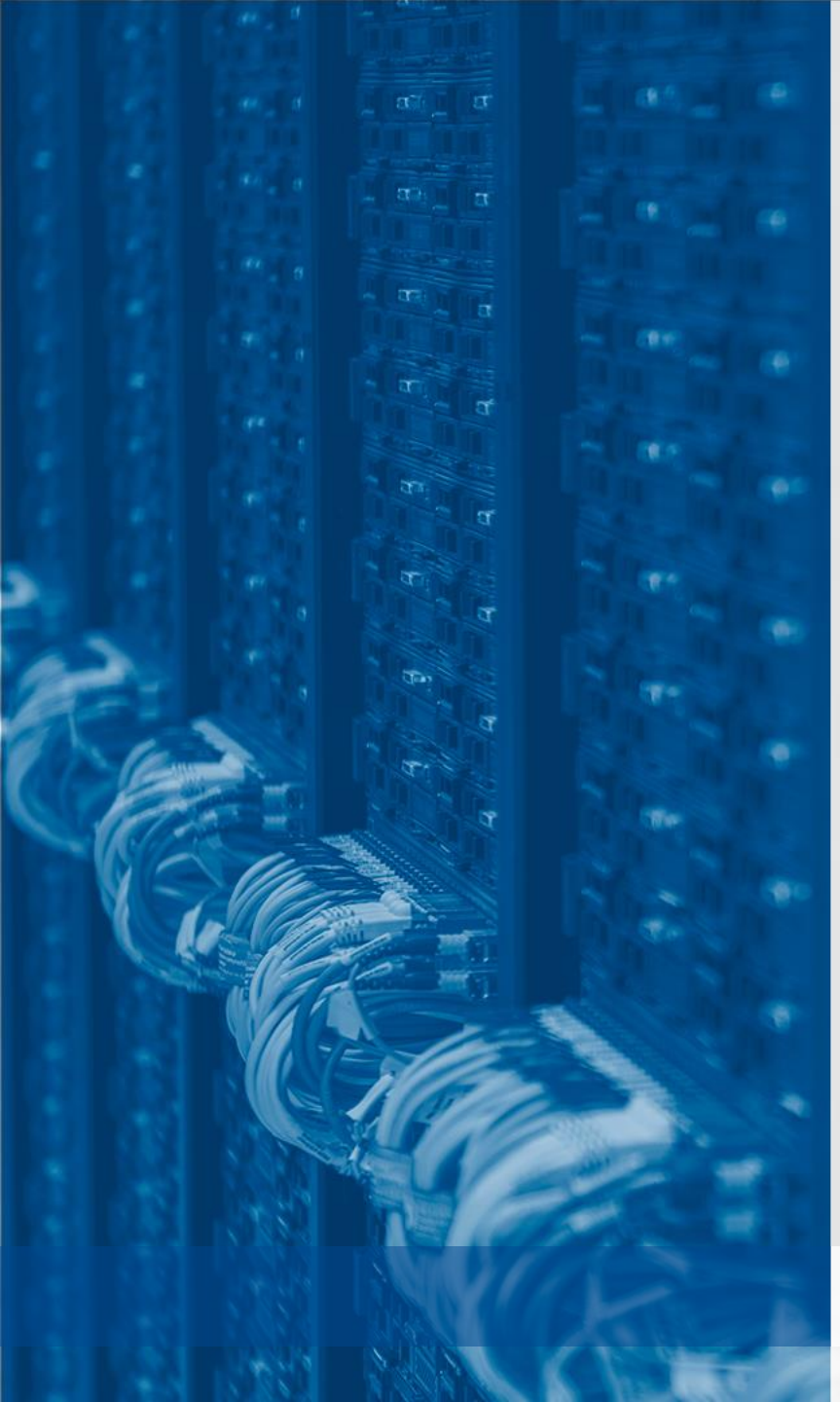


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

UC SANTA BARBARA

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Past, Present and Future of Designing, Integrating and Simulating RTL Models

Guillem López Paradís, Jonathan Balkind,
Adrià Armejach, Miquel Moretó

OSCAR 2024

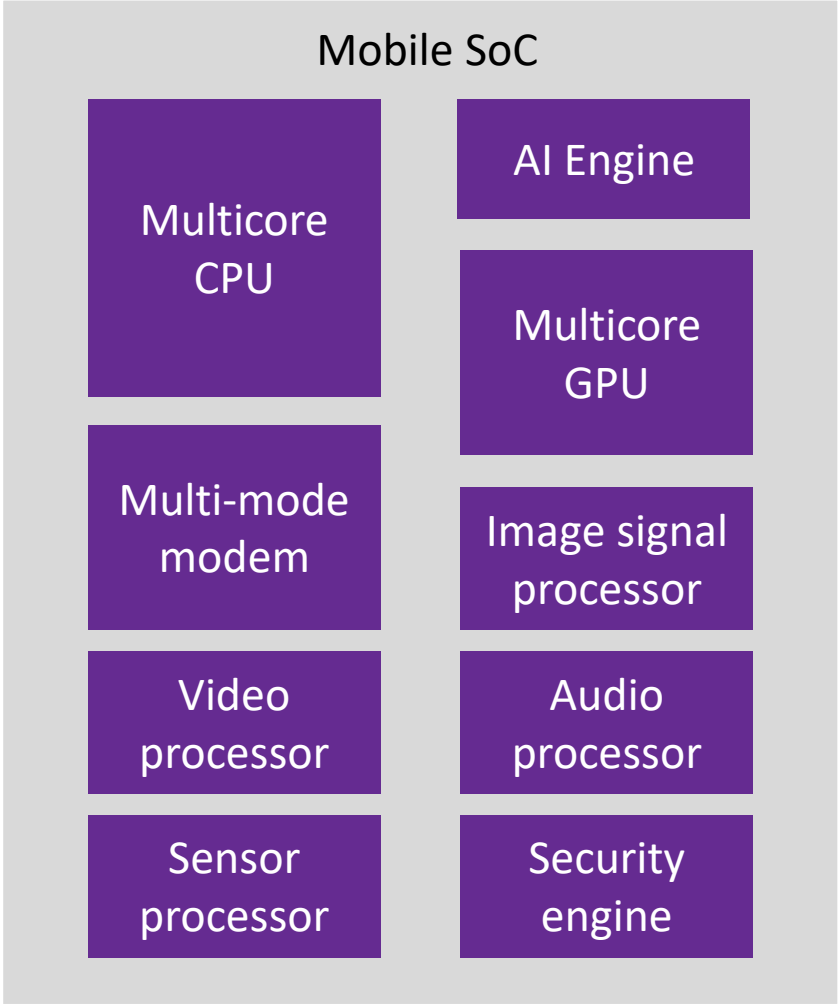
Unió Europea
Fons Social Europeu
L'FSE inverteix en el teu futur

Motivation

- **Boom in fabricating new hw**

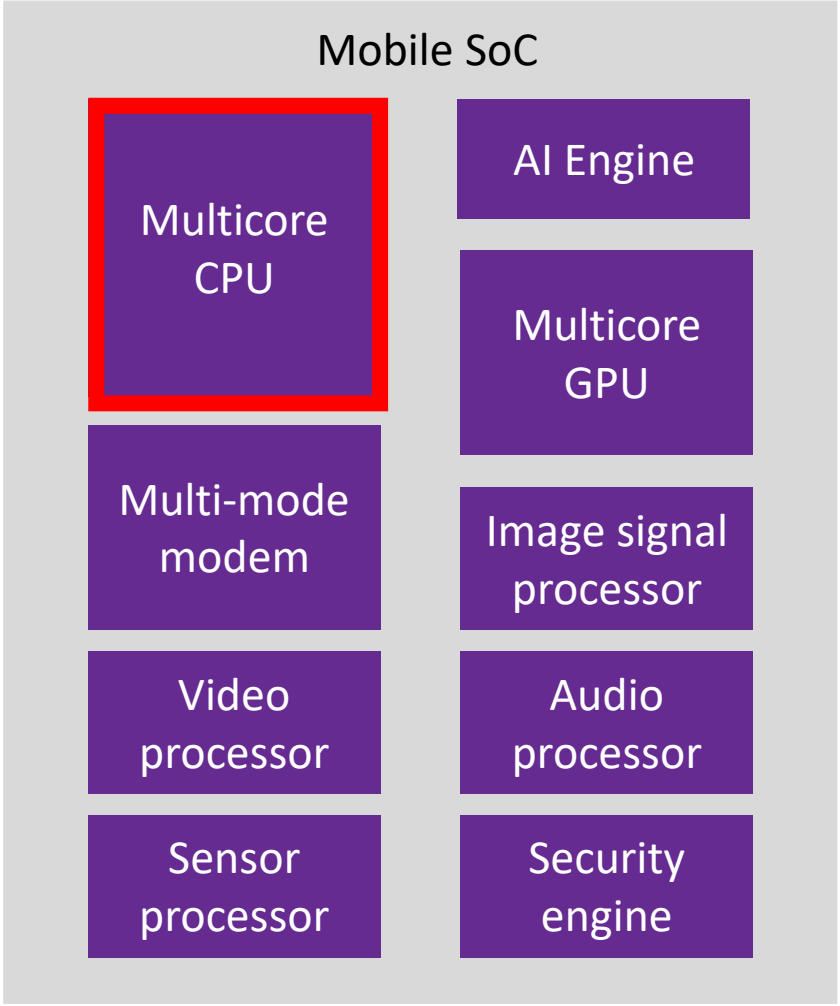
Motivation

- Boom in fabricating new hw
- **Heterogeneous hardware on the same SoC requires complex integration and verification processes**



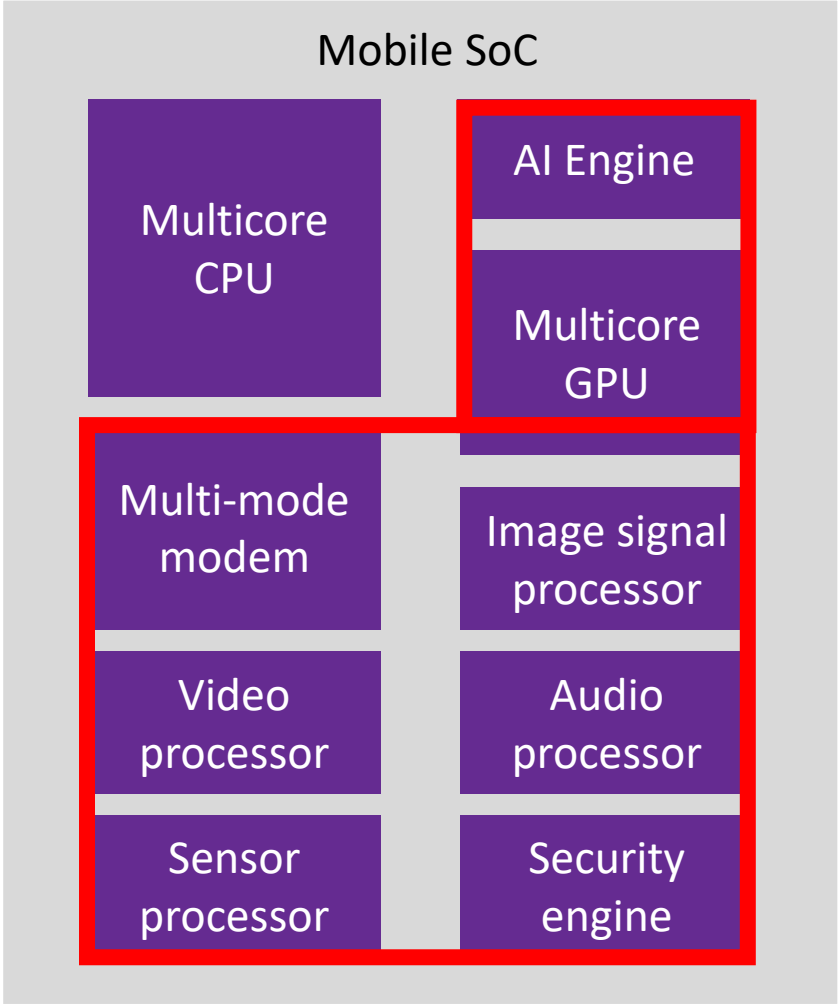
Motivation

- Boom in fabricating new hw
- **Heterogeneous hardware on the same SoC requires complex integration and verification processes**



Motivation

- Boom in fabricating new hw
- **Heterogeneous hardware on the same SoC requires complex integration and verification processes**



Motivation

- Boom in fabricating new hw
- Heterogeneous hardware on the same SoC requires complex integration and verification processes
- **Improve the tools to verify large-scale hardware designs!**

Outline

- **Gem5+RTL: A Full-System RTL Simulation Infrastructure**

- Fast Behavioural RTL Simulation of 10B Transistor SoC Designs with Metro-MPI

Gem5+RTL Objectives

- Framework that **enables easy integration of existing RTL hardware** blocks within a **SoC** for full-system simulations

Gem5+RTL Objectives

- Framework that enables easy integration of existing RTL hardware blocks within a SoC for full-system simulations

- Deliver a **comprehensive hardware/software ecosystem** where **all** the **main components** of the **SoC** are **present** with a complete software stack

Gem5+RTL Objectives

- Framework that enables easy integration of existing RTL hardware blocks within a SoC for full-system simulations

- Deliver a comprehensive hardware/software ecosystem where all the main components of the SoC are present with a complete software stack

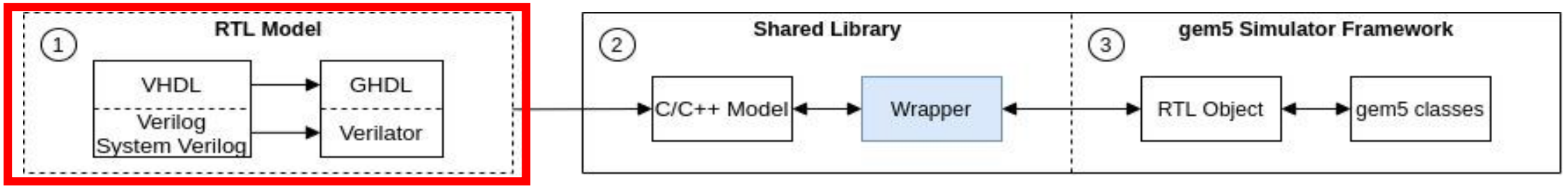
- **Enable testing** the **implemented functionality** of these hardware blocks and also, the **expected performance** they will provide on an existing **SoC design**

Framework Design

1. **We use Verilator and GHDL to obtain a C++ model from an RTL model written in Verilog/SystemVerilog and VHDL**

2. We provide a wrapper to interact with it and gem5. Then, the wrapper and the C++ model are combined into a shared library

3. In gem5, a generic framework is provided to ease the integration of a wide range of potential hardware designs: generic RTLObject class

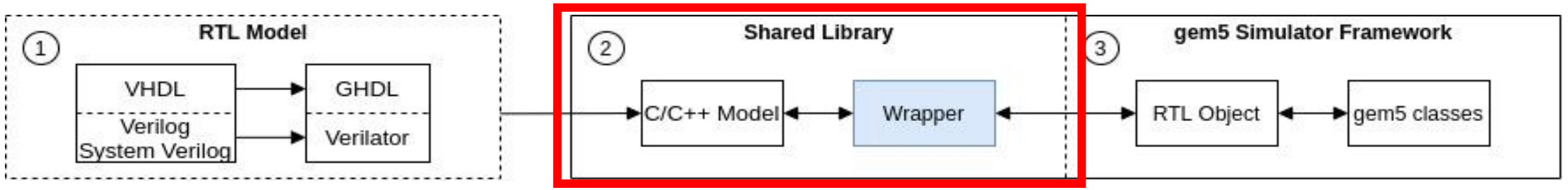


Framework Design

1. We use Verilator and GHDL to obtain a C++ model from an RTL model written in Verilog/SystemVerilog and VHDL

2. **We provide a wrapper to interact with it and gem5. Then, the wrapper and the C++ model are combined into a shared library**

3. In gem5, a generic framework is provided to ease the integration of a wide range of potential hardware designs: generic RTLObject class

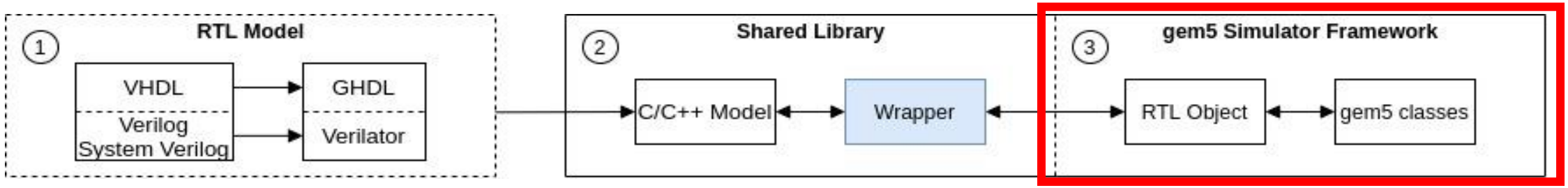


Framework Design

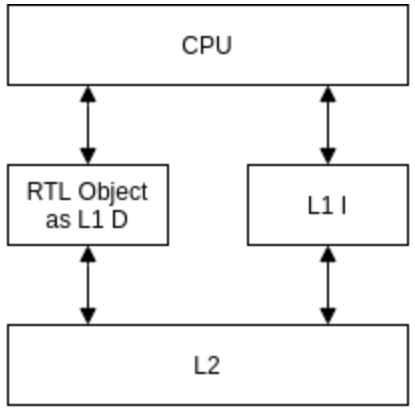
1. We use Verilator and GHDL to obtain a C++ model from an RTL model written in Verilog/SystemVerilog and VHDL

2. We provide a wrapper to interact with it and gem5. Then, the wrapper and the C++ model are combined into a shared library

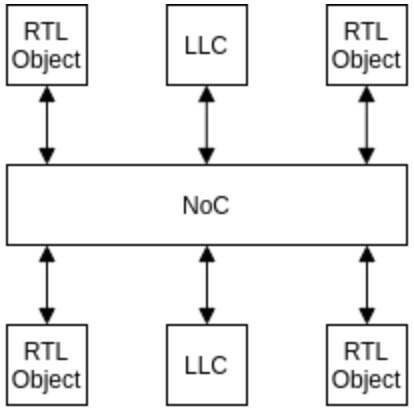
3. **In gem5, a generic framework is provided to ease the integration of a wide range of potential hardware designs: generic RTLObject class**



Connectivity Examples



(a) Cache configuration



(b) NoC design exploration

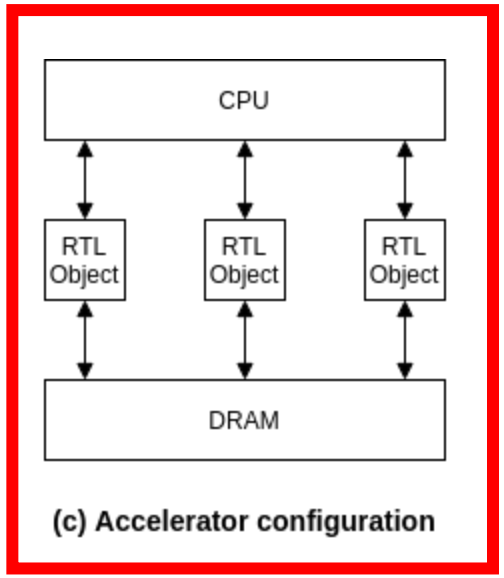
(c) Accelerator configuration

(d) PMU configuration

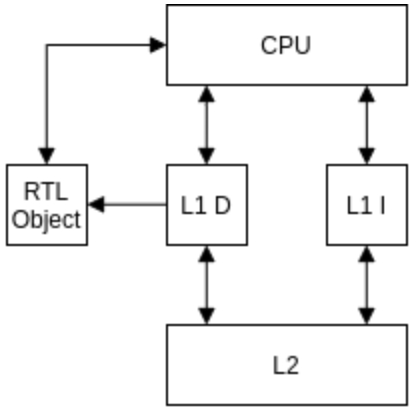
Connectivity Examples

(a) Cache configuration

(b) NoC design exploration



(c) Accelerator configuration



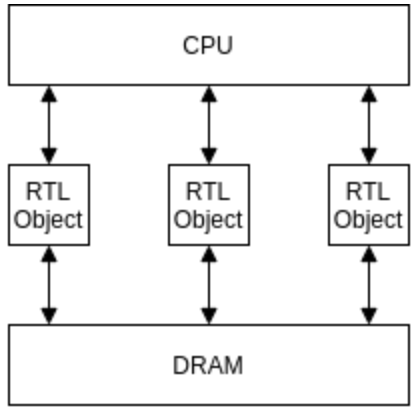
(d) PMU configuration

Accelerator Use Case

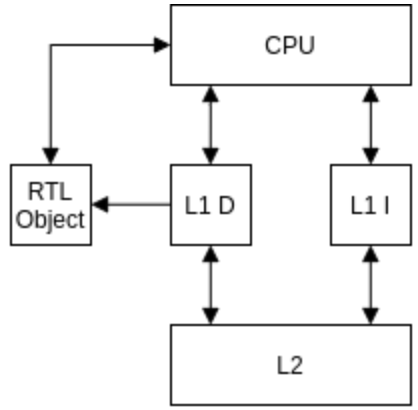
Connectivity Examples

(a) Cache configuration

(b) NoC design exploration



(c) Accelerator configuration



(d) PMU configuration

PMU Use Case

Acknowledgments

arm

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Gem5+RTL

https://gitlab.bsc.es/glopez/gem5-rtl

guillem.lopez@bsc.es

Unió Europea
Fons Social Europeu
L'FSE inverteix en el teu futur

Outline

- Gem5+RTL: A Full-System RTL Simulation Infrastructure

- **Fast Behavioural RTL Simulation of 10B Transistor SoC Designs with Metro-MPI**

Motivation: RTL Simulation Performance

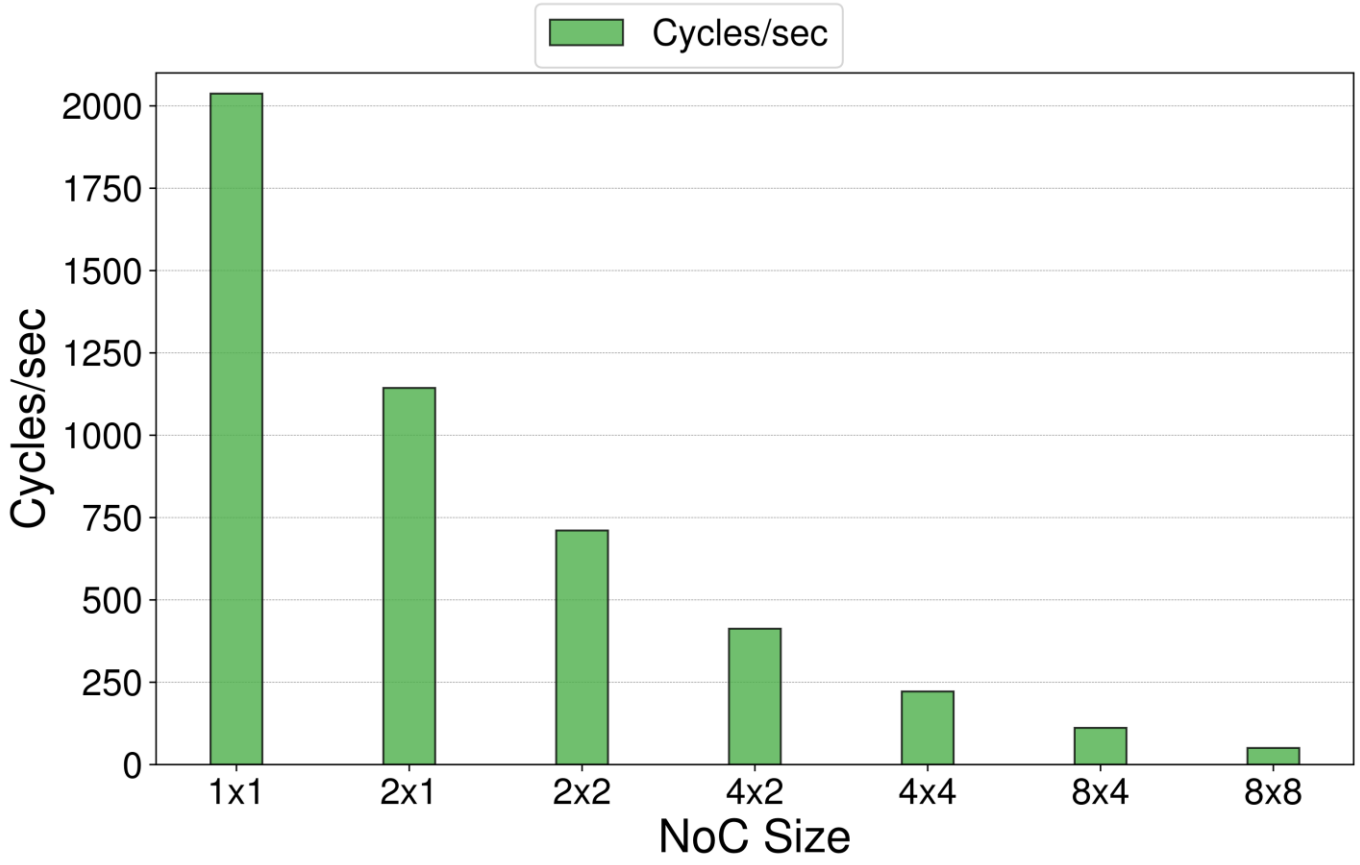
- **SoCs today are reaching 10B+ transistors in scale**

Name	#Cores	#Billions Transistors
Apple M2	8-12	20 - 67
AWS Graviton 3	64	55
Esperanto ET-SoC-1	1024	24

Motivation: RTL Simulation Performance

- SoCs today are reaching 10B+ transistors in scale

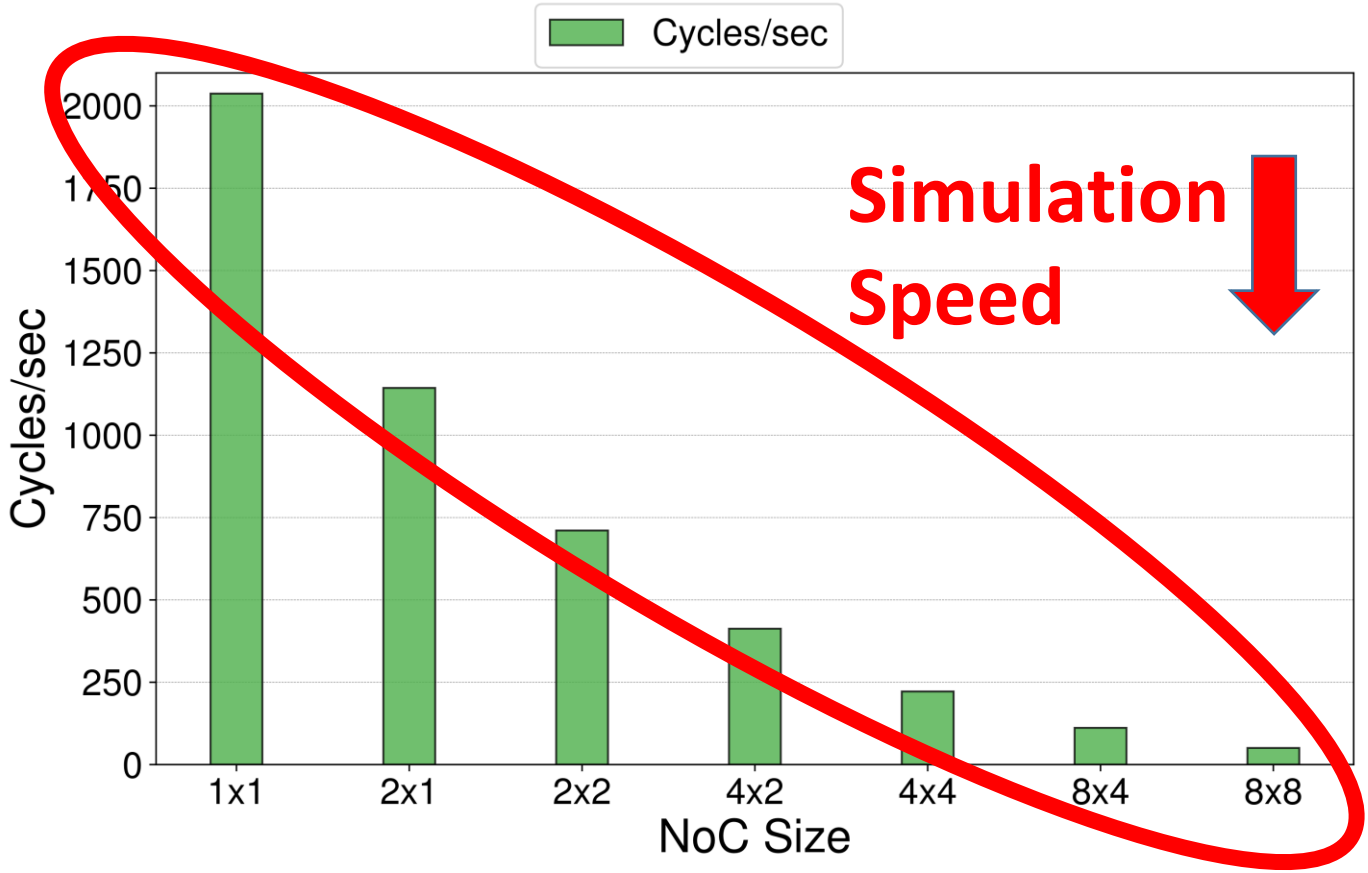
- **RTL Simulation performance drops as the design grows**



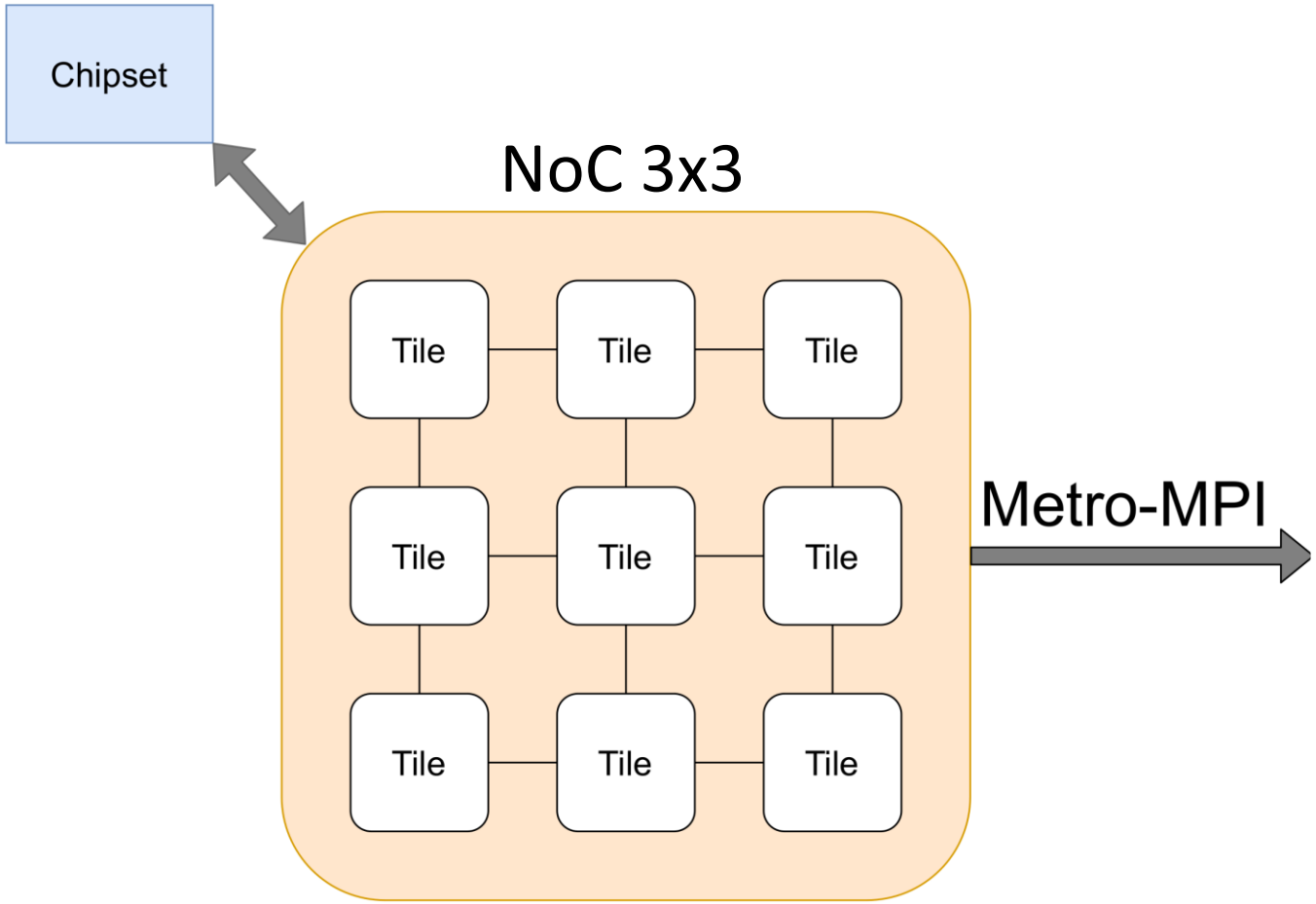
Motivation: RTL Simulation Performance

- SoCs today are reaching 10B+ transistors in scale

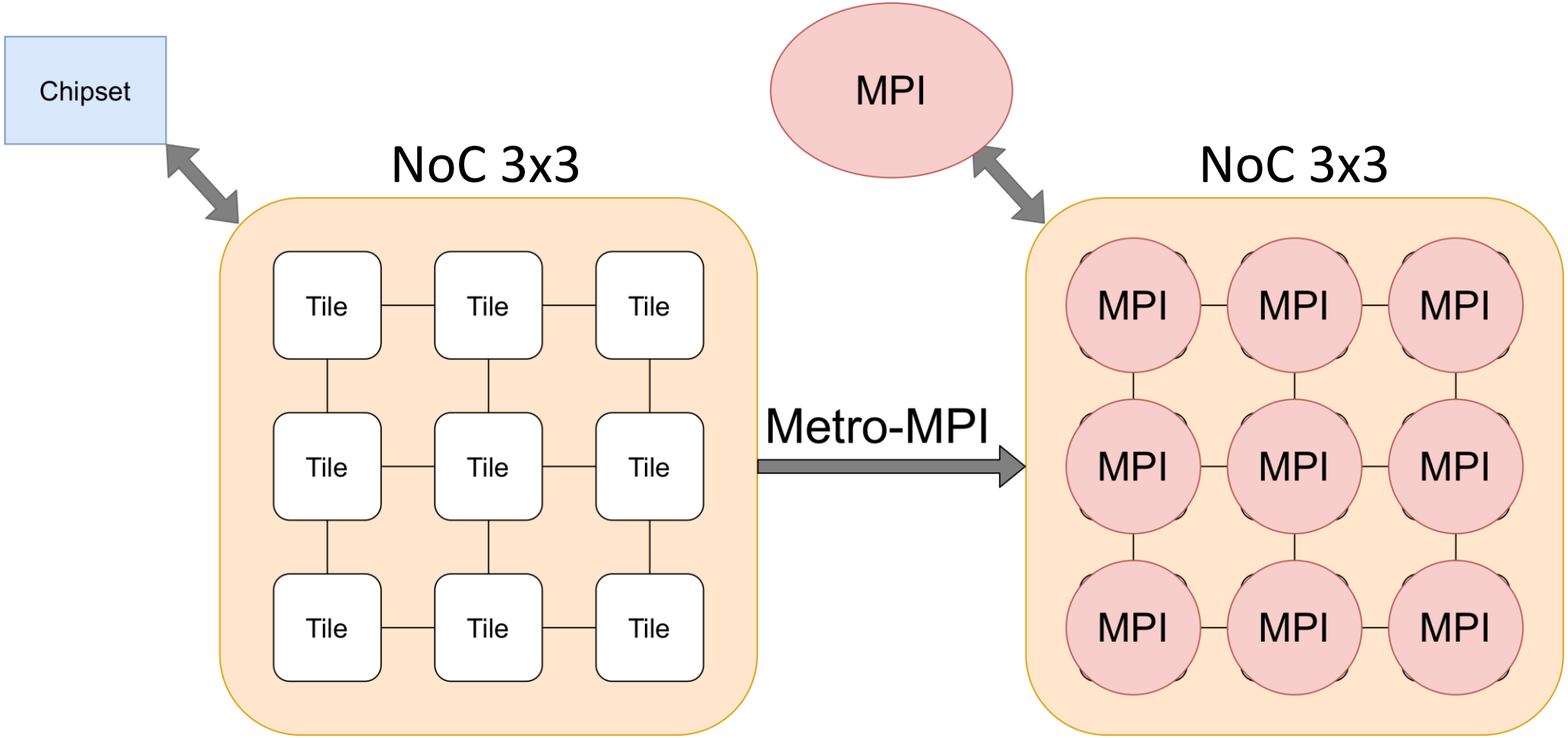
- **RTL Simulation performance drops as the design grows**



NoC RTL-Simulation with MPI

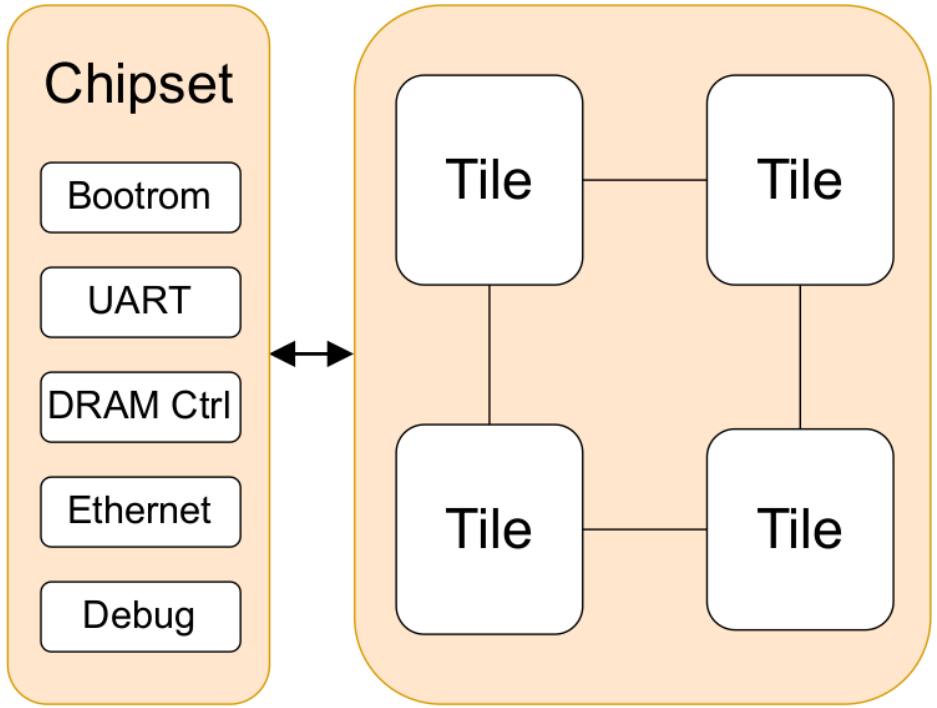


NoC RTL-Simulation with MPI



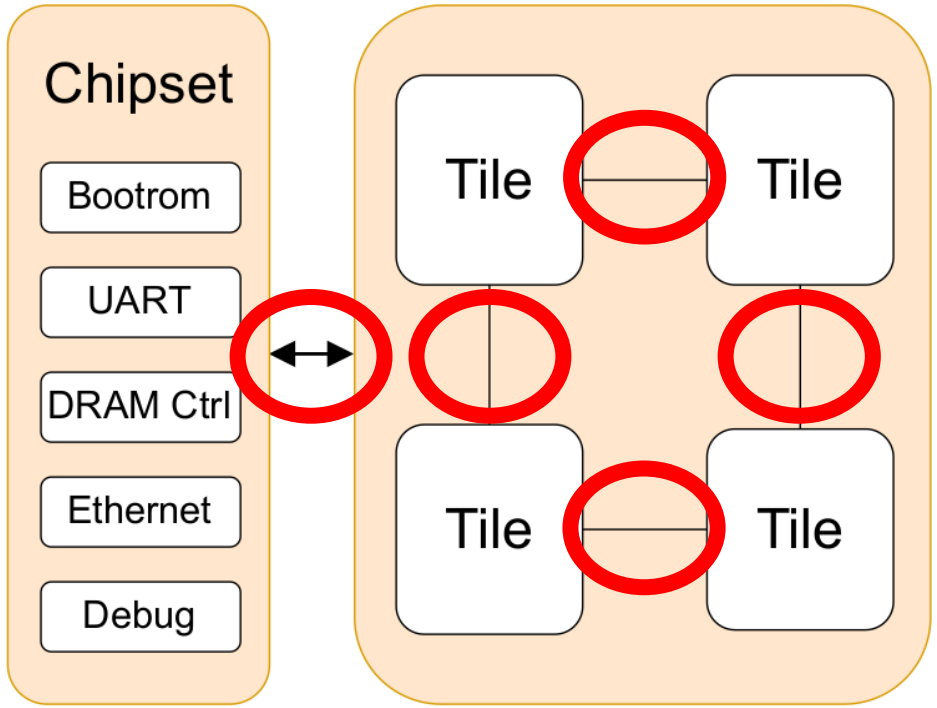
Exploting SoCs' Natural Boundaries

- **Partition the design at "latency-insensitive" interfaces (NoCs, AXI, etc)**



Exploting SoCs' Natural Boundaries

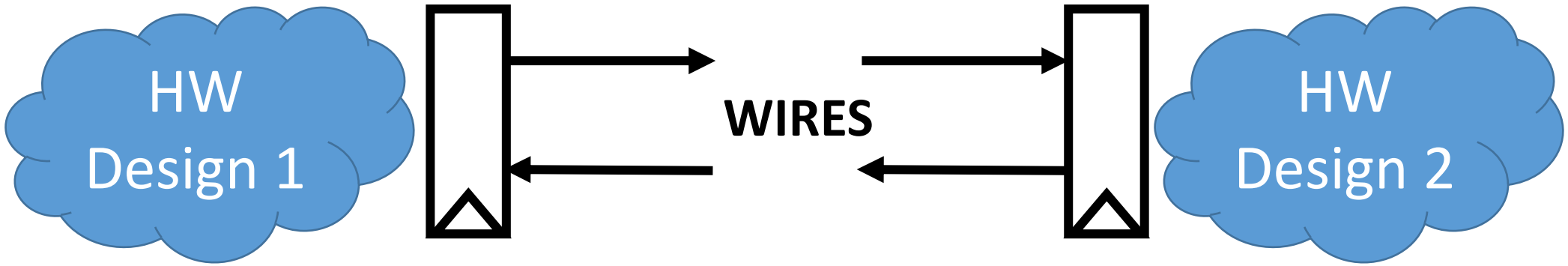
- **Partition the design at "latency-insensitive" interfaces (NoCs, AXI, etc)**



Exploting SoCs' Natural Boundaries

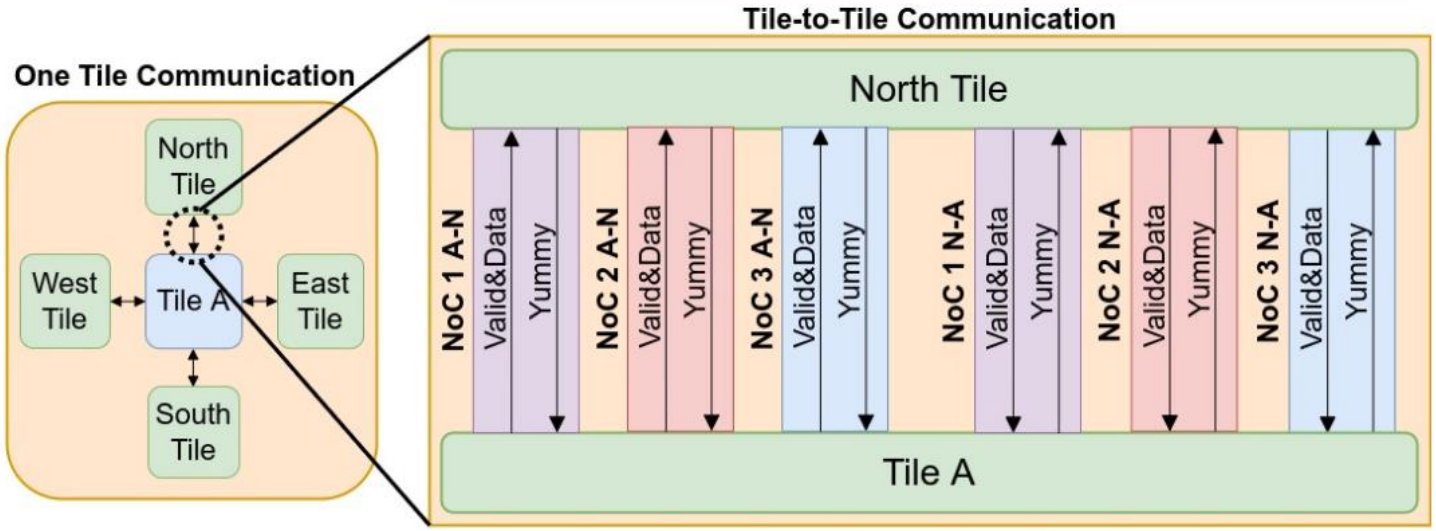
- Partition the design at "latency-insensitive" interfaces (NoCs, AXI, etc)

- **Replace NoC wires with MPI messages**

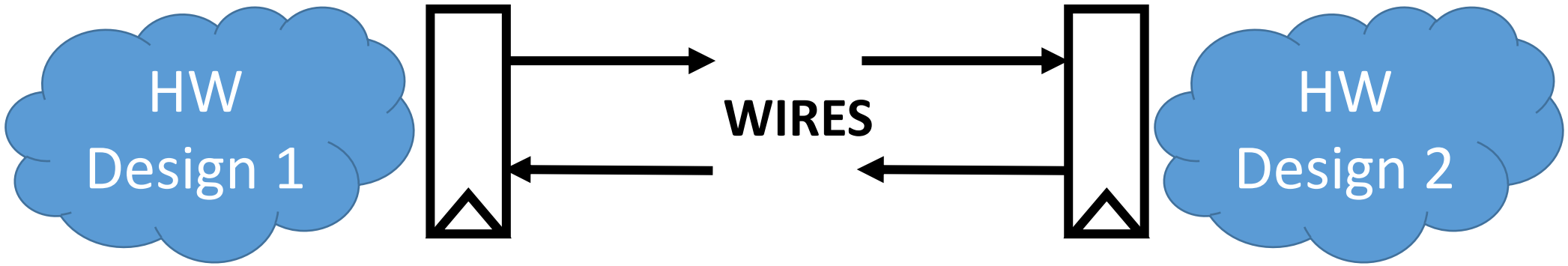


Exploting SoCs' Natural Boundaries

- Partition the design at "latency-insensitive" interfaces (NoCs, AXI, etc)

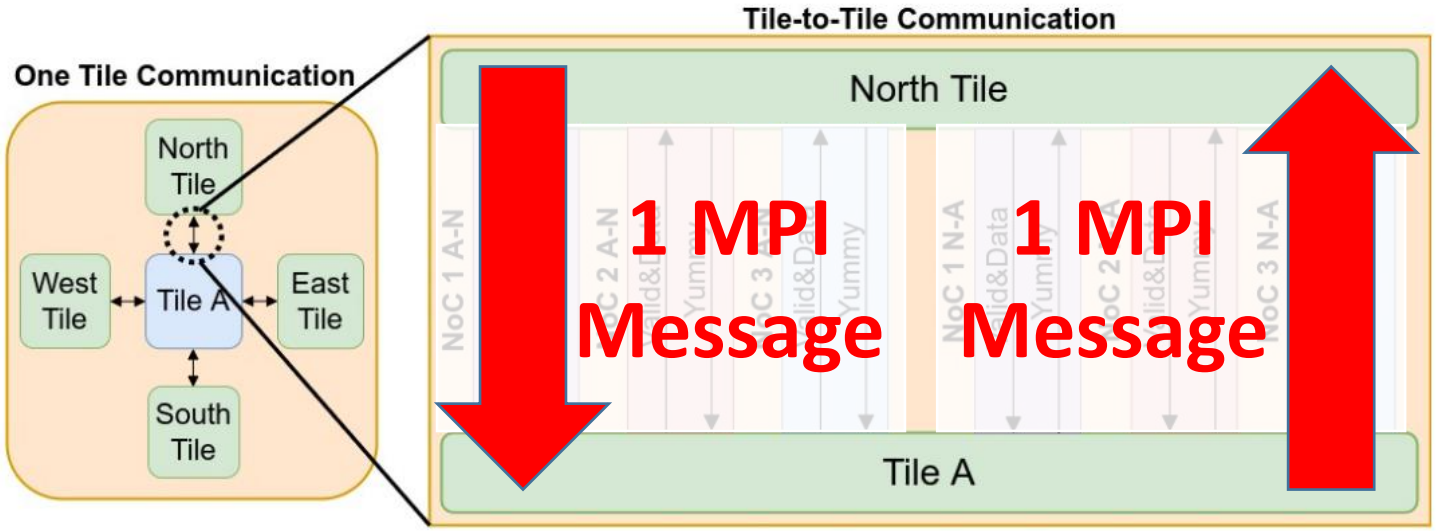


- **Replace NoC wires with MPI messages**



Exploting SoCs' Natural Boundaries

- Partition the design at "latency-insensitive" interfaces (NoCs, AXI, etc)



- **Replace NoC wires with MPI messages**

Methodology

- **We use OpenPiton+Ariane → chip sizes from 1x1 to 32x32 (1024 tiles)**

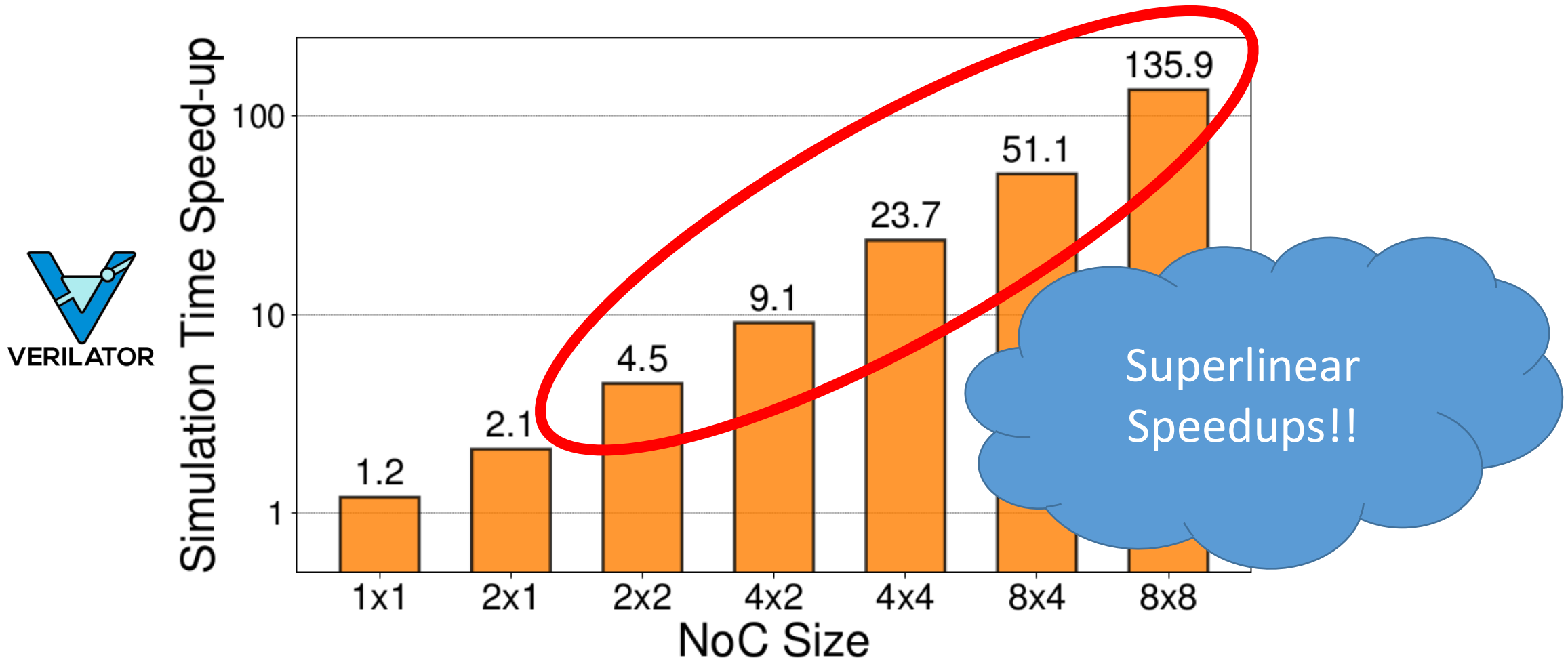
- **Testbench: Atomic synchronized token passing app**

Methodology

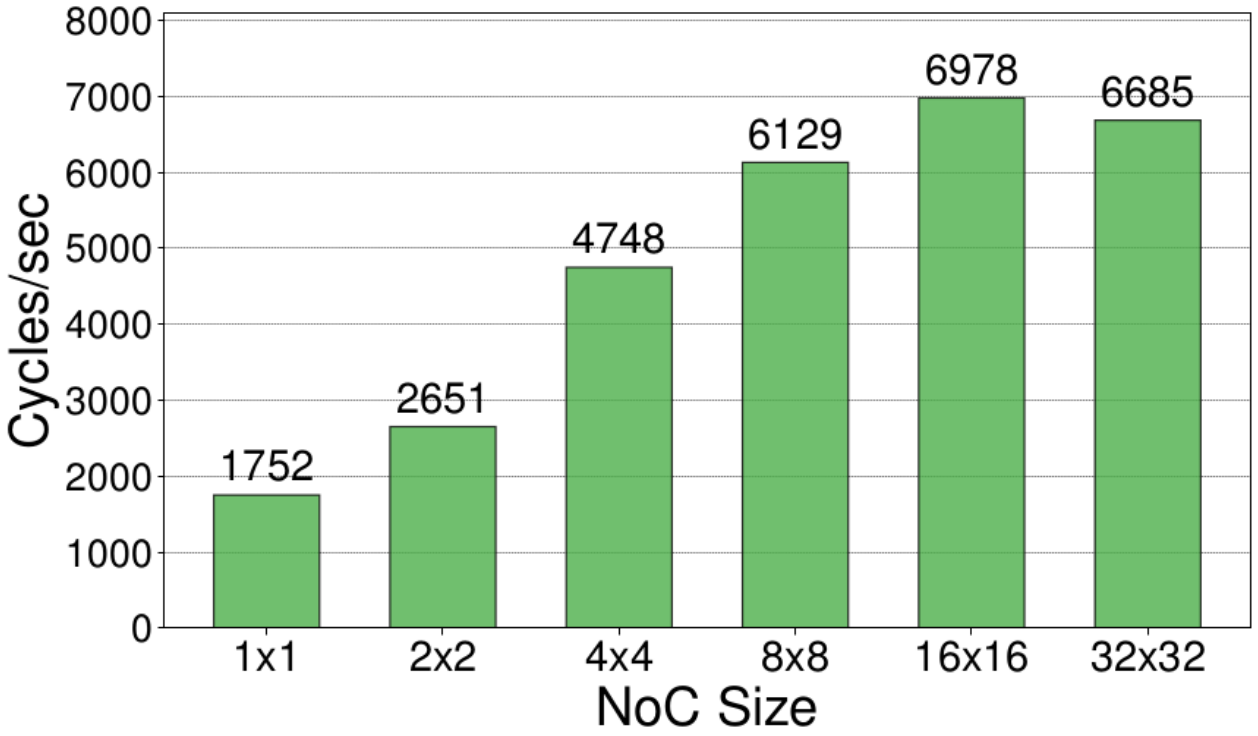
- We use OpenPiton+Ariane → chip sizes from 1x1 to 32x32 (1024 tiles)
- Testbench: Atomic synchronized token passing app
- **Simulators: Verilator and a "*Big 3*" RTL Simulator**
- **We use MareNostrum 4 Supercomputer with 100Gbs Network**
  - **1 Node has 48 cores**

NoC Size	1x1	4x4	32x32
#MPI/Cores	2	17	1025
#Nodes	1	1	22

Metro-MPI Simulation Time Speedup

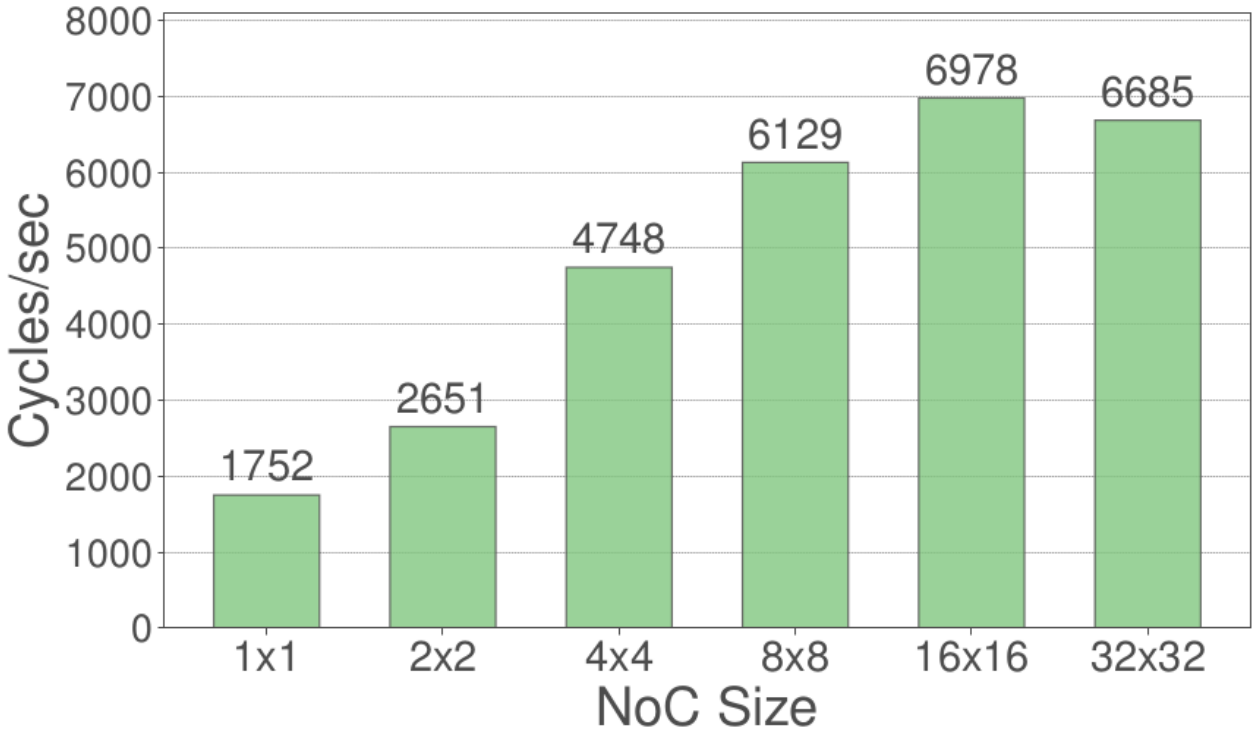


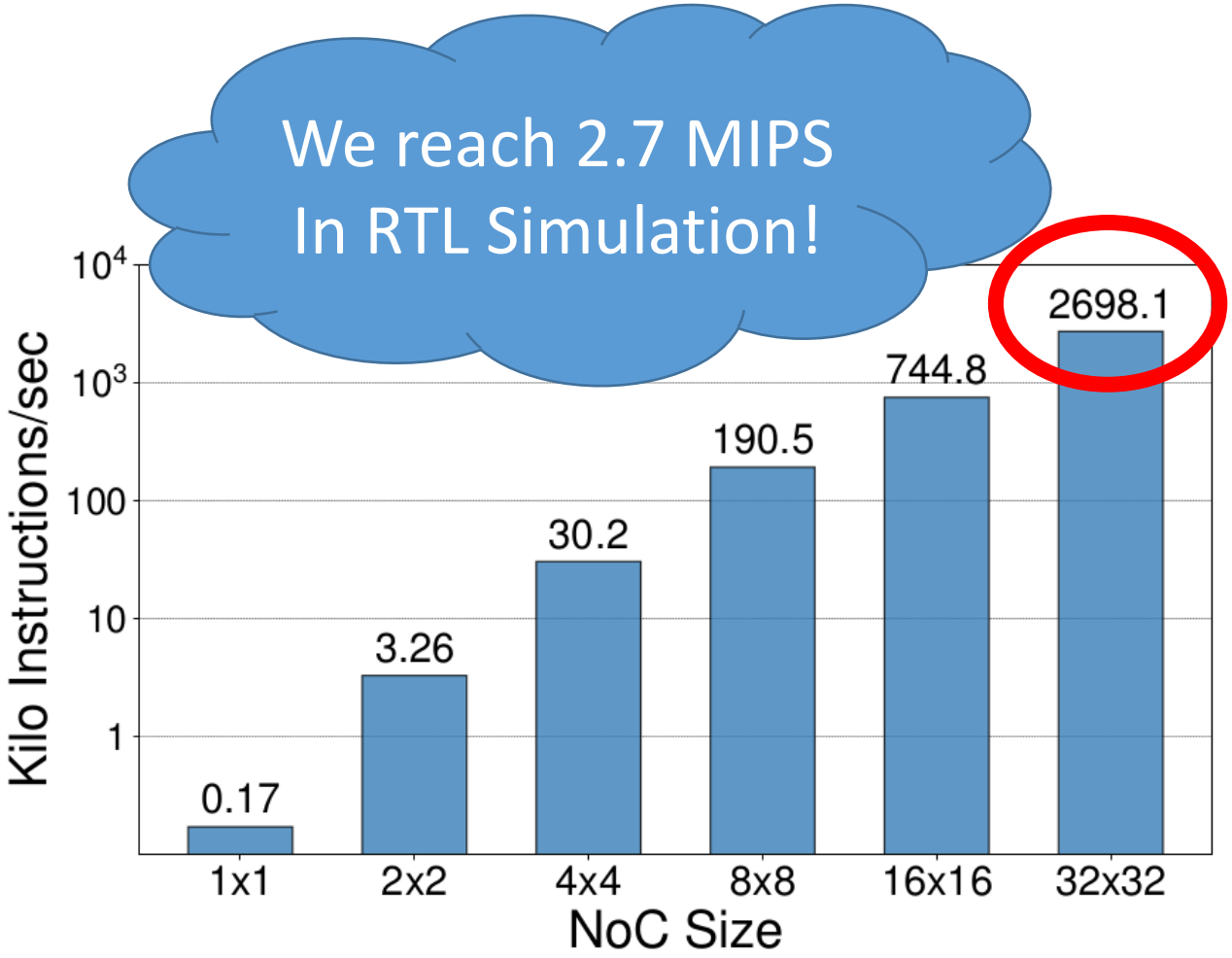
Metro-MPI Simulated Cycles/sec



VERILATOR

Metro-MPI Simulated Instruction/sec

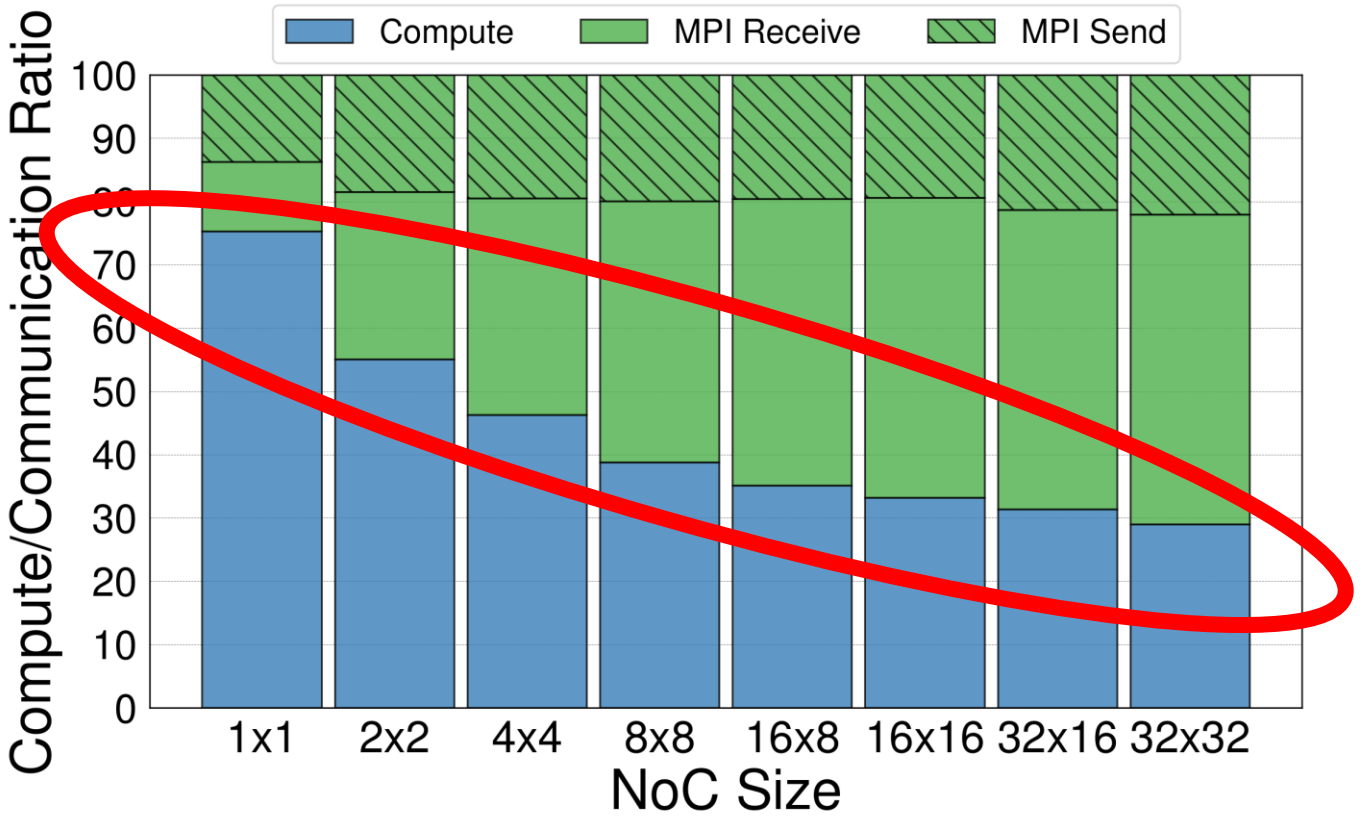




VERILATOR

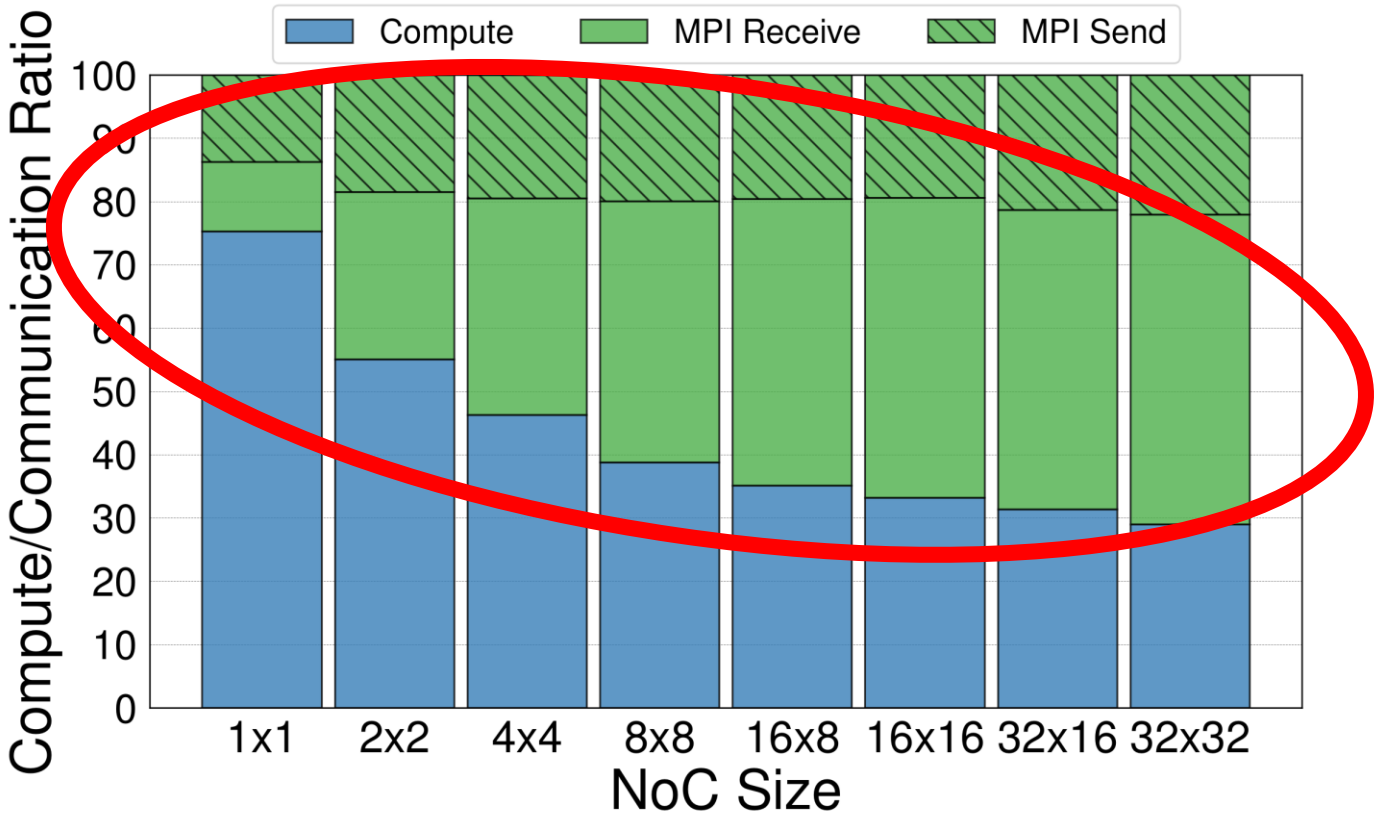
MPI Overhead

- **Compute Ratio decreases as the size of the simulated design increases (#MPI processes increases)**



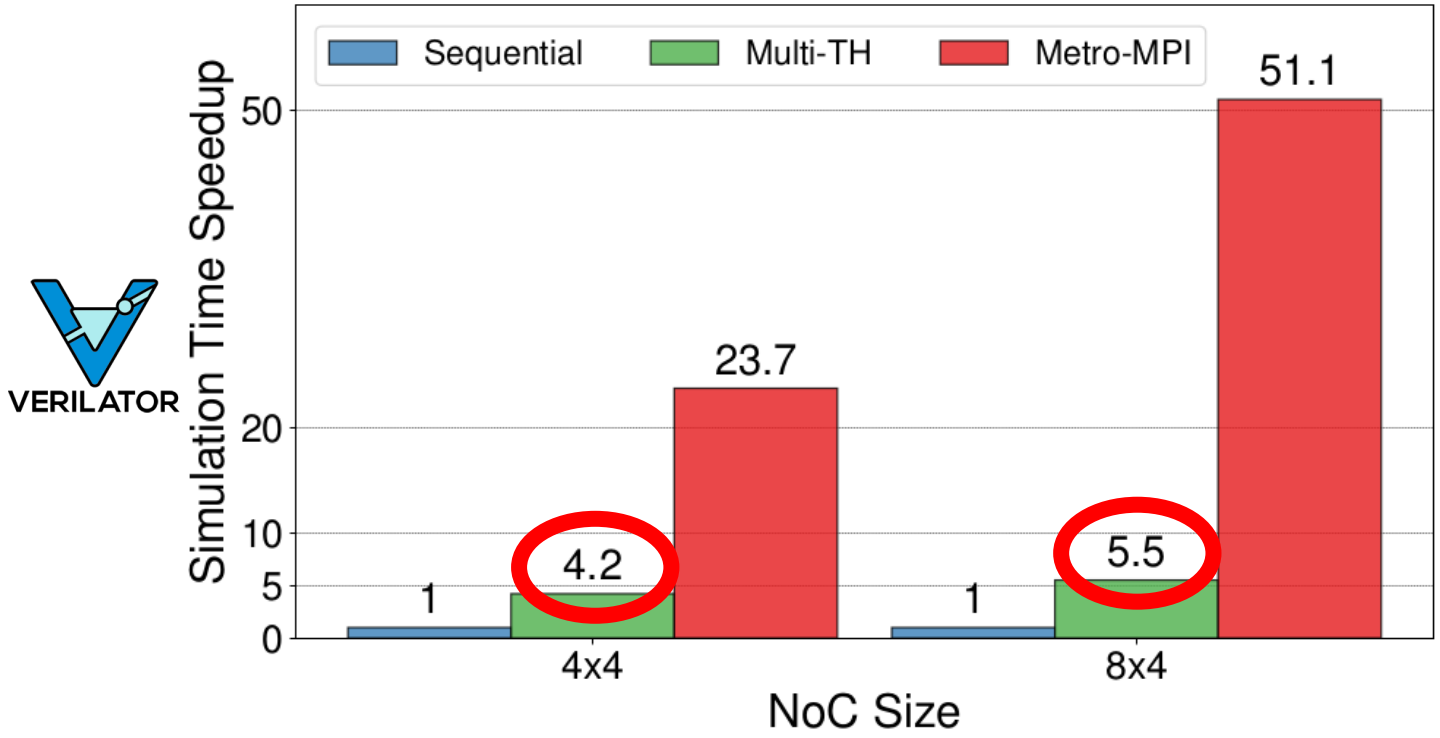
MPI Overhead

- Compute Ratio decreases as the size of the simulated design increases (#MPI processes increases)
- **MPI Receive increases (sync)**



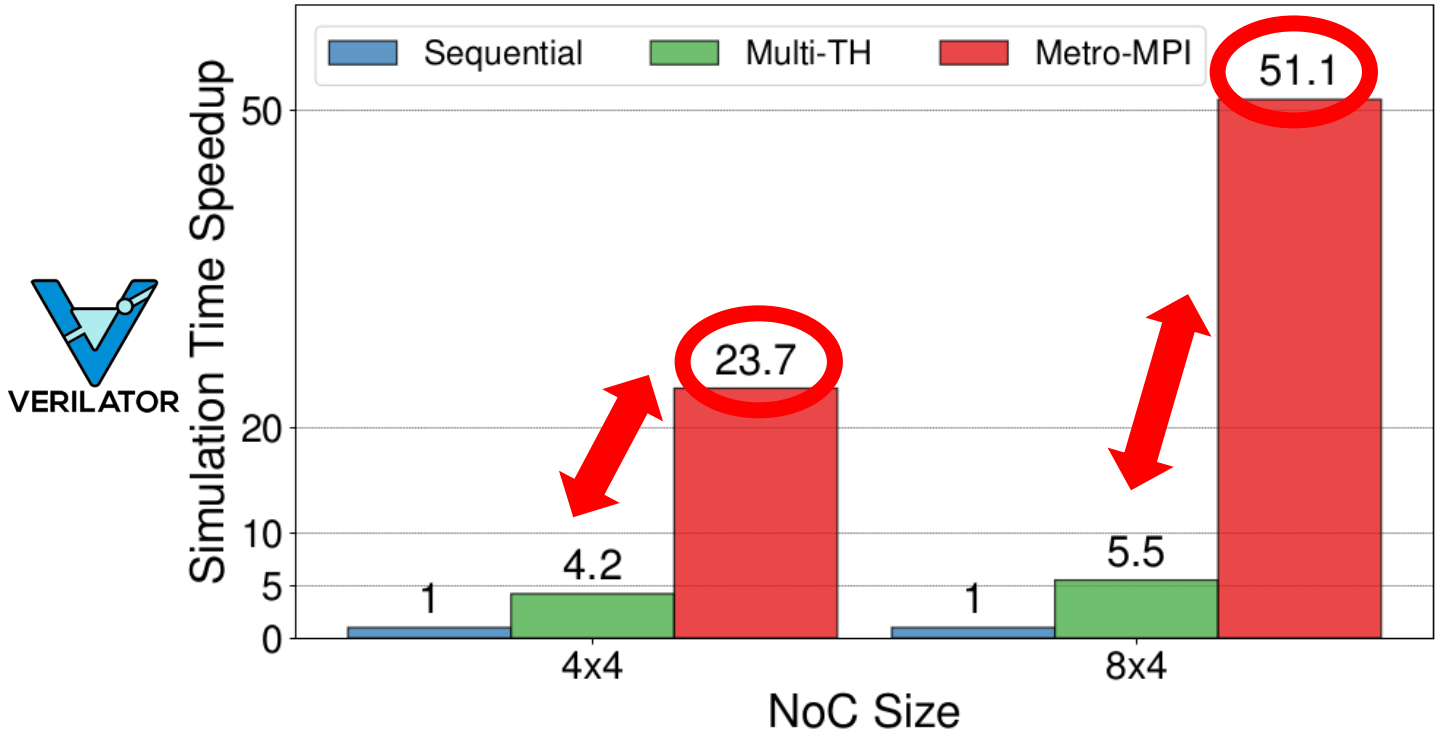
Metro-MPI vs Verilator Multithreading

- **Verilator natively supports acceleration with pthreads**
- **This speeds up simulation by 4.2x and 5.5x, using as many threads as tiles**



Metro-MPI vs Verilator Multithreading

- Verilator natively supports acceleration with pthreads
- This speeds up simulation by 4.2x and 5.5x, using as many threads as tiles
- **Metro-MPI outperforms Verilator multithreading by a further 5.64x and 9.29x**

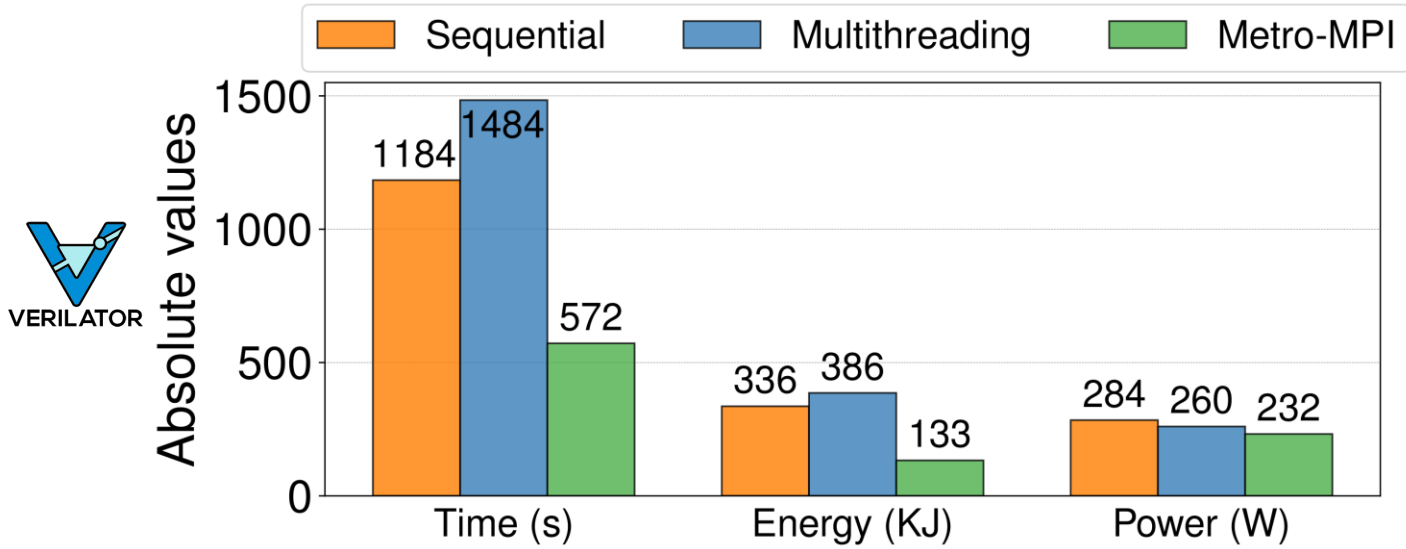


Night Regression Time & Energy results

- **We fix the amount of work (32 simulations of a 8x4 NoC Simulation) and compare three options**

- Metro-MPI outperforms in Time by 2.06x and 2.59x

- Metro-MPI outperforms in Energy by 2.52x and 2.90x

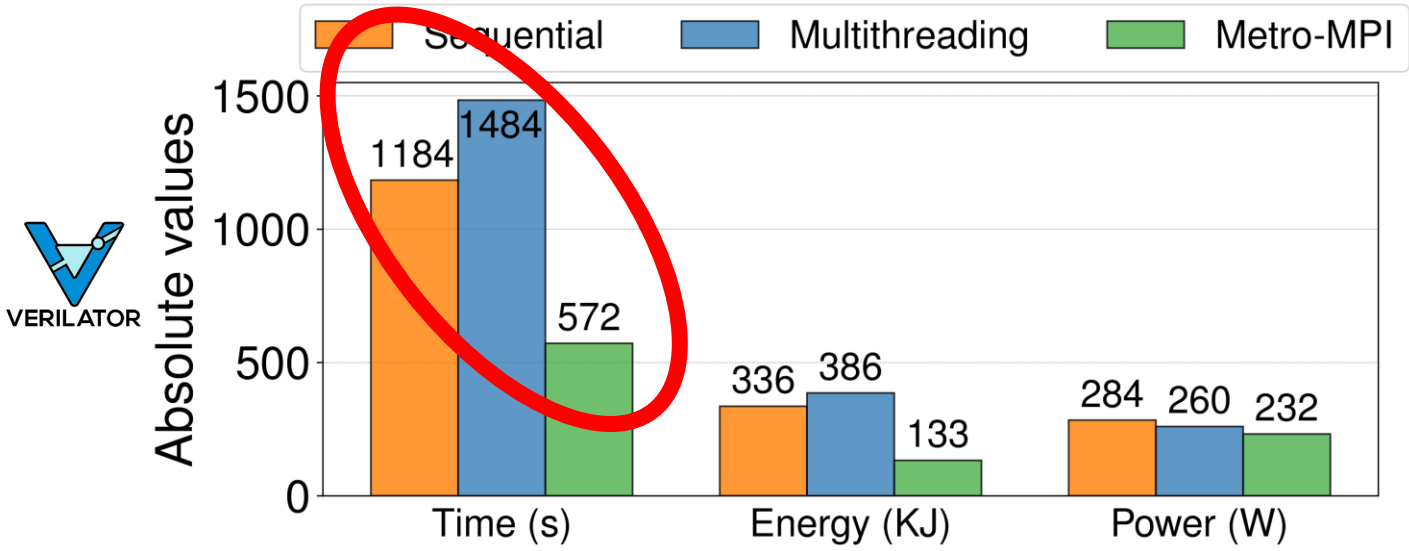


Night Regression Time & Energy results

- We fix the amount of work (32 simulations of a 8x4 NoC Simulation) and compare three options

- **Metro-MPI outperforms in Time by 2.06x and 2.59x**

- Metro-MPI outperforms in Energy by 2.52x and 2.90x

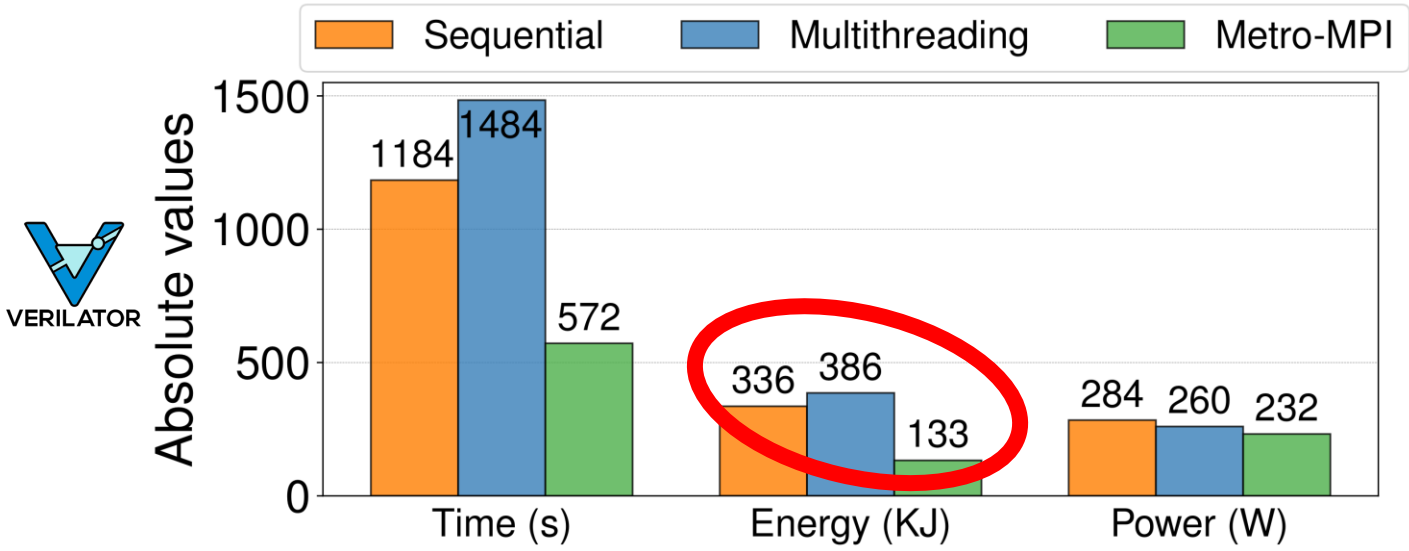


Night Regression Time & Energy results

- We fix the amount of work (32 simulations of a 8x4 NoC Simulation) and compare three options

- Metro-MPI outperforms in Time by 2.06x and 2.59x

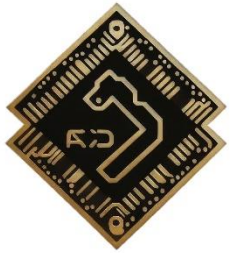
- **Metro-MPI outperforms in Energy by 2.52x and 2.90x**



Metro-MPI at BSC

- **Many projects at BSC working towards a heterogenous multi-core chip**

MEEP
MareNostrum Experimental Exascale Platform



- Metro-MPI has been successfully used in these projects to perform a design space exploration of a 64 cores NoC

- Personally, I am using metro-MPI in my research, executing simulations that takes several days and Millions of cycles

Metro-MPI at BSC

- Many projects at BSC working towards a heterogenous multi-core chip

- **Metro-MPI has been successfully used in these projects to perform a design space exploration of a 64 cores NoC**

- Personally, I am using metro-MPI in my research, executing simulations that takes several days and Millions of cycles

Metro-MPI at BSC

- Many projects at BSC working towards a heterogenous multi-core chip

MEEP
MareNostrum Experimental Exascale Platform

European Processor Initiative

- Metro-MPI has been successfully used in these projects to perform a design space exploration of a 64 cores NoC

- **Personally, I am using metro-MPI in my research, executing simulations that takes several days and Millions of cycles**

Future Work

- **Metro-MPI is open-source and we are considering trying other multi-core platforms**
- **We would like to perform the same experiments done with Verilator with Commercial Simulator such as Synopsis VCS → license problem**
- Metro-MPI could be automatically applied in some designs that show repeated hardware blocks e.g. `*a la Verilator multi-threaded*`
- Try newer versions of Verilator v5 that supports the Verilog timing model

Future Work

- Metro-MPI is open-source and we are considering trying other multi-core platforms

- We would like to perform the same experiments done with Verilator with Commercial Simulator such as Synopsis VCS → license problem

- **Metro-MPI could be automatically applied in some designs that show repeated hardware blocks e.g. `*a la Verilator multi-threaded*`**

- **Try newer versions of Verilator v5 that supports the Verilog timing model**

Conclusions

- General methodology that can be applied to multiple designs
  - Exploiting natural boundaries (“latency-insensitive” interfaces)

Conclusions

- General methodology that can be applied to multiple designs
  - Exploiting natural boundaries (“latency-insensitive” interfaces)

- **Overcomes problems found in RTL Simulators:**
  - **Binary size, ITLB and ICache MPKI**

Conclusions

- General methodology that can be applied to multiple designs
  - Exploiting natural boundaries (“latency-insensitive” interfaces)

- Overcomes problems found in RTL Simulators:
  - Binary size, ITLB and ICache MPKI

- **Exceptional scaling:**
  - **Simulation scales up to 1024 tiles**
  - **In simulation throughput, reaching 2.7 MIPS on a 1024 tile chip**
  - **In simulation time speedup, up to 136x with respect to sequential**

Acknowledgments

arm

Google Summer of Code

DRAC

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

UC SANTA BARBARA

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

HM

Metro-MPI

Open Source at: github.com/metro-mpi

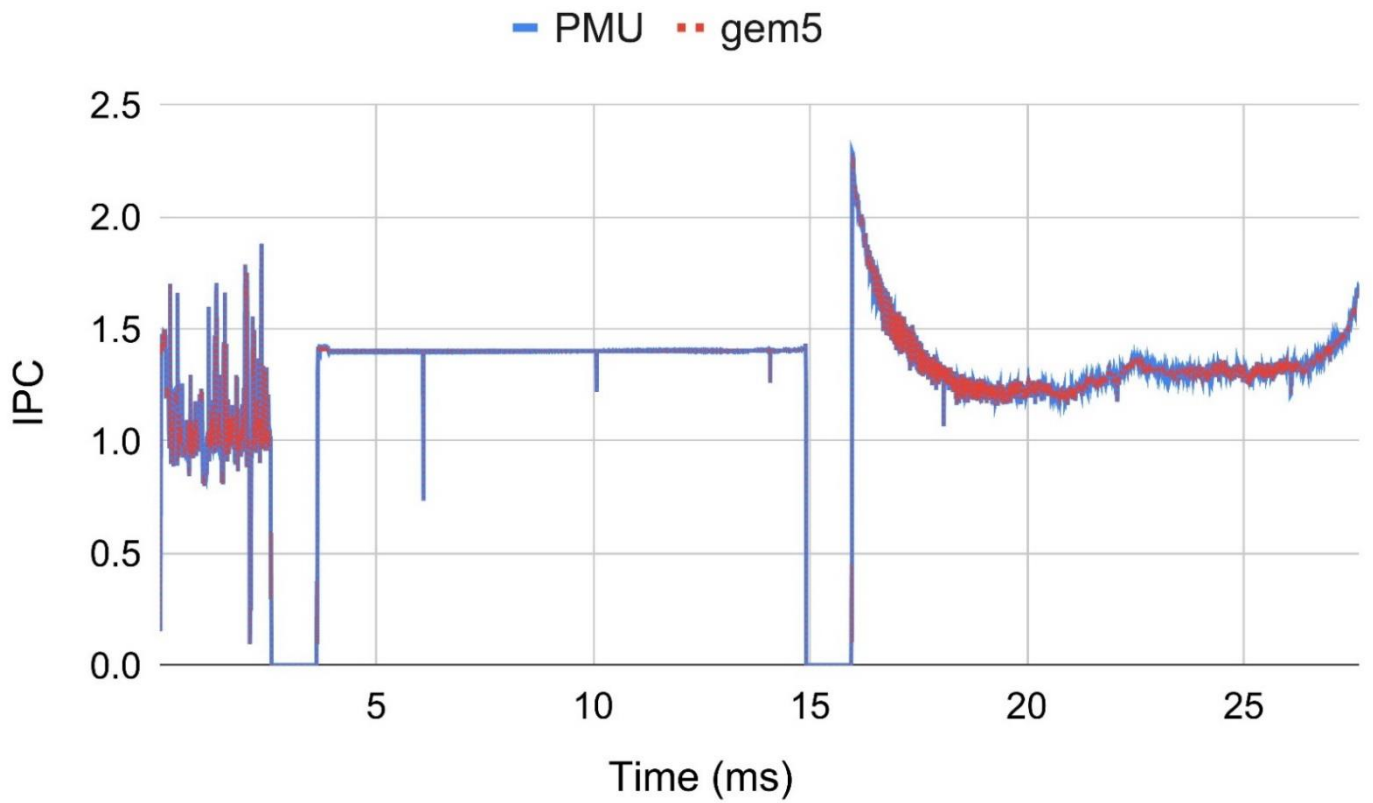
OSCAR 2024

guillem.lopez@bsc.es

Unió Europea
Fons Social Europeu
L'FSE inverteix en el teu futur

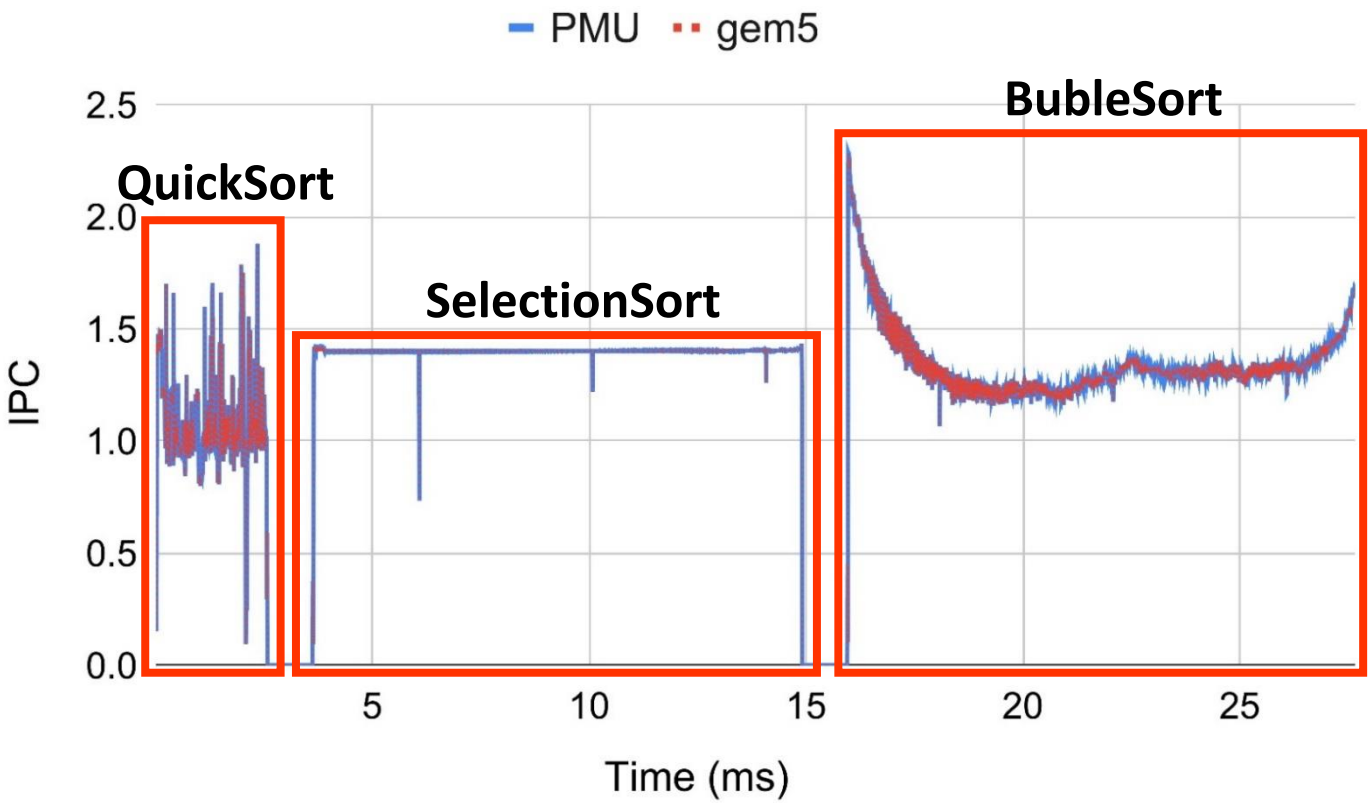
Evaluation PMU: IPC

- **Comparison stats gem5 vs PMU:**
  - **Every 1k cycles, compare IPC stats (y-axis)**
  - **X-axis Time in ms**
- Executed three sorting algorithms
  - 3k elements for QuickSort
  - 30k elements rest
- Separated with a sleep call of 1 ms



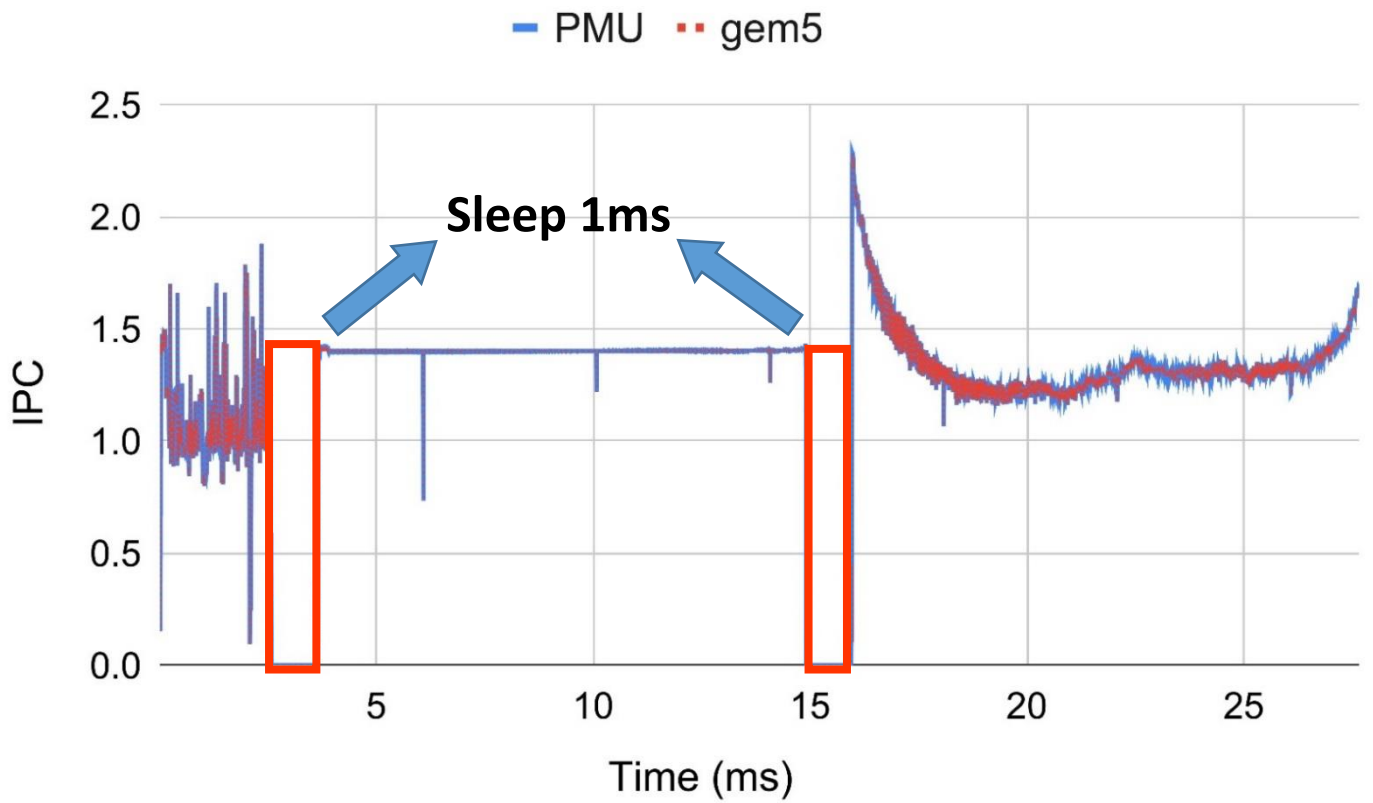
Evaluation PMU: IPC

- **Comparison stats gem5 vs PMU:**
  - **Every 1k cycles, compare IPC stats (y-axis)**
  - **X-axis Time in ms**
- Executed three sorting algorithms
  - 3k elements for QuickSort
  - 30k elements rest
- Separated with a sleep call of 1 ms



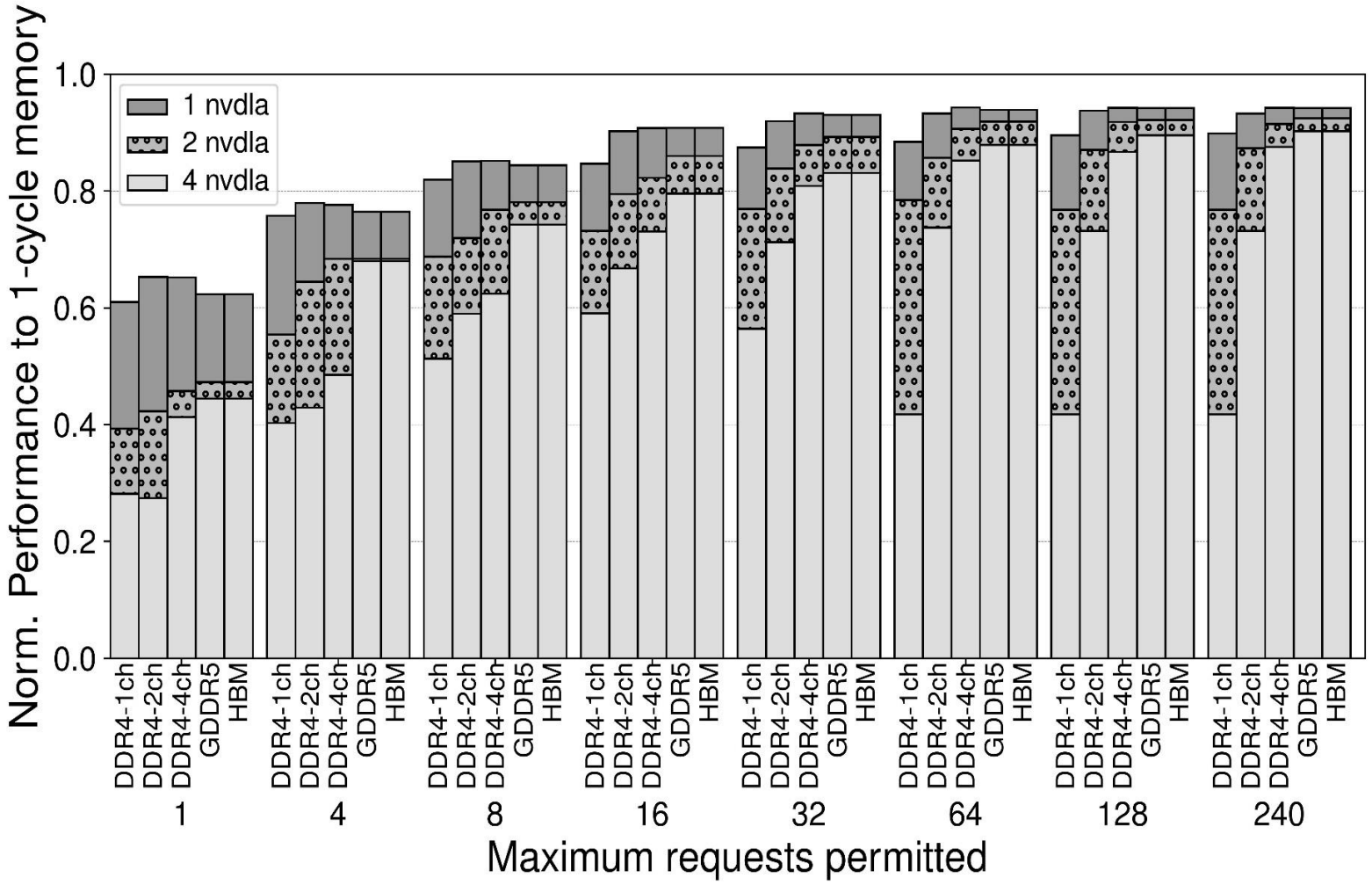
Evaluation PMU: IPC

- Comparison stats gem5 vs PMU:
  - Every 1k cycles, compare IPC stats (y-axis)
  - X-axis Time in ms
- Executed three sorting algorithms
  - 3k elements for QuickSort
  - 30k elements rest
- **Separated with a sleep call of 1 ms**



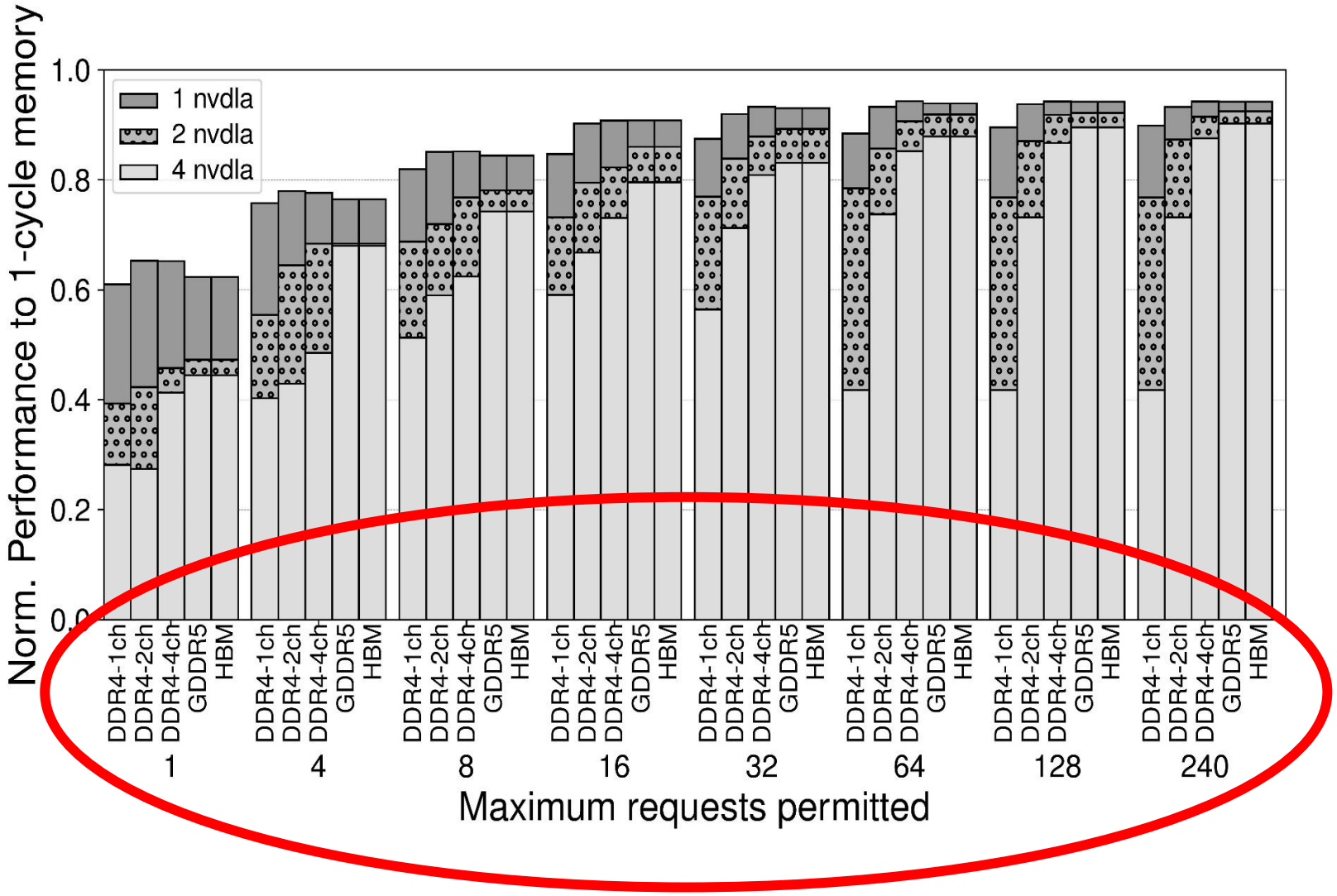
Evaluation NVDLA: GoogleNet

- **Evaluation of NVDLA performed by executing traces of real applications provided by NVIDIA**
- Parameters for the design space exploration (x-axis)
- Performance (y-axis) is normalized to an ideal 1 cycle memory latency



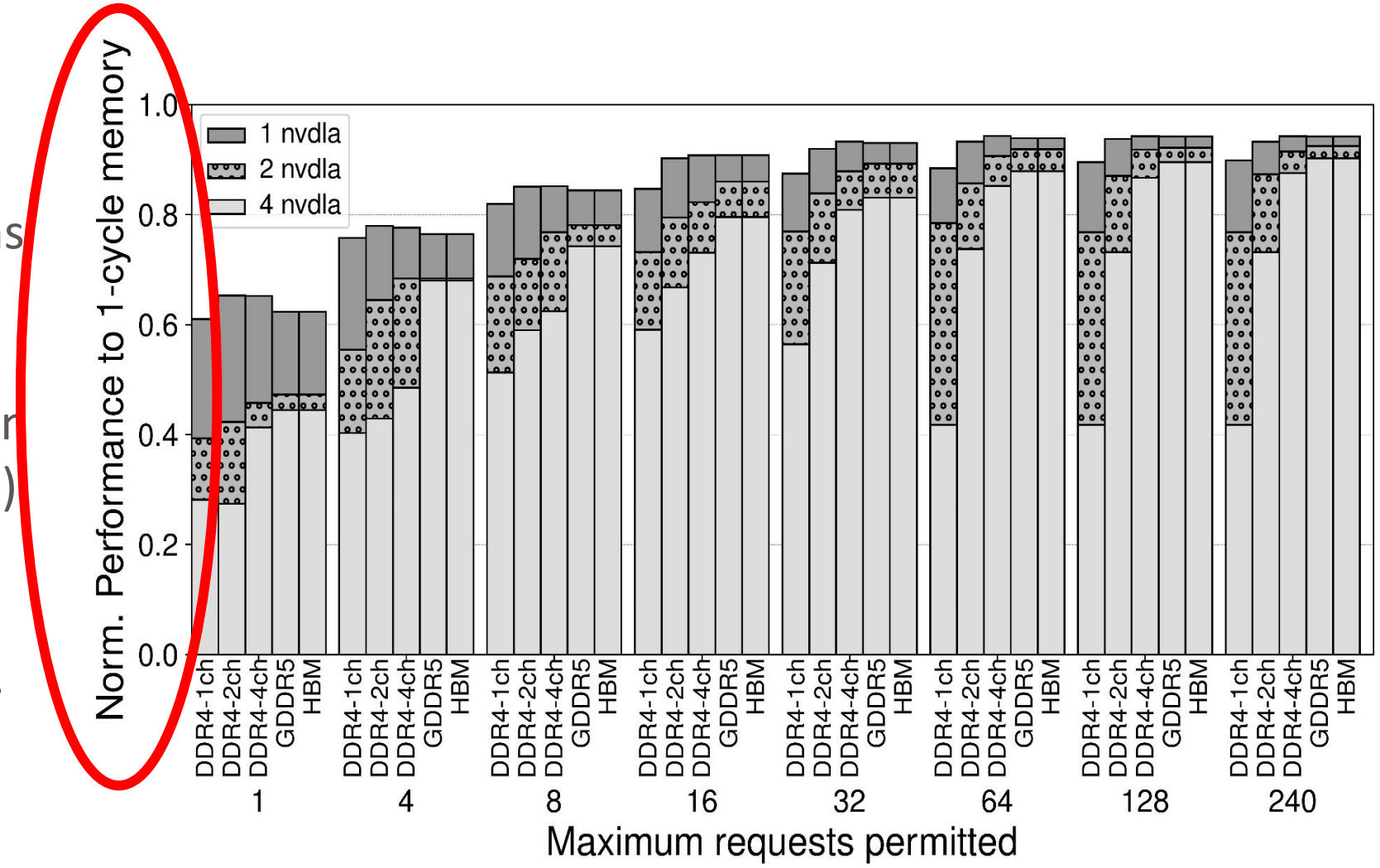
Evaluation NVDLA: GoogleNet

- Evaluation of NVDLA performed by executing traces of real applications provided by NVIDIA
- **Parameters for the design space exploration (x-axis)**
- Performance (y-axis) is normalized to an ideal 1 cycle memory latency



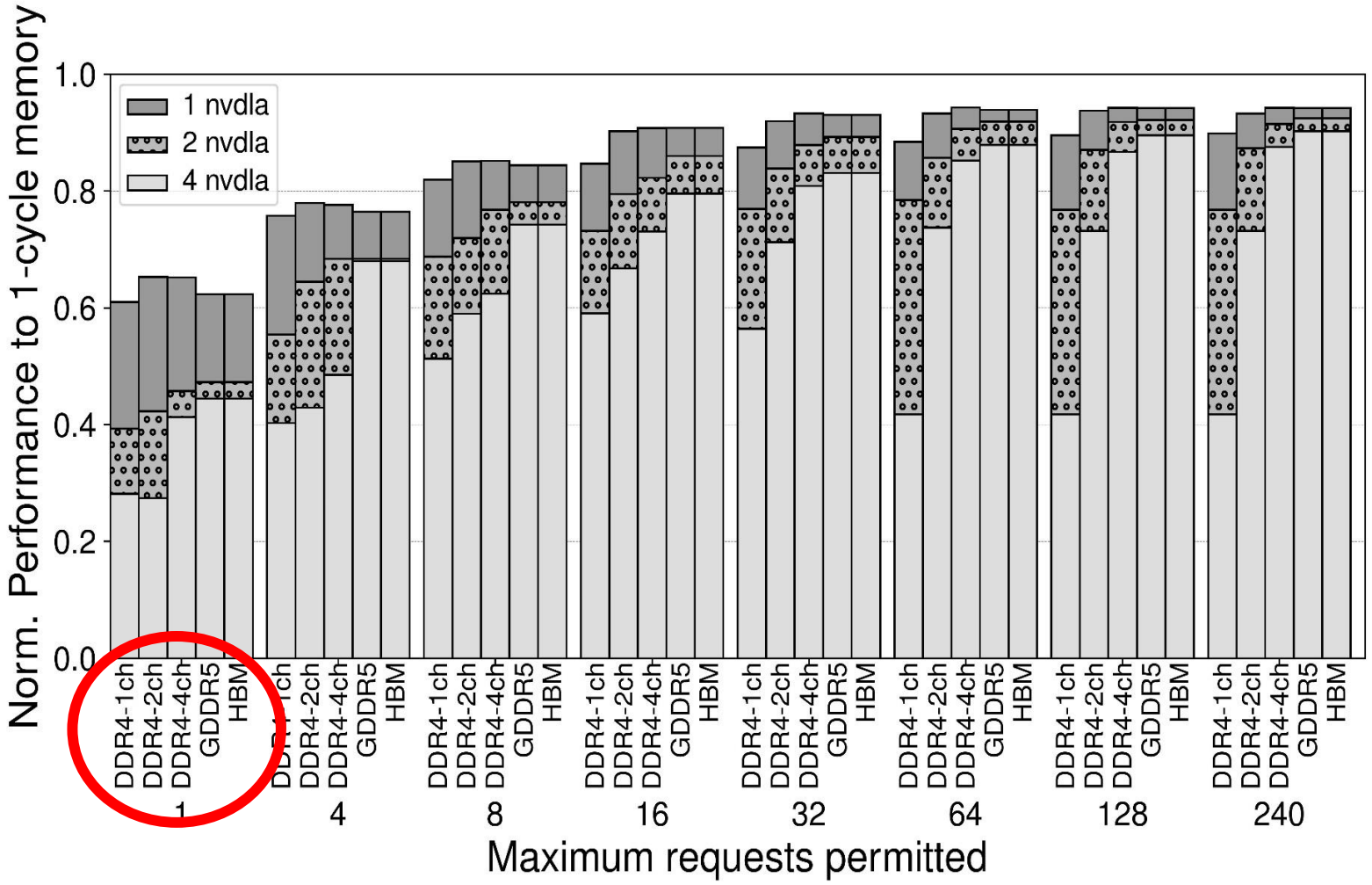
Evaluation NVDLA: GoogleNet

- Evaluation of NVDLA performed by executing traces of real applications provided by NVIDIA
- Parameters for the design space exploration (x-axis)
- **Performance (y-axis) is normalized to an ideal 1 cycle memory latency**



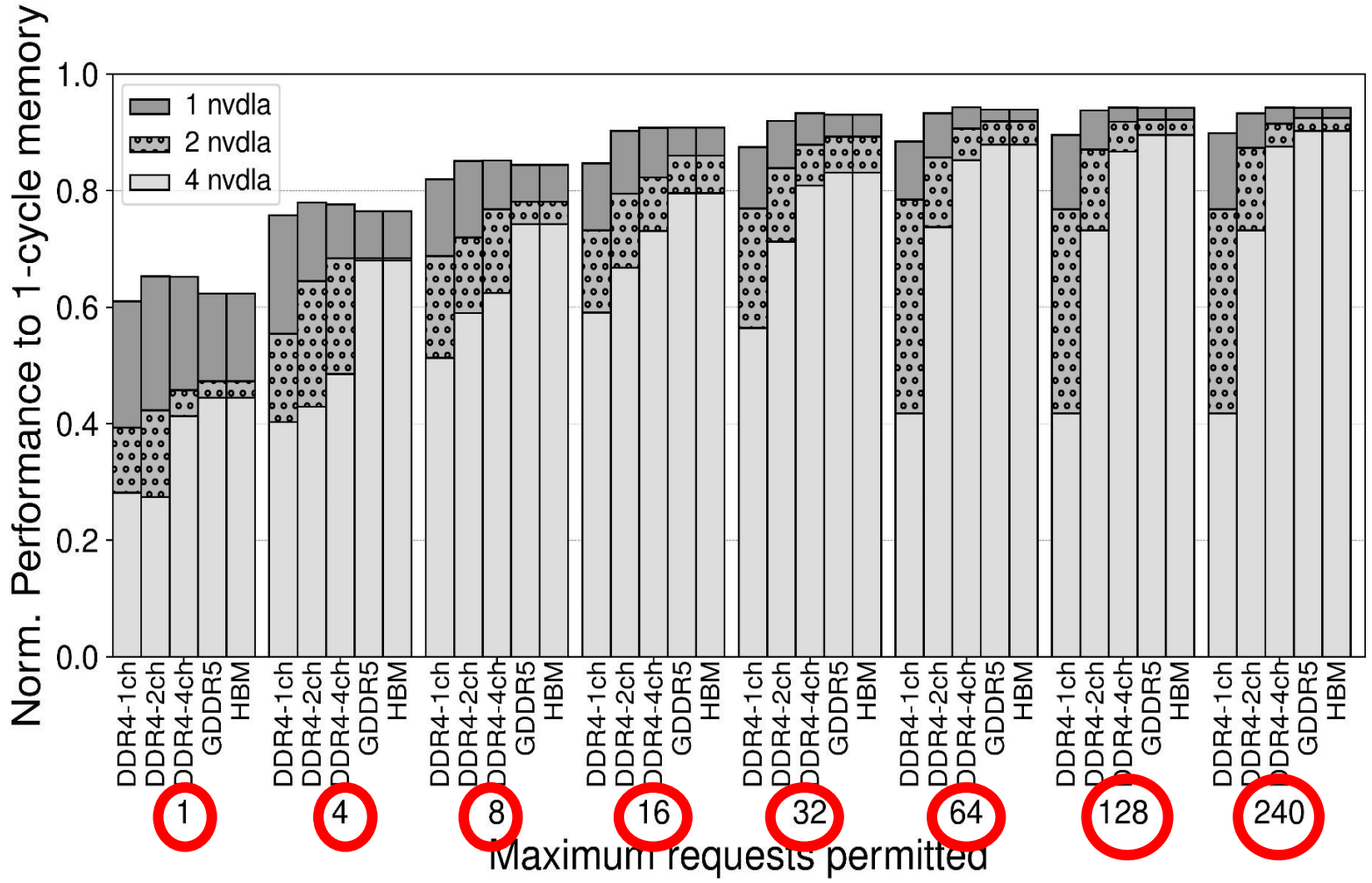
Evaluation NVDLA: GoogleNet

- **Using several memory configurations**
- Different number of maximum requests from NVDLA to main memory
- Different number of nvdla in the system: 1, 2 and 4 nvdla's configurations



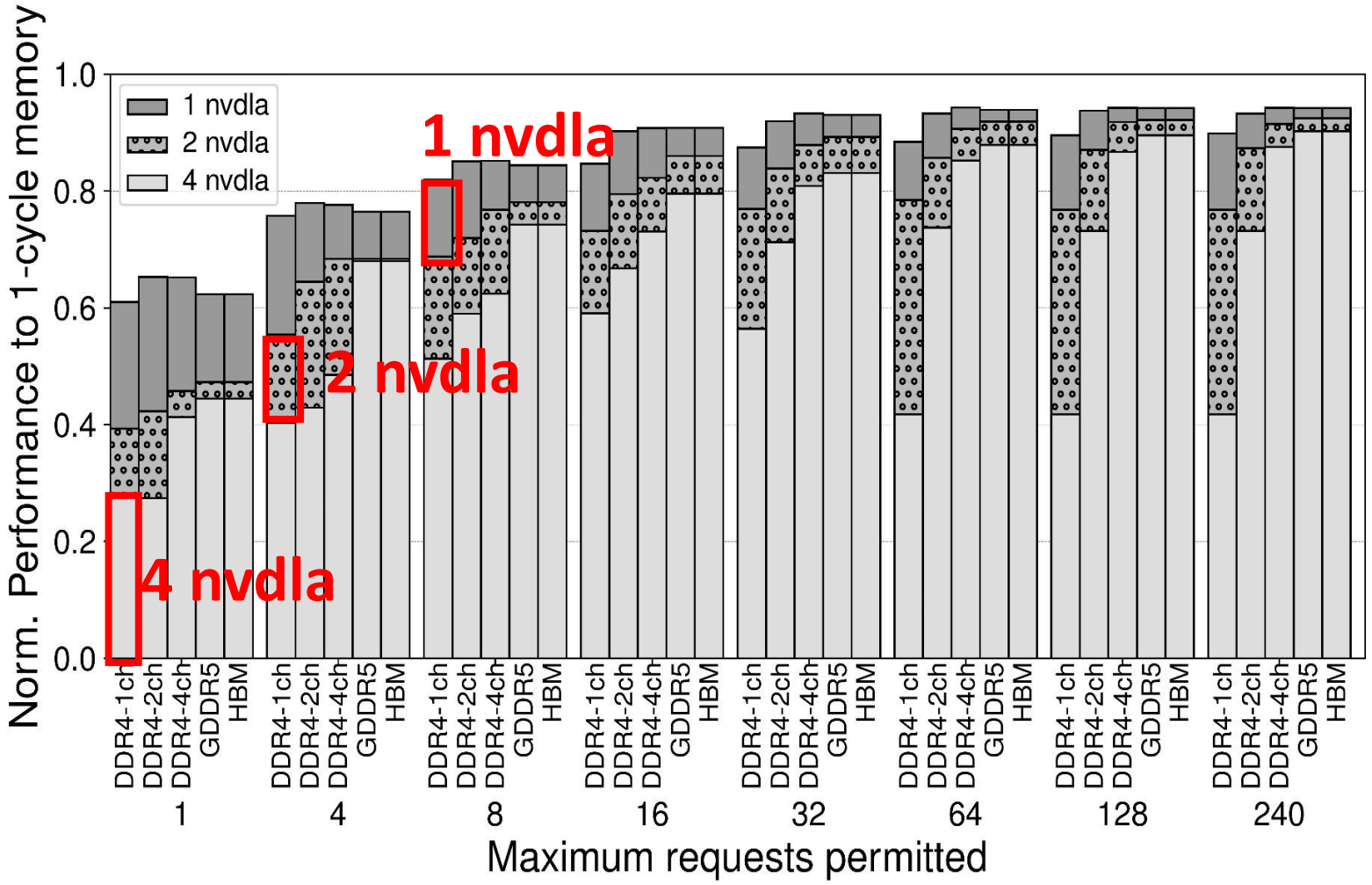
Evaluation NVDLA: GoogleNet

- Using several memory configurations
- **Different number of maximum requests from NVDLA to main memory**
- Different number of nvdla in the system: 1, 2 and 4 nvdla's configurations



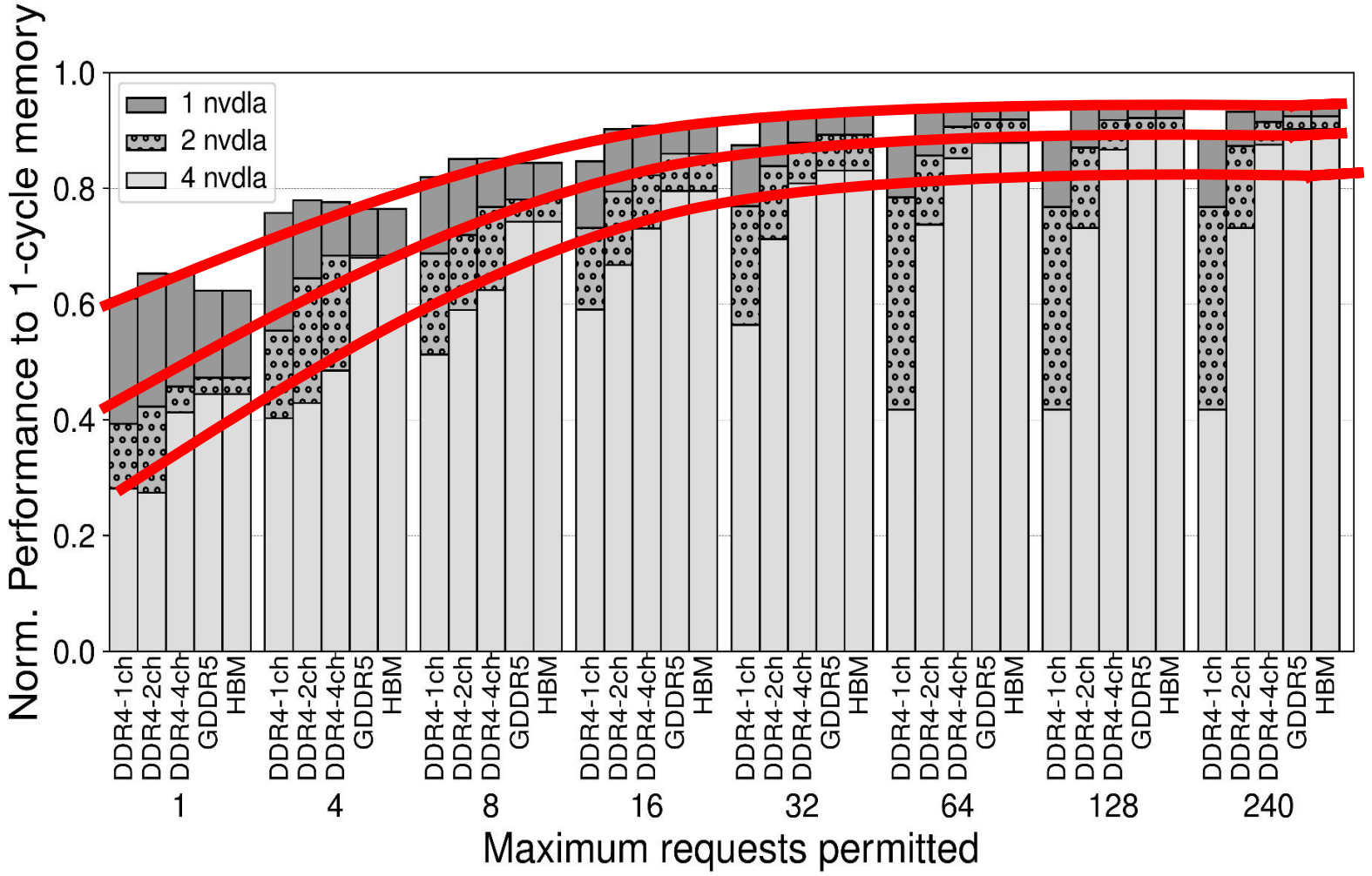
Evaluation NVDLA: GoogleNet

- Using several memory configurations
- Different number of maximum requests from NVDLA to main memory
- **Different number of nvdla in the system: 1, 2 and 4 nvdla's configurations**



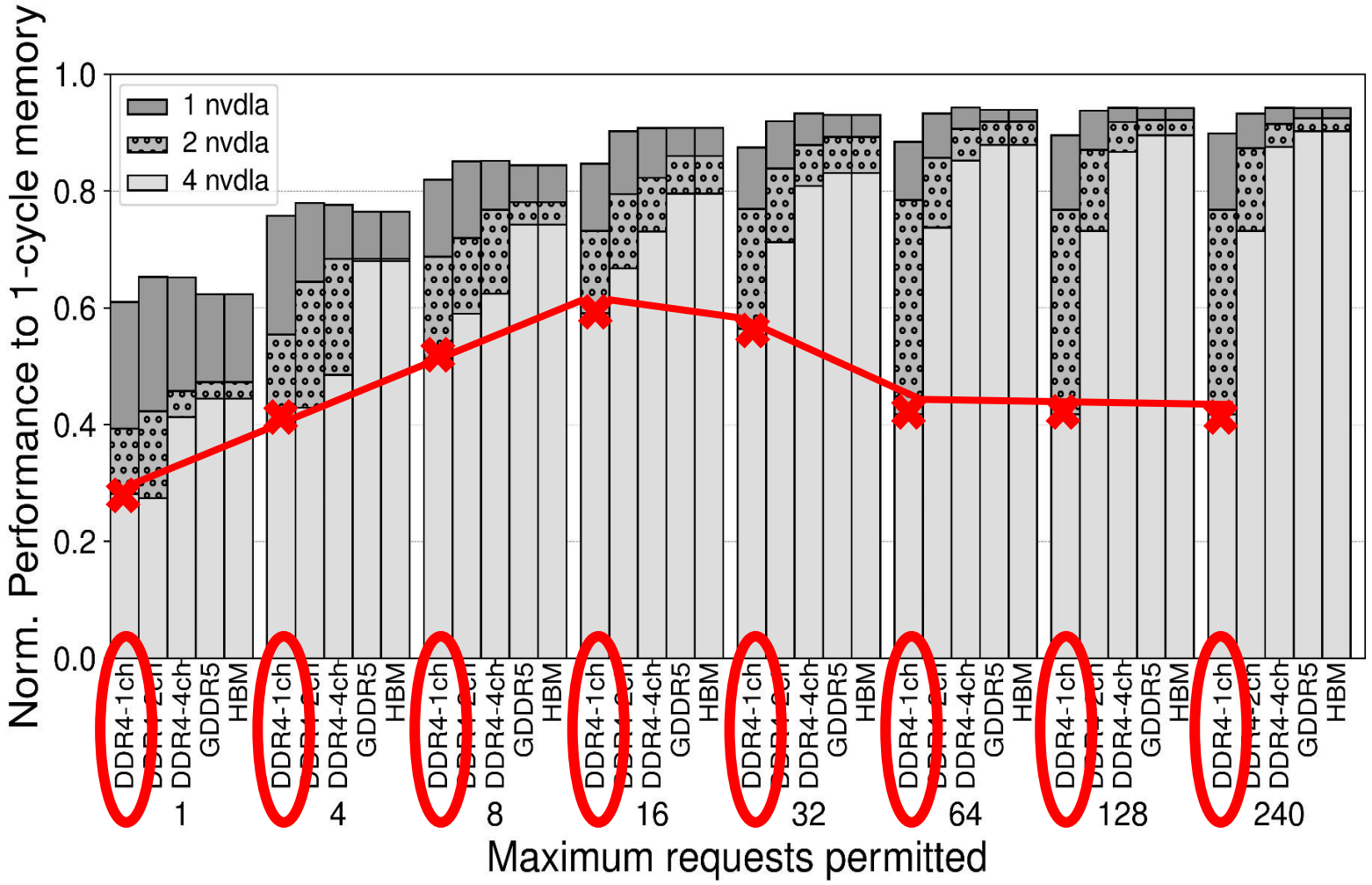
Evaluation NVDLA: GoogleNet

- **Maximum number of requests affects dramatically**
- Some memory configs cannot deliver enough bw for 2 and 4 nvdlas
- We recommend HBM or GDDR5 when more than 2 NVDLAs are in the system

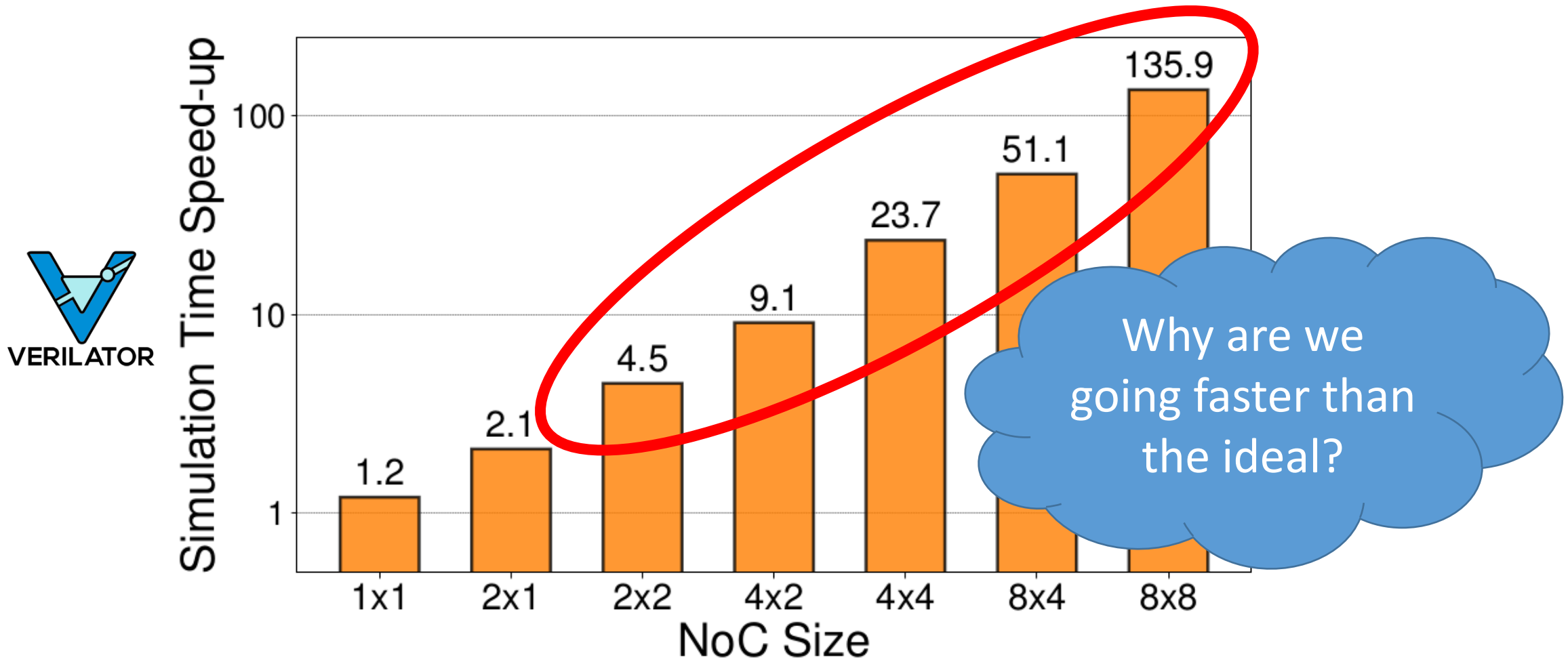


Evaluation NVDLA: GoogleNet

- Maximum number of requests affects dramatically
- **Some memory configs cannot deliver enough bw for 2 and 4 nvdlas**
- We recommend HBM or GDDR5 when more than 2 NVDLAs are in the system

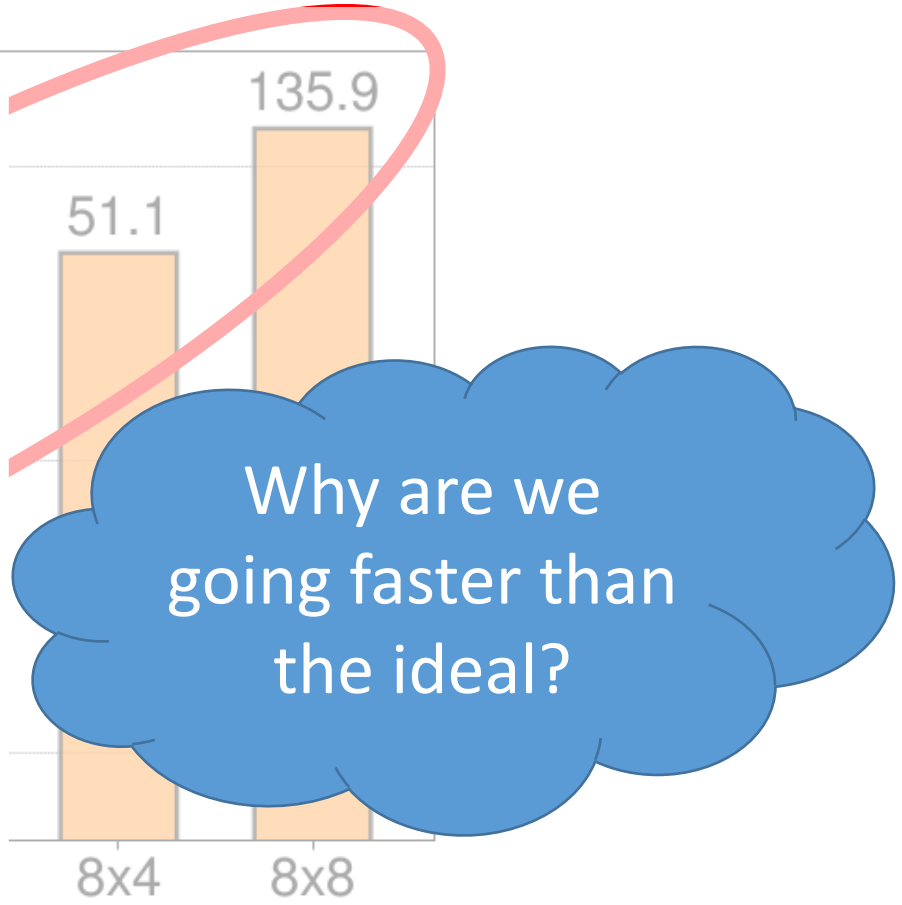


Metro-MPI Simulation Time Speedup



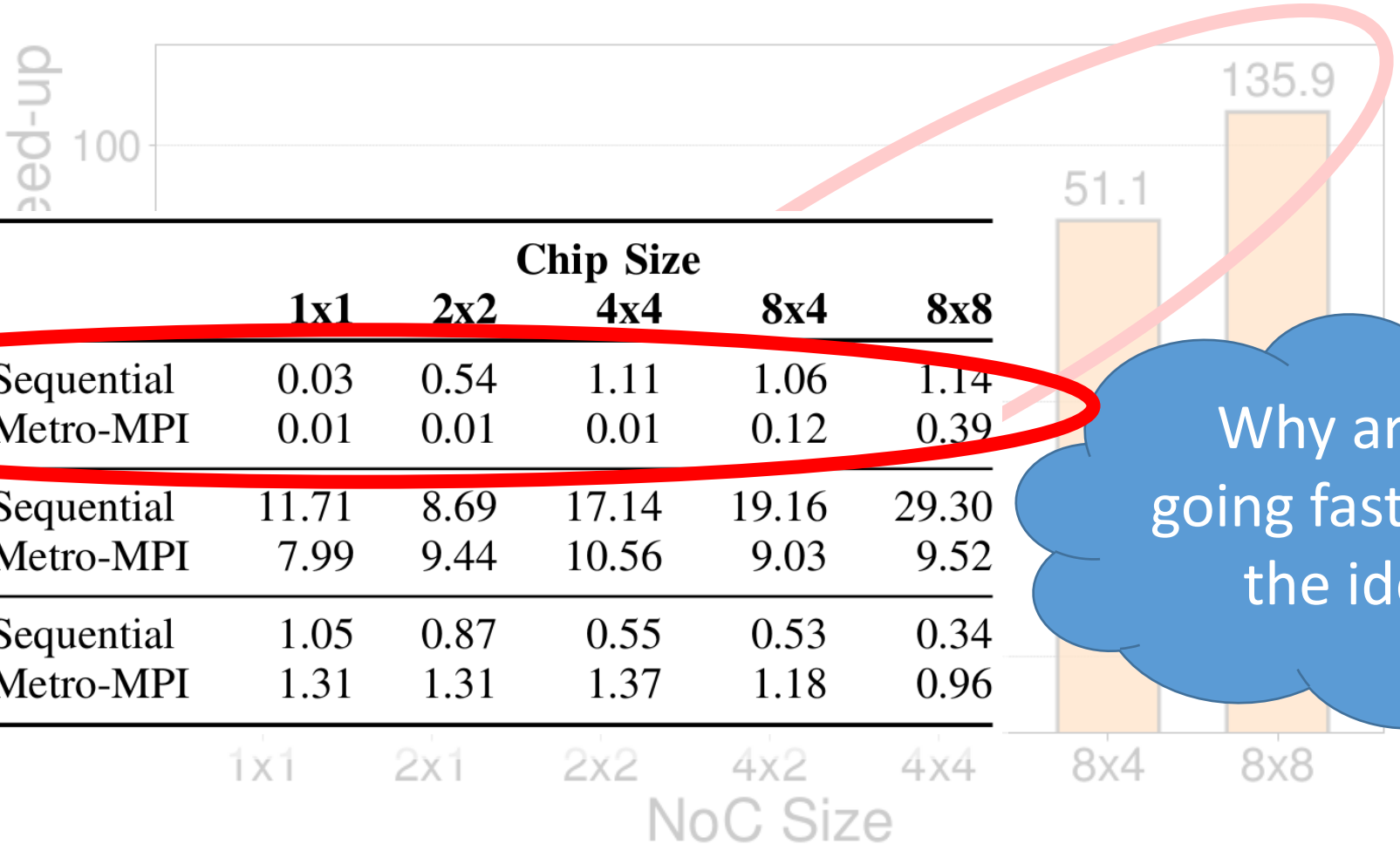
Metro-MPI Simulation Time Speedup

- **We did a Profiling Analysis**
  - **ITLB** Misses per 1k instr (**MPKI**)
  - **Icache** Misses per 1k instr (**MPKI**)
  - **IPC**



Metro-MPI Simulation Time Speedup

		Chip Size 1x1	2x2	4x4	8x4	8x8
ITLB MPKI	Sequential	0.03	0.54	1.11	1.06	1.14
	Metro-MPI	0.01	0.01	0.01	0.12	0.39
ICache MPKI	Sequential	11.71	8.69	17.14	19.16	29.30
	Metro-MPI	7.99	9.44	10.56	9.03	9.52
IPC	Sequential	1.05	0.87	0.55	0.53	0.34
	Metro-MPI	1.31	1.31	1.37	1.18	0.96



Why are we going faster than the ideal?

Metro-MPI Simulation Time Speedup

		Chip Size 1x1	2x2	4x4	8x4	8x8
ITLB MPKI	Sequential	0.03	0.54	1.11	1.06	1.14
	Metro-MPI	0.01	0.01	0.01	0.12	0.39
ICache MPKI	Sequential	11.71	8.69	17.14	19.16	29.30
	Metro-MPI	7.99	9.44	10.56	9.03	9.52
IPC	Sequential	1.05	0.87	0.55	0.53	0.34
	Metro-MPI	1.31	1.31	1.37	1.18	0.96

Why are we going faster than the ideal?

Metro-MPI Big-3 Performance

- In a system with 8 cores
- Big 3 Simulator scales almost linearly
- Big 3 is already using threads in the default (non metro-MPI version)

TABLE III: Metro-MPI scaling with a commercial simulator.

Speedups	1x1	2x1	2x2	4x2	4x4	8x4
Simulation time	0.93	1.54	3.20	6.17	8.44	7.81
CPS	0.91	1.26	2.73	5.15	7.08	6.75
IPS	1.41	1.36	2.82	5.36	7.35	6.90

Metro-MPI Big-3 Performance

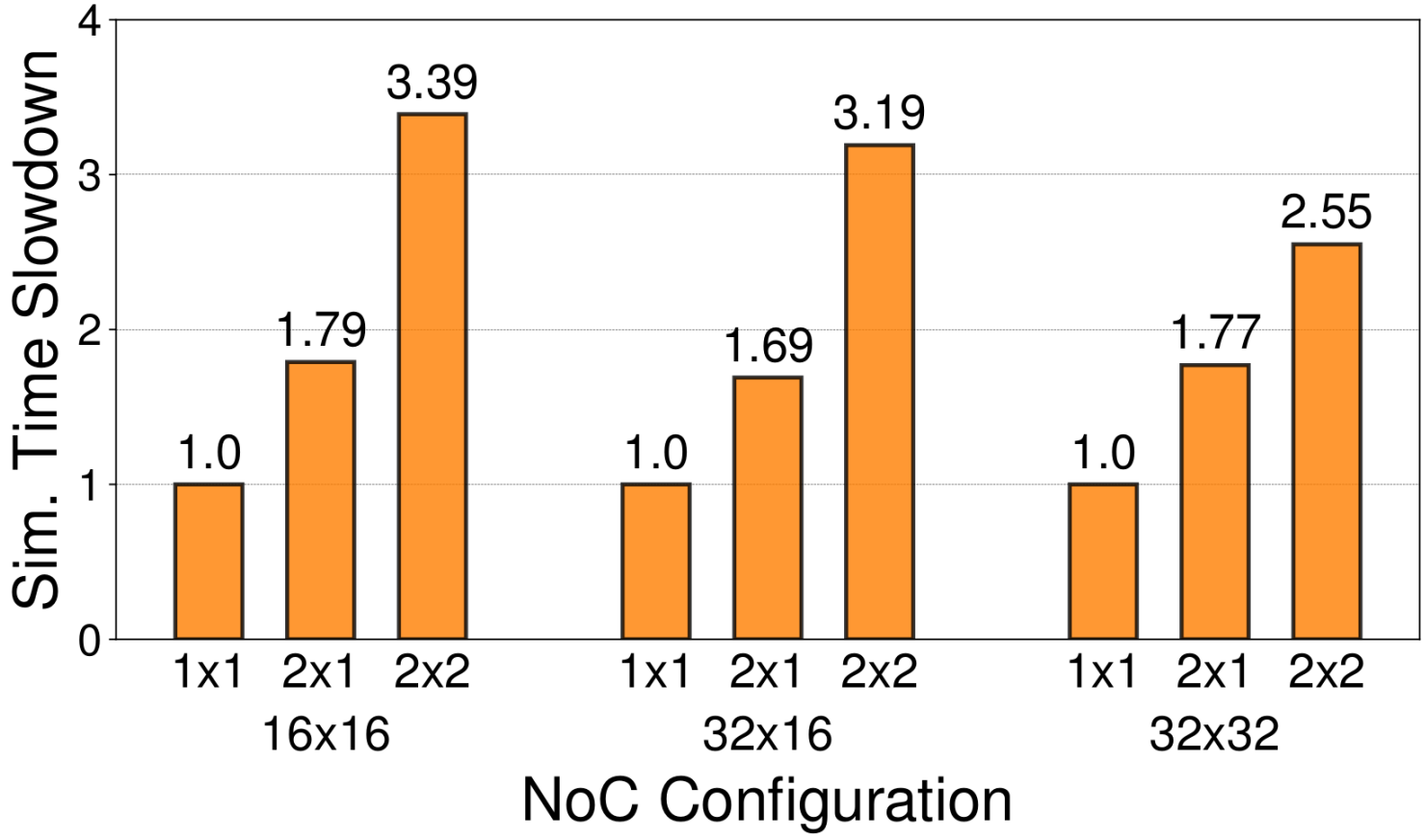
- In a system with 8 cores
- Big 3 Simulator scales almost linearly
- Big 3 is already using threads in the default (non metro-MPI version)

TABLE III: Metro-MPI scaling with a commercial simulator.

Speedups	1x1	2x1	2x2	4x2	4x4	8x4
Simulation time	0.93	1.54	3.20	6.17	8.44	7.81
CPS	0.91	1.26	2.73	5.15	7.08	6.75
IPS	1.41	1.36	2.82	5.36	7.35	6.90

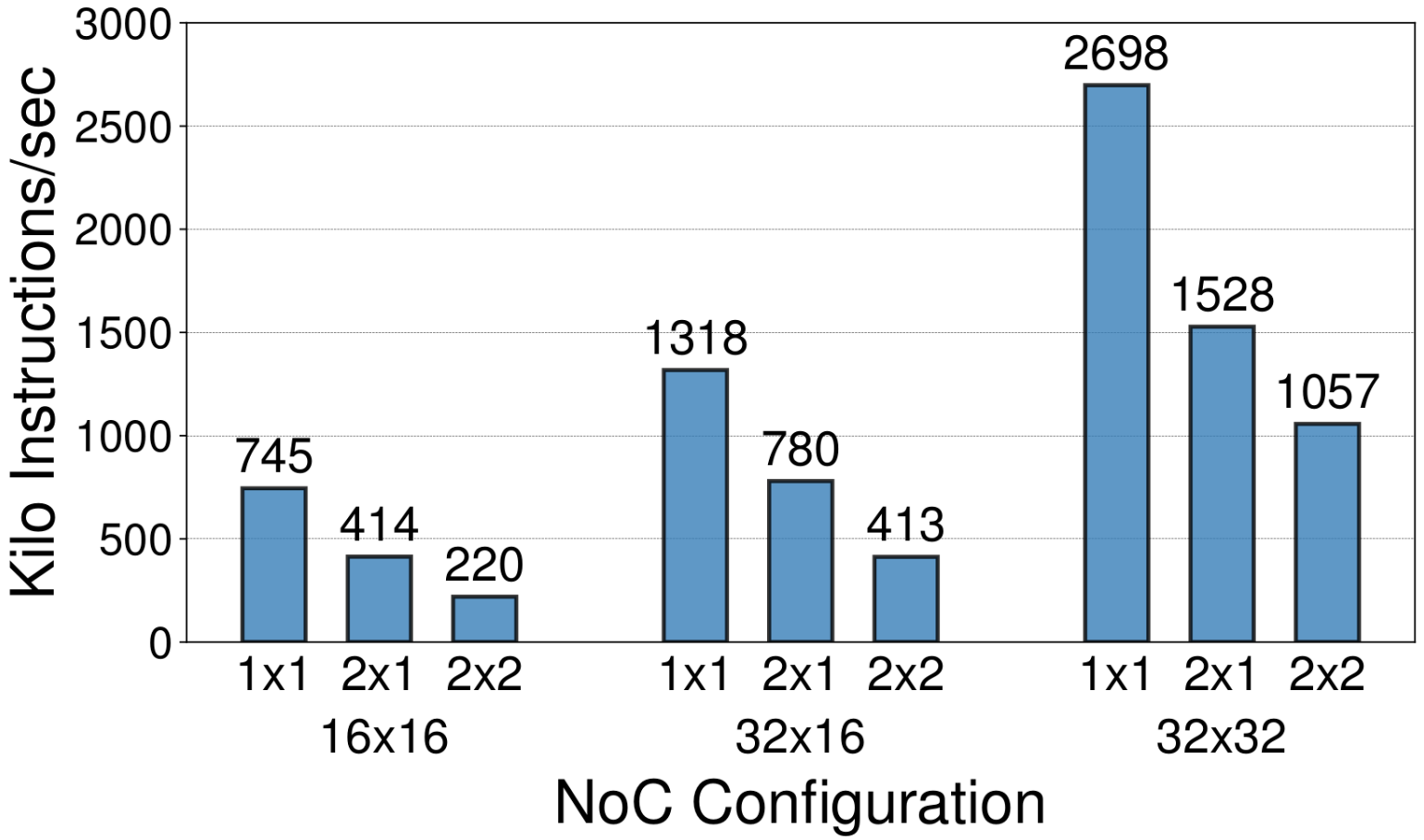
Multi Tile Granule Performance

- **Slow-down in simulation time**
- Consequently, KIPS also slow downs
- But, work per core increases, hence to run regressions is better ☺



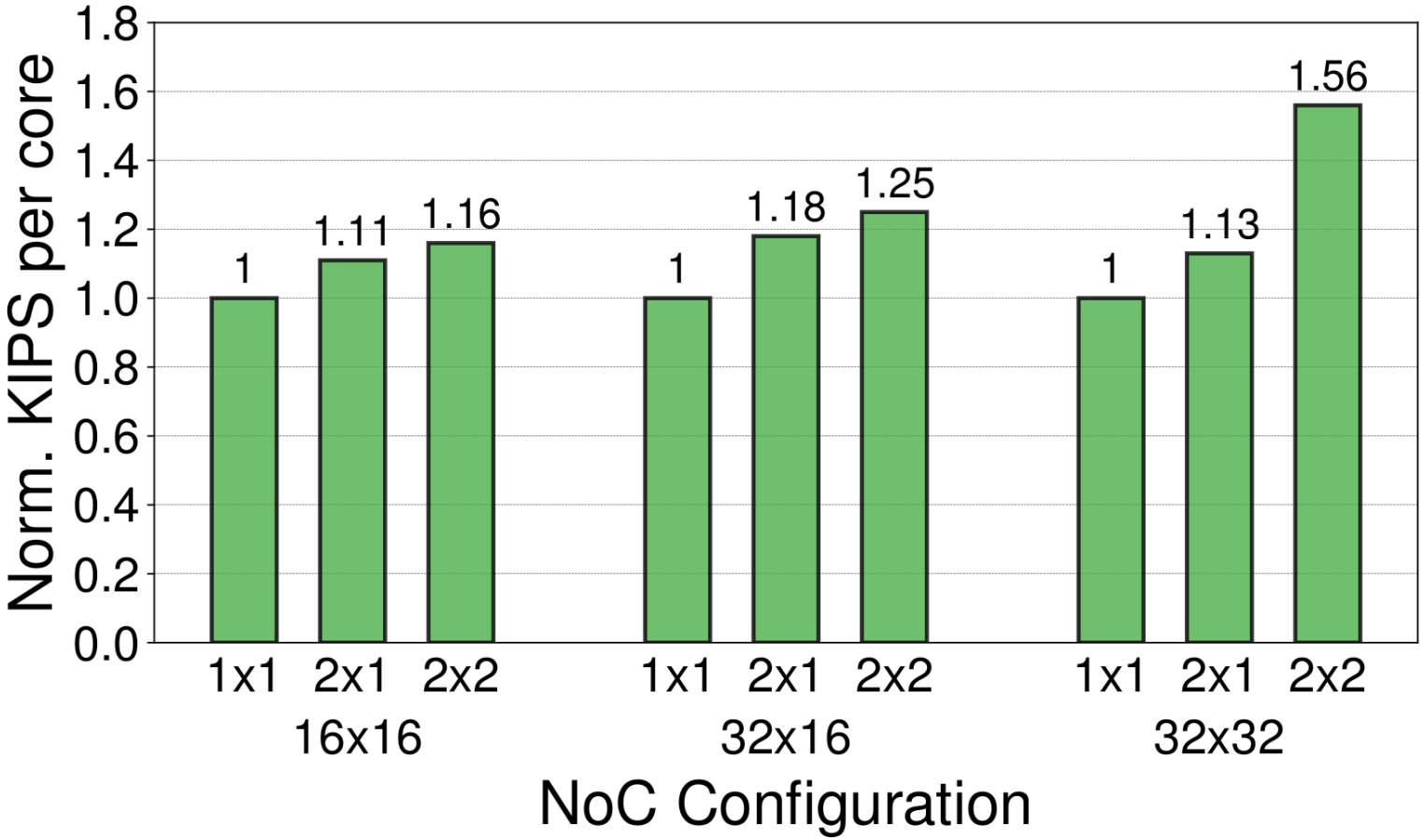
Multi Tile Granule Performance

- Slow-down in simulation time
- **Consequently, KIPS also slow down**
- But, work per core increases, hence to run regressions is better ☺



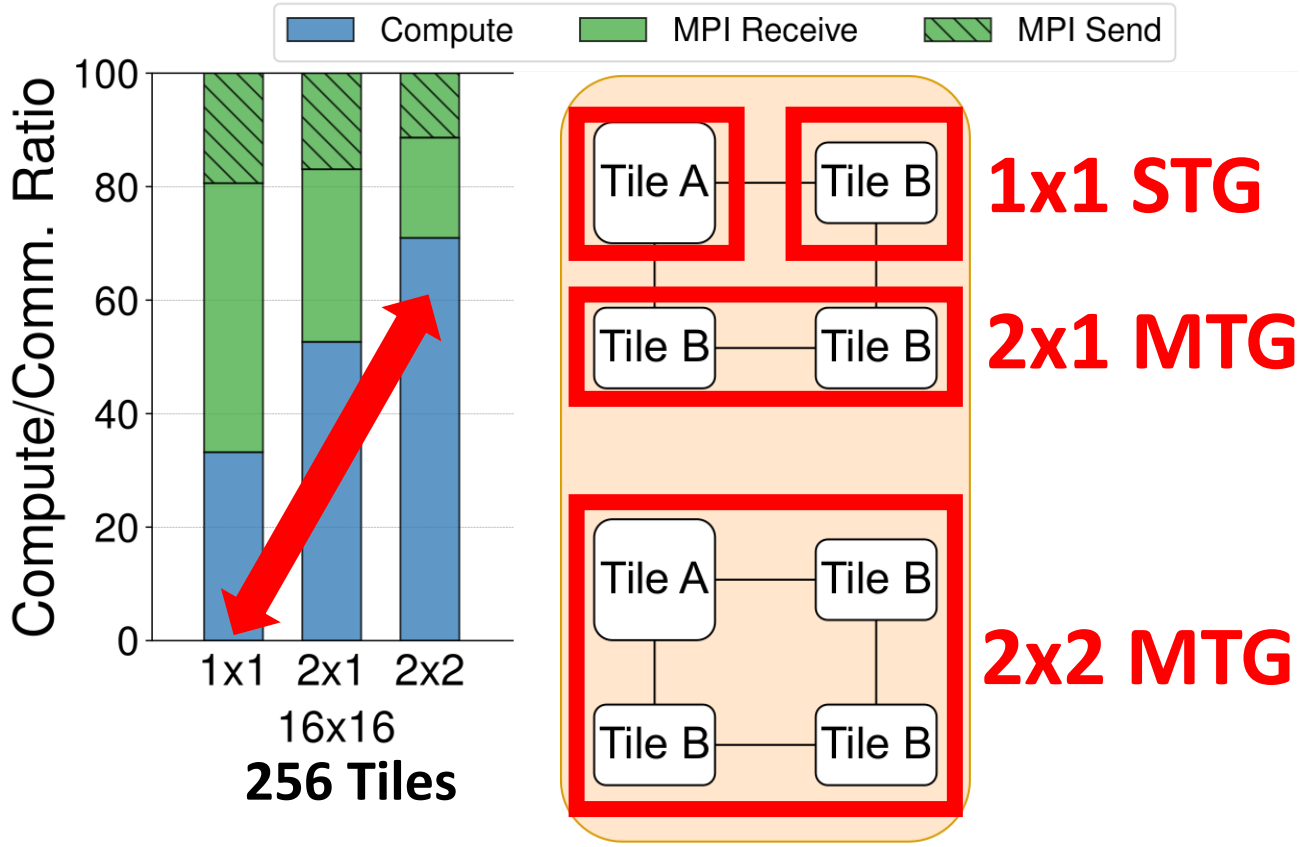
Multi Tile Granule Performance

- Slow-down in simulation time
- Consequently, KIPS also slow downs
- **But, work per core increases, hence to run regressions is better ☺**



MPI Overhead (Multi Tile)

- **Multi-Tile granules (MTG) help to increase the compute ratio**



MPI Overhead (Multi Tile)

- Multi-Tile granules (MTG) help to increase the compute ratio

- **MPI Receive decreases as the number of MPI processes is reduced**