BSC Barcelona Supercomputing Center
Centro Nacional de Supercomputación

INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING

gem5+RTL: A Framework to Enable RTL Models Inside a Full-System Simulator

Author: Guillem López Paradís

Co-Authors: Miquel Moretó and Adrià Armejach

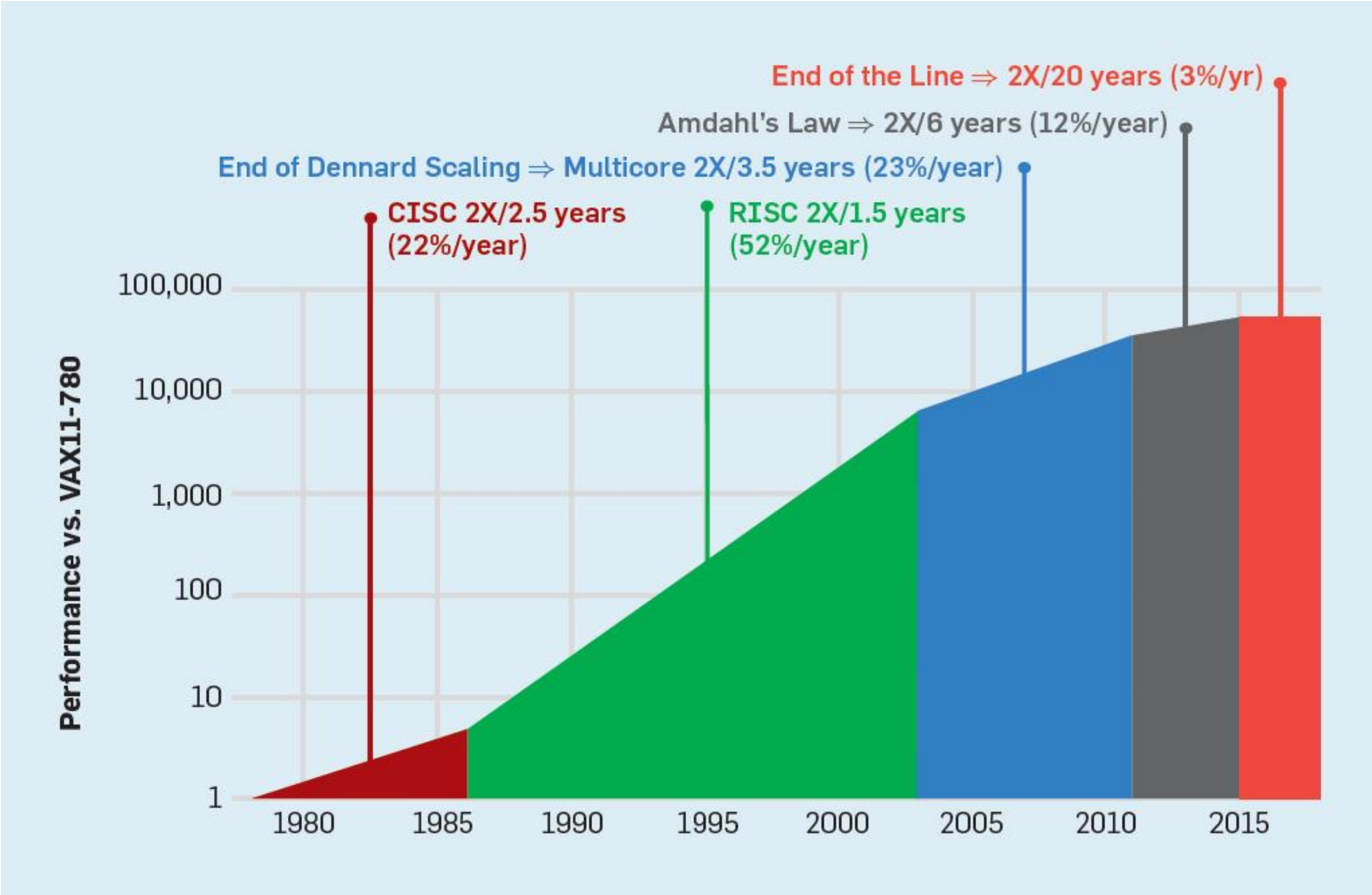
10/August/2021

50th International Conference on Parallel Processing (ICPP) August 9-12, 2021 in Virtual Chicago, IL

Heritage of *Moore's law*

- **Y-axis → CPU performance in a logarithmic scale**

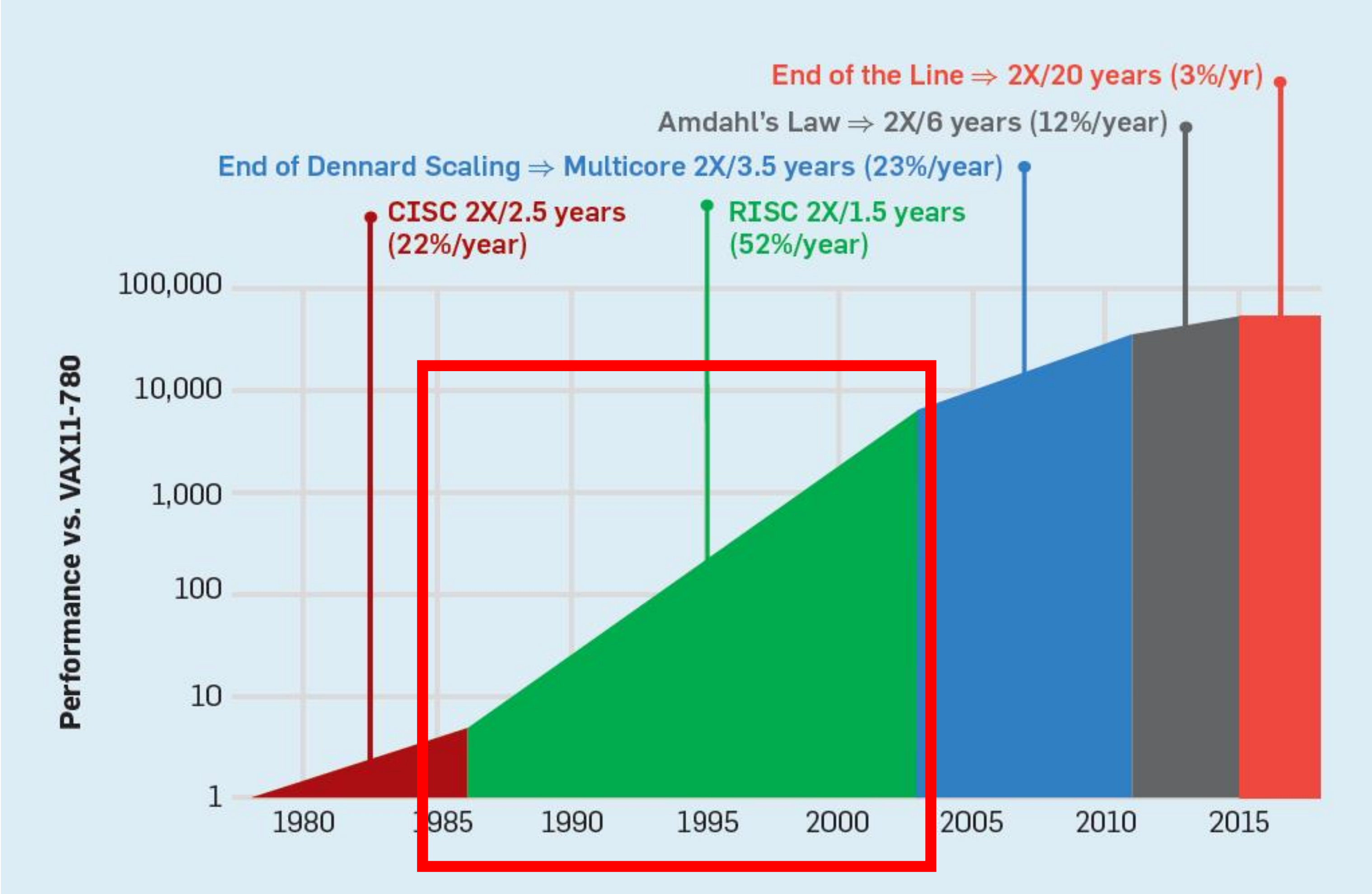
- **X-axis → Time in years**



Source: jj.github.io

Heritage of *Moore's law*

- **Moore's law and Dennard scaling ruled an exponential phase (green) and created a whole industry**
- These golden "*rules*" stop delivering the same speed-up in performance in early 2000 (blue)
- Last 20 years (blue, grey, red phase), we have added more hardware modules into the SoC
- Red phase curve is flat!

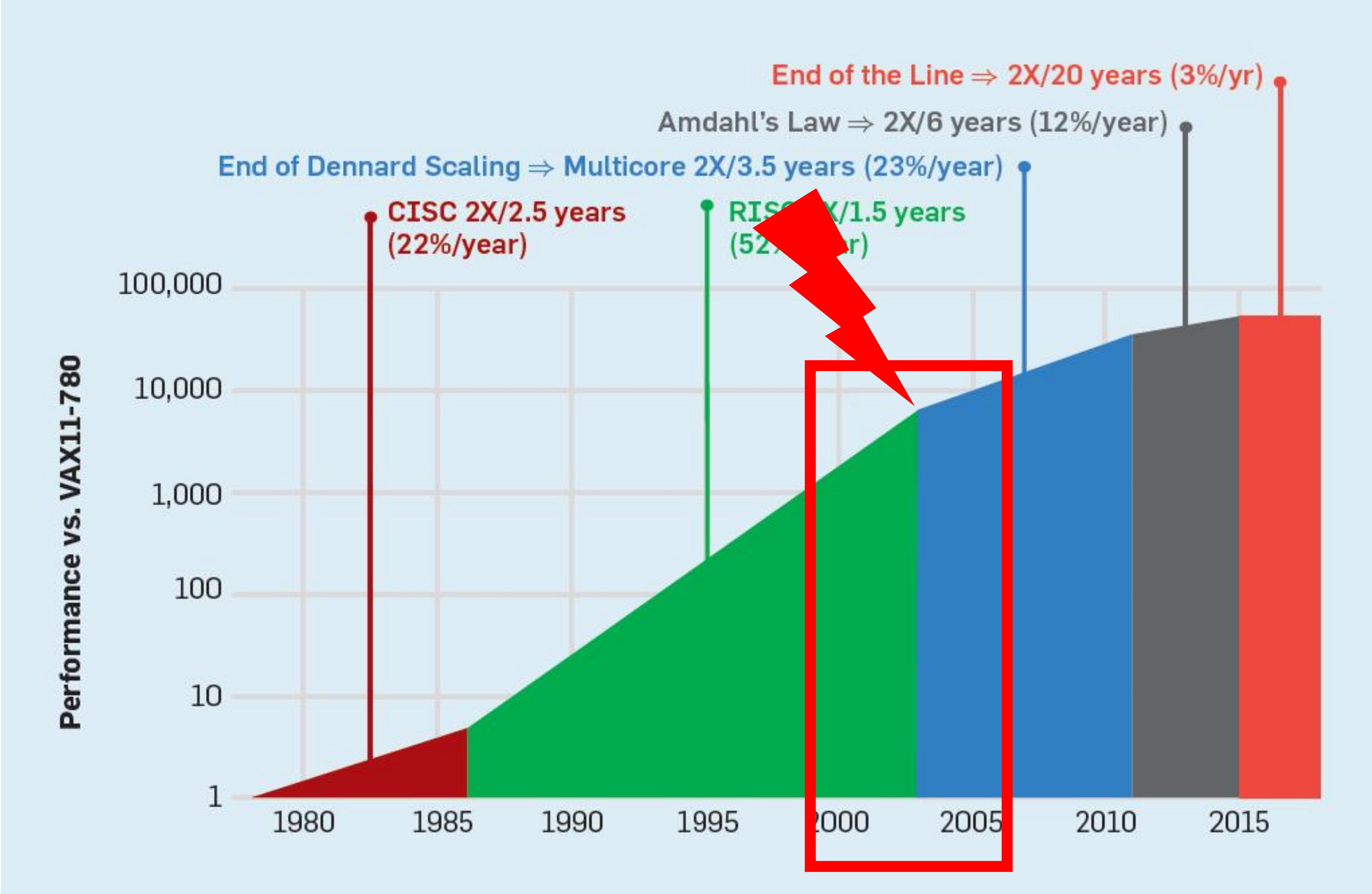


Source: jj.github.io

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Heritage of *Moore's law*

- Moore's law and Dennard scaling ruled an exponential phase (green) and created a whole industry
- **These golden "*rules*" stop delivering the same speed-up in performance in early 2000 (blue)**
- Last 20 years (blue, grey, red phase), we have added more hardware modules into the SoC
- Red phase curve is flat!



Source: jj.github.io

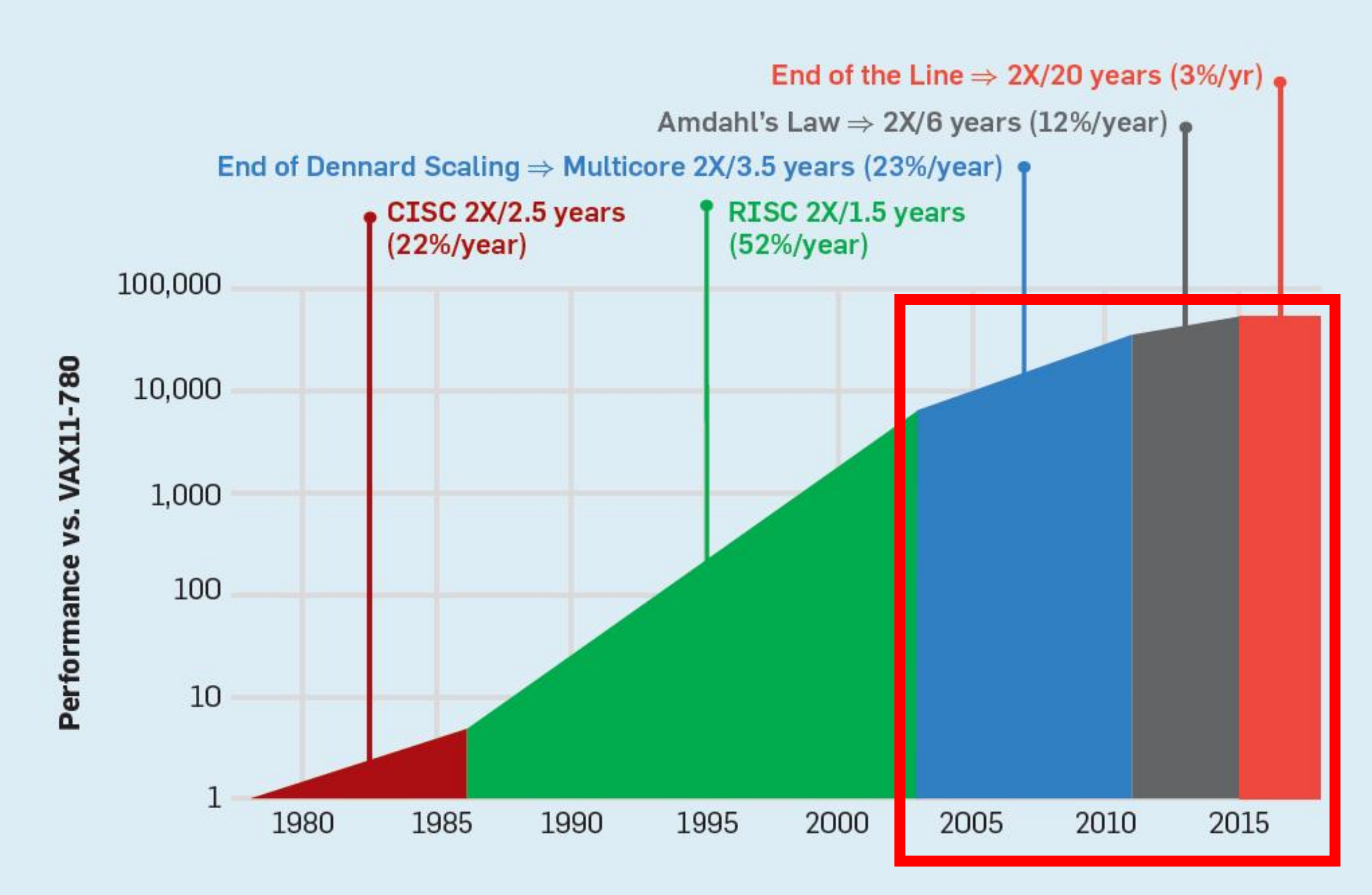
Heritage of *Moore's law*

- Moore's law and Dennard scaling ruled an exponential phase (green) and created a whole industry

- These golden "*rules*" stop delivering the same speed-up in performance in early 2000 (blue)

- **Last 20 years (blue, grey, red phase), we have added more hardware modules into the SoC**

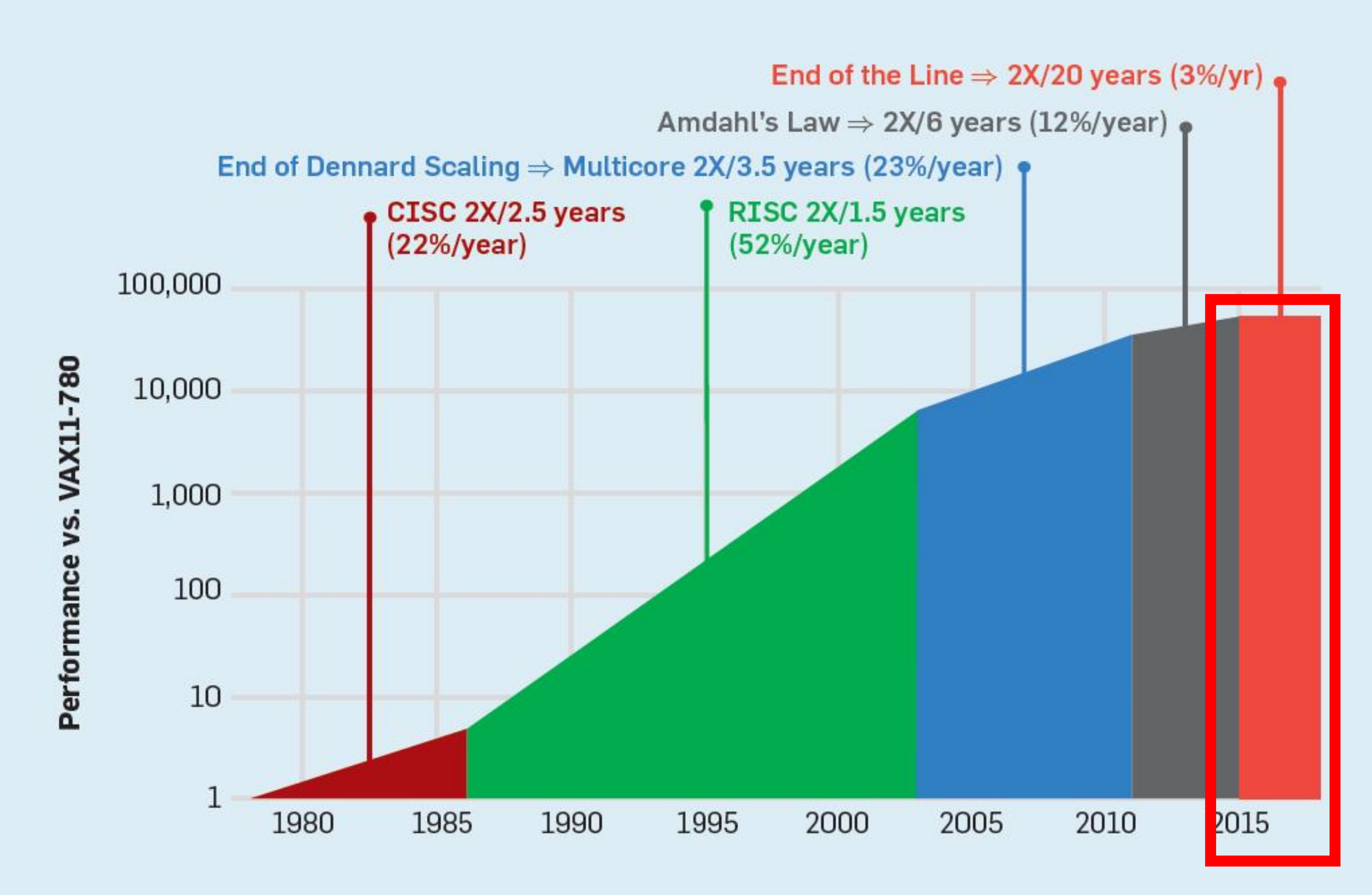
- Red phase curve is flat!



Source: jj.github.io

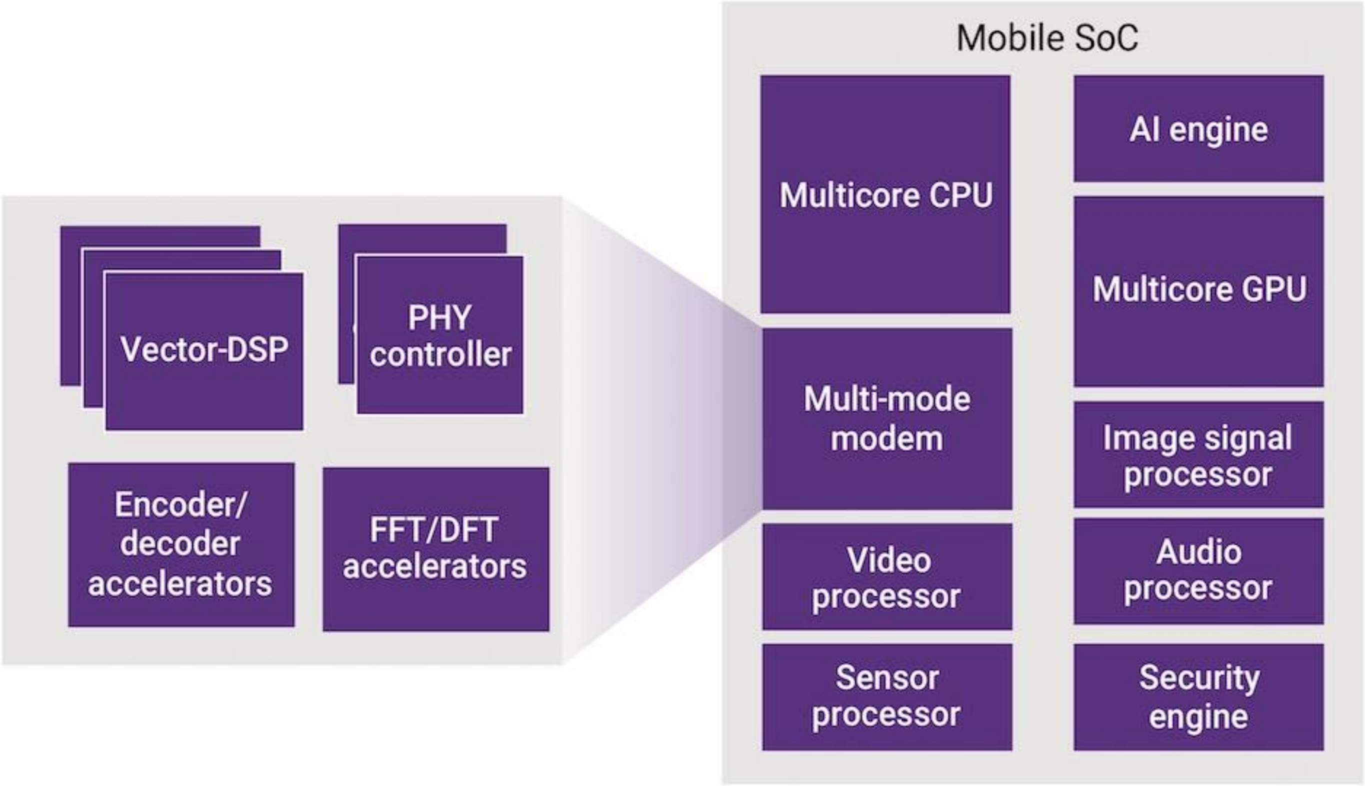
Heritage of *Moore's law*

- Moore's law and Dennard scaling ruled an exponential phase (green) and created a whole industry
- These golden "*rules*" stop delivering the same speed-up in performance in early 2000 (blue)
- Last 20 years (blue, grey, red phase), we have added more hardware modules into the SoC
- **Red phase curve is flat!**



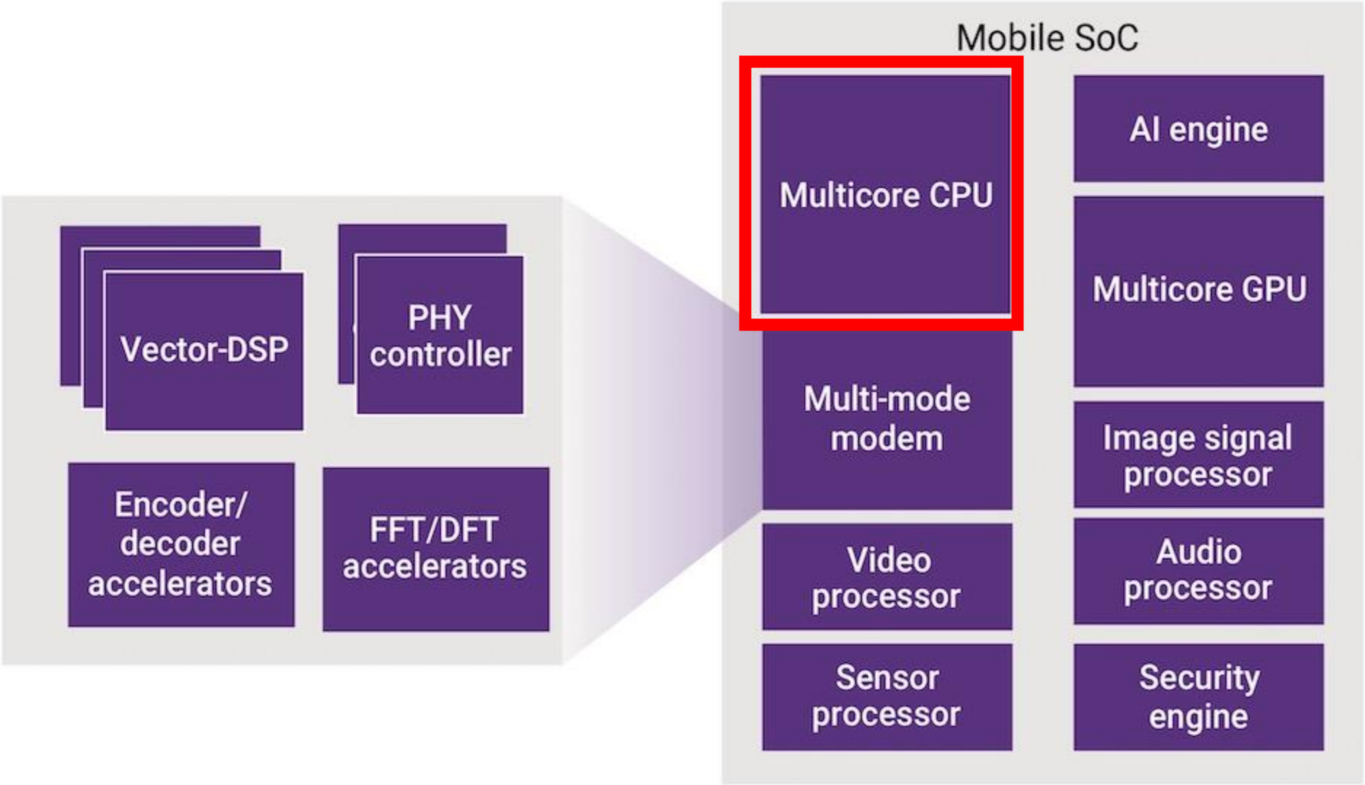
Source: jj.github.io

Current Situation



Source: www.techdesignforums.com

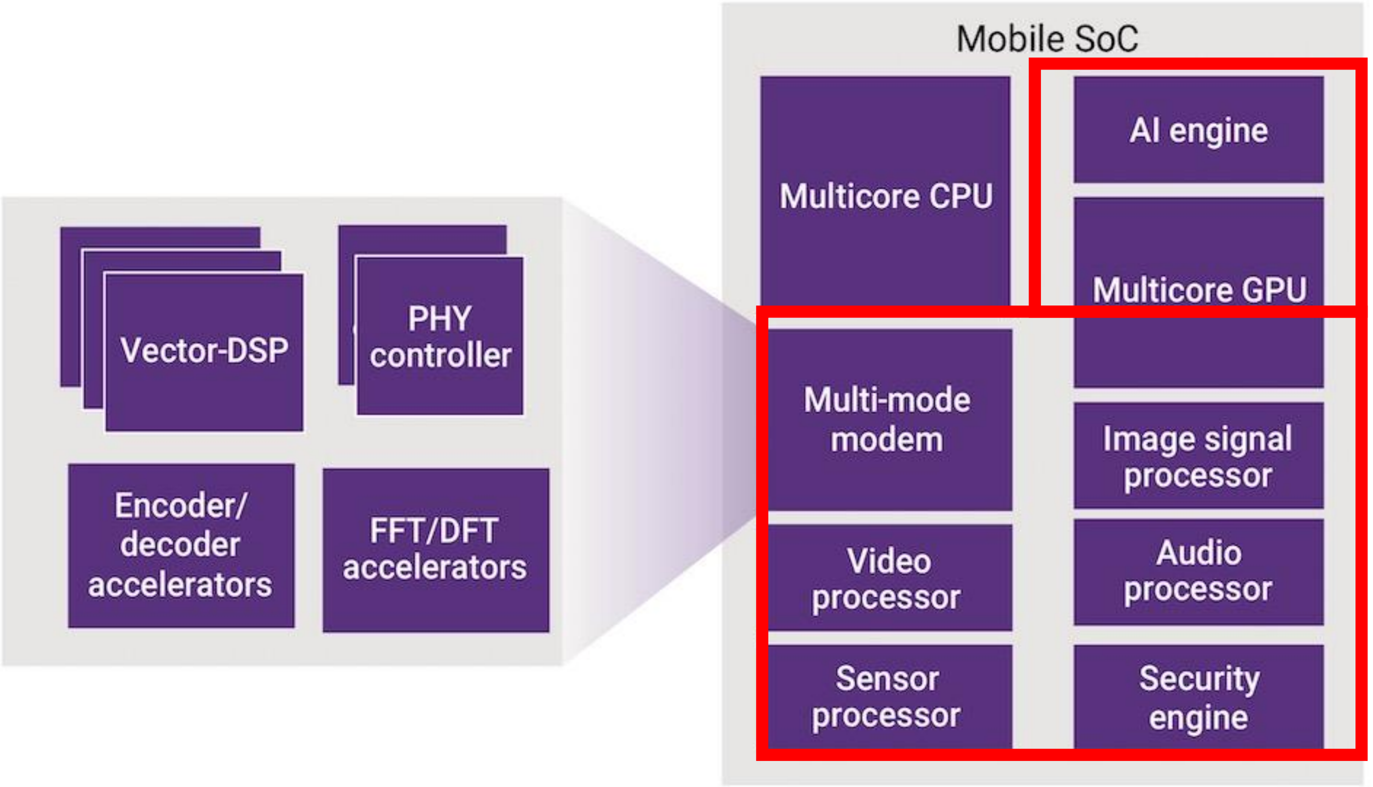
Current Situation



Source: www.techdesignforums.com

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Current Situation



Source: www.techdesignforums.com

Motivation

- **Boom in fabricating new chips**

Sources: riscv.org , lowrisc.org and graphcore.ai

Motivation

- Boom in fabricating new chips

- **Multiple and heterogeneous hardware modules on the same SoC requires complex integration and verification processes**

Motivation

- Boom in fabricating new chips
- Multiple and heterogeneous hardware modules on the same SoC requires complex integration and verification processes
- **Improve the tools to verify large-scale hardware designs**

Outline

- Introduction and Motivation
- **Gem5+RTL: A Full-System RTL Simulation Infrastructure**
- Use-case and Evaluation: NVDLA
- Conclusions and Future Work

Gem5+RTL Design Objectives

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

Gem5+RTL Design Objectives

- Provide a 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 Design Objectives

- Provide a 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

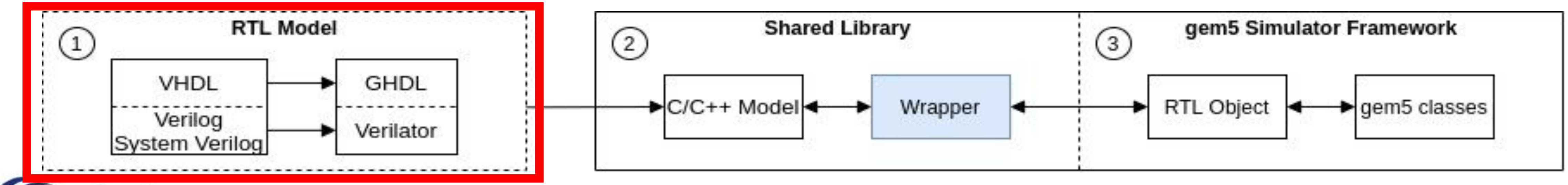
- **Enable testing** the **implemented functionality** of these hardware blocks and also, the **expected performance** they will provide on an existing **SoC design**

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

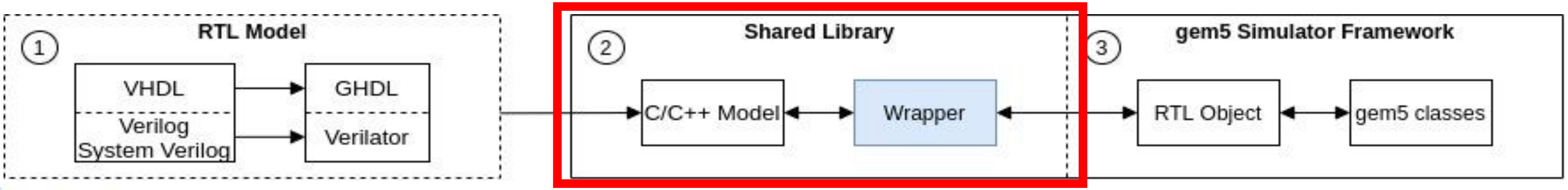


Framework Design

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

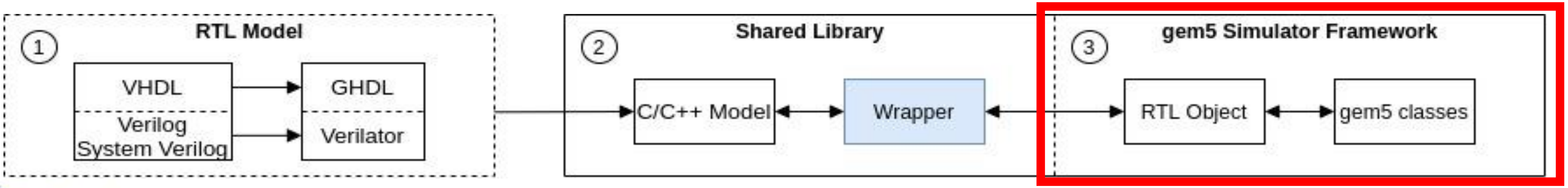


Framework Design

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**



Outline

- Introduction and Motivation
- Gem5+RTL: A Full-System RTL Simulation Infrastructure
- **Use-case and Evaluation: NVDLA**
- Conclusions and Future Work

Use Cases: NVDLA

- **NVDLA is the NVDIA Deep Learning Accelerator**

- **Open Source → on GitHub, Good Documentation**

Use Cases: NVDLA

- NVDLA is the NVDIA Deep Learning Accelerator

- Open Source → on GitHub, good documentation

- **Jetson Family of Products have some of these units in the SoC**

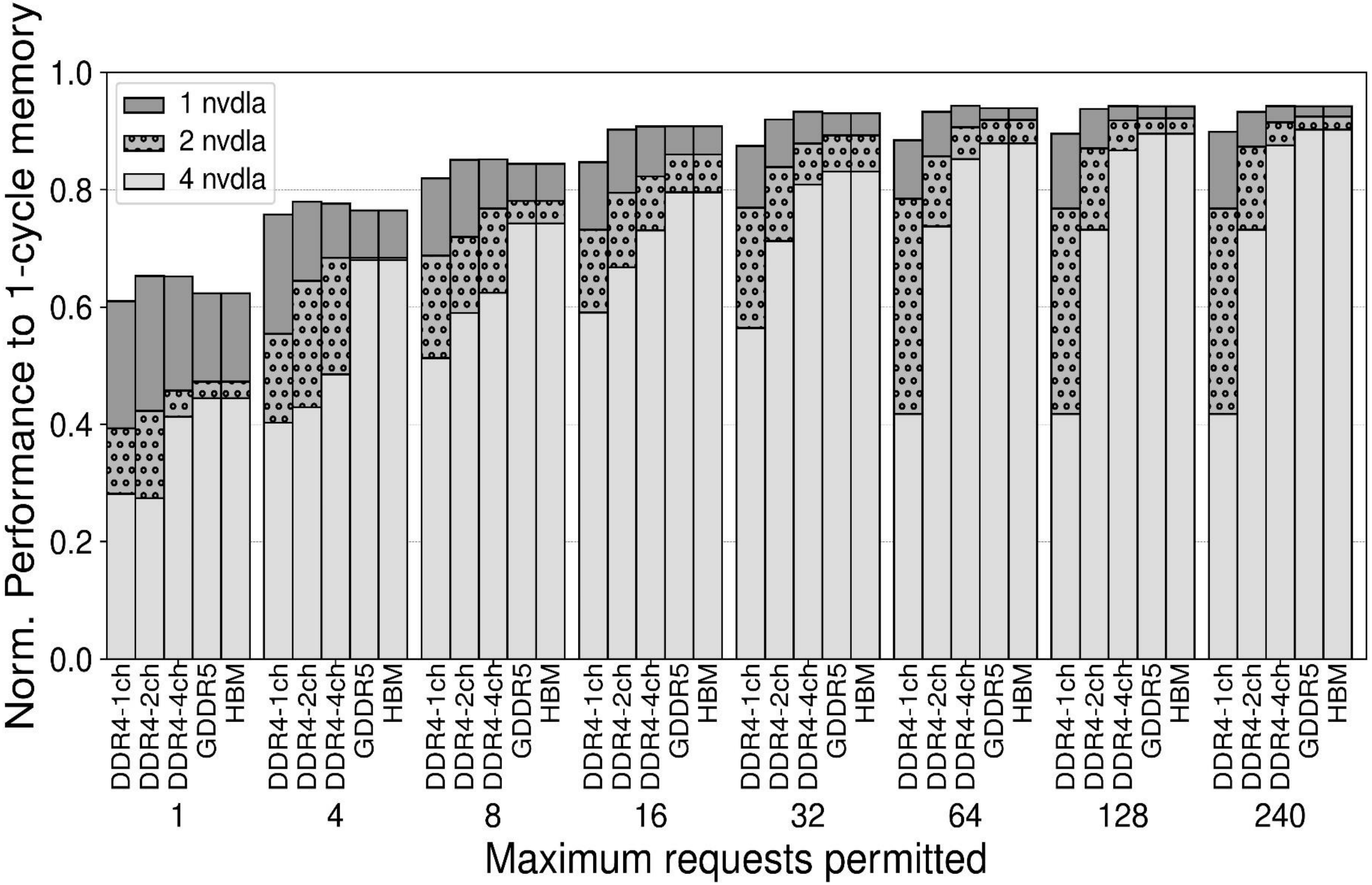
- **Perform a Design Space Exploration of which type of main memory is suitable**

Evaluation NVDLA: GoogleNet

- **Evaluation of NVDLA performed by executing traces of real applications provided by NVIDIA**

- Parameters for the design space exploration (x-axis)

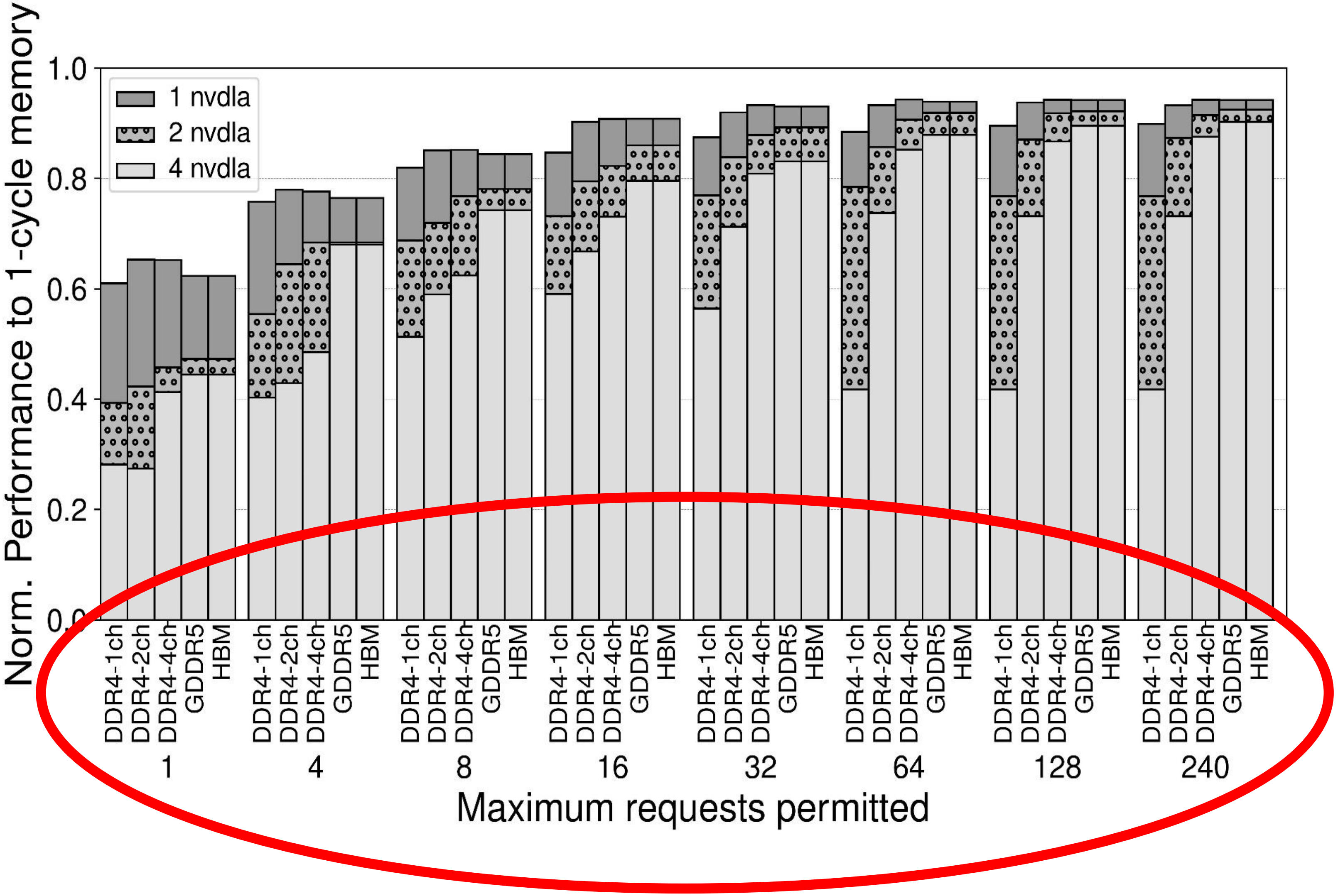
- Performance (y-axis) is normalized to an ideal 1 cycle memory latency



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

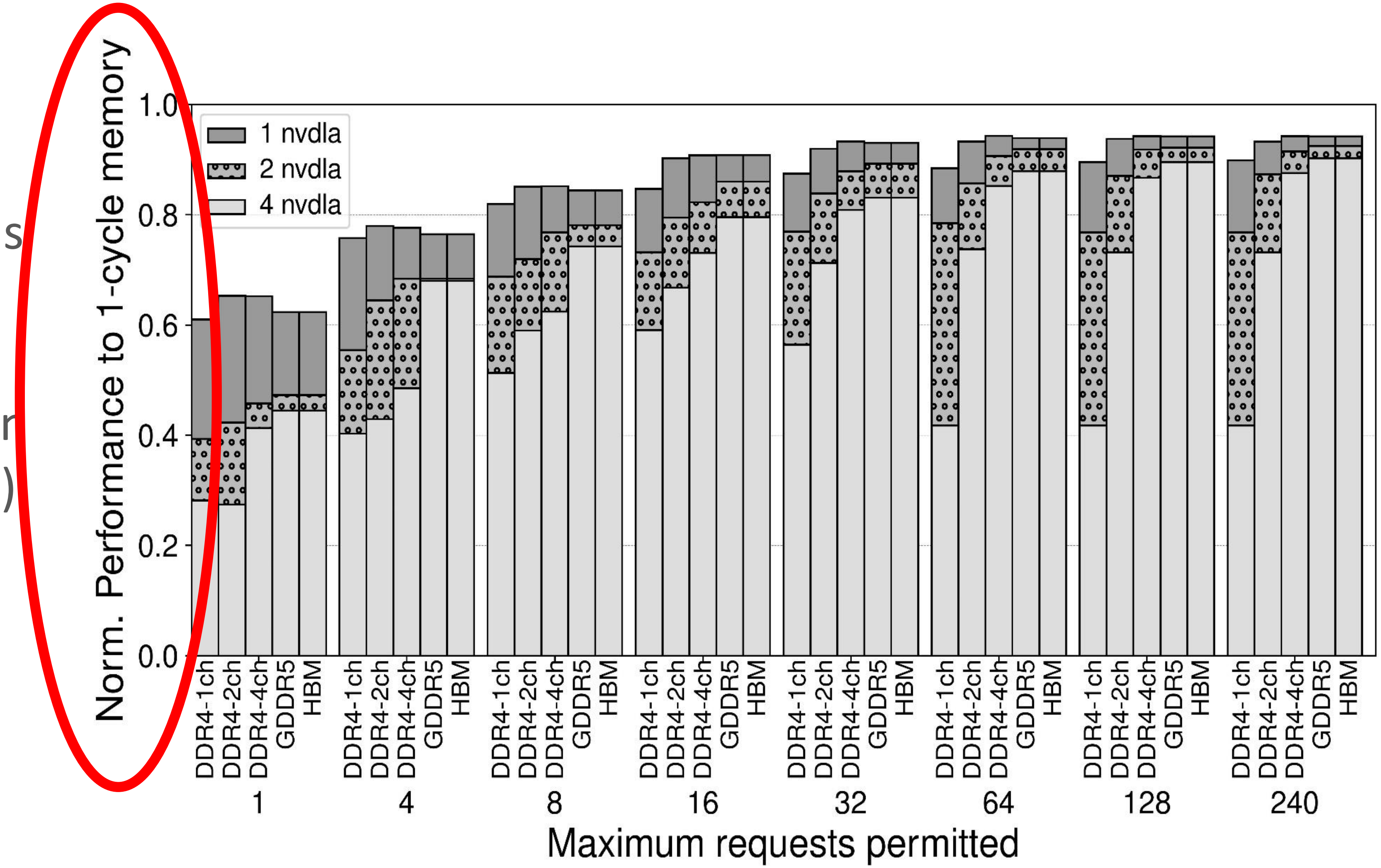
Evaluation NVDLA: GoogleNet

- 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



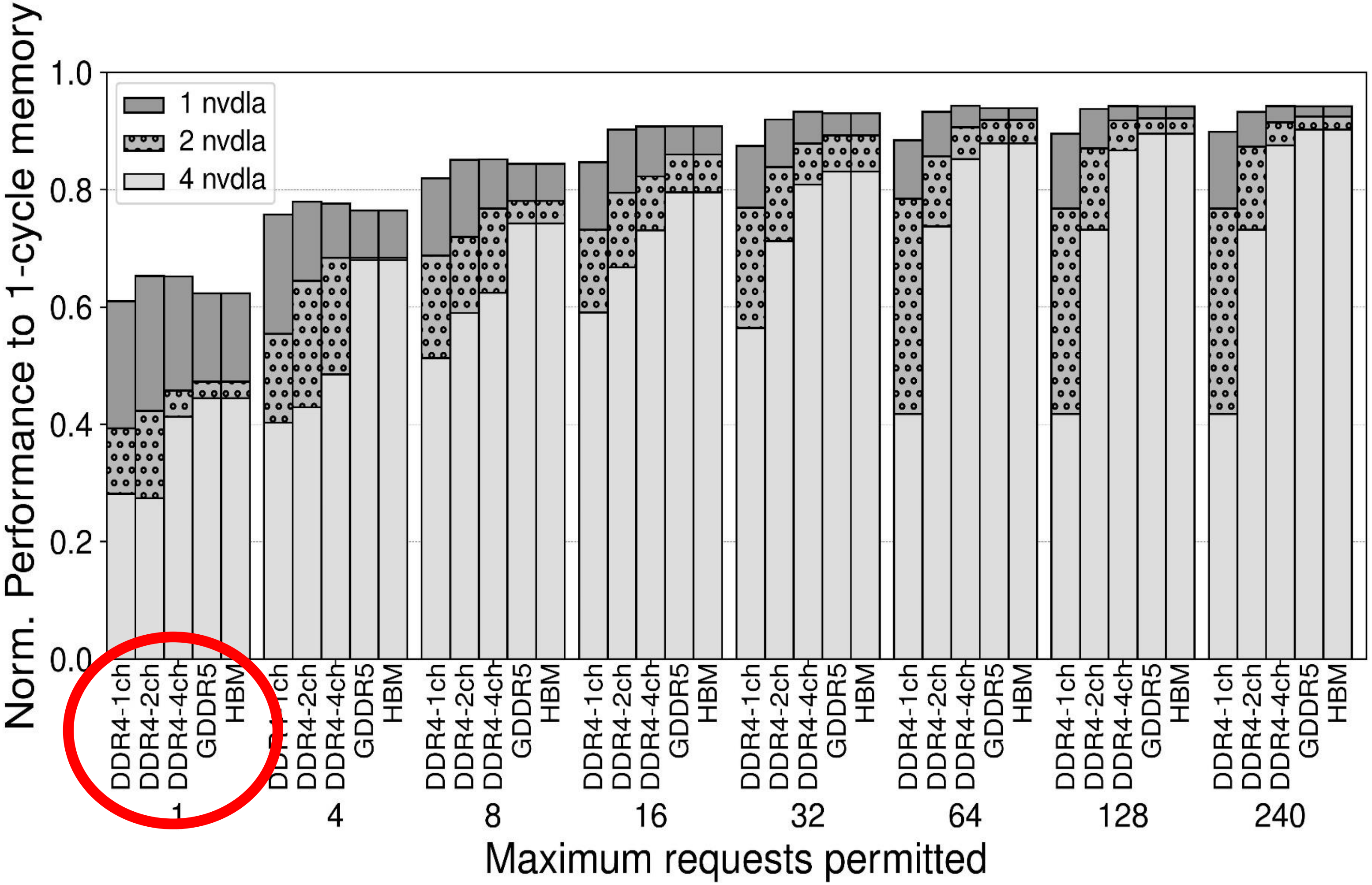
Evaluation NVDLA: GoogleNet

- 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

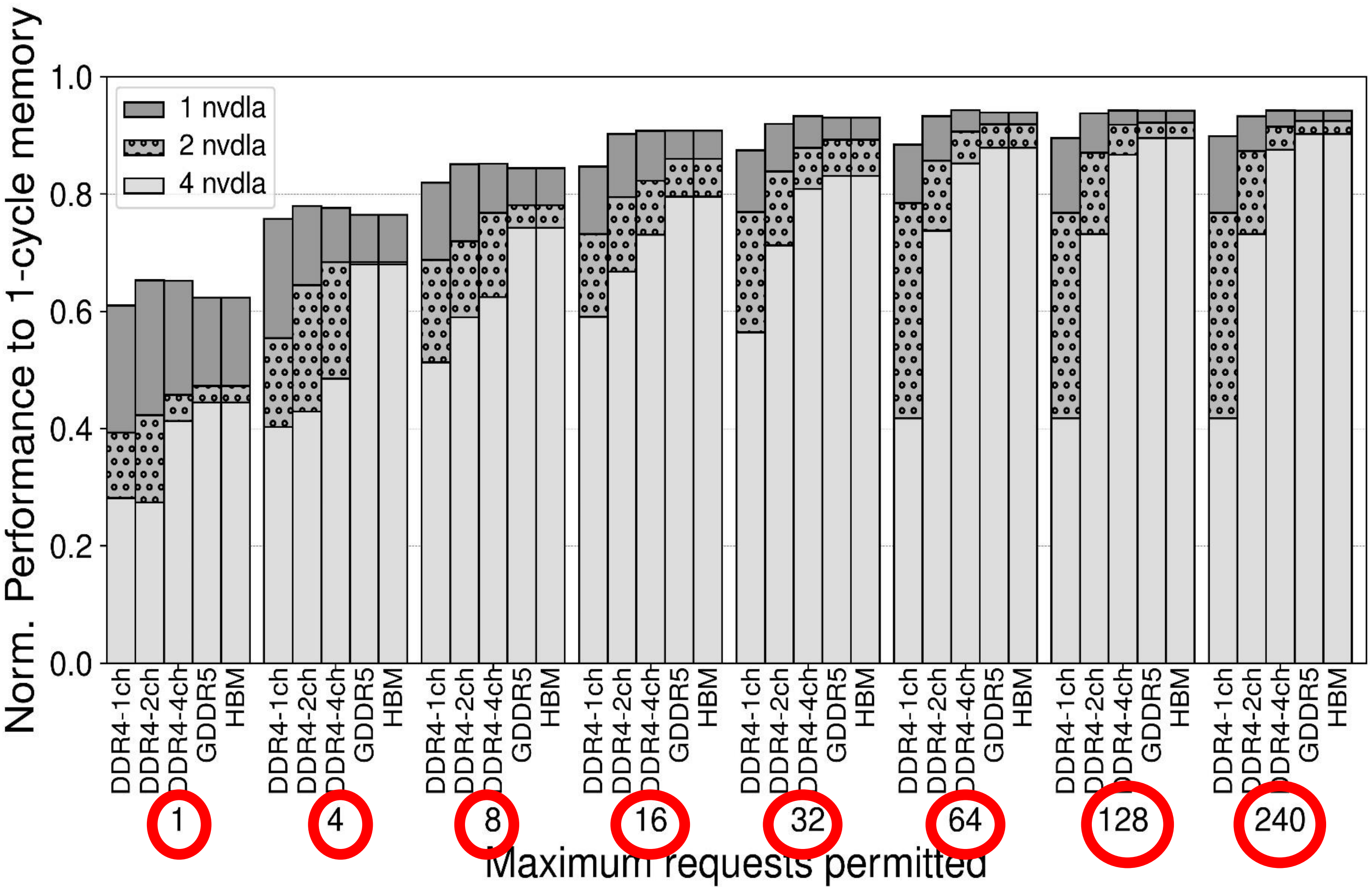


Evaluation NVDLA: GoogleNet

- Using several memory configurations

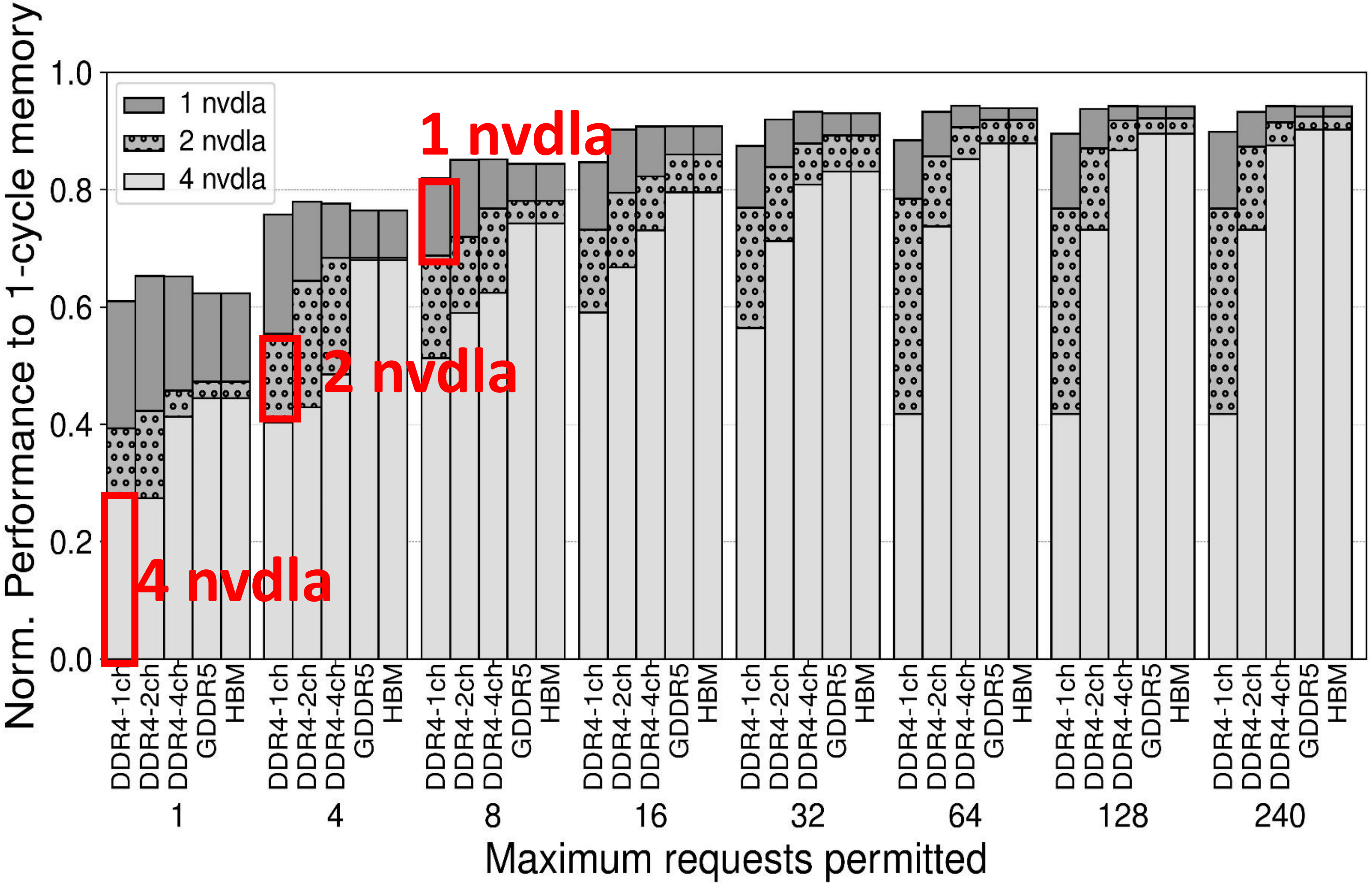
- **Different number of maximum requests from NVDLA to main memory**

- Different number of nvdla in the system: 1, 2 and 4 nvdla's configurations



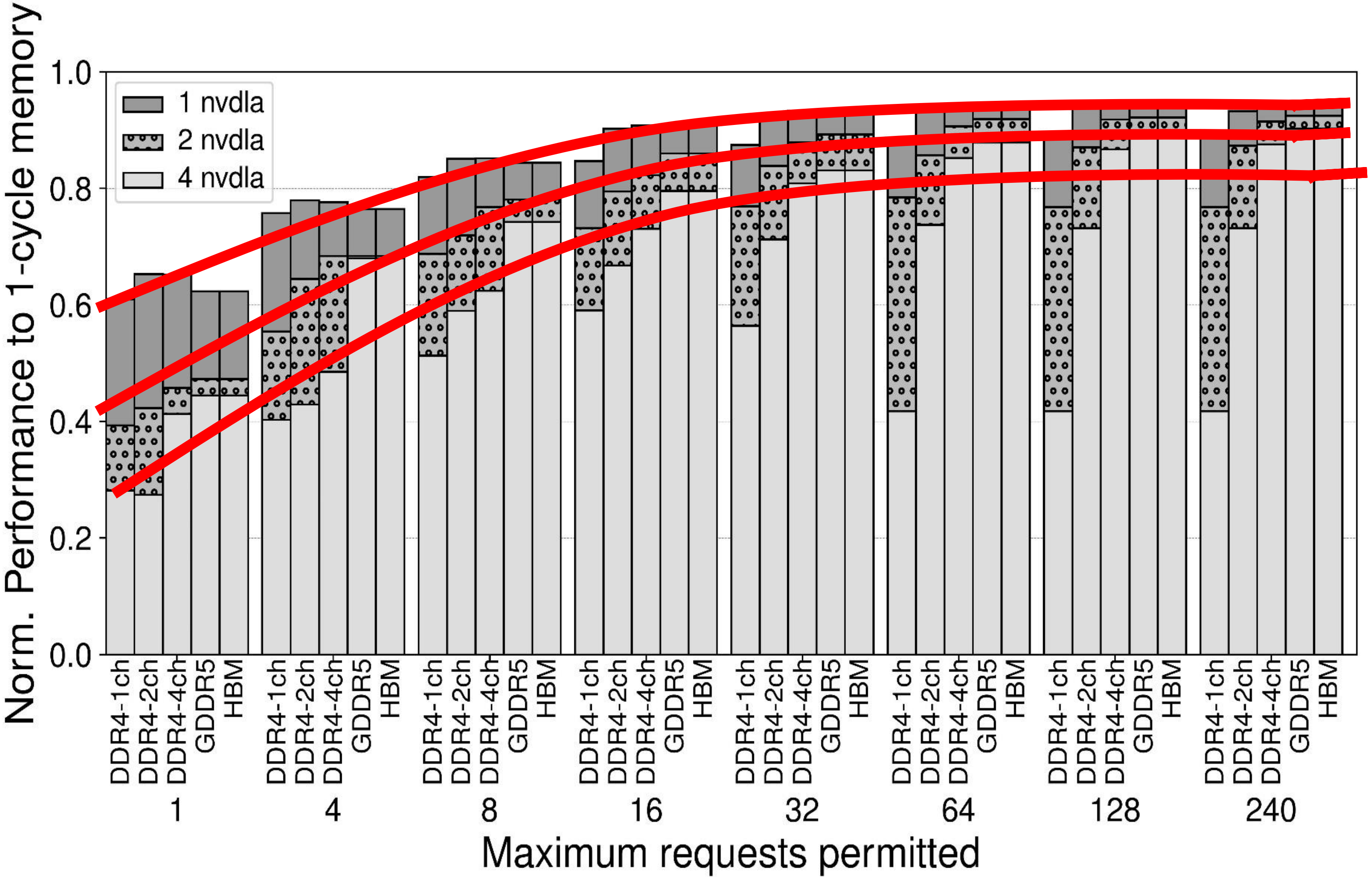
Evaluation NVDLA: GoogleNet

- 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



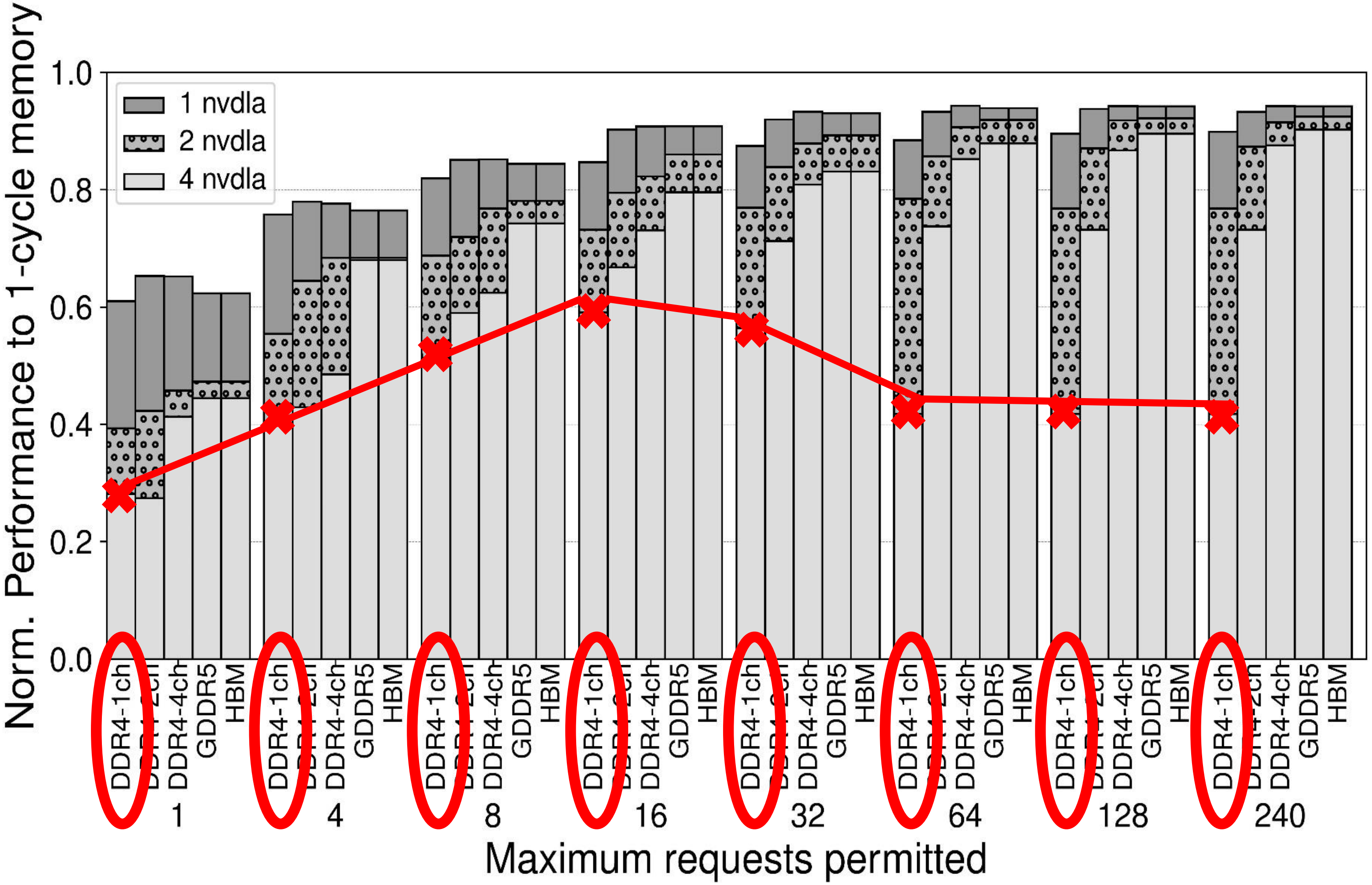
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Evaluation NVDLA: GoogleNet

- Maximum number of requests affects dramatically

- **Some memory configs cannot deliver enough bw for 2 and 4 nvdlas**

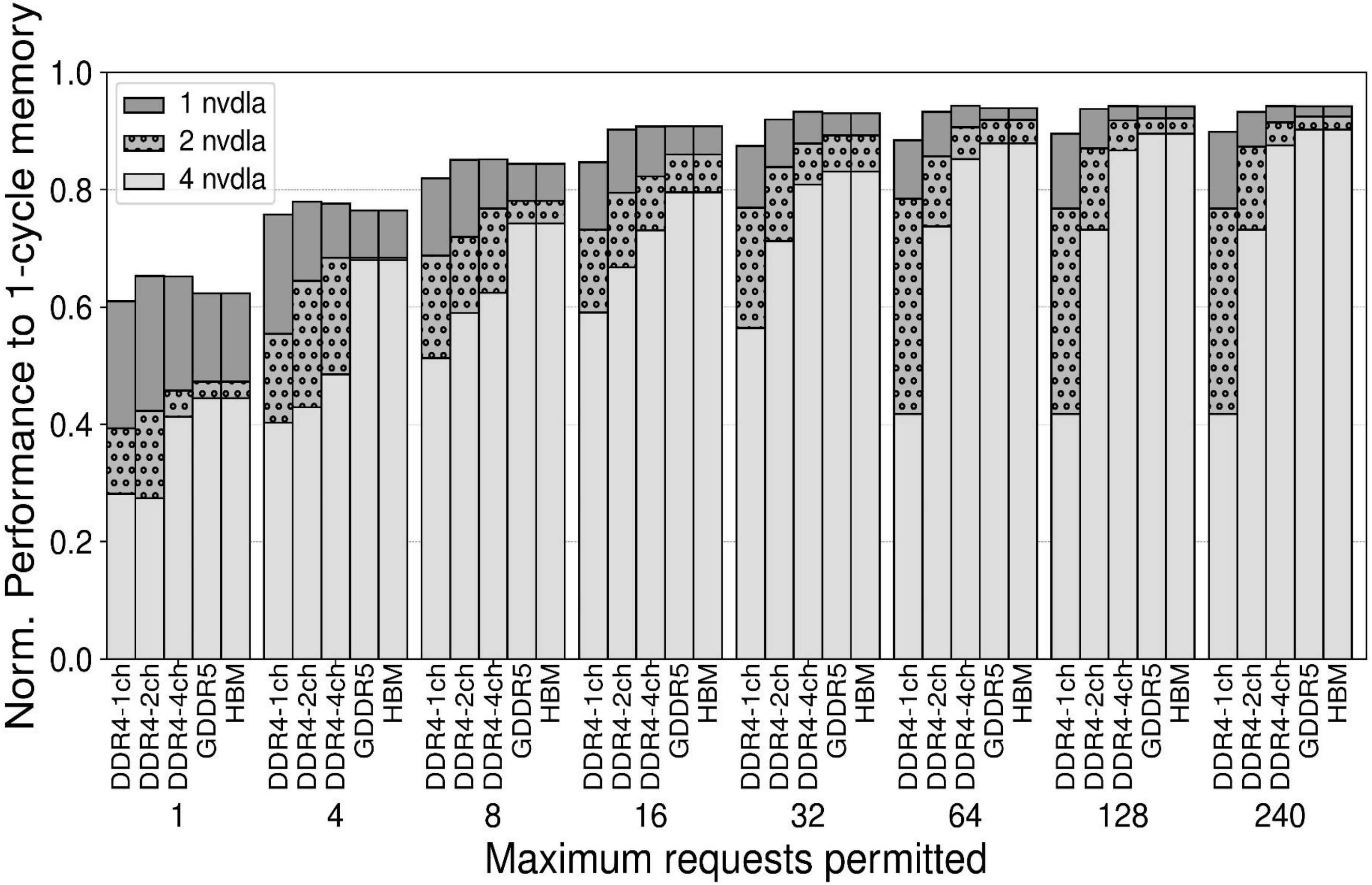
- We recommend HBM or GDDR5 when more than 2 NVDLAs are in the system



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

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**



Outline

- Introduction and Motivation
- Gem5+RTL: A Full-System RTL Simulation Infrastructure
- Use-case and Evaluation: NVDLA
- **Conclusions and Future Work**

Conclusions

- **We provide an infrastructure able to integrate RTL models inside a full system simulator**
  - **Boots unmodified Linux**
  - **Complete software stack**
  - **Models interactions will all the SoC components**

- We provide two relevant use-cases evaluation
  - Debugging
  - Performance

- We believe our tool is suitable for SoC designers to make informed design decisions

Conclusions

- We provide an infrastructure able to integrate RTL models inside a full system simulator
    - Boots unmodified Linux
    - Complete software stack
    - Models interactions will all the SoC components

- **We provide two relevant use-cases evaluation**
    - **Debugging**
    - **Performance**

- We believe our tool is suitable for SoC designers to make informed design decisions

Conclusions

- We provide an infrastructure able to integrate RTL models inside a full system simulator
  - Boots unmodified Linux
  - Complete software stack
  - Models interactions will all the SoC components

- We provide two relevant use-cases evaluation
  - Debugging
  - Performance

- **We believe our tool is suitable for SoC designers to make informed design decisions**

Future Work

- **Improving the connectivity of the NVDLA with gem5, using an IOMMU**

- Adding more RTL models and explore, for example, interesting re-programmable hardware that can be placed on the pipeline

- Add more features to the framework, for example, allow checkpointing of RTL models connected to the regular checkpoints of gem5

Future Work

- Improving the connectivity of the NVDLA with gem5, using an IOMMU
- **Adding more RTL models and explore, for example, interesting re-programmable hardware that can be placed on the pipeline**
- Add more features to the framework, for example, allow checkpointing of RTL models connected to the regular checkpoints of gem5

Future Work

- Improving the connectivity of the NVDLA with gem5, using an IOMMU

- Adding more RTL models and explore, for example, interesting re-programmable hardware that can be placed on the pipeline

- **Add more features to the framework, for example, allow checkpointing of RTL models connected to the regular checkpoints of gem5**

BSC
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

INTERNATIONAL
CONFERENCE ON
PARALLEL
PROCESSING

Check it out!

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

INTERNATIONAL
CONFERENCE ON
PARALLEL
PROCESSING
ICPP/2021/CHICAGO/USA

acm In-Cooperation
sighpc
AUGUST 9–12, 2021

guillem.lopez@bsc.es

Outline

- Introduction and Motivation
- Gem5+RTL: A Full-System RTL Simulation Infrastructure
- **Use-case and Evaluation: PMU**
- Use-case and Evaluation: NVDLA
- Conclusions and Future Work

Use Cases: PMU

- PMU is Performance Monitor Unit: Takes statistics of the core

- Developed in Verilog at BSC

Use Cases: PMU

- PMU is Performance Monitor Unit: Takes statistics of the core

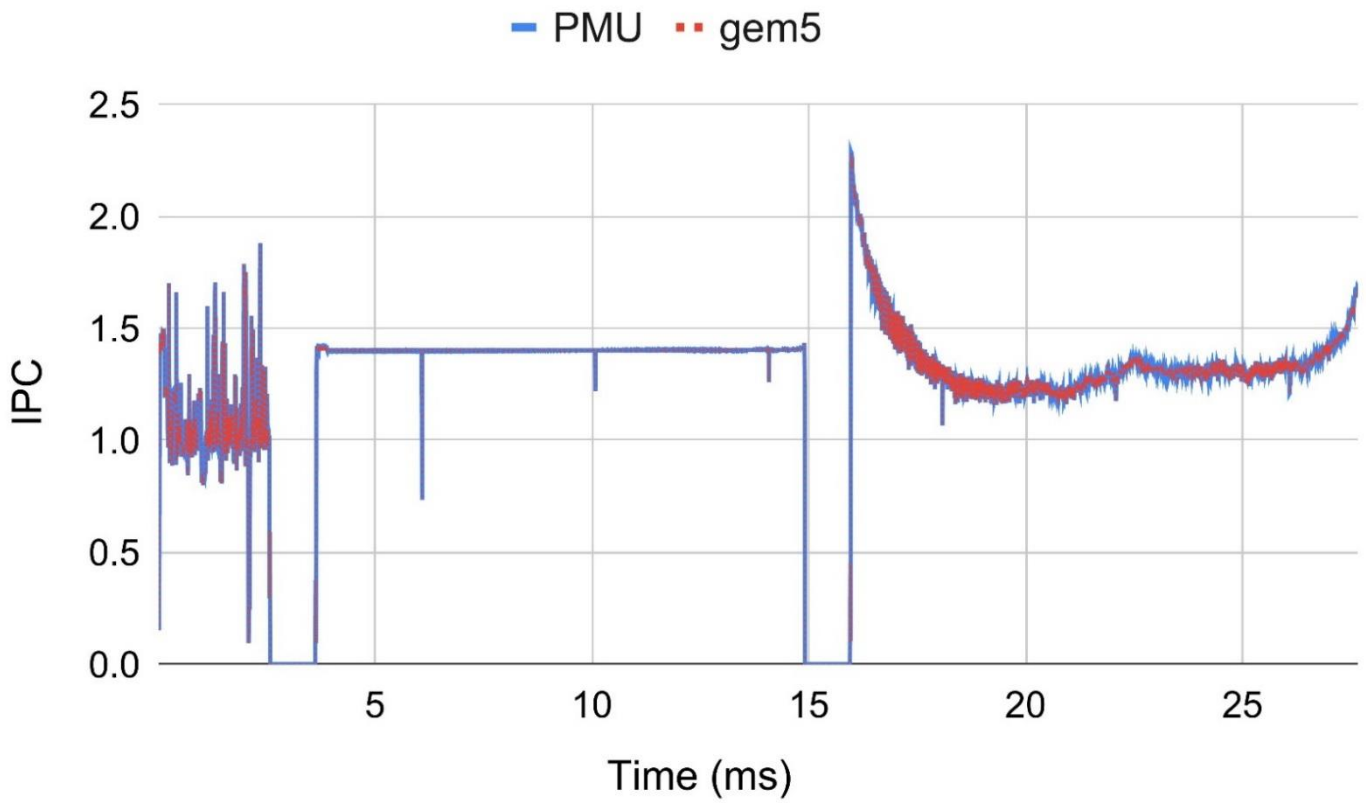
- Developed in Verilog at BSC

- **Has programmability features to trigger thresholds**

- **Debug functionally the hardware block**

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



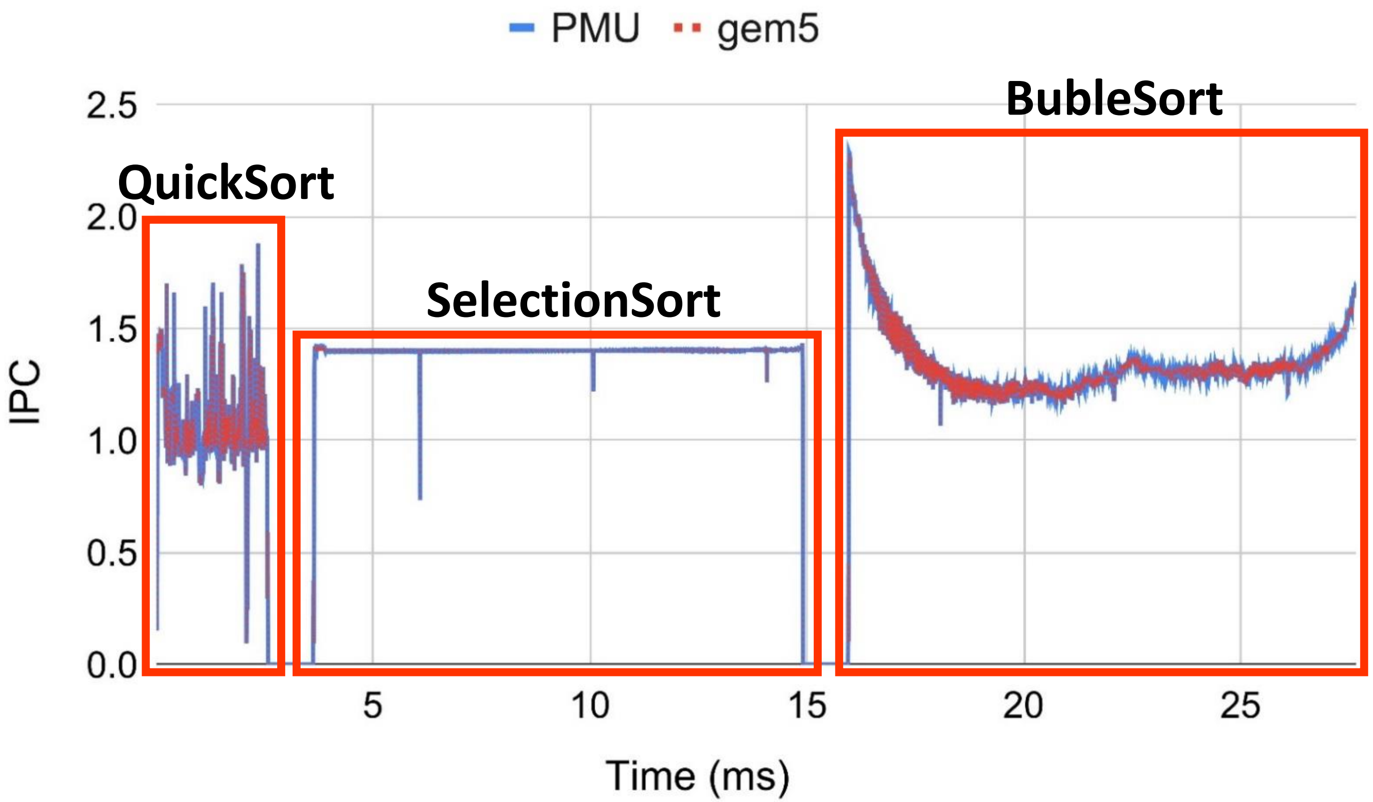
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

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



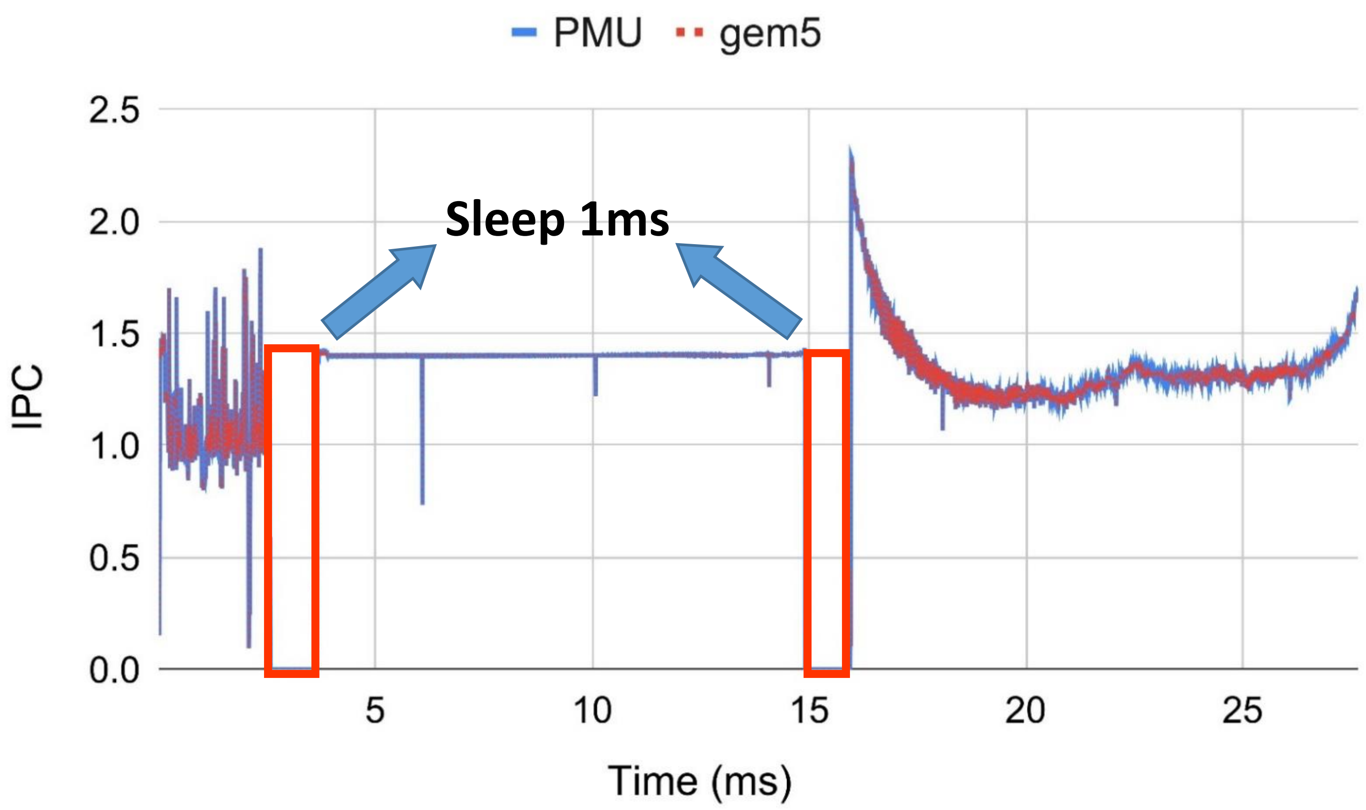
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

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**



Our Solution

- **gem5+RTL framework: a flexible infrastructure** that **enables** easy **integration of existing RTL** models with the popular full-system **gem5** simulator

Our Solution

- gem5+ RTL framework: a flexible infrastructure that enables easy integration of existing RTL models with the popular full-system gem5 simulator

- Enables to perform **functional testing and design space exploration studies** of existing RTL models on a **full-system environment** that models an entire SoC

Our Solution

- gem5+ RTL framework: a flexible infrastructure that enables easy integration of existing RTL models with the popular full-system gem5 simulator

- Enables to perform functional testing and design space exploration studies of existing RTL models on a full-system environment that models an entire SoC

- We show **two different use-cases** and evaluate their performance

Problem

- **Existing Systems-on-Chip (SoCs) have become incredibly complex,** incorporating a large number of hardware blocks in their designs.

Problem

- Existing Systems-on-Chip (SoCs) have become incredibly complex, incorporating a large number of hardware blocks in their designs.

- **Current tools do not model all the potential interactions** and restrictions that may arise **when the hardware block is integrated into a complex SoC** with a complete software stack.

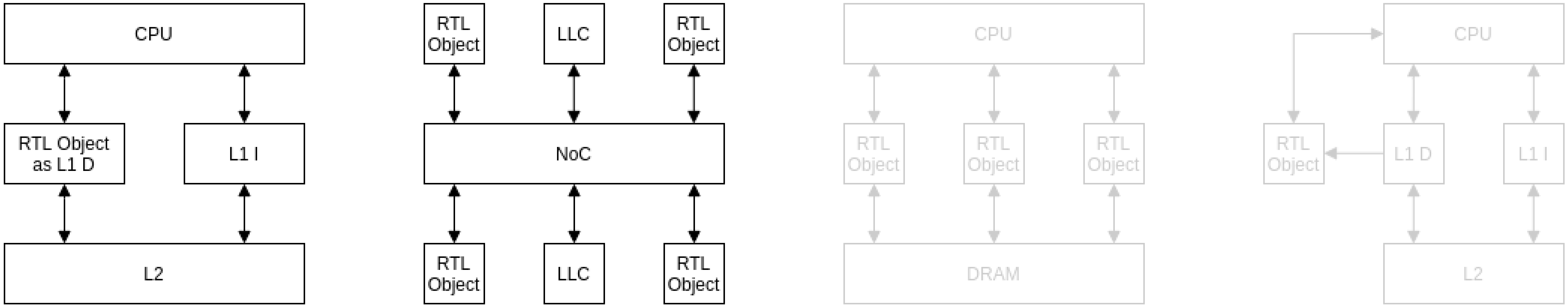
Problem

- Existing Systems-on-Chip (SoCs) have become incredibly complex, incorporating a large number of hardware blocks in their designs.

- Current tools do not model all the potential interactions and restrictions that may arise when the hardware block is integrated into a complex SoC with a complete software stack.

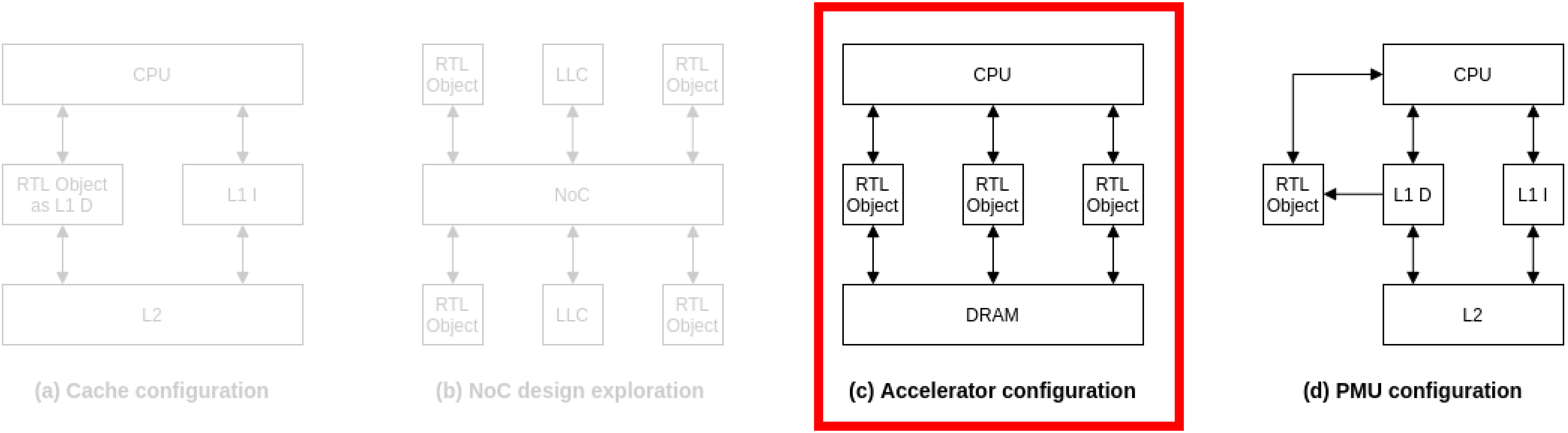
- **Need for tools that enable testing the functionality these hardware blocks**, but **also in terms of the expected performance** they will provide on an existing SoC design.

Connectivity Examples



(a) Cache configuration **(b) NoC design exploration** **(c) Accelerator configuration** **(d) PMU configuration**

Connectivity Examples



(a) Cache configuration

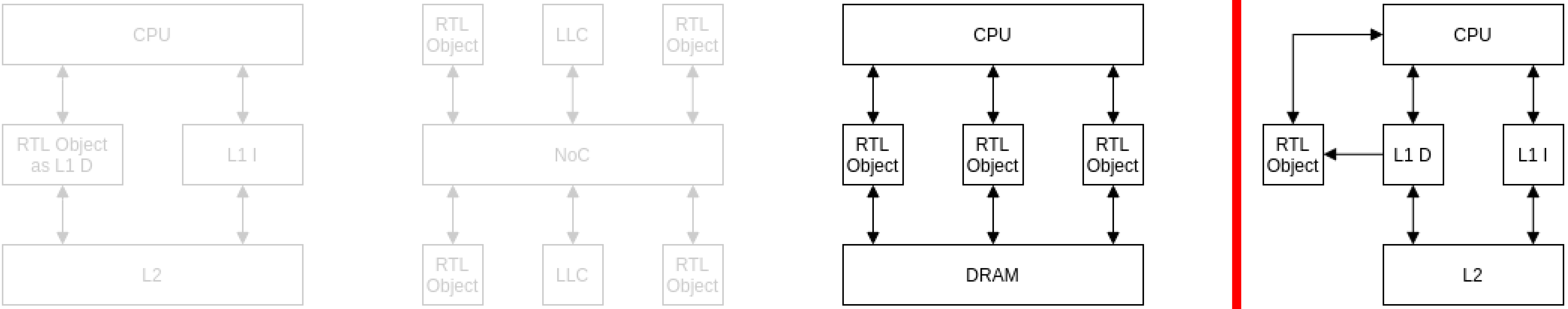
(b) NoC design exploration

(c) Accelerator configuration

(d) PMU configuration

NVDLA Use Case

Connectivity Examples



(a) Cache configuration

(b) NoC design exploration

(c) Accelerator configuration

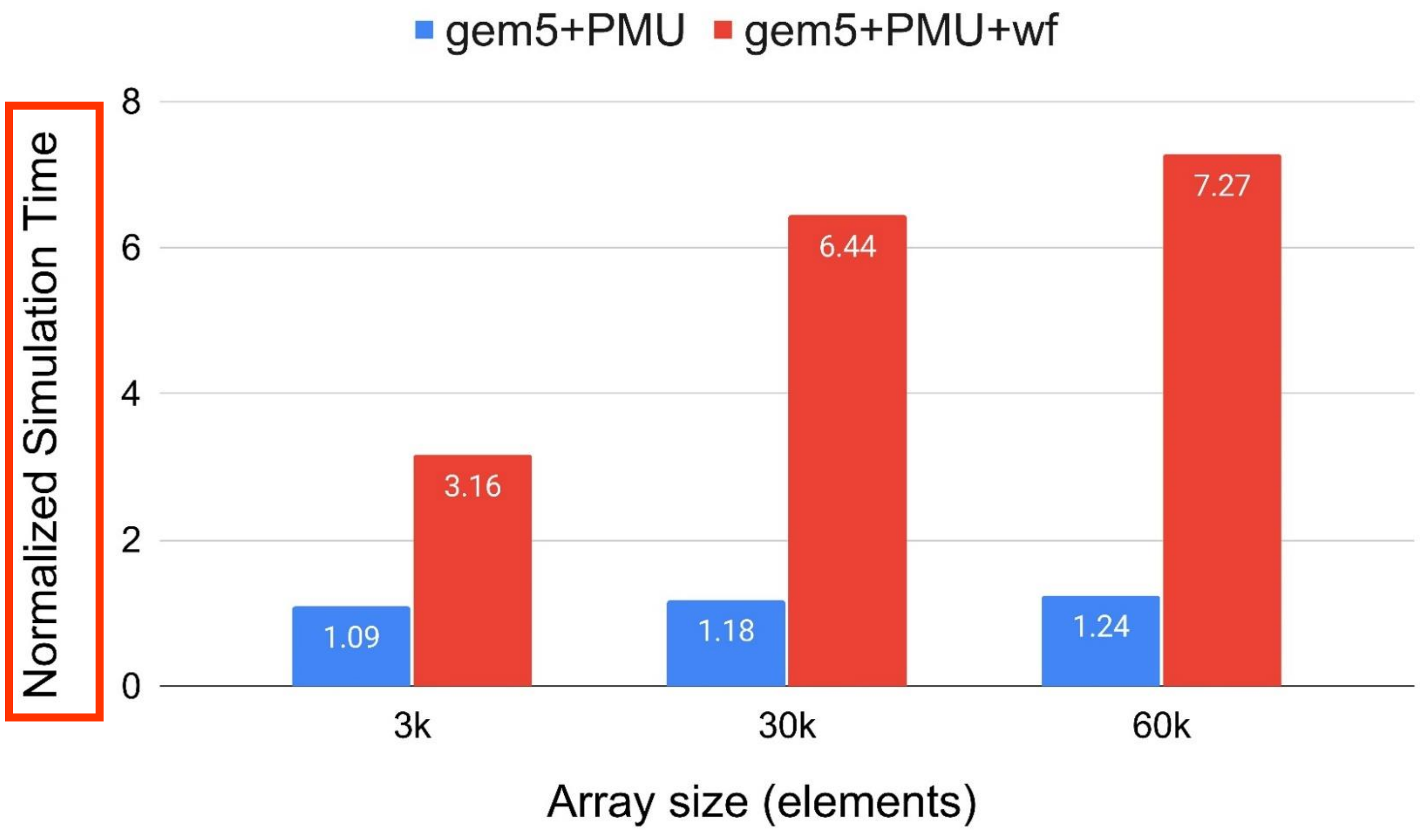
(d) PMU configuration

PMU Use Case

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

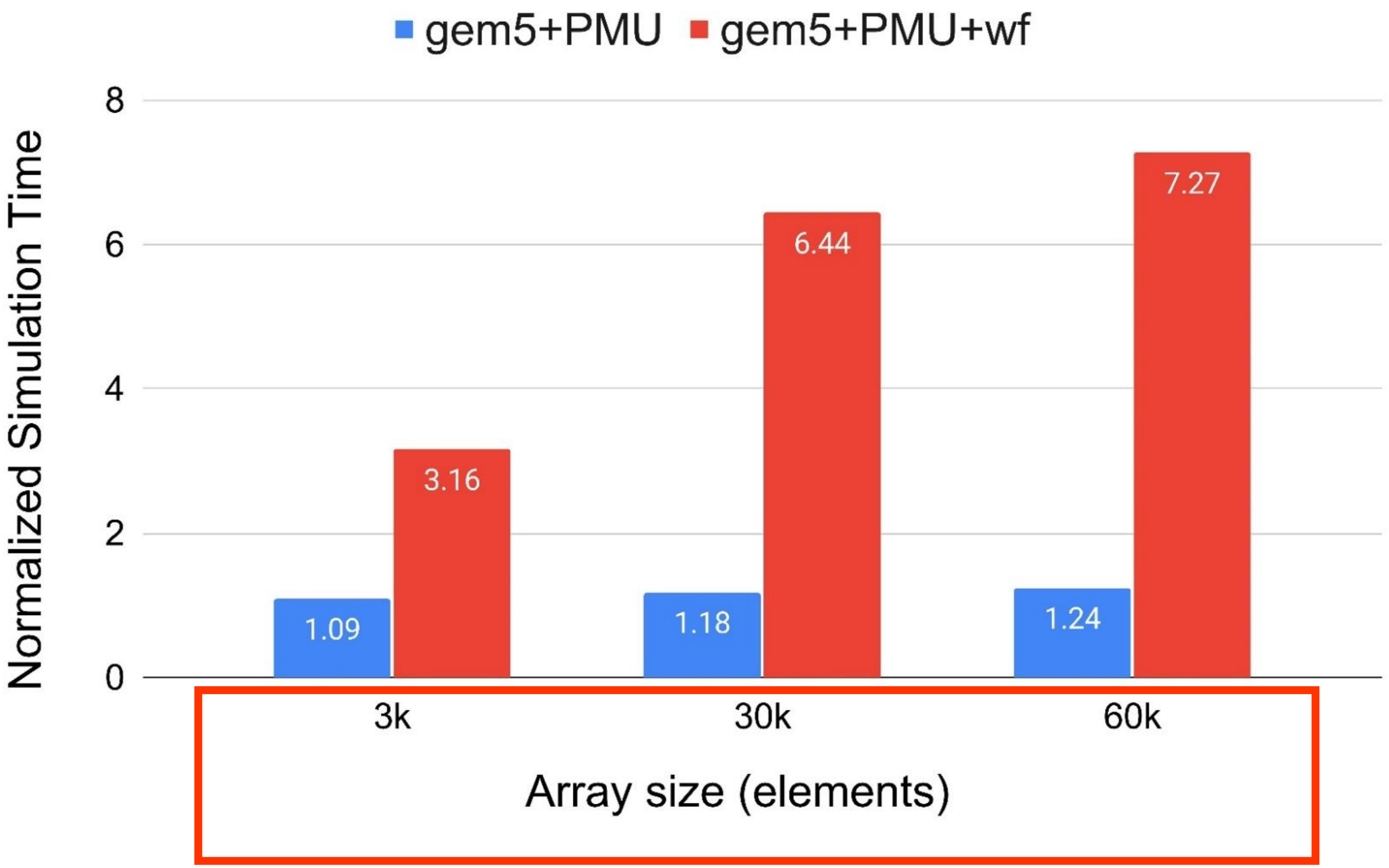
Evaluation PMU: Timing

- **Evaluated the timing overhead of the gem5+RTL (PMU) against gem5 alone with different array sizes**
- On avg. 20% overhead
- Tracing a waveform has a huge overhead as expected



Evaluation PMU: Timing

- **Evaluated the timing overhead of the gem5+RTL (PMU) against gem5 alone with different array sizes**
- On avg. 20% overhead
- Tracing a waveform has a huge overhead as expected

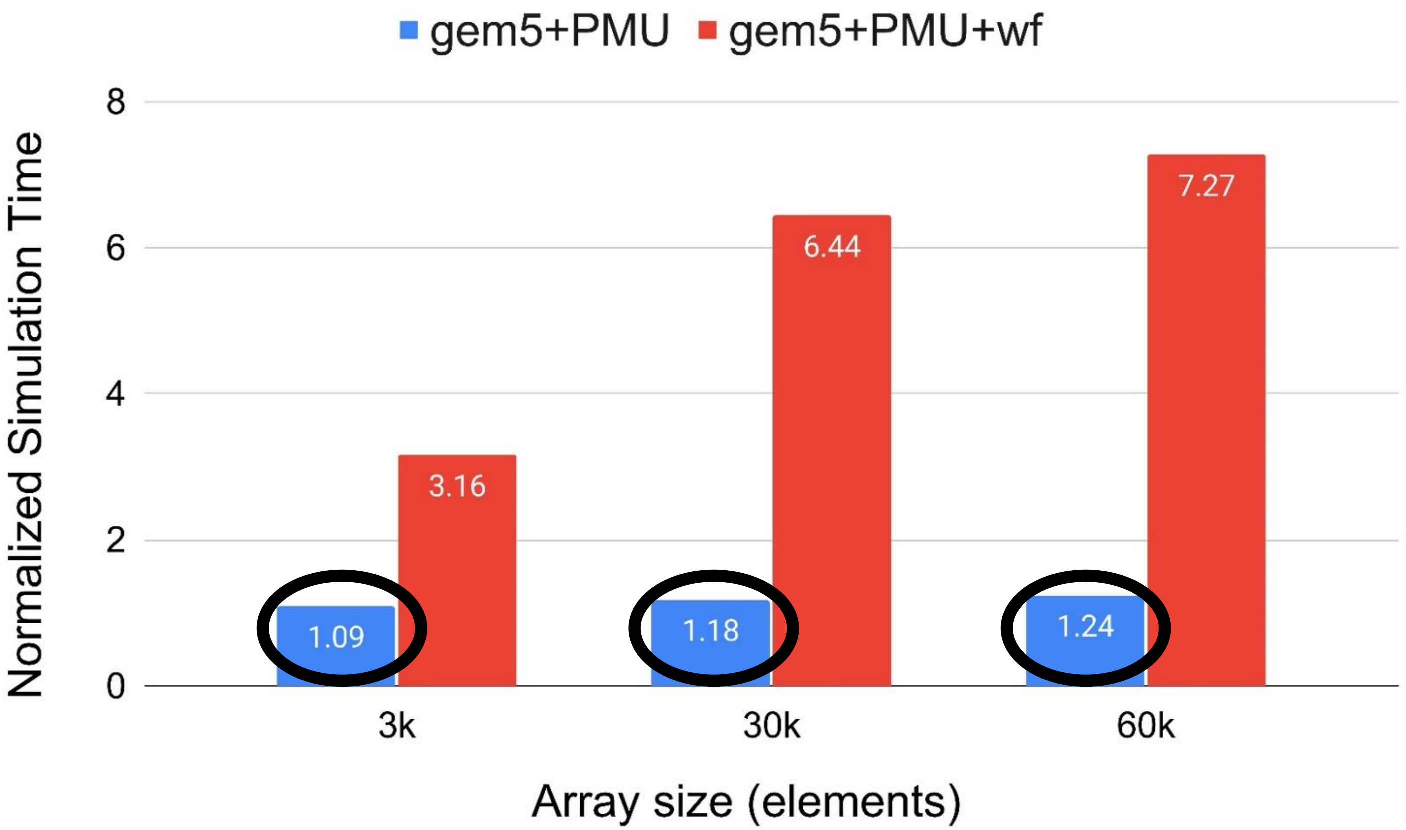


Evaluation PMU: Timing

- Evaluated the timing overhead of the gem5+RTL (PMU) against gem5 alone with different array sizes

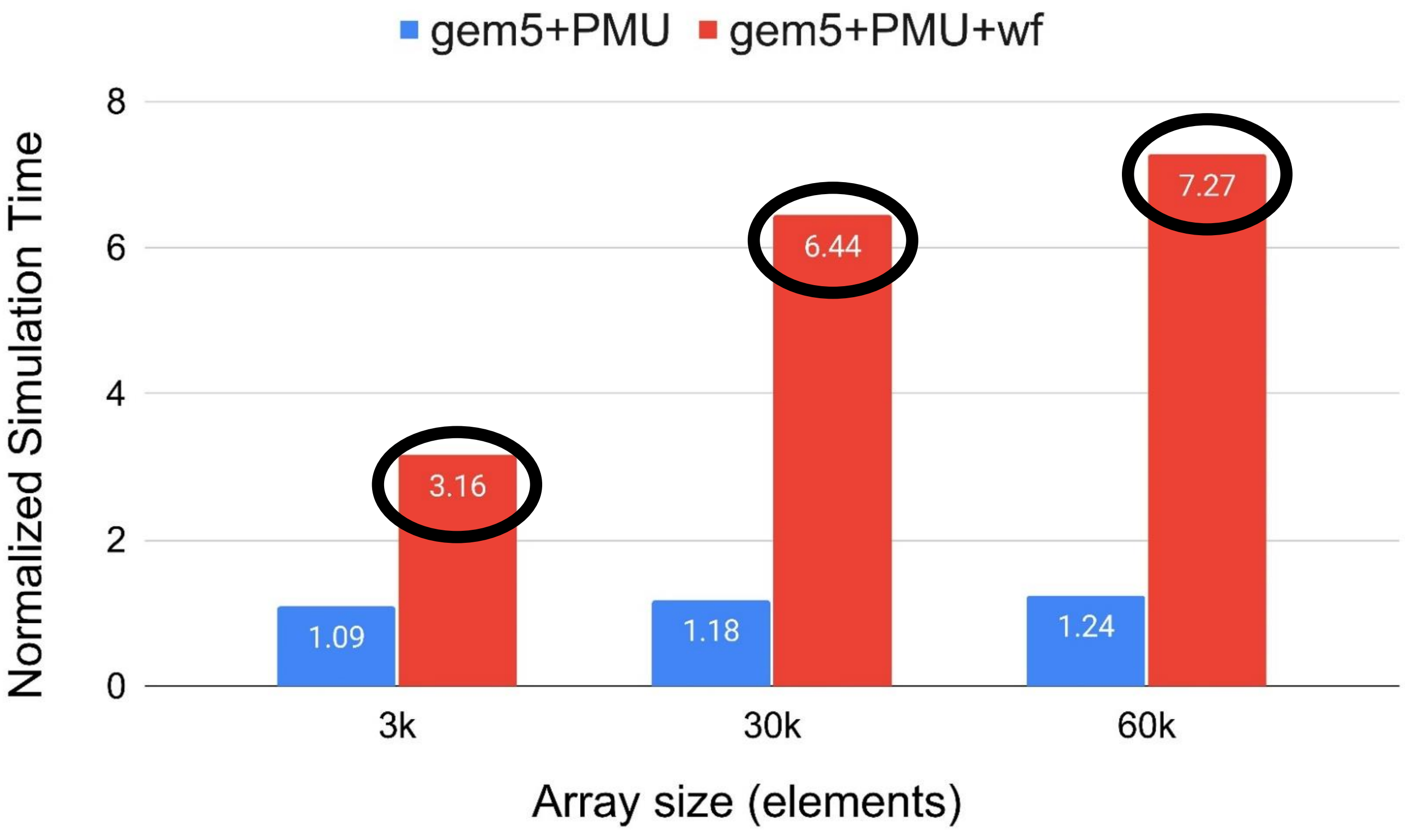
- **On avg. 20% overhead**

- Tracing a waveform has a huge overhead as expected



Evaluation PMU: Timing

- Evaluated the timing overhead of the gem5+RTL (PMU) against gem5 alone with different array sizes
- On avg. 20% overhead
- **Tracing a waveform has a huge overhead as expected**



Software Infrastructure

Full-System Simulator

- Widely used on **Academia** and **Industry**
- **Multiple ISA's** such as Armv8, x86_64, and RISC-V
- **Multi-level cache hierarchies** and **different memory technologies**.
- **Support** an **Operating System (OS)** like a **Linux** kernel and run **multi-threaded and multi-process applications**

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Logo obtained from https://www.gem5.org/

Software Infrastructure

Full-System Simulator

- Widely used on Academia and Industry
- Multiple ISA's such as Armv8, x86_64, and RISC-V
- Multi-level cache hierarchies and different memory technologies.
- Support an Operating System (OS) like a Linux kernel and run multi-threaded and multi-process applications

HDL Simulator

- Used both in **Academia** and **Industry**
- **High speed** by compiling synthesizable **Verilog to** multi-threaded **C++/SystemC**
- **Good level of performance** when **compared** to the **commercial solutions**

Logos obtained from https://www.gem5.org/ and https://www.veripool.org/wiki/verilator

Extra ch2: State Of Art

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Related Work

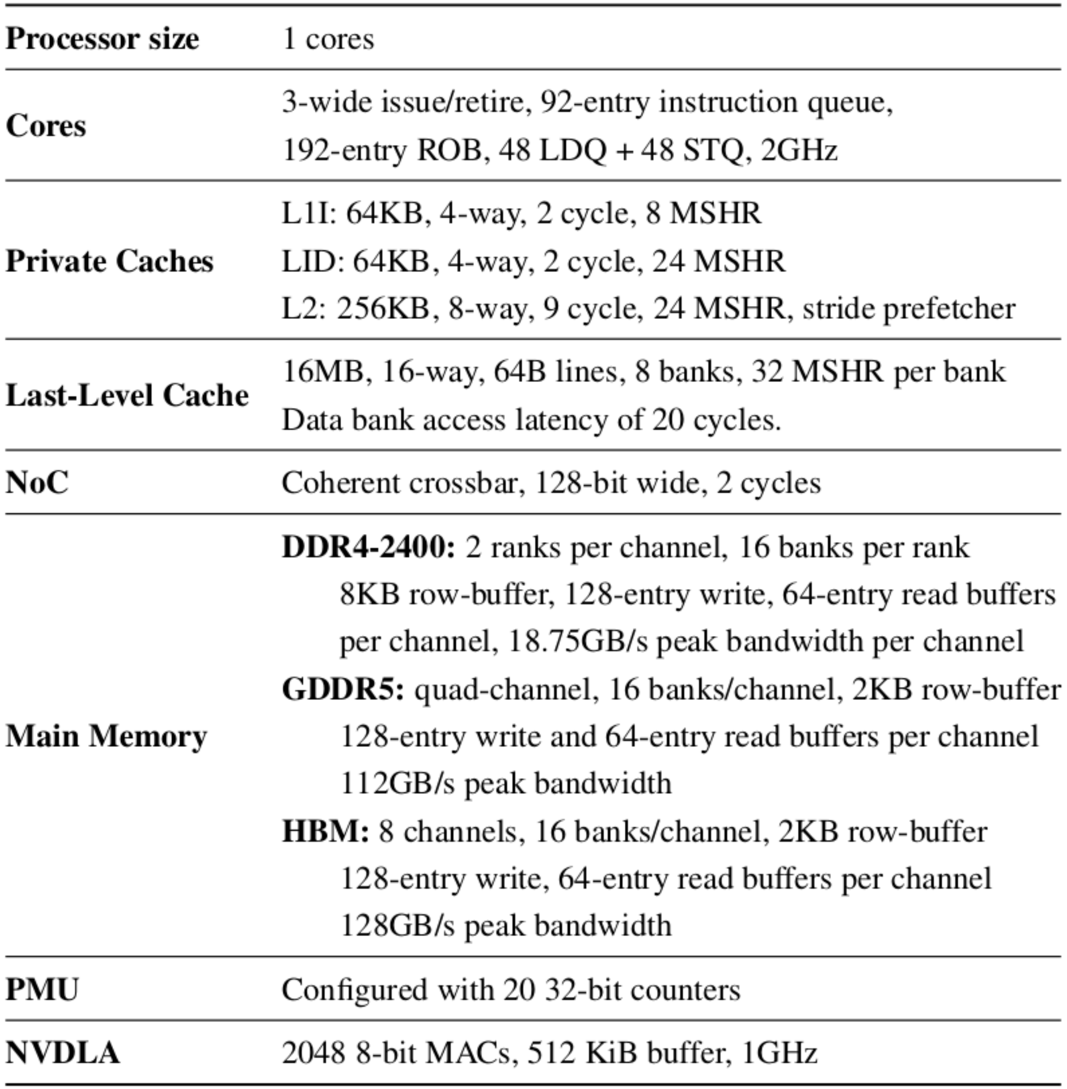
1. Bridge between Full-System Simulators and Verilator
    1. Gem5+Verilator focusing on FPGA → **PAAS**
    2. **Muli2Sim+Verilator** focusing on FPGA

2. **SynFull**: Synthetic traces made with gem5
    1. Markov Chains
    2. Clustering techniques to group phases of applications

Extra ch3: Methodology

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

Methodology Second Part



Processor size	1 cores
Cores	3-wide issue/retire, 92-entry instruction queue, 192-entry ROB, 48 LDQ + 48 STQ, 2GHz
Private Caches	L1I: 64KB, 4-way, 2 cycle, 8 MSHR LID: 64KB, 4-way, 2 cycle, 24 MSHR L2: 256KB, 8-way, 9 cycle, 24 MSHR, stride prefetcher
Last-Level Cache	16MB, 16-way, 64B lines, 8 banks, 32 MSHR per bank Data bank access latency of 20 cycles.
NoC	Coherent crossbar, 128-bit wide, 2 cycles
Main Memory	**DDR4-2400:** 2 ranks per channel, 16 banks per rank 8KB row-buffer, 128-entry write, 64-entry read buffers per channel, 18.75GB/s peak bandwidth per channel **GDDR5:** quad-channel, 16 banks/channel, 2KB row-buffer 128-entry write and 64-entry read buffers per channel 112GB/s peak bandwidth **HBM:** 8 channels, 16 banks/channel, 2KB row-buffer 128-entry write, 64-entry read buffers per channel 128GB/s peak bandwidth
PMU	Configured with 20 32-bit counters
NVDLA	2048 8-bit MACs, 512 KiB buffer, 1GHz

Table 3.2 Parameters for gem5+RTL full-system simulations.

Extra ch5: Cocnlusions

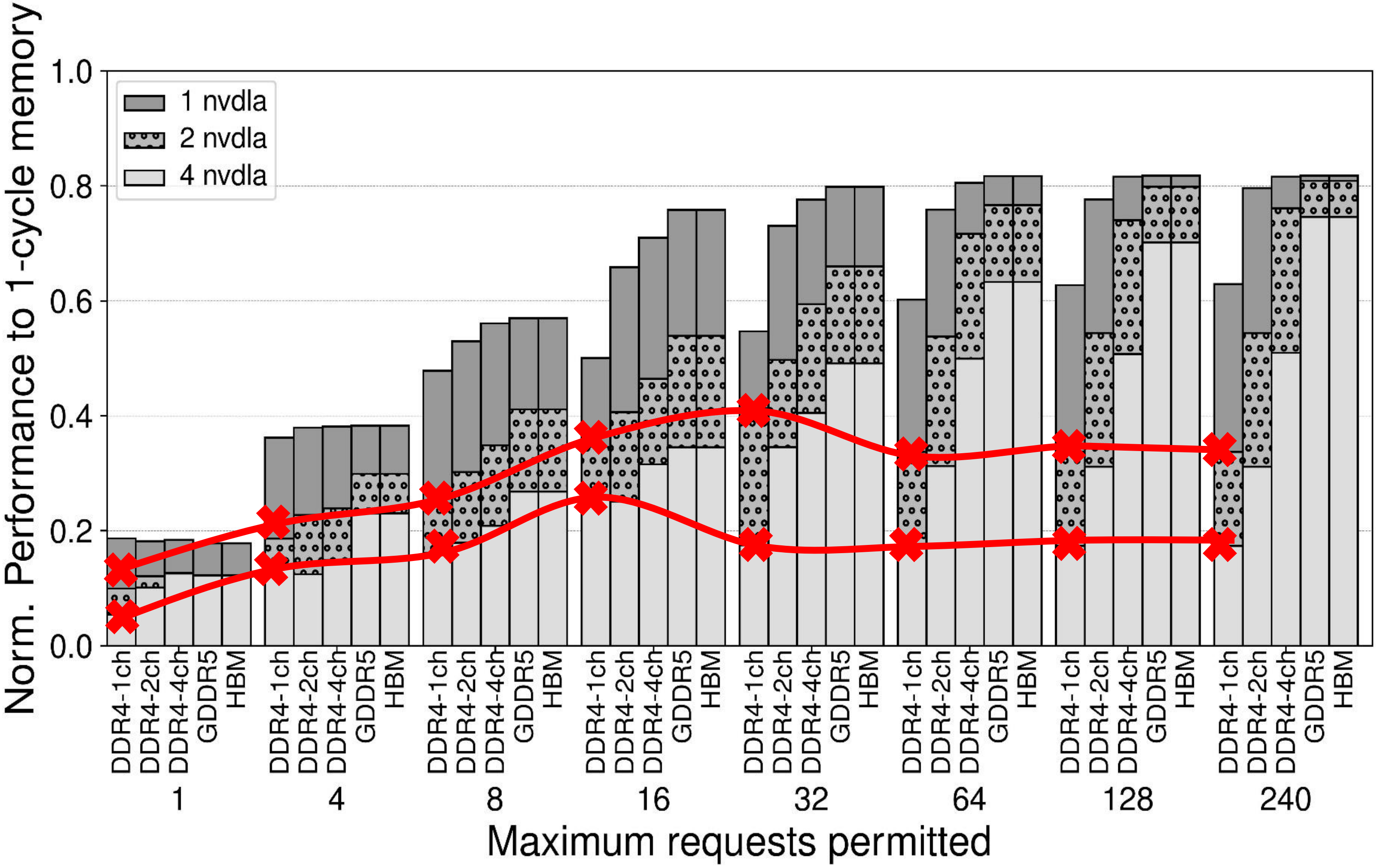
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

Evaluation NVDLA: Sanity3

- Same evaluation like before but with a more memory intensive app (also shorter)

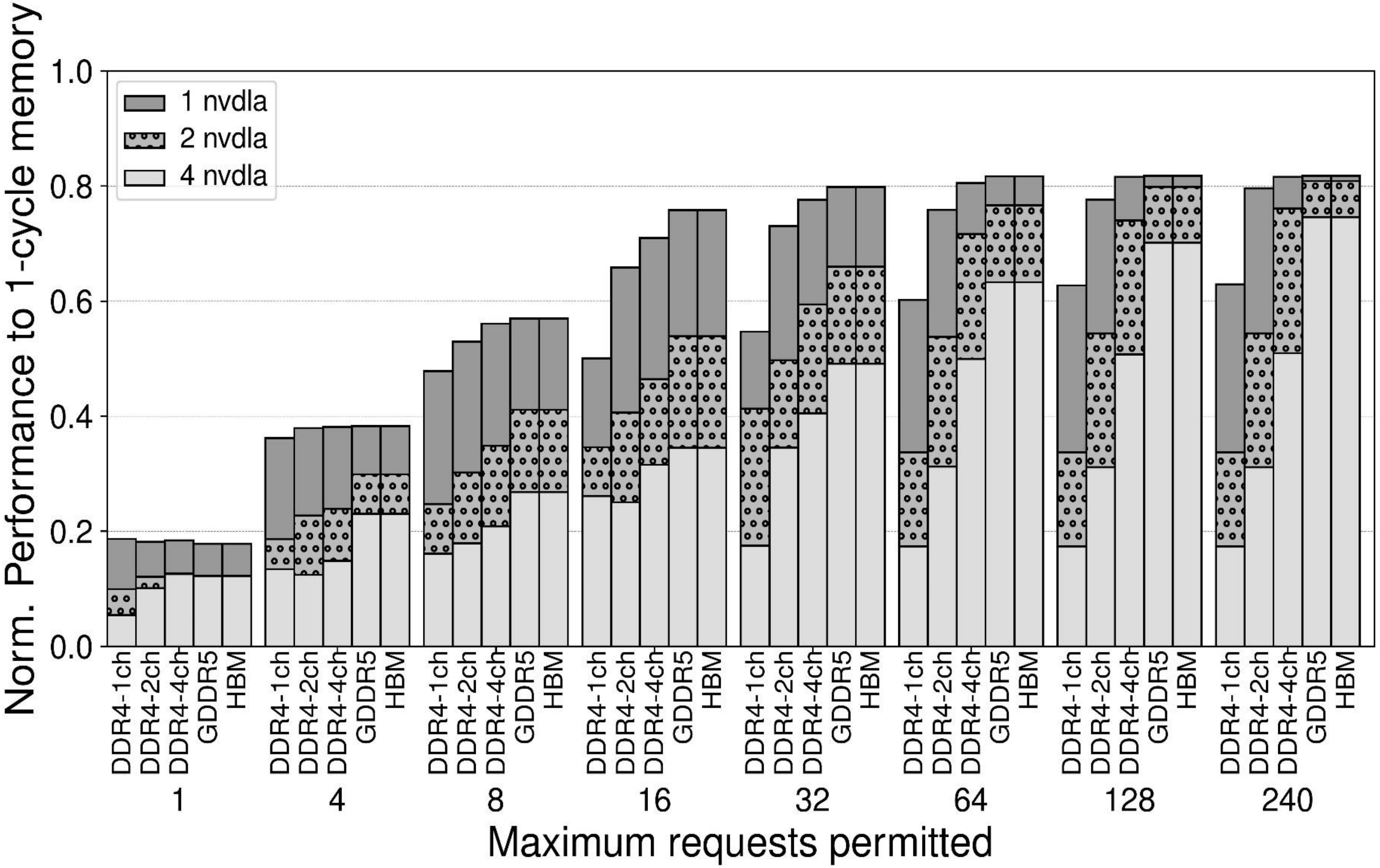
- Maximum number of requests is the key parameter again

- **Same situation of DDR4-1ch that cannot handle enough bw (also DDR4-2ch)**



Evaluation NVDLA: Sanity3

- **Same evaluation like before but with a more memory intensive app (also shorter)**
- Maximum number of requests is the key parameter again
- Same situation of DDR4-1ch that cannot handle enough bw (also DDR4-2ch)



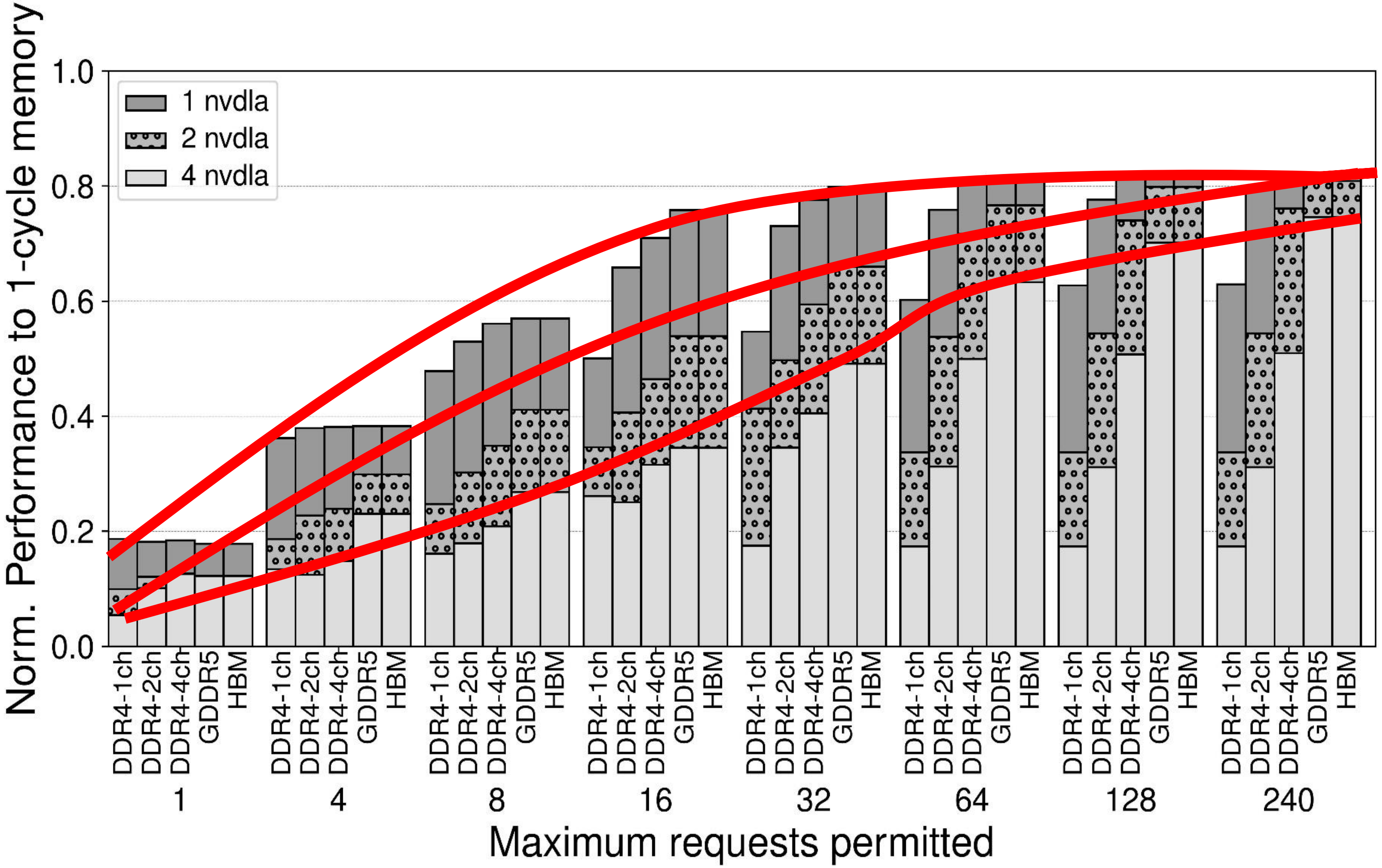
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Evaluation NVDLA: Sanity3

- Same evaluation like before but with a more memory intensive app (also shorter)

- **Maximum number of requests is the key parameter again**

- Same situation of DDR4-1ch that cannot handle enough bw (also DDR4-2ch)



Conclusions: gem5+RTL

- **Challenging Compilation, NVDLA:**
  - **NVDLA design is large (1 Million LUTS)**
  - **Needs more than 24 GB of RAM to create the C++ Model (300MB)**
  - **Depending on the optimization level, takes several hours**

- Tool suitable for SoC designers to make informed design decisions

- Increase the knowledge of Verilator

Conclusions: gem5+RTL

- Challenging Compilation, NVDLA:
  - NVDLA design is large (1 Million LUTS)
  - Needs more than 24 GB of RAM to create the C++ Model (300MB)
  - Depending on the optimization level, takes several hours
- **Tool suitable for SoC designers to make informed design decisions**
- Increase the knowledge of Verilator

Conclusions: gem5+RTL

- Challenging Compilation, NVDLA:
  - NVDLA design is large (1 Million LUTS)
  - Needs more than 24 GB of RAM to create the C++ Model (300MB)
  - Depending on the optimization level, takes several hours

- Tool suitable for SoC designers to make informed design decisions

- **Increase the knowledge of Verilator**

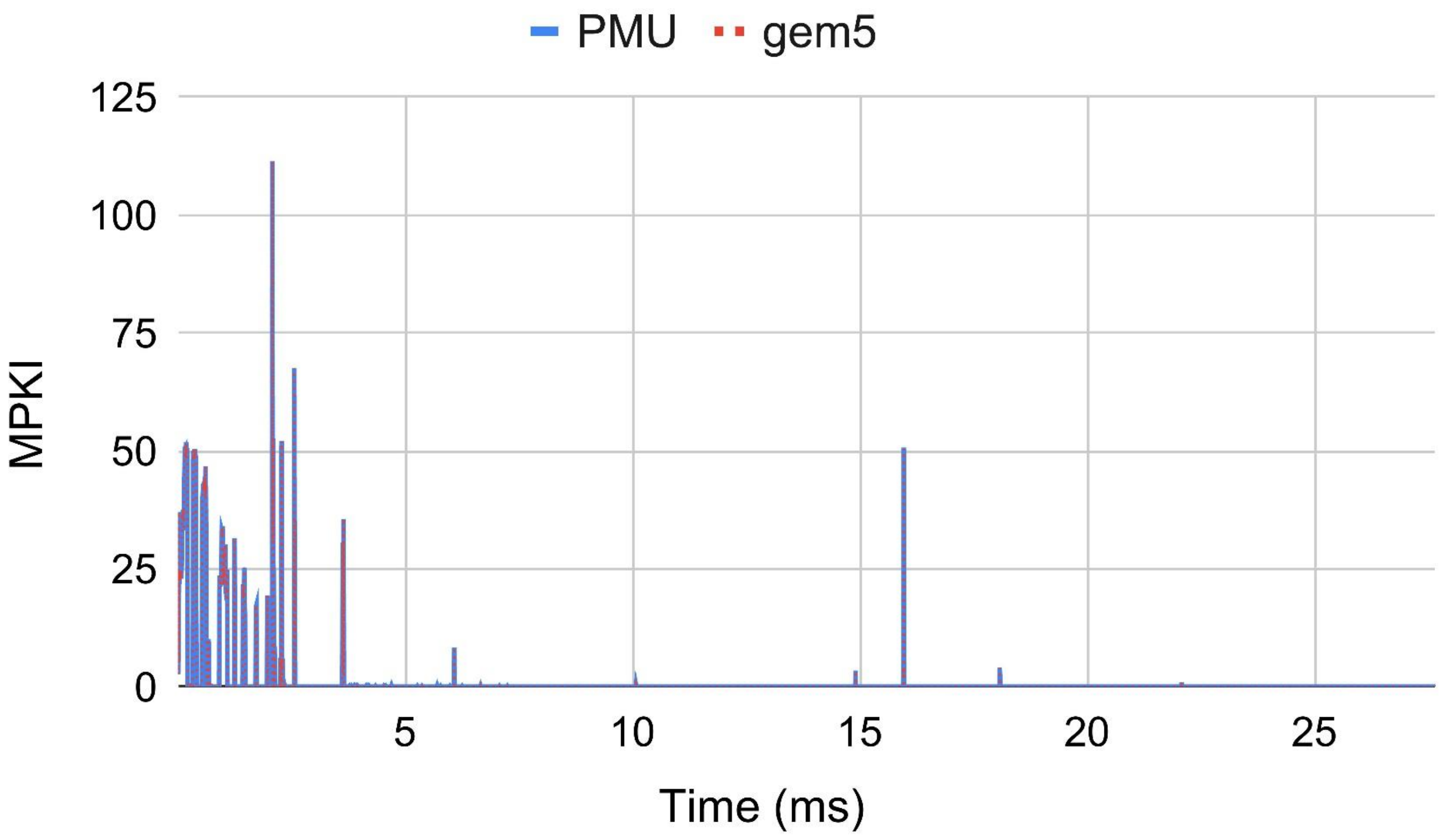
Future Work: gem5+RTL

- Improving the connectivity of the NVDLA with gem5, making use of a IOMMU
- Adding more RTL models and explore, for example, interesting re-programmable hardware that can be placed on the pipeline
- Add more features to the framework, for example, allow checkpointing of RTL models connected to the regular checkpoints of gem5
- Make a better study of which optimizations can be applied to Verilator to improve the final performance of the generated C++ model
- Add more memory models like scratchpads to offer more flexibility
- Add support for VHDL, the other well-known RTL language used in industry

Extra ch5: PMU

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

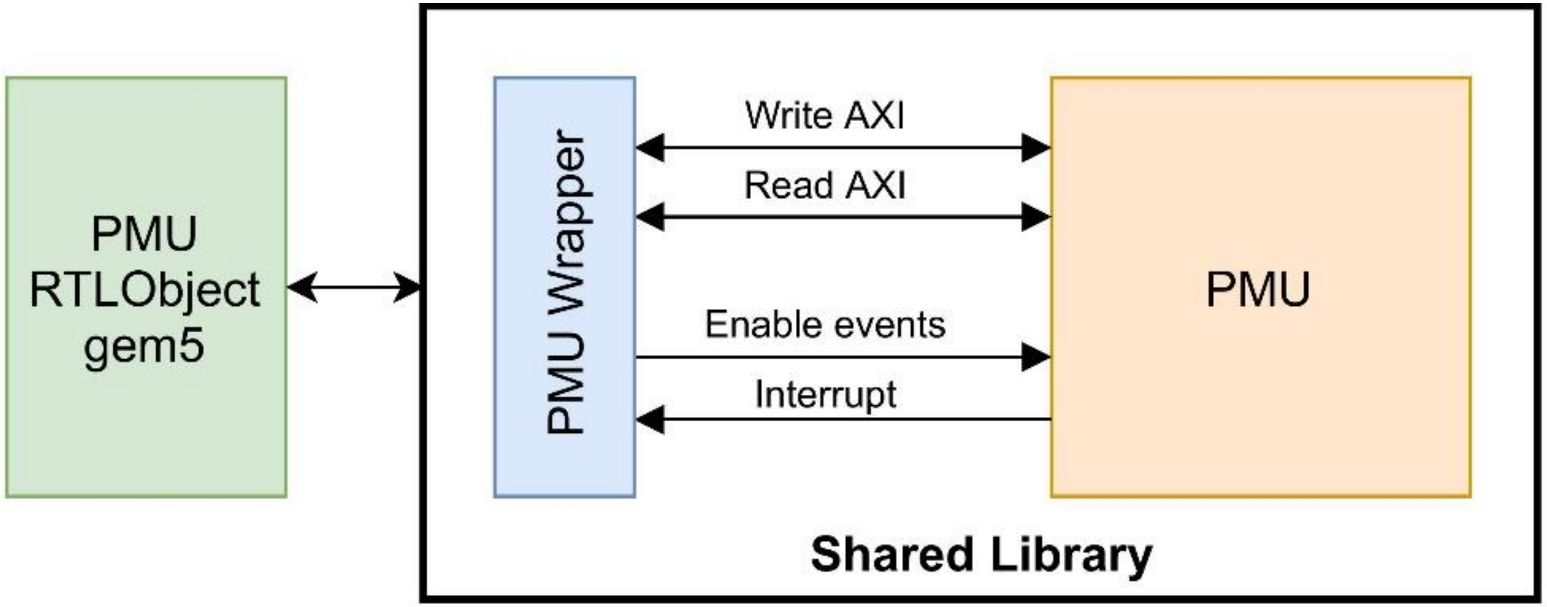
Evaluation PMU: MPKI



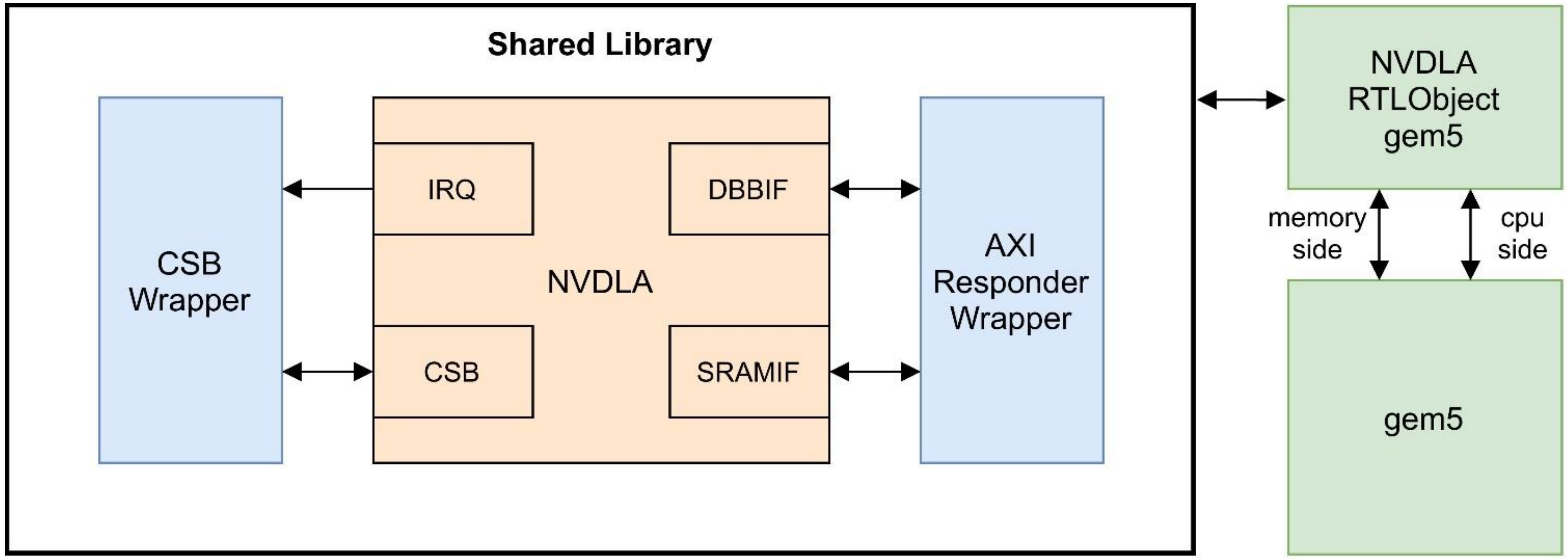
Extra ch5: NVDLA

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

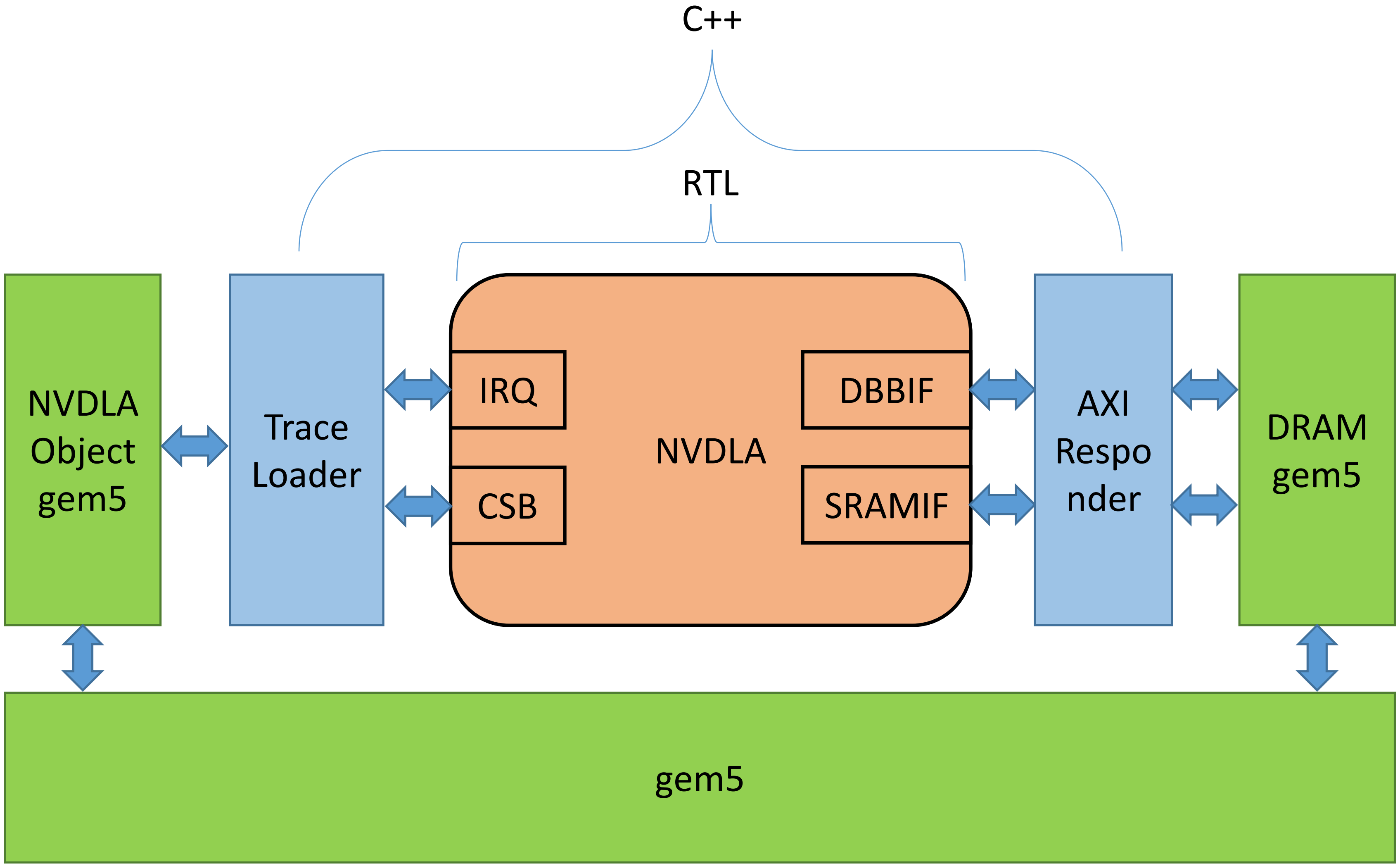
Use Cases: PMU Connection



Use Cases: NVDLA Connection



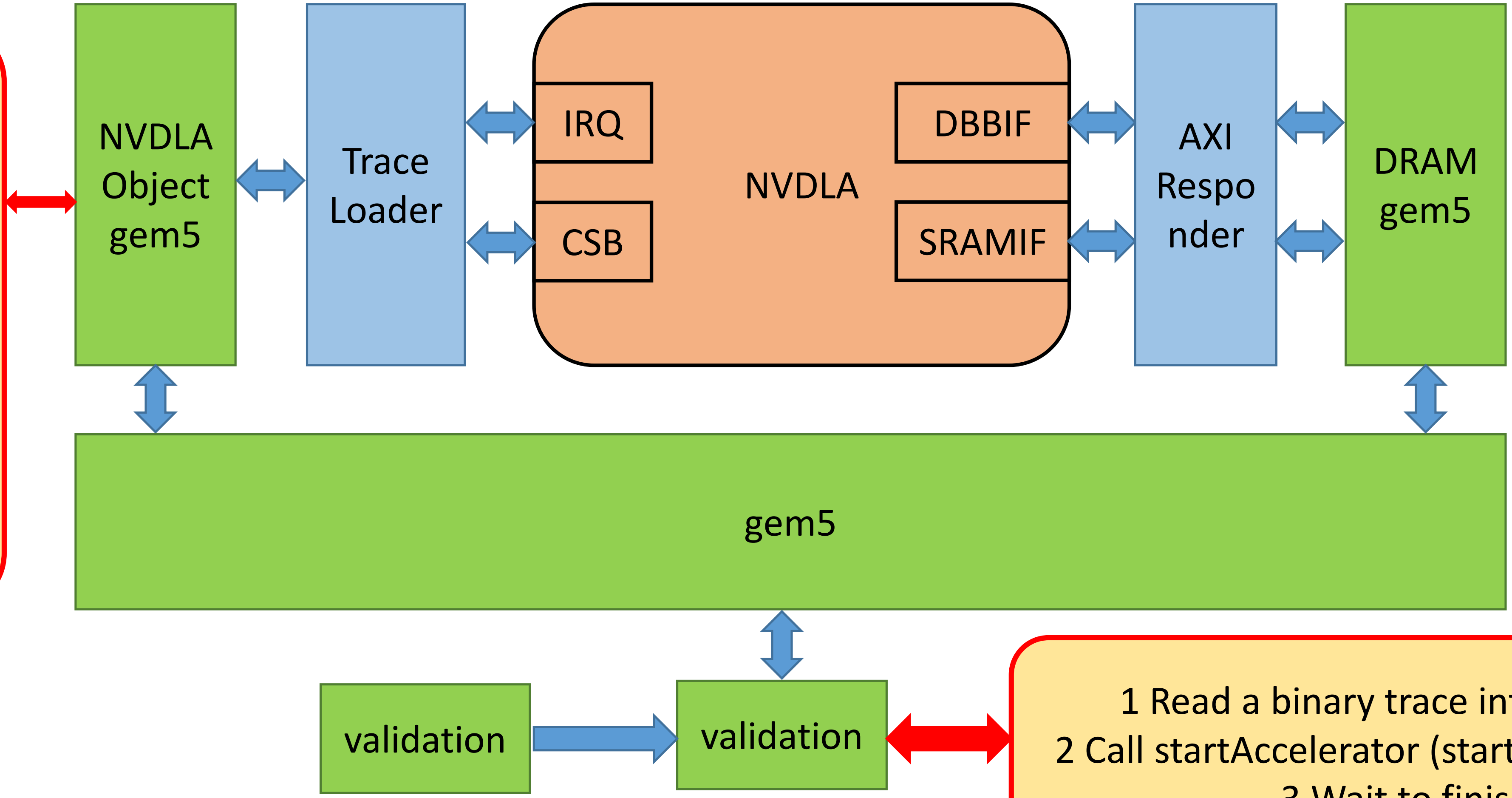
NVDLA inside gem5



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

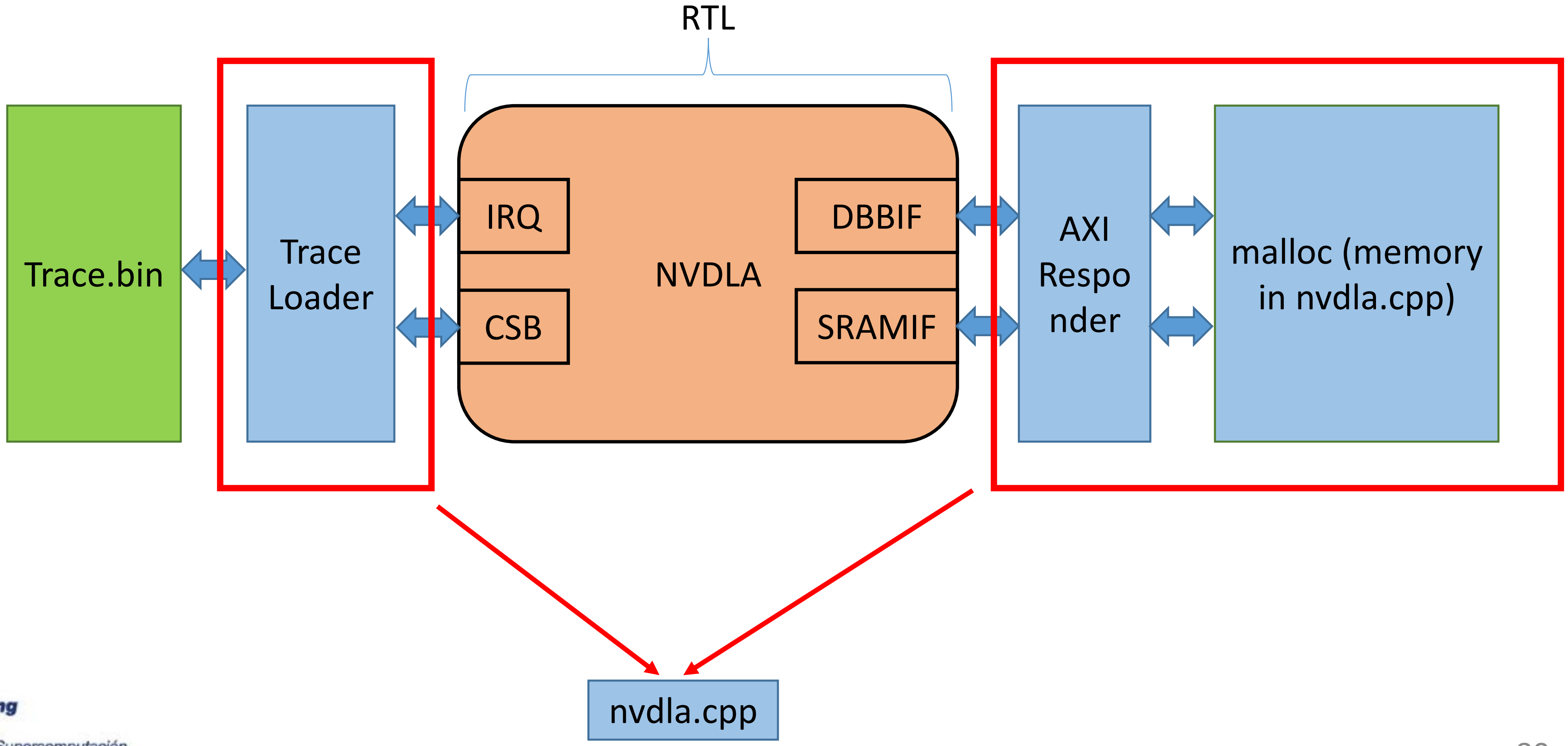
NVDLA inside gem5

1 Get all the data through memory petitions
2 Load the trace into the Traceloader
3 When there is an AXI write to DRAM, do it atomicly or functioanly



1 Read a binary trace into memory
2 Call startAccelerator (start @, # of bytes)
3 Wait to finish

NVDLA Testbench Verilator



Extra ch5: NVDLA Verilator Optimization

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Optimizing NVDLA Verilator Model

Trace (No optz)	Cycles	Sim time	Sim speed
Sanity3	10k	46.18 s	217.39 Hz
GoogleNet	49k	392.08 s	124.97 Hz
AlexNet	158k	1459.79 s	108.23 Hz

Traces (Os)	Cycles	Sim time	Sim speed
Sanity3	10k	18.5 s	541.1 Hz
GoogleNet	49k	103 s	472.9 Hz
AlexNet	158k	363 s	458 Hz

Compilation time Baseline: ~30 min
Compilation Time Os : ~2h
Compilation Time O3: It failed needing more than 16 GB, TODO

Optimizing NVDLA Verilator Model

Traces (Os)	Cycles	Sim time	Sim speed
Sanity3	10k	18.5 s	541.1 Hz
GoogleNet	49k	103 s	472.9 Hz
AlexNet	158k	363 s	458 Hz

Can we do better?

- O3 → **High compilation time** and **RAM** requirements **> 16GB**
- **Threads** option in Verilator
- **Verilator** is very sensitive to **UNOPT** and warnings, code of NVDLA has lots of warnings

However

- NVDLA needs **2M LUTS**, only fits on the highest FPGA in the market by Xilinx, which takes up to 82% of capacity of VU-440 (2018)
- Meaning NVDLA is huge.

Extra ch5: NVDLA Traces

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

AVAILABLE TRACES

- Basic sanity tests
  - **sanity0** - basic register write and compare read-back value
  - **sanity1** - memory copy test using bdma (dbb to dbb), test ends using register polling
  - **sanity2** - sanity1 waiting on interrupts instead of register polling
  - **sanity3** - convolution test, test ends using register polling and compares output mem region to determine passing
  - **sanity3_cvsram** - convolution test, uses cvsram path instead of dbb, test ends using register polling and compares output mem region to determine passing
- Short single function tests using dbb
  - **conv_8x8_fc_int16**
  - **pdp_max_pooling_int16**
  - **sdp_relu_int16**
- Long layer tests
  - **googlenet_conv2_3x3_int16** - uses cvsram, 30 min runtime
  - **cc_alexnet_conv5_relu5_int16_dtest_cvsram** - uses cvsram, 156 min runtime

Extra ch5: NVDLA Spec
from NVIDIA Documentation

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

NVDLA

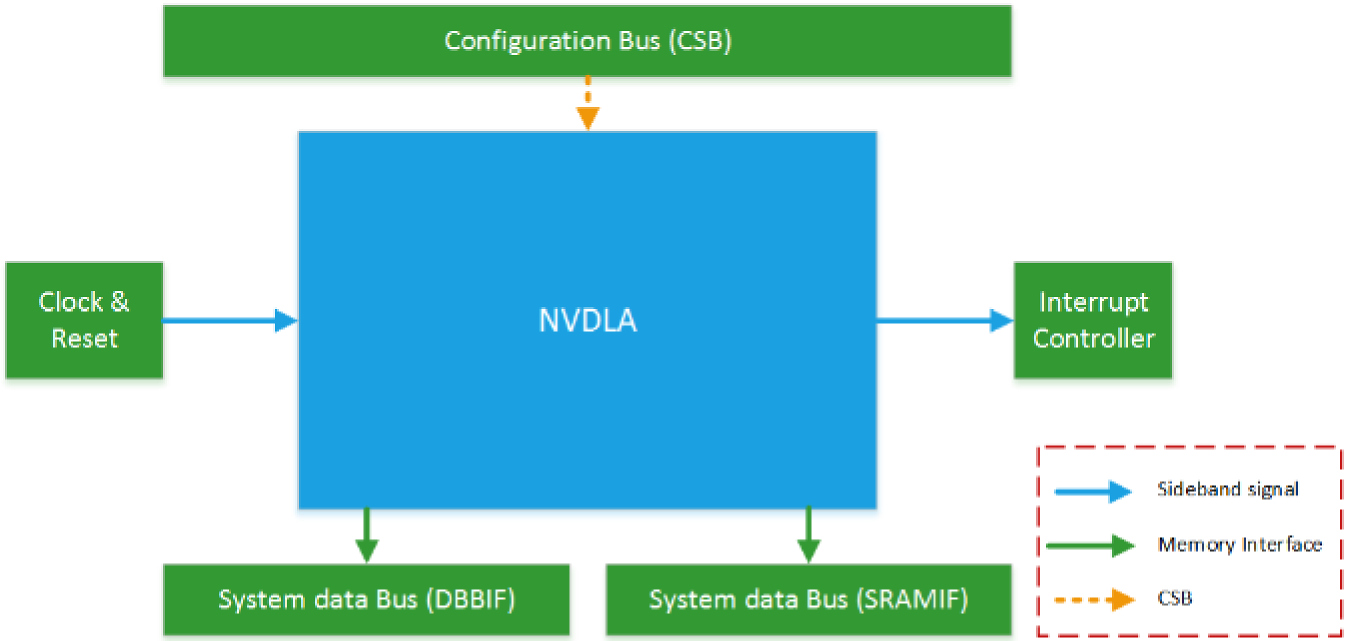


Image taken from [1]

- CSB: Commands
- IRQ: When a task finishes interrupt
- DBB: System memory

NVDLA Internal Block Diagram

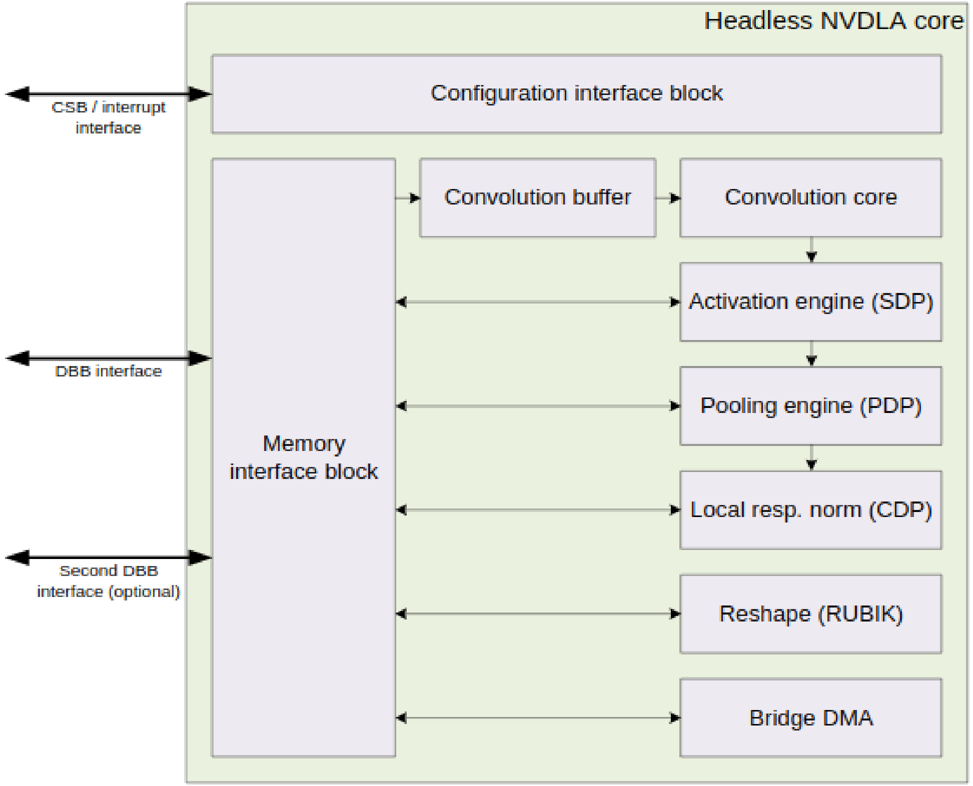


Image taken from [2]

- Each Block/Engine is separate and independently configurable
- Scheduling operations for each unit are delegated to a co-processor or CPU

Software

- NVDIA offers two tools:
  - **Compilations tool:** Convert existing models into a NVDLA usable model.
  - **Runtime environment:** Run-time software to load and execute networks on NVDLA.

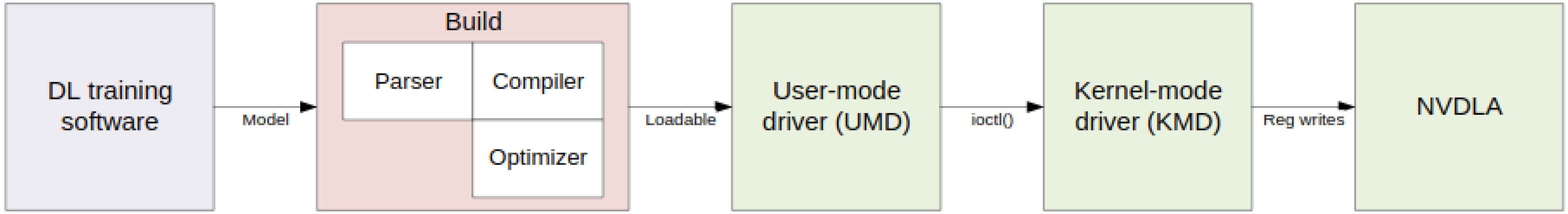
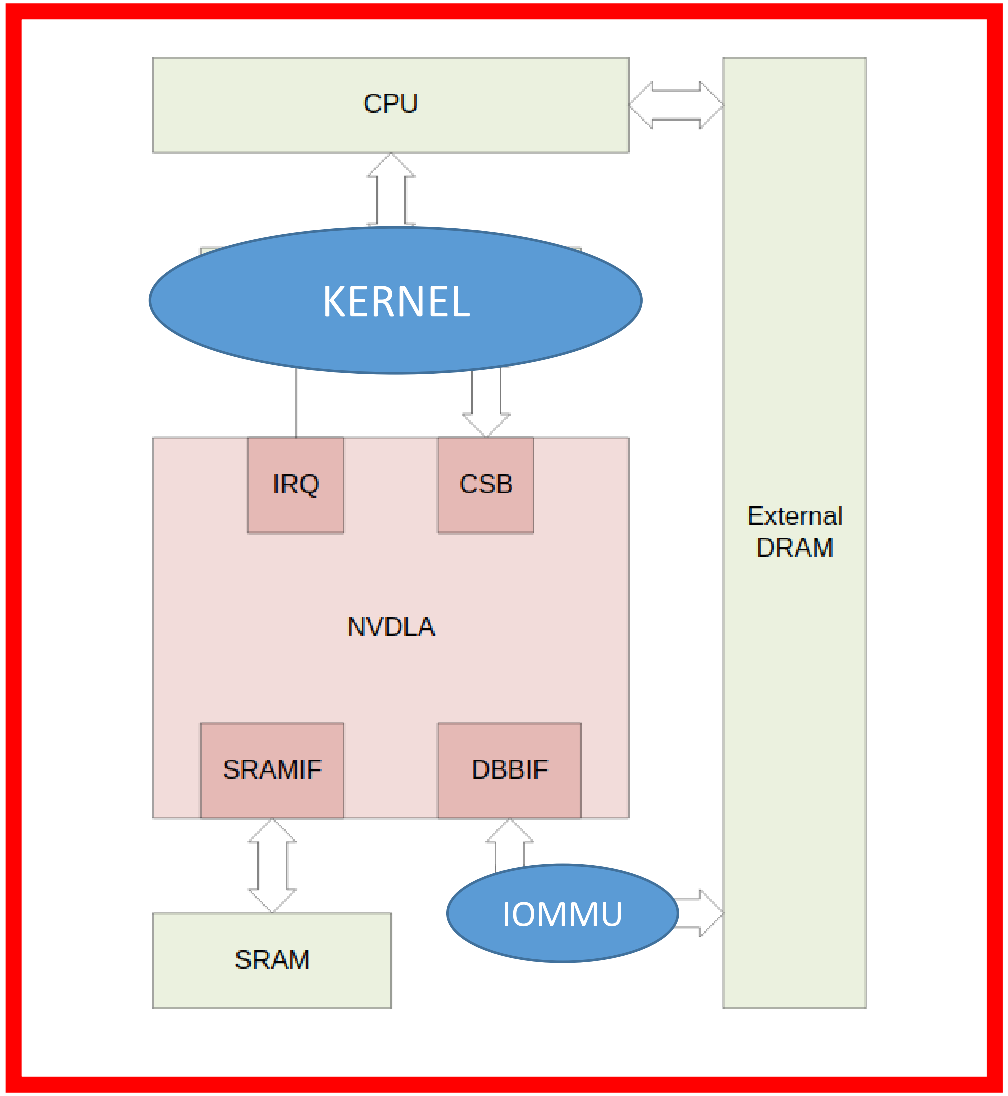


Image taken from [5]

NVDLA real situation



Barcelona Supercomputing Center Centro Nacional de Supercomputación

Images taken from [2]

Contributions and Publications

- **Guillem López-Paradís**, Adria Armejach, Miquel Moreto, ***gem5+RTL: A Framework to Enable RTL Models Inside a Full-System Simulator,*** **Paper Under Review on DATE 21'**

- **Guillem López-Paradís**, Adria Armejach, Miquel Moreto, ***Enable RTL models inside the gem5 simulator***, **ACACES 19**: Advanced Computer Architecture and Compilation for Embedded Systems 2019 Poster Abstracts, Fiuggi, Italy, 2019

Contributions and Publications

- **Guillem López-Paradís**, Adria Armejach, Miquel Moreto, ***gem5+RTL: A Framework to Enable RTL Models Inside a Full-System Simulator,*** **Paper Under Review on DATE 21'**

- **Guillem López-Paradís**, Adria Armejach, Miquel Moreto, ***Enable RTL models inside the gem5 simulator***, **ACACES 19**: Advanced Computer Architecture and Compilation for Embedded Systems 2019 Poster Abstracts, Fiuggi, Italy, 2019