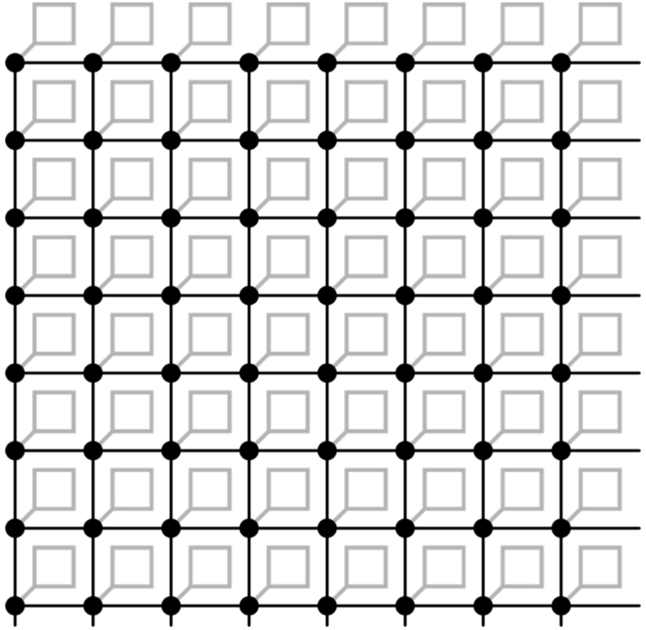
- Directly skips one or more routers
- Reduces network diameter
- Increases the number of bisection channels
- Increases router radix

Concurrently proposed with T. Jung et al,
*Ruche Networks: Wire-Maximal No-Fuss NoCs*

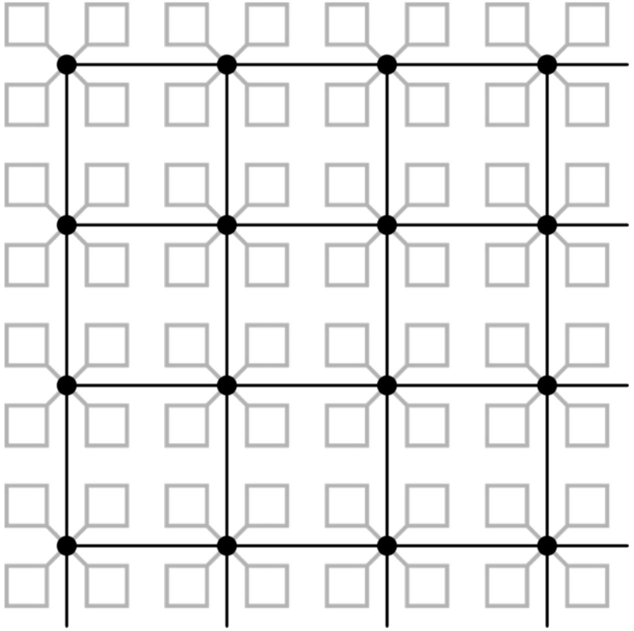
# CONCENTRATION TO REDUCE THE OCN DIAMETER



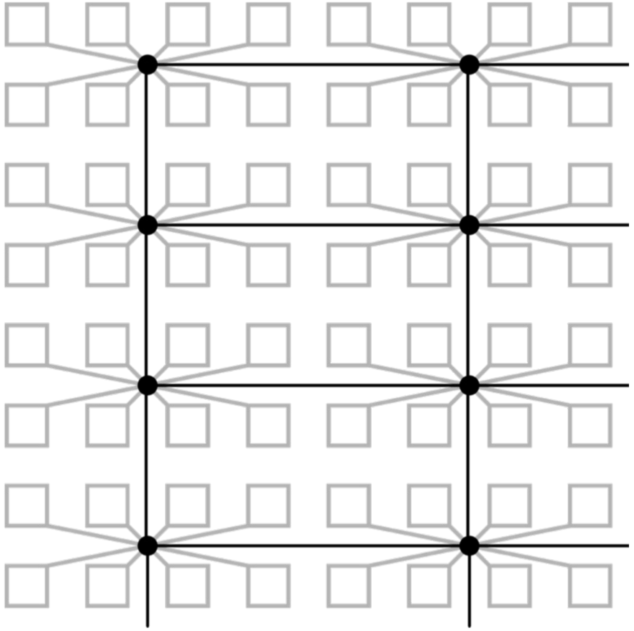
16x16 manycore



No concentration



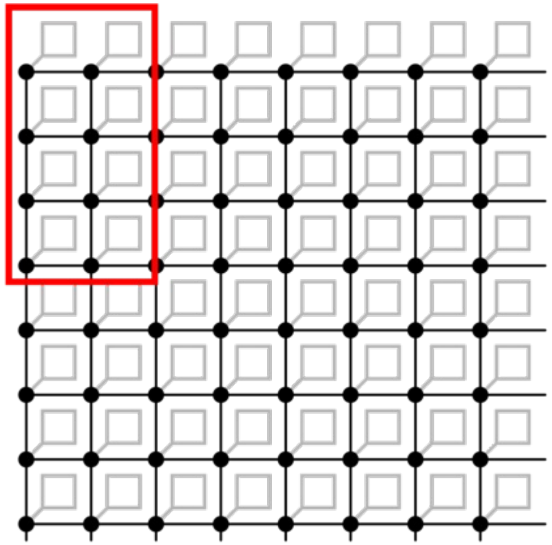
Concentration factor of four



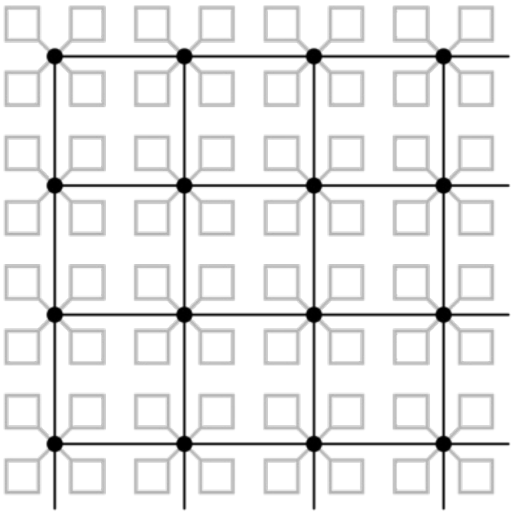
Concentration factor of eight

- Groups multiple cores together to share one router
- Reduces network diameter
- Reduces the number of routers
- Reduces the number of bisection channels
- Increases router radix

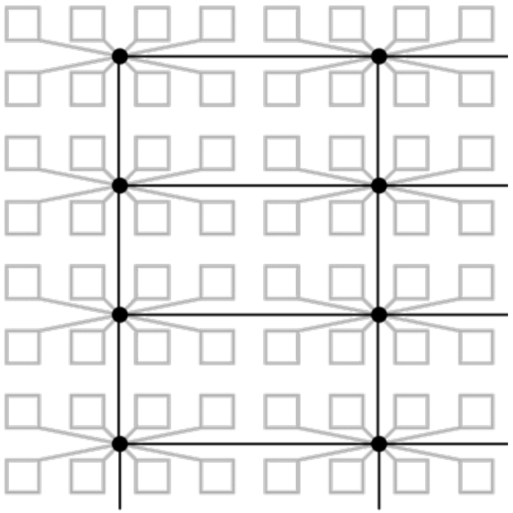
# Tiled Physical Design – No Ruche Channels



mesh-c1r0

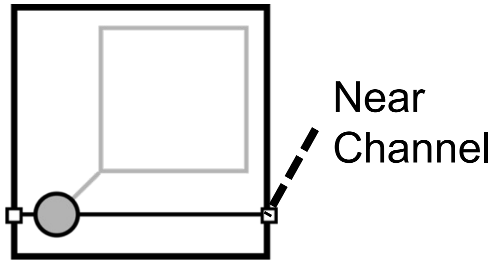


mesh-c4r0

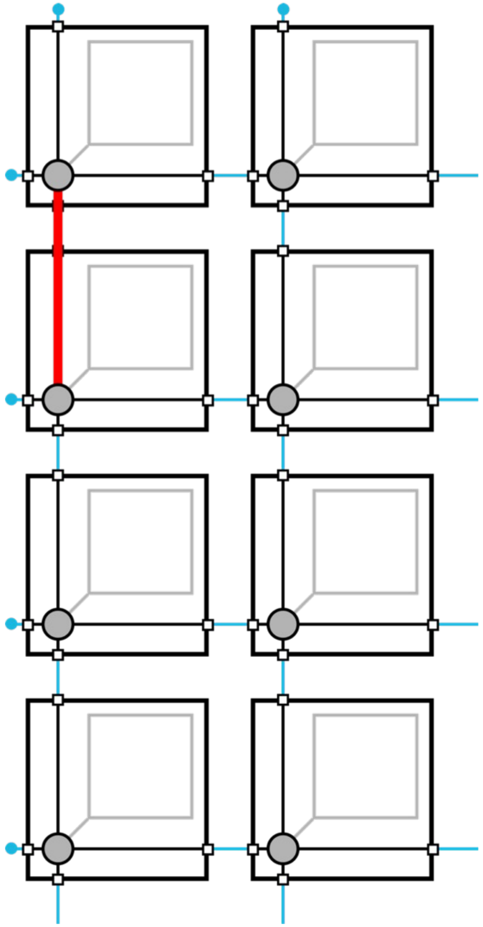


mesh-c8r0

- Only has near channel in both dimensions
- Pins are aligned to ensure short global routing

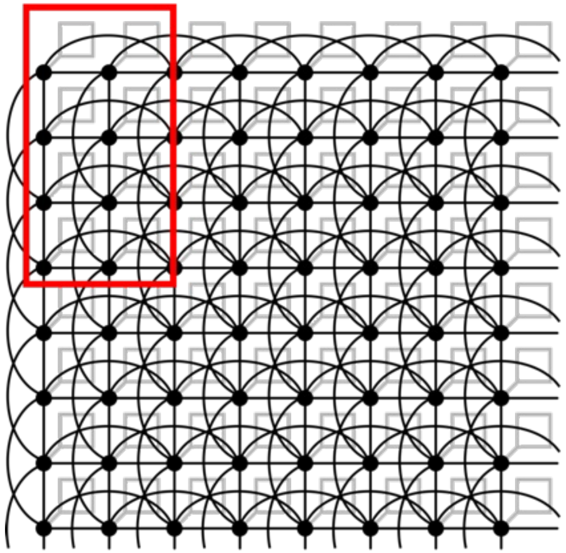


mesh-c1r0 hard macro in 1D

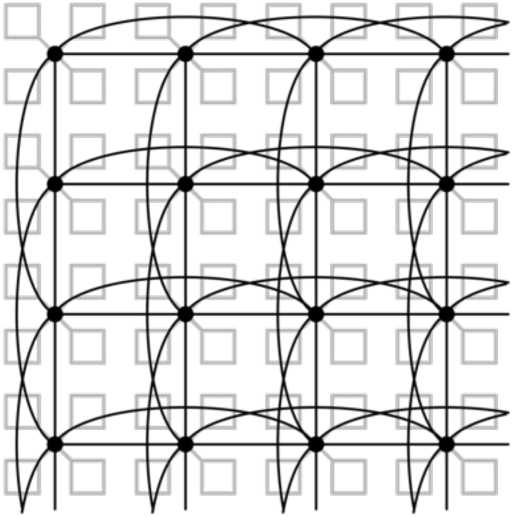


mesh-c1r0 tiled physical design

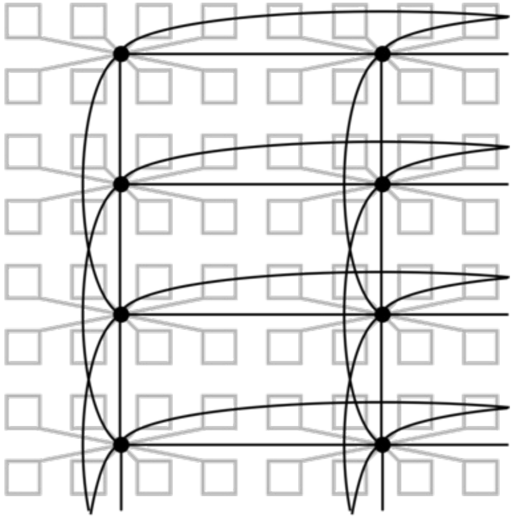
# Tiled Physical Design – Ruche Factor of Two



mesh-c1r2

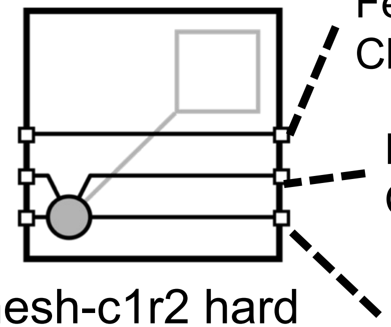


mesh-c4r2

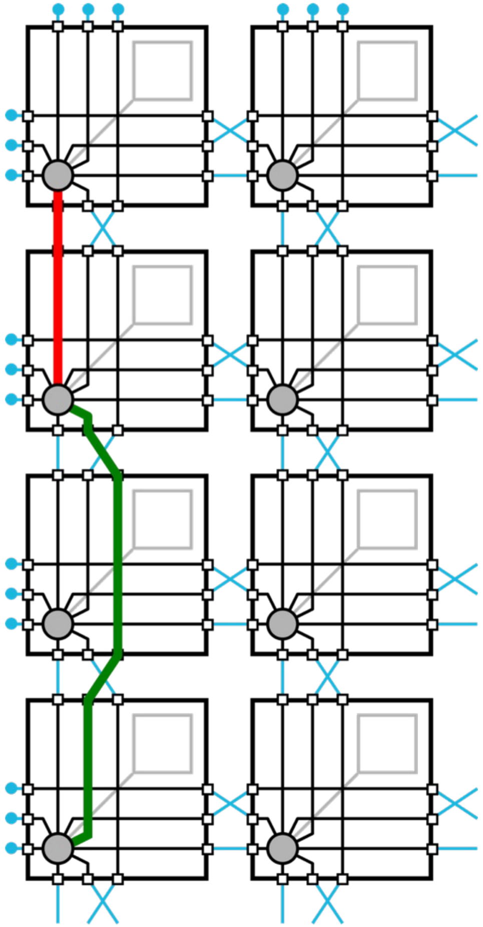


mesh-c8r2

- Near channel, far channel, and one feedthrough channel in one dimension
- Short cross-over routing between feedthrough channel and far channel

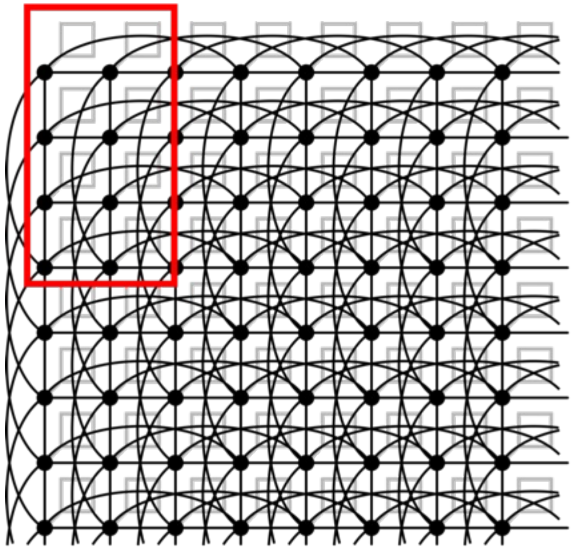


mesh-c1r2 hard macro in 1D

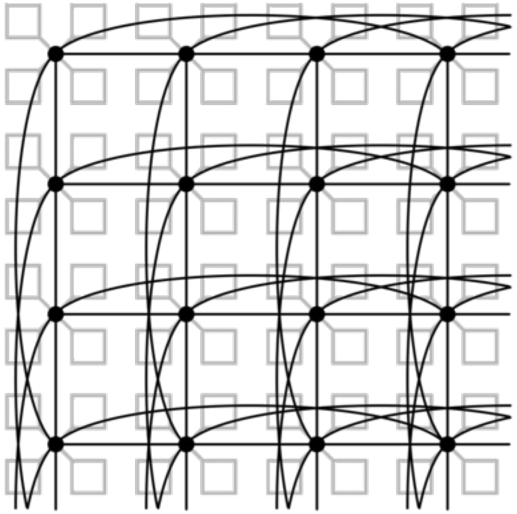


mesh-c1r2 tiled physical design

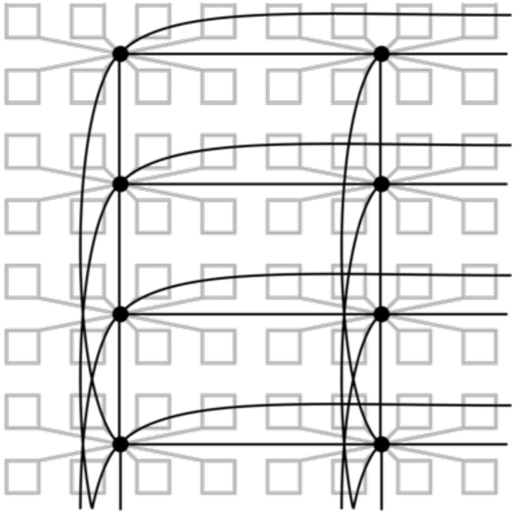
# Tiled Physical Design – Ruche Factor of Three



mesh-c1r3

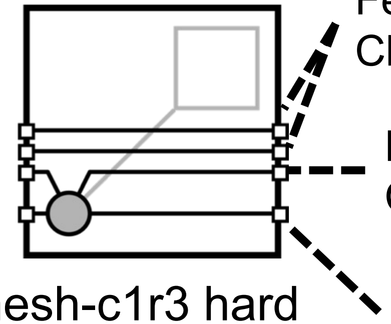


mesh-c4r3

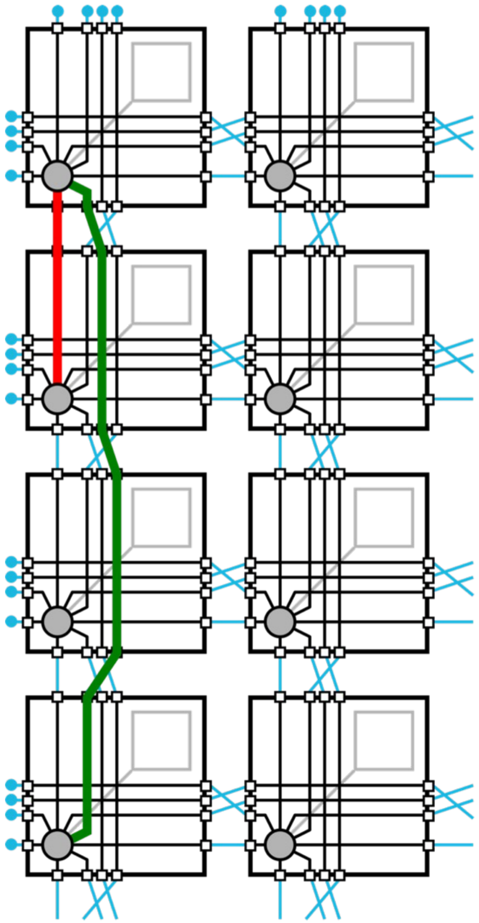


mesh-c8r3

- Near channel, far channel, and two feedthrough channels in one dimension
- Short cross-over routing between feedthrough channels and far channel

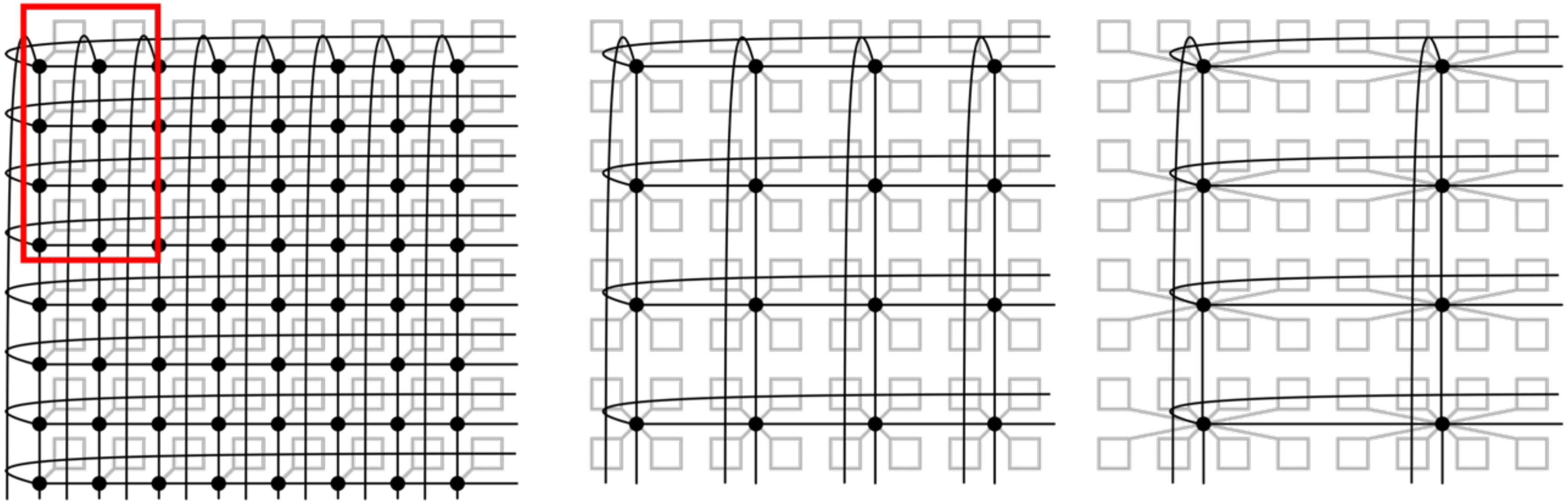


mesh-c1r3 hard macro in 1D

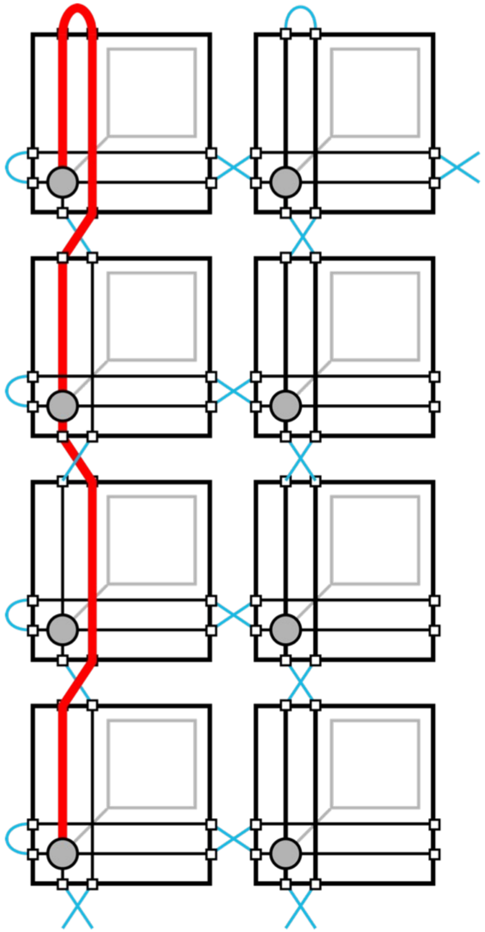


mesh-c1r3 tiled physical design

# Tiled Physical Design – Folded Torus

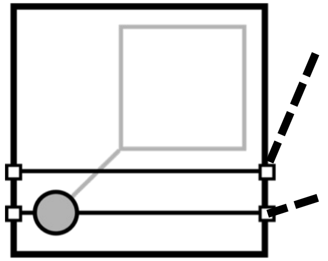


torus-c1r0 torus-c4r0 torus-c8r0

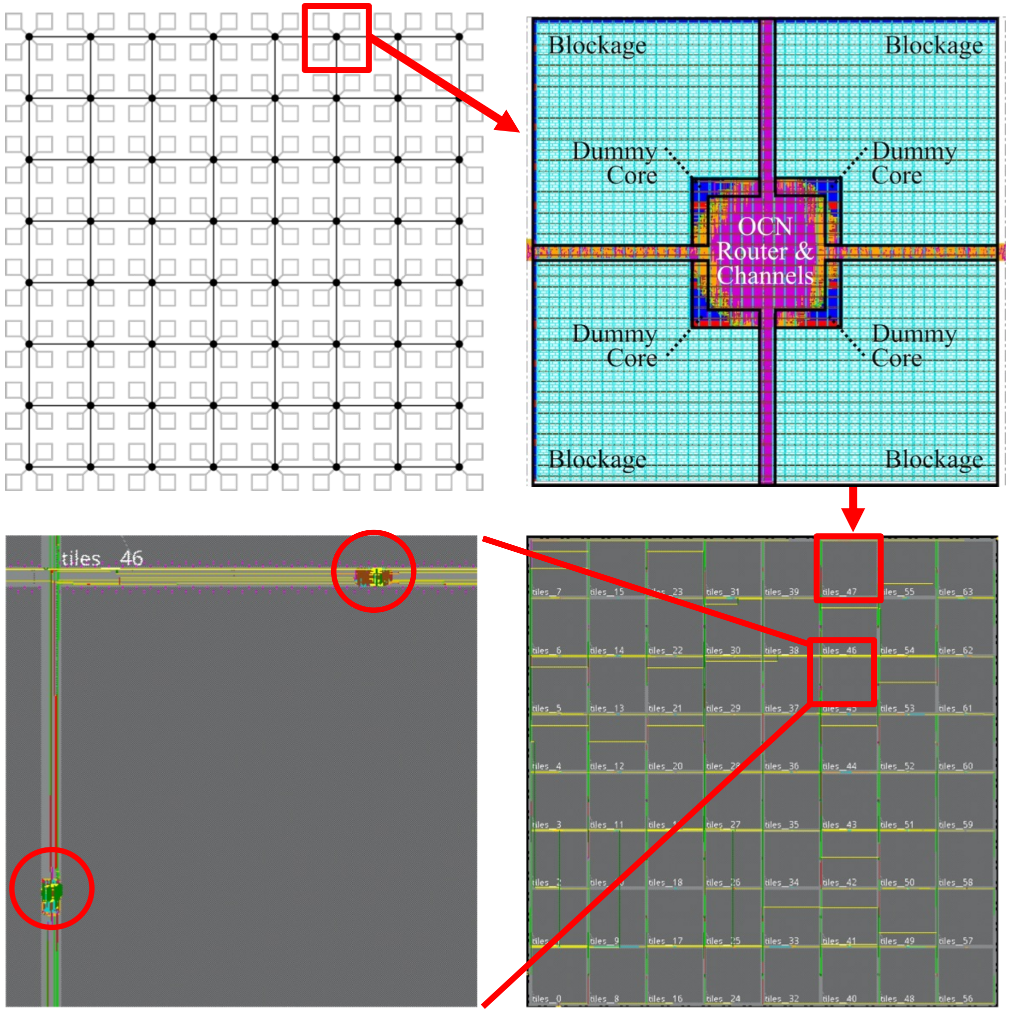


torus-c4r0 tiled physical design

- Only far channel and feedthrough channel in one dimension
- Short cross-over routing between feedthrough channels and far channel
- Short wrap-around routing at the edge



torus-c1r0 hard macro in 1D



# Implementing Low-Diameter OCN for Manycore Processors Using A Tiled Physical Design Methodology

Motivation

Manycore OCN Topologies

**Manycore OCN Analytical Modeling**

Manycore OCN Physical Design

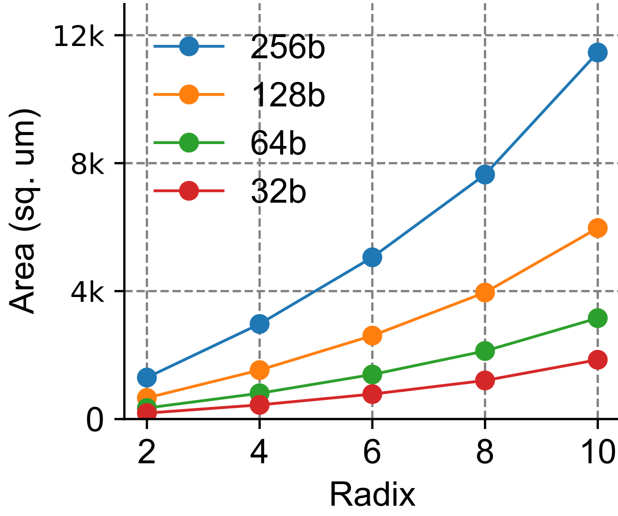
PyOCN Framework

# ANALYTICAL MODELING METHODOLOGY

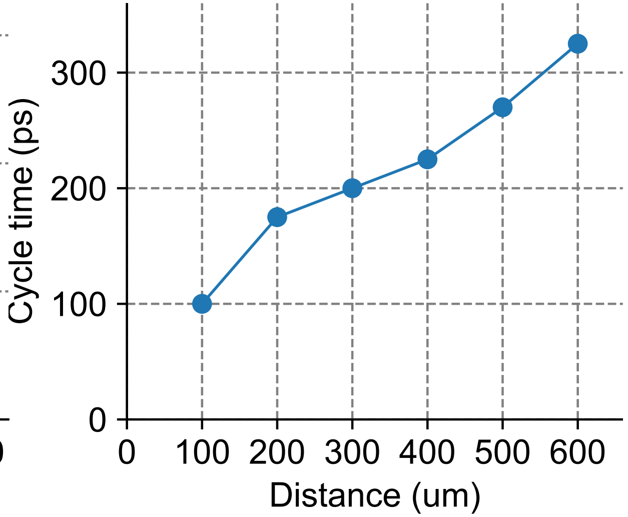
- Model the latency, area, and bandwidth analytically before doing physical design to narrow down our focus
- Router area model and channel latency model are constructed based on physical results and floorplans
- Zero-load latency is calculated analytically

$$T_{\o} = H_R t_R + H_C t_C + \frac{L}{b}$$

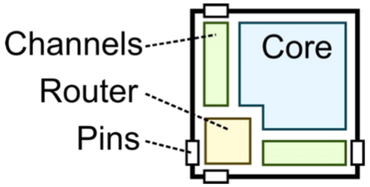
- Observation
  - Router area **does not scale quadratically** as radix increases
  - A packet can travel a very long distance on the channel in one cycle



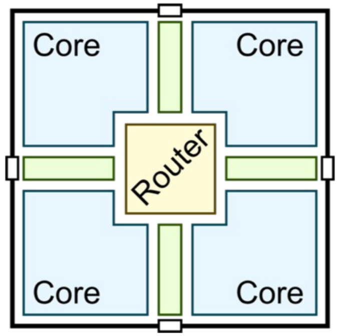
Router Area Results



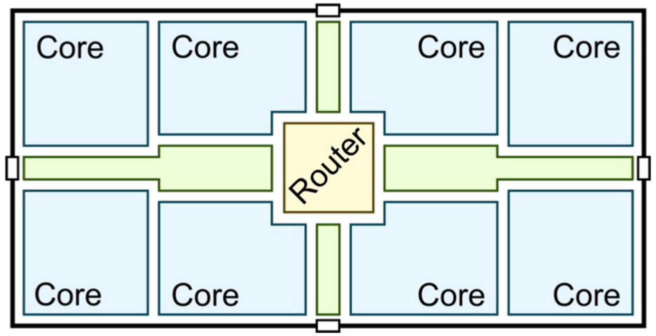
Channel Latency Results



No Concentration

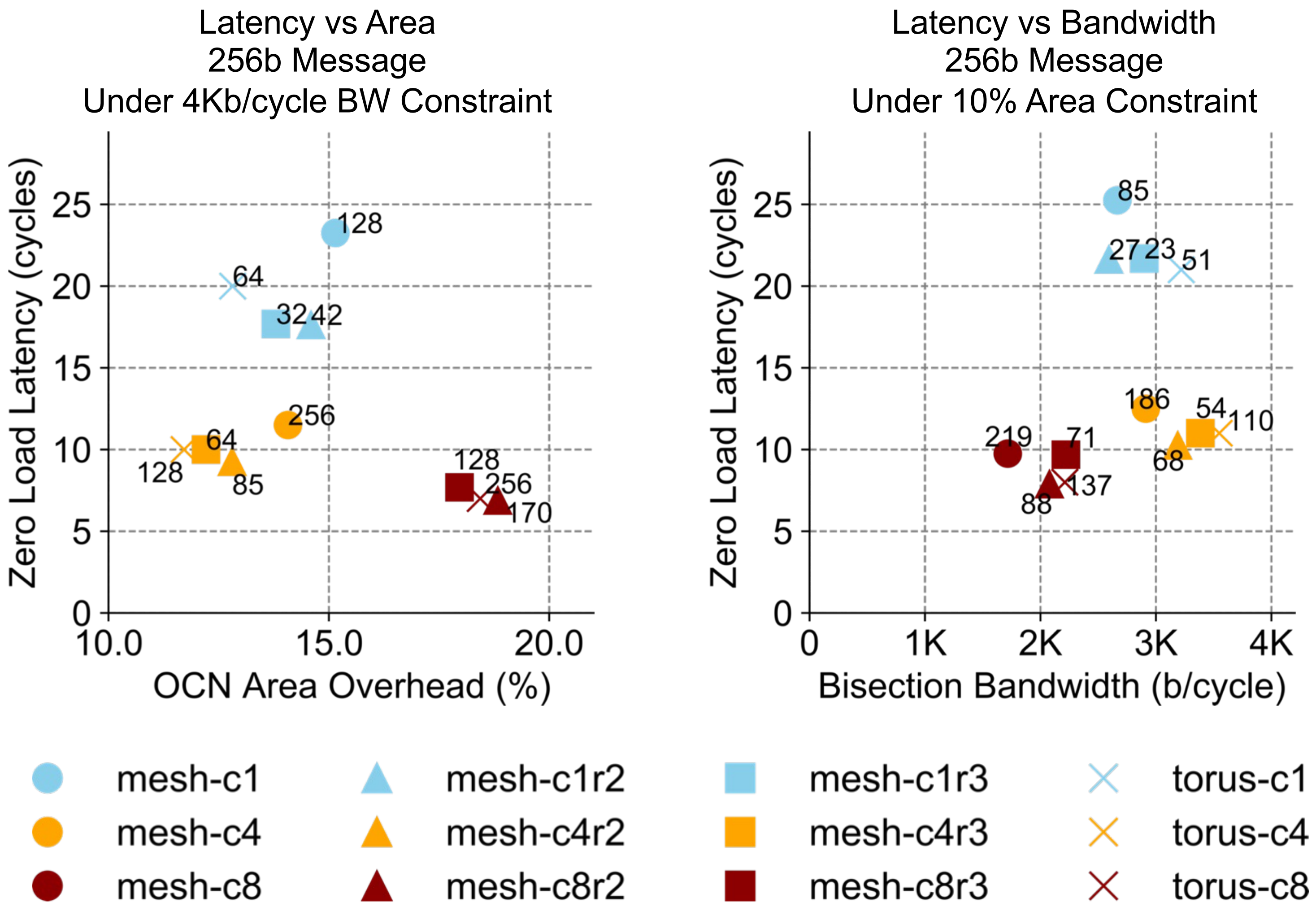


Concentration factor of four



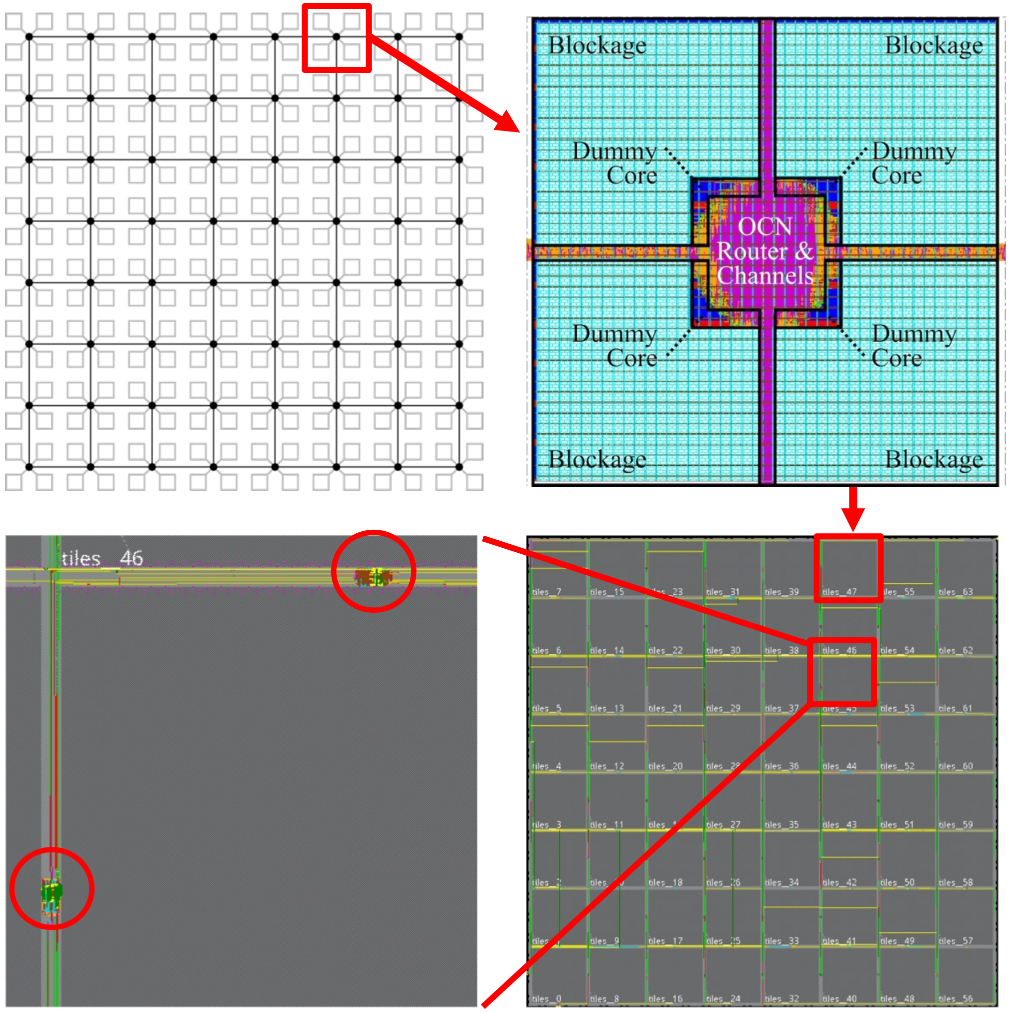
Concentration factor of eight

# ANALYTICAL MODELING RESULTS


Latency vs Area
256b Message
Under 4Kb/cycle BW Constraint

Latency vs Bandwidth
256b Message
Under 10% Area Constraint

- Moderate ruche factor improves bandwidth and/or reduces area
- Moderate concentration reduces latency at similar bandwidth and area
- Increasing ruche factor does not necessarily improves latency as it may lead to narrower channels which **increases serialization latency**



# Implementing Low-Diameter OCN for Manycore Processors Using A Tiled Physical Design Methodology

Motivation

Manycore OCN Topologies

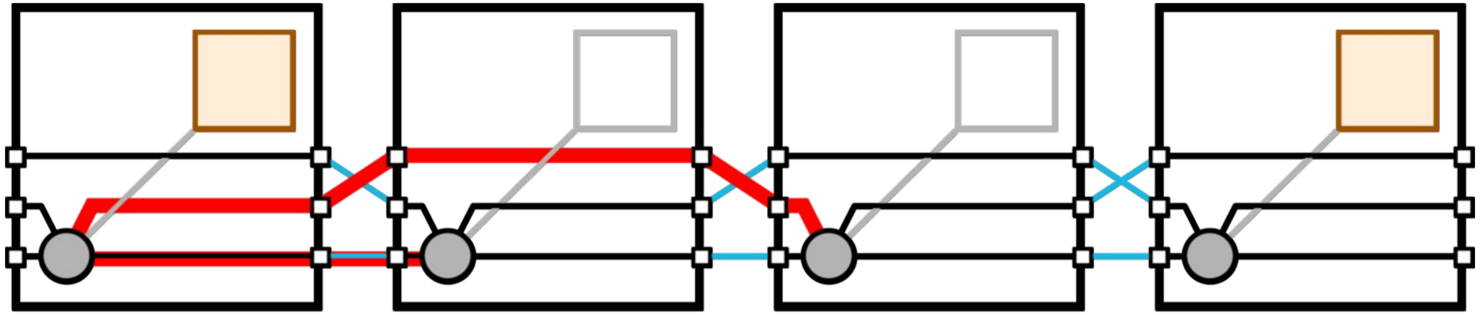
Manycore OCN Analytical Modeling

**Manycore OCN Physical Design**

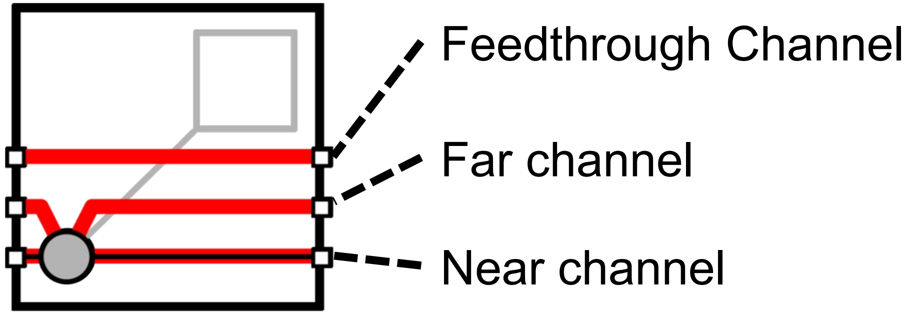
PyOCN Framework

# Hard Macro Design Methodology

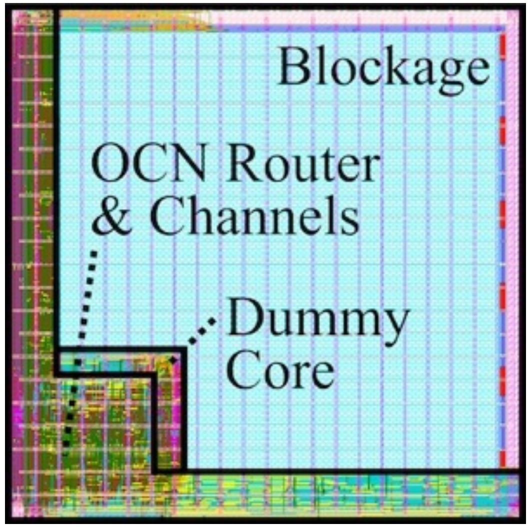
- Map global timing constraints to local timing constraints
- Use three metal layers for local horizontal routing (M2, M4, M6), three layers for vertical routing (M3, M5, M7)
- Connect “dummy cores” to the injection and ejection ports of the router to prevent ASIC toolflow from optimizing away any logic
- Use routing and placement blockages to prevent the ASIC toolflow from using the routing resources reserved for the real cores



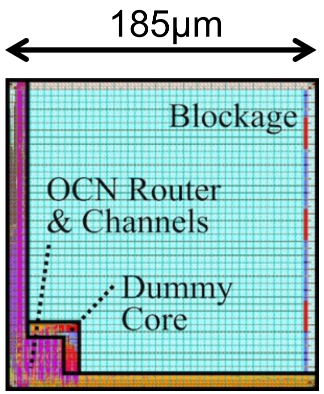
mesh-c1r2 global constraints



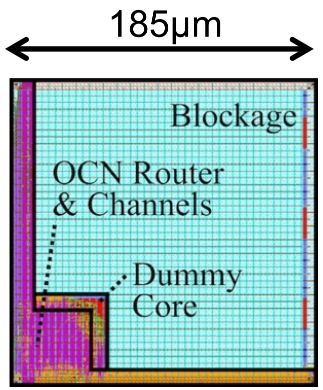
mesh-c1r2 local constraints



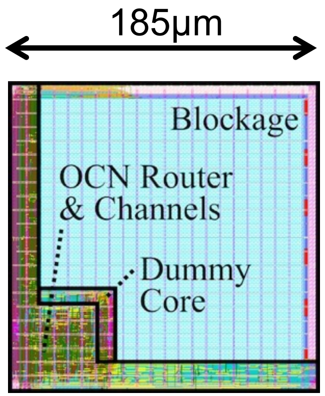
# Example Hard Macros



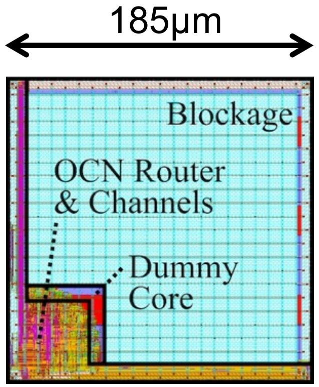
mesh-c1r0-b32



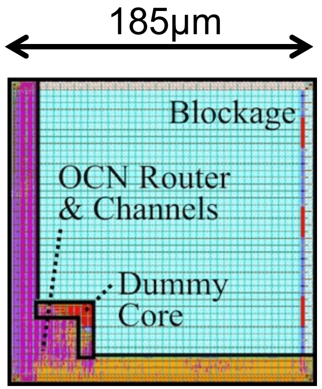
mesh-c1r0-b64



mesh-c1r0-b128



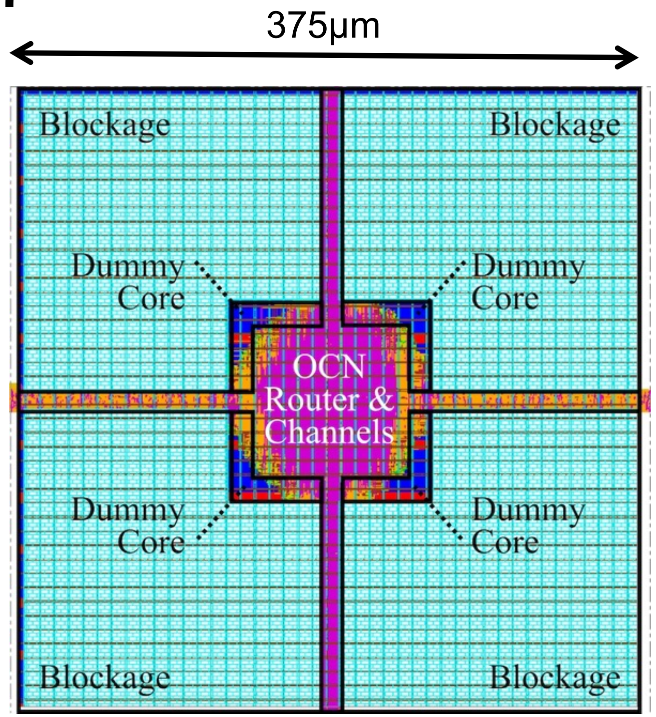
mesh-c1r0q0-b32



torus-c1r0-b32

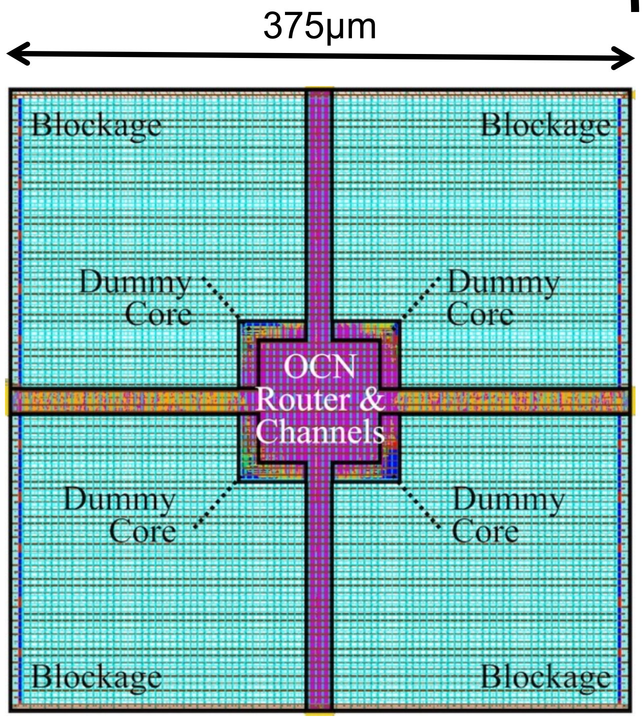
No Concentration & Ruche Channels

Concentration Factor of Four



mesh-c4r0-b128

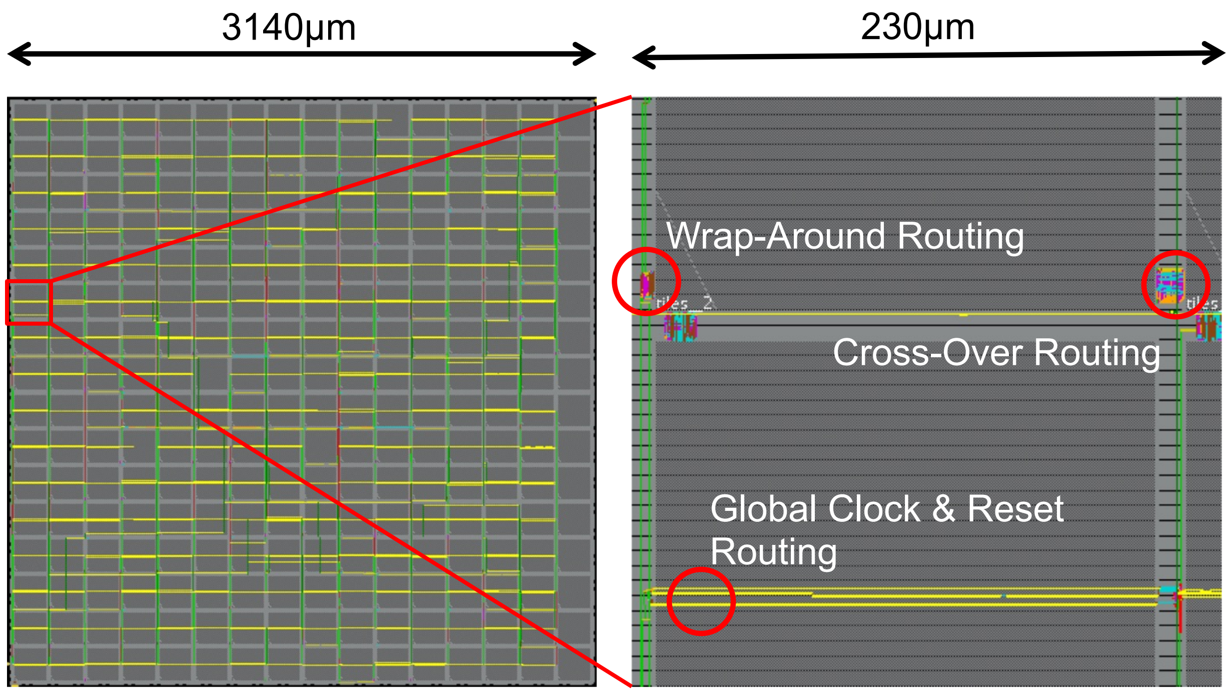
No Ruche Channels



mesh-c4r2-b64

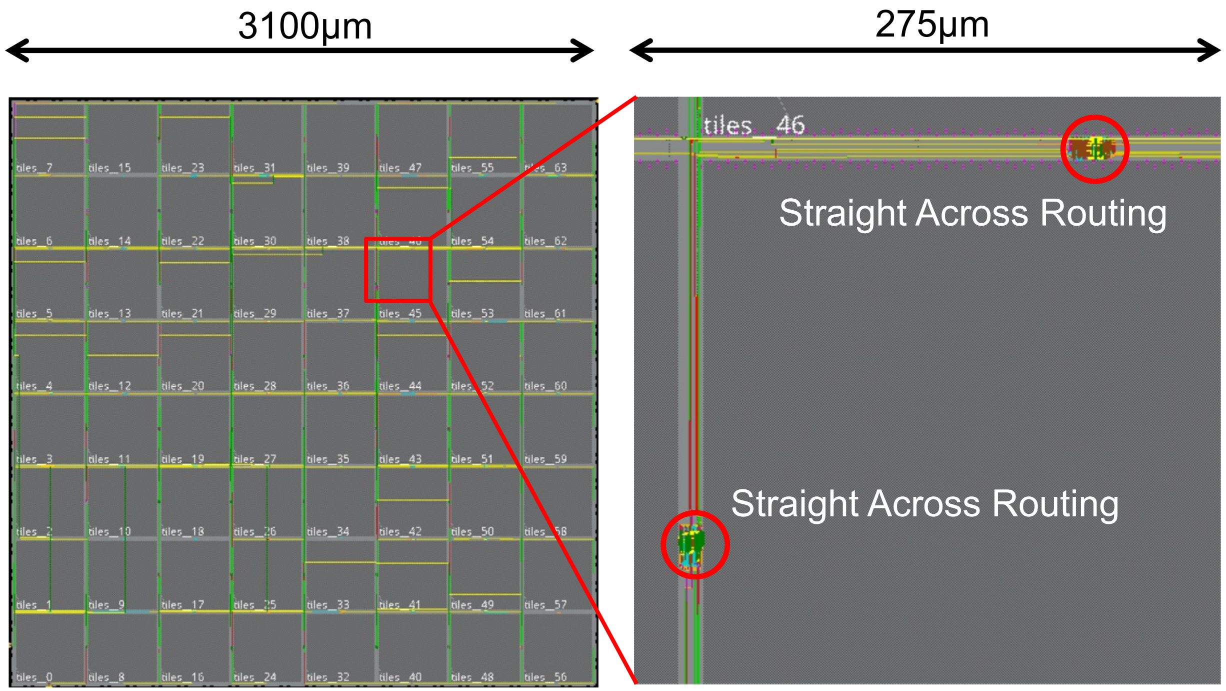
Ruche Factor of Two

# Composing Hard Macros at Chip Top-Level



torus-c1r0-b32 Full Chip

torus-c1r0-b32 Close Up

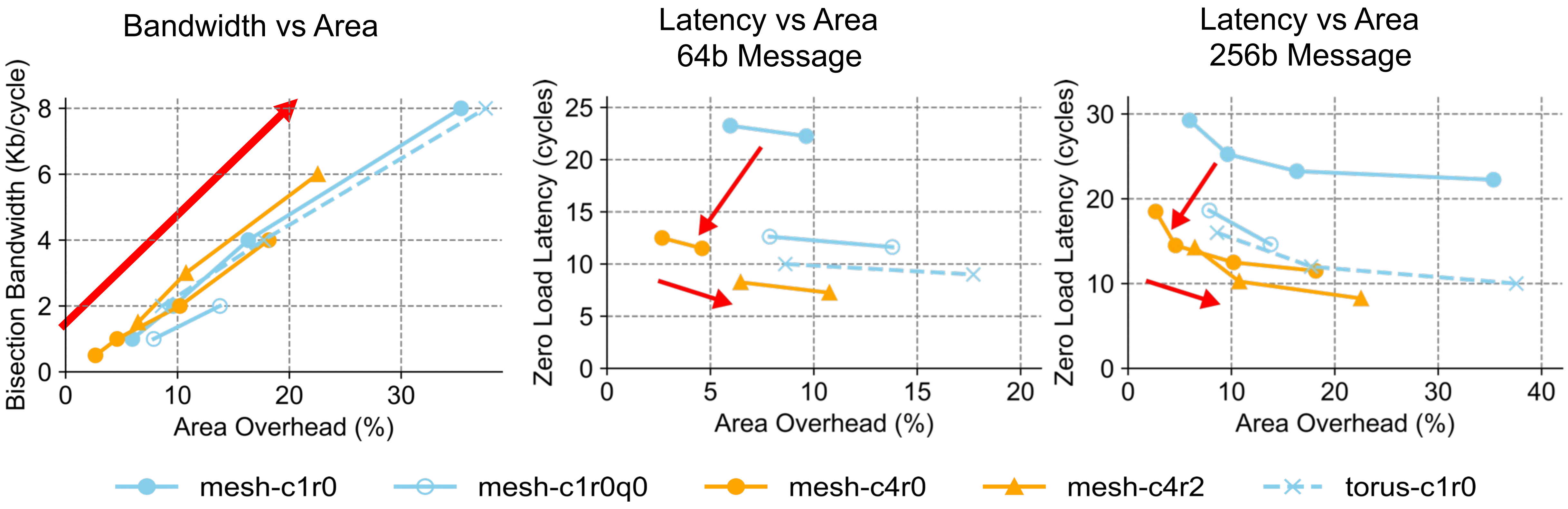


mesh-c4r2-b64 Full Chip

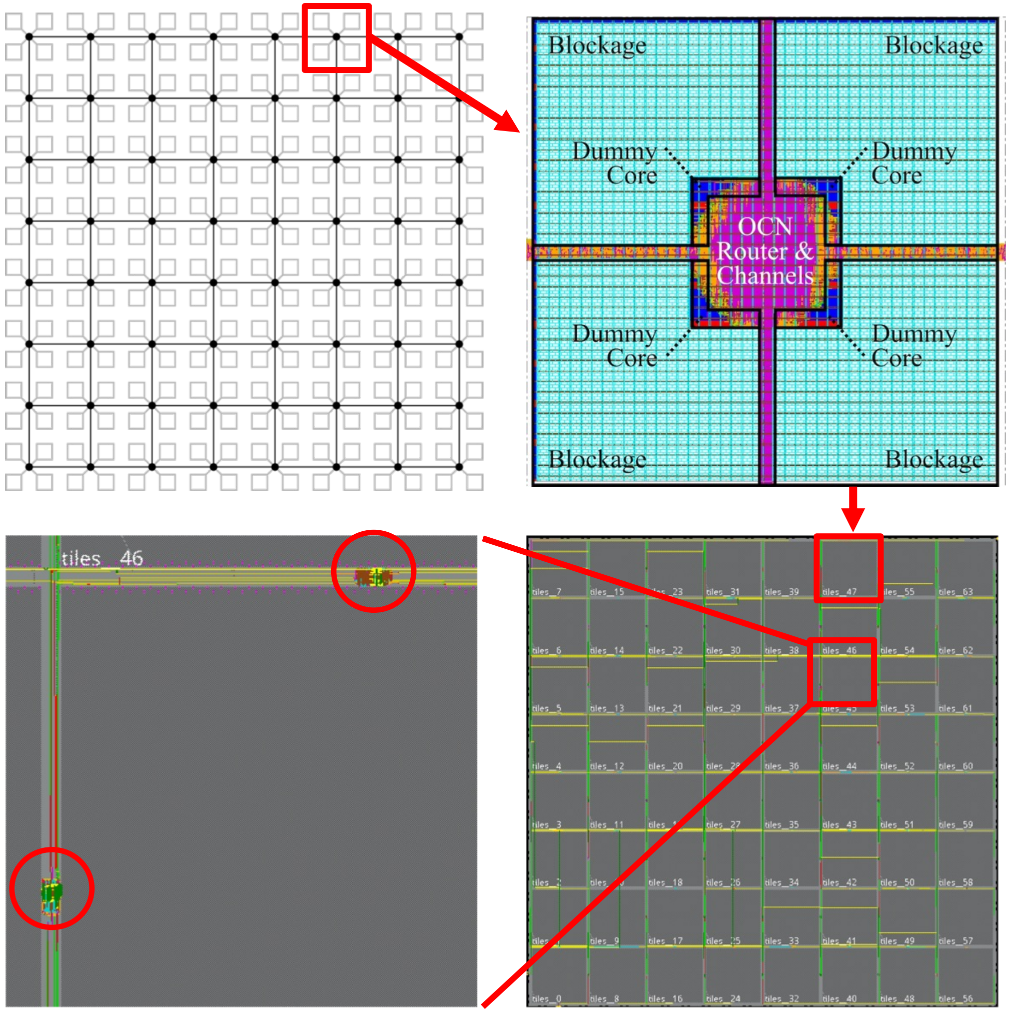
mesh-c4r2-b64 Close Up

1. 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
3. Timing closure for the hard macro must imply timing closure at the chip level

# Macro-Level Results for Promising Topologies



- Increasing bandwidth leads to increase in area for all topologies
- Increasing concentration and ruche factor leads to lower latency & lower Area



# Implementing Low-Diameter OCN for Manycore Processors Using A Tiled Physical Design Methodology

Motivation

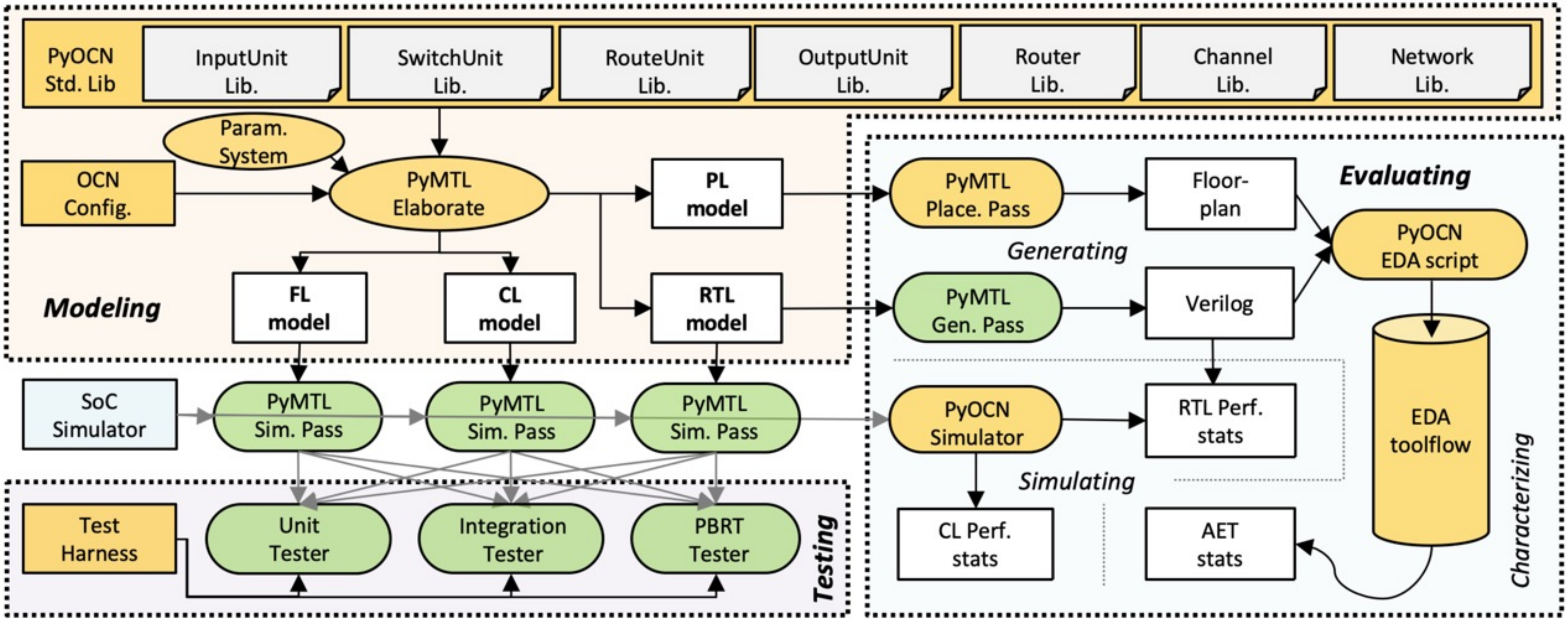
Manycore OCN Topologies

Manycore OCN Analytical Modeling

Manycore OCN Physical Design

**PyOCN Framework**

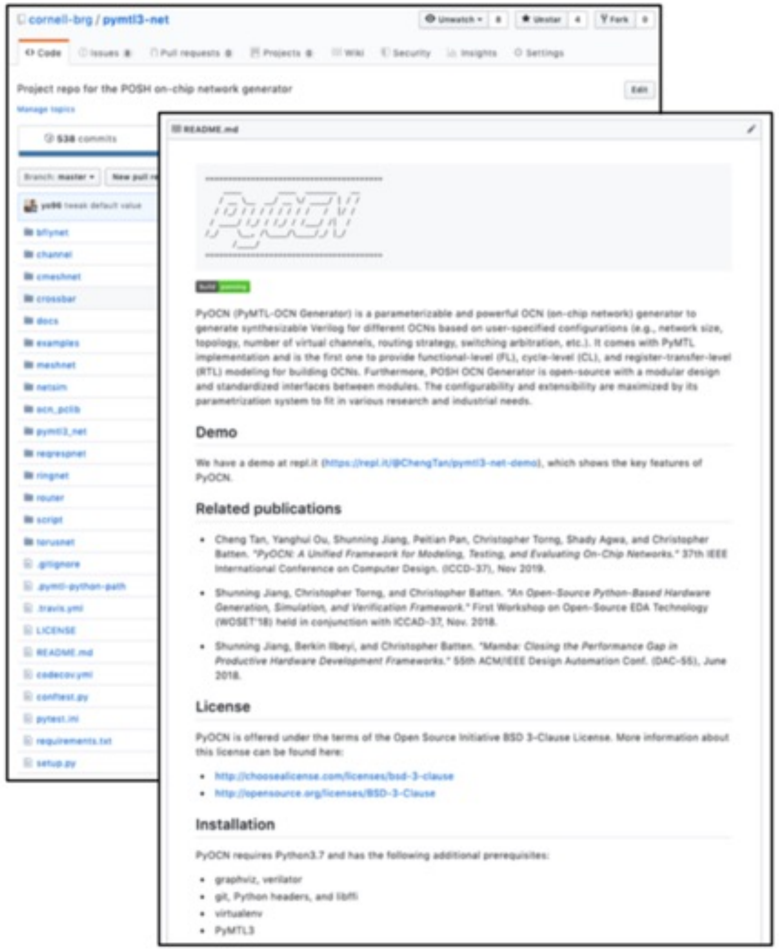
# PyOCN: A Unified Framework for Modeling, Testing, and Evaluating OCN



# PyOCN is Open-Sourced, Packaged, and Published

## Open-Sourced on GitHub

https://github.com/cornell-brg/pymtl3-net



Project repo for the POSH on-chip network generator

PyOCN (PyMTL-OCN Generator) is a parameterizable and powerful OCN (on-chip network) generator to generate synthesizable Verilog for different OCNs based on user-specified configurations (e.g., network size, topology, number of virtual channels, routing strategy, switching arbitration, etc.). It comes with PyMTL implementation and is the first one to provide functional-level (FL), cycle-level (CL), and register-transfer-level (RTL) modeling for building OCNs. Furthermore, POSH OCN Generator is open-source with a modular design and standardized interfaces between modules. The configurability and extensibility are maximized by its parametrization system to fit in various research and industrial needs.

### Demo

We have a demo at repl.it (https://repl.it/@ChengTan/pymtl3-net-demo), which shows the key features of PyOCN.

### Related publications

- Cheng Tan, Yanghui Ou, Shunning Jiang, Peitian Pan, Christopher Torng, Shady Agwa, and Christopher Batten. *"PyOCN: A Unified Framework for Modeling, Testing, and Evaluating On-Chip Networks."* 37th IEEE International Conference on Computer Design. (ICCD-37), Nov 2019.
- Shunning Jiang, Christopher Torng, and Christopher Batten. *"An Open-Source Python-Based Hardware Generation, Simulation, and Verification Framework."* First Workshop on Open-Source EDA Technology (WOSET'18) held in conjunction with ICCAD-37, Nov. 2018.
- Shunning Jiang, Berkin Ilbeyi, and Christopher Batten. *"Mamba: Closing the Performance Gap in Productive Hardware Development Frameworks."* 55th ACM/IEEE Design Automation Conf. (DAC-55), June 2018.

### License

PyOCN is offered under the terms of the Open Source Initiative BSD 3-Clause License. More information about this license can be found here:

- http://choosealicense.com/licenses/bsd-3-clause
- http://opensource.org/licenses/BSD-3-Clause

### Installation

PyOCN requires Python3.7 and has the following additional prerequisites:

- graphviz, verilator
- git, Python headers, and libffi
- virtualenv
- PyMTL3

## Packaged on PyPi

https://pypi.org/project/pymtl3-net

**To create a virtual environment and install pymtl3-net along with all of its dependencies:**

```
% python3 -m venv ${HOME}/venv/pymtl3
% source ${HOME}/venv/pymtl3/bin/activate
% pip install pymtl3-net
```

**To test a 4-terminal ring with with a single packet:**

```
% pymtl3-net test ring --nterminals 4 \
                      --dump-vcd
```

**To simulate a 2x2 mesh with specific injection rate:**

```
% pymtl3-net sim mesh --ncols 2 --nrows 2 \
                     --injection-rate 10 -v
```

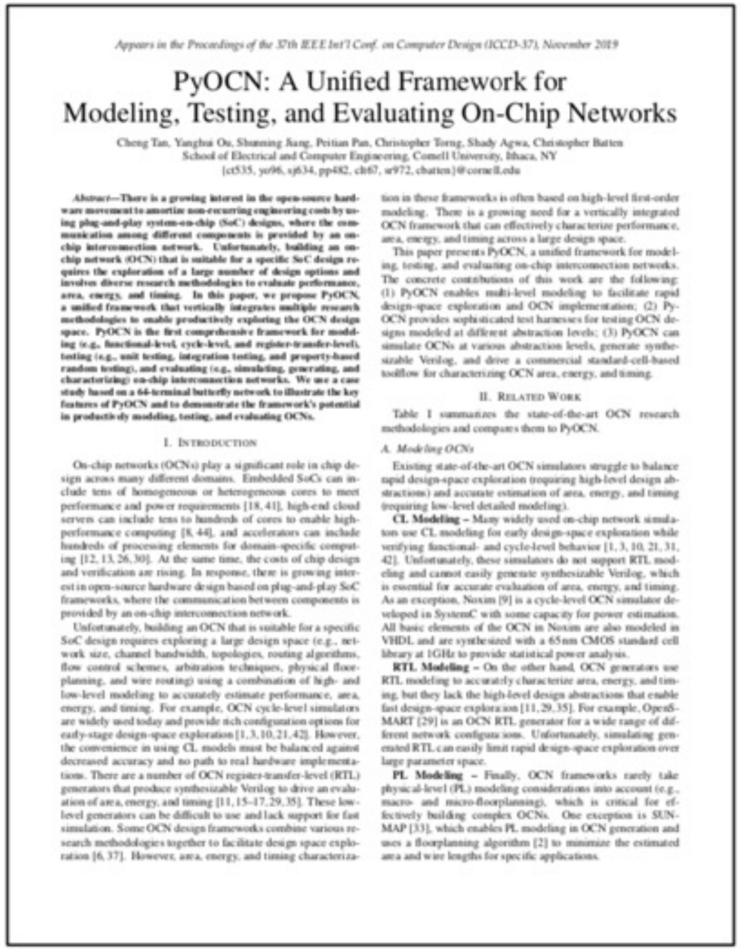
**To simulate a 2x2 mesh across injection rates:**

```
% pymtl3-net sim mesh --ncols 2 --nrows 2 \
                     --sweep -v
```

**To generate a 4x4 mesh Verilog RTL:**

```
% pymtl3-net gen mesh --ncols 4 --nrows 4
```

## IEEE Int'l Conf. on Computer Design (ICCD-37), November 2019



*Appears in the Proceedings of the 37th IEEE Int'l Conf. on Computer Design (ICCD-37), November 2019*

### PyOCN: A Unified Framework for Modeling, Testing, and Evaluating On-Chip Networks

Cheng Tan, Yanghui Ou, Shunning Jiang, Peitian Pan, Christopher Torng, Shady Agwa, Christopher Batten
School of Electrical and Computer Engineering, Cornell University, Ithaca, NY
{ct535, yo96, sj634, pp482, ct467, sr972, cbatten}@cornell.edu

***Abstract*—There is a growing interest in the open-source hardware movement to amortize non-recurring engineering costs by using plug-and-play system-on-chip (SoC) designs, where the communication among different components is provided by an on-chip interconnection network. Unfortunately, building an on-chip network (OCN) that is suitable for a specific SoC design requires the exploration of a large number of design options and involves diverse research methodologies to evaluate performance, area, energy, and timing. In this paper, we propose PyOCN, a unified framework that vertically integrates multiple research methodologies to enable productively exploring the OCN design space. PyOCN is the first comprehensive framework for modeling (e.g., functional-level, cycle-level, and register-transfer-level), testing (e.g., unit testing, integration testing, and property-based random testing), and evaluating (e.g., simulating, generating, and characterizing) on-chip interconnection networks. We use a case study based on a 64-terminal butterfly network to illustrate the key features of PyOCN and to demonstrate the framework's potential in productively modeling, testing, and evaluating OCNs.**

#### I. Introduction

On-chip networks (OCNs) play a significant role in chip design across many different domains. Embedded SoCs can include tens of homogeneous or heterogeneous cores to meet performance and power requirements [18, 41], high-end cloud servers can include tens to hundreds of cores to enable high-performance computing [8, 44], and accelerators can include hundreds of processing elements for domain-specific computing [12, 13, 26, 30]. At the same time, the costs of chip design and verification are rising. In response, there is growing interest in open-source hardware design based on plug-and-play SoC frameworks, where the communication between components is provided by an on-chip interconnection network.

Unfortunately, building an OCN that is suitable for a specific SoC design requires exploring a large design space (e.g., network size, channel bandwidth, topologies, routing algorithms, flow control schemes, arbitration techniques, physical floorplanning, and wire routing) using a combination of high- and low-level modeling to accurately estimate performance, area, energy, and timing. For example, OCN cycle-level simulators are widely used today and provide rich configuration options for early-stage design-space exploration [1, 3, 10, 21, 42]. However, the convenience in using CL models must be balanced against decreased accuracy and no path to real hardware implementations. There are a number of OCN register-transfer-level (RTL) generators that produce synthesizable Verilog to drive an evaluation of area, energy, and timing [11, 15–17, 29, 35]. These low-level generators can be difficult to use and lack support for fast simulation. Some OCN design frameworks combine various research methodologies together to facilitate design space exploration [6, 37]. However, area, energy, and timing characterization in these frameworks is often based on high-level first-order modeling. There is a growing need for a vertically integrated OCN framework that can effectively characterize performance, area, energy, and timing across a large design space.

This paper presents PyOCN, a unified framework for modeling, testing, and evaluating on-chip interconnection networks. The concrete contributions of this work are the following: (1) PyOCN enables multi-level modeling to facilitate rapid design-space exploration and OCN implementation; (2) PyOCN provides sophisticated test harnesses for testing OCN designs modeled at different abstraction levels; (3) PyOCN can simulate OCNs at various abstraction levels, generate synthesizable Verilog, and drive a commercial standard-cell-based toolflow for characterizing OCN area, energy, and timing.

#### II. Related Work

Table I summarizes the state-of-the-art OCN research methodologies and compares them to PyOCN.

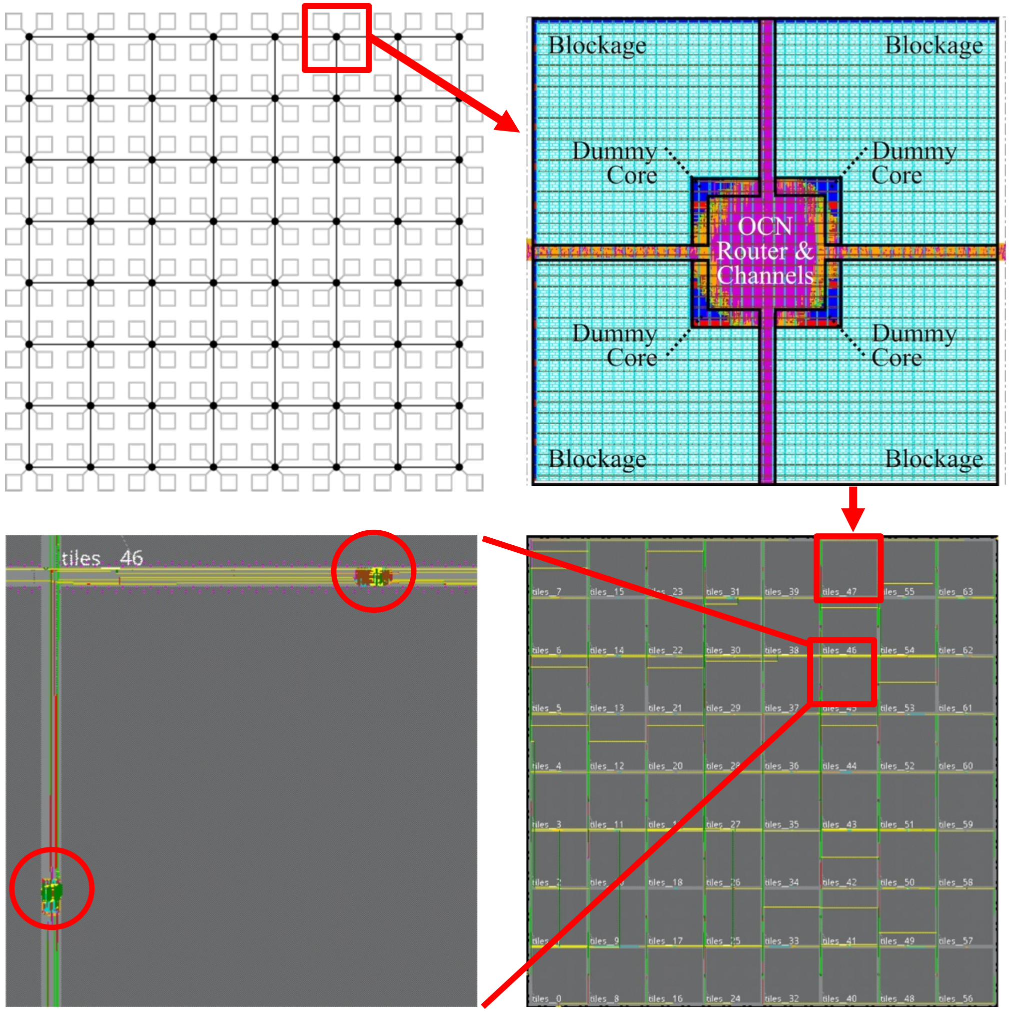
*A. Modeling OCNs*

Existing state-of-the-art OCN simulators struggle to balance rapid design-space exploration (requiring high-level design abstractions) and accurate estimation of area, energy, and timing (requiring low-level detailed modeling).

**CL Modeling** – Many widely used on-chip network simulators use CL modeling for early design-space exploration while verifying functional- and cycle-level behavior [1, 3, 10, 21, 31, 42]. Unfortunately, these simulators do not support RTL modeling and cannot easily generate synthesizable Verilog, which is essential for accurate evaluation of area, energy, and timing. As an exception, Noxim [9] is a cycle-level OCN simulator developed in SystemC with some capacity for power estimation. All basic elements of the OCN in Noxim are also modeled in VHDL and are synthesized with a 65nm CMOS standard cell library at 1GHz to provide statistical power analysis.

**RTL Modeling** – On the other hand, OCN generators use RTL modeling to accurately characterize area, energy, and timing, but they lack the high-level design abstractions that enable fast design-space exploraion [11, 29, 35]. For example, OpenSMART [29] is an OCN RTL generator for a wide range of different network configuraions. Unfortunately, simulating generated RTL can easily limit rapid design-space exploration over large parameter space.

**PL Modeling** – Finally, OCN frameworks rarely take physical-level (PL) modeling considerations into account (e.g., macro- and micro-floorplanning), which is critical for effectively building complex OCNs. One exception is SUNMAP [33], which enables PL modeling in OCN generation and uses a floorplanning algorithm [2] to minimize the estimated area and wire lengths for specific applications.



# Implementing Low-Diameter OCN for Manycore Processors Using A Tiled Physical Design Methodology

- We present a tiled physical design methodology to implement low-diameter OCNs for manycore processors
- We analyze the latency, area, and bandwidth tradeoffs of 12 topologies with different concentration and ruche factor
- Long channels are the key to fully exploiting the VLSI wiring capability but must leverage a tiled physical design methodology
- Moderate concentration and ruching can reduce latency at similar area and bisection bandwidth

This work was supported in part by NSF CRI Award #1512937, DARPA POSH Award #FA8650-18-2-7852, and equipment, tool, and/or physical IP donations from Intel, Synopsys, and Cadence.