



HIPEAC2021 TUTORIAL



SESAM/VPSIM: FAST AND HIGHLY-CONFIGURABLE SIMULATION FRAMEWORK FOR THE EPI PROCESSOR

Lilia Zaourar, Tanguy Sassolas

Mohamed Benazouz, Gabriel Busnot, Amir Charif, Fatma Jebali, Oumaima Matoussi, Rania Mameesh, Salah Eddine Saidi, Nicolas Ventroux, Arief Wicaksana

CEA LIST

Lilia.zaourar@cea.fr , Tanguy.sassolas@cea.fr

sesam-vpsim@cea.fr

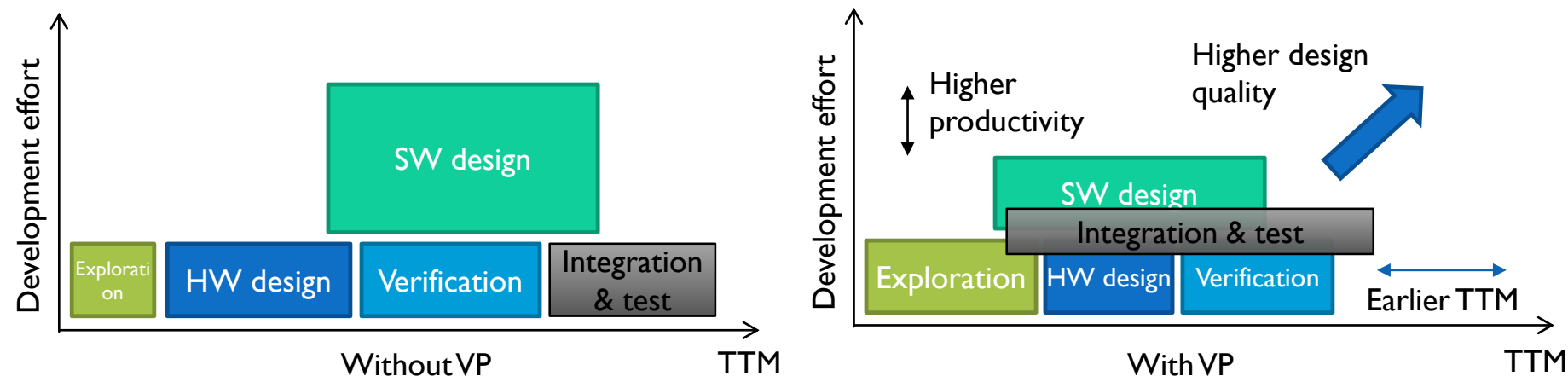


INTRODUCTION TO VP AND EPI NEEDS

BENEFITS OF ARCHITECTURE VIRTUAL PROTOTYPING

- Virtual prototyping can help designing complex SoCs and Electronic Systems
 - Allow fast design exploration and improve design quality and performance while encompassing power, reliability ...
 - Bring a virtual HW platform to SW designers to start development in complex environments (e.g. external sensor data) prior to silicon
 - Provide unlimited SdK seats, and Continuous Integration capabilities throughout product cycle

- VPs parallelize design phases and increase design productivity



OUTLINE

- VPSim for European Processor Initiative
- VPSim background overview
- Recent EPI evolutions VPSim 0.3.0
 - Cache
 - Noc
 - Numerical results
 - Co-simulation for verification of safety critical applications
- Conclusion & perspectives
- References
- Video demonstration

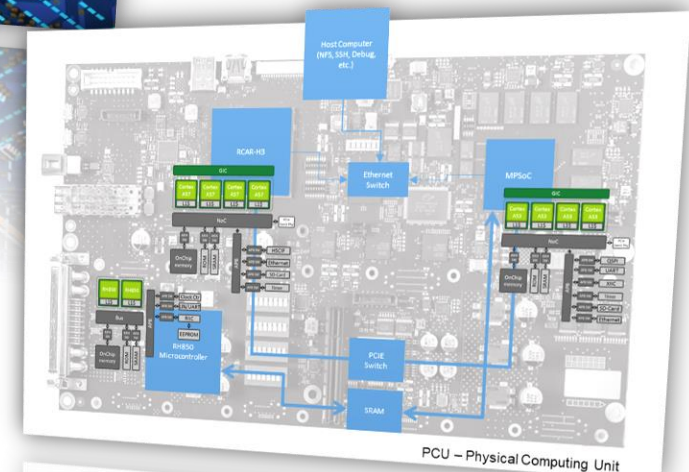
SESAM/VPSIM



- A virtual prototyping framework
 - Early software development
 - Performance profiling and debug
 - Design space exploration
 - Hardware validation
 - CPS prototyping
- Easy interfacing thx to SystemC/TLM 2.0 and FMI
- Fast platform description with Python
 - With large and flexible IP portfolio
- Rapid simulation able to run full software stacks
 - From hypervisor, to full-fledged applications with standard debugging features

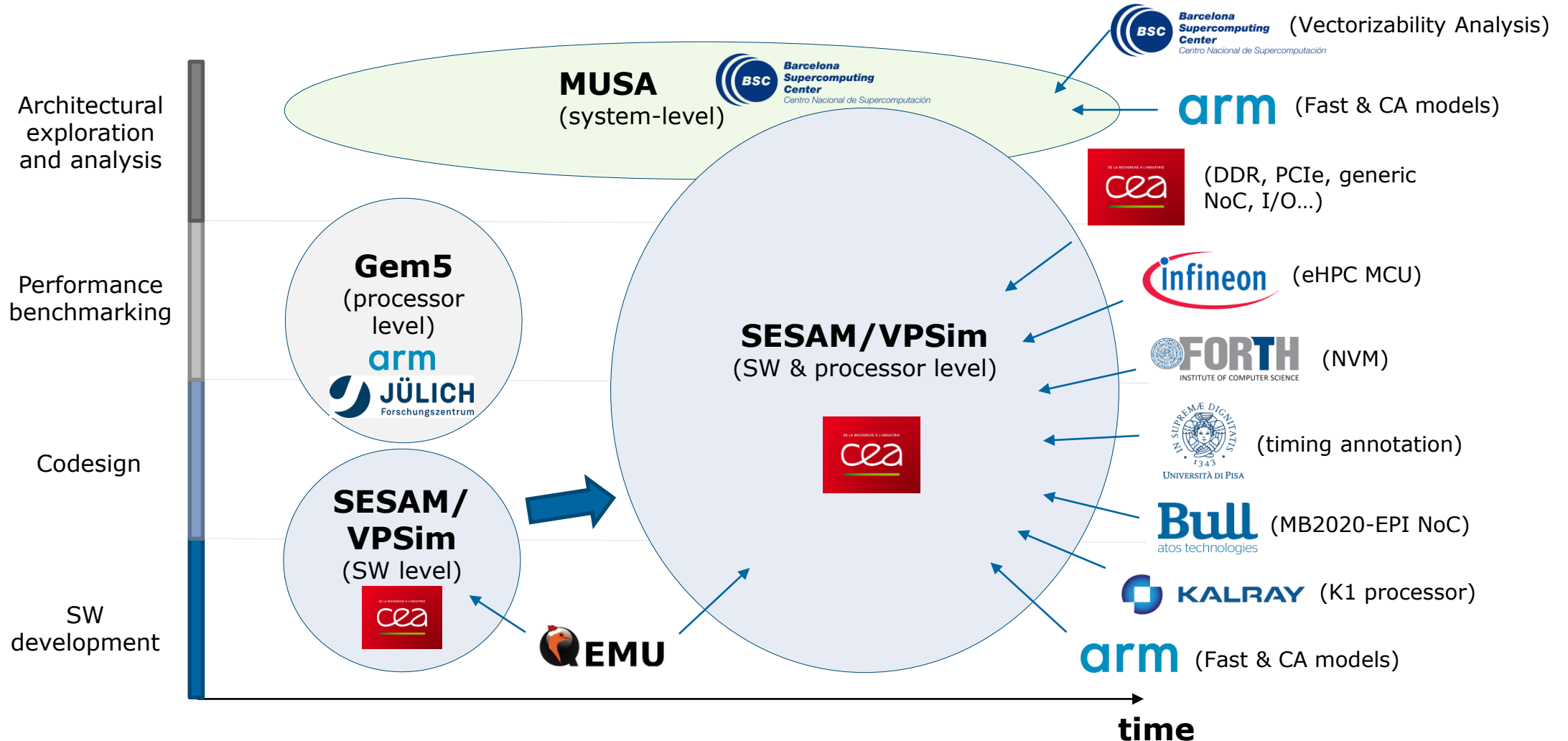


From SoC to CPS design



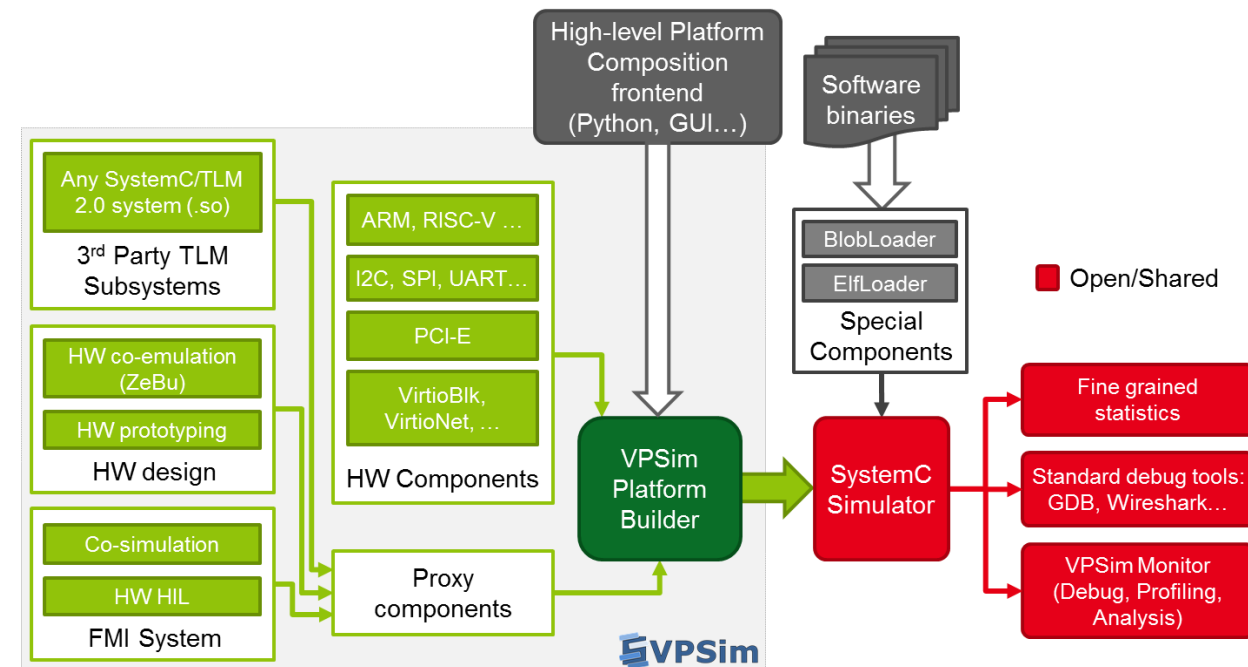
VPSim FOR EUROPEAN PROCESSOR INITIATIVE

POSITIONING OF SIMULATION ACTIVITIES



EPI VIRTUAL PLATFORM USING SESAM/VPSIM

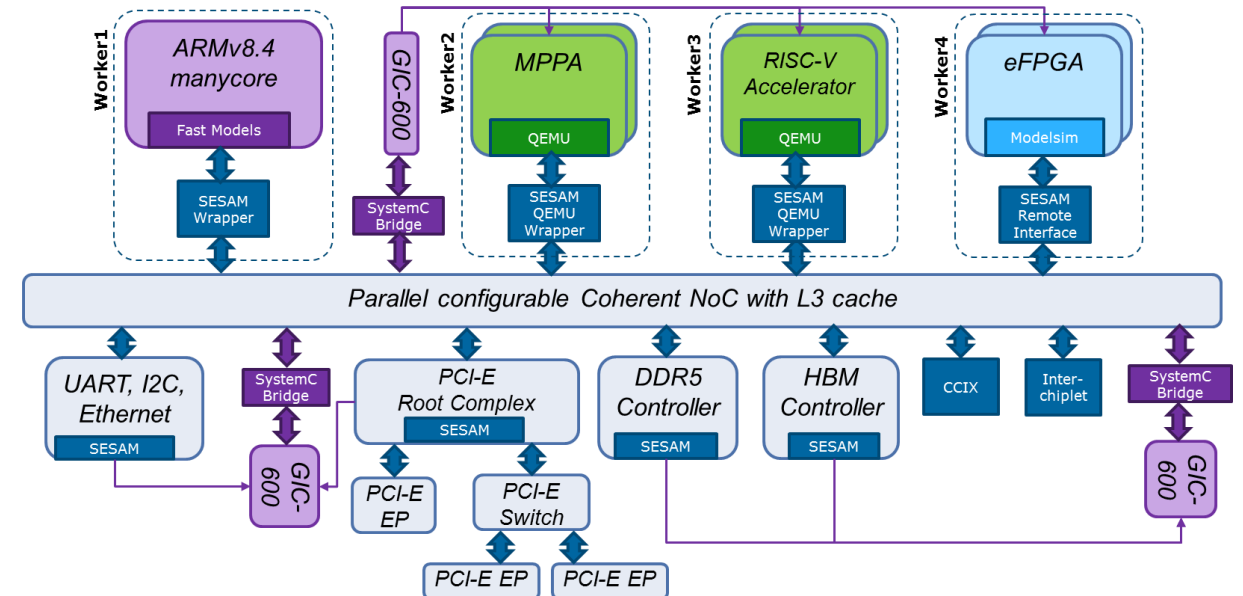
- Virtual Environment
 - Software development
 - Hardware/Software Co-Design
- Help software teams get started
 - Early testing and preparation of software environment and test suites
 - Support all levels from BIOS to OpenMPI
 - EPI processor specifications
 - Early identification of performance issues



EPI VIRTUAL PLATFORM USING SESAM/VPSIM

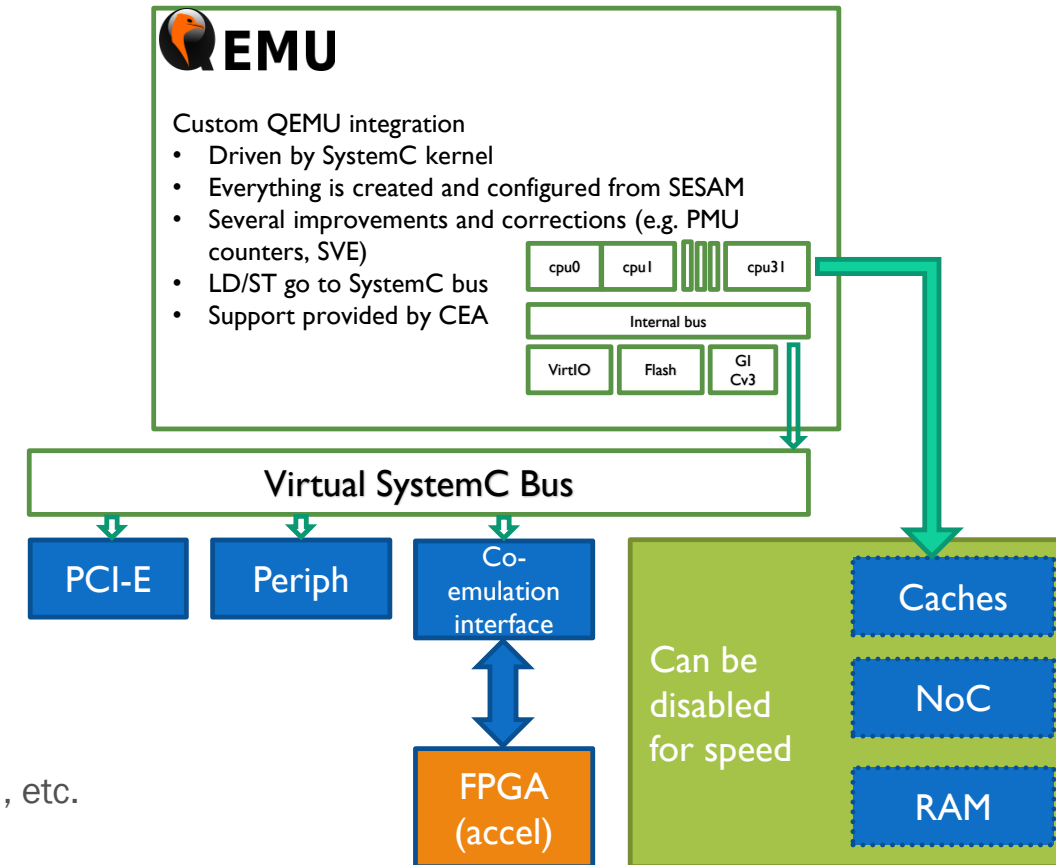
Challenges

- Highly complex heterogeneous platform
 - Manycore processor (GPP), accelerators (RISC-V, Kalray), Microcontrollers (Infineon)
- Require latest processor features to enable software development
 - ARM SVE (*Scalable Vector Extensions*)
 - ARM PMU (*Performance Monitoring Units*)
- Need for speed
(Full server OS and compilation environment)
- Need for accuracy
(focus on memory subsystem with NoC to identify bottlenecks)



GPP (GENERAL PURPOSE PROCESSOR) VIRTUAL PLATFORM

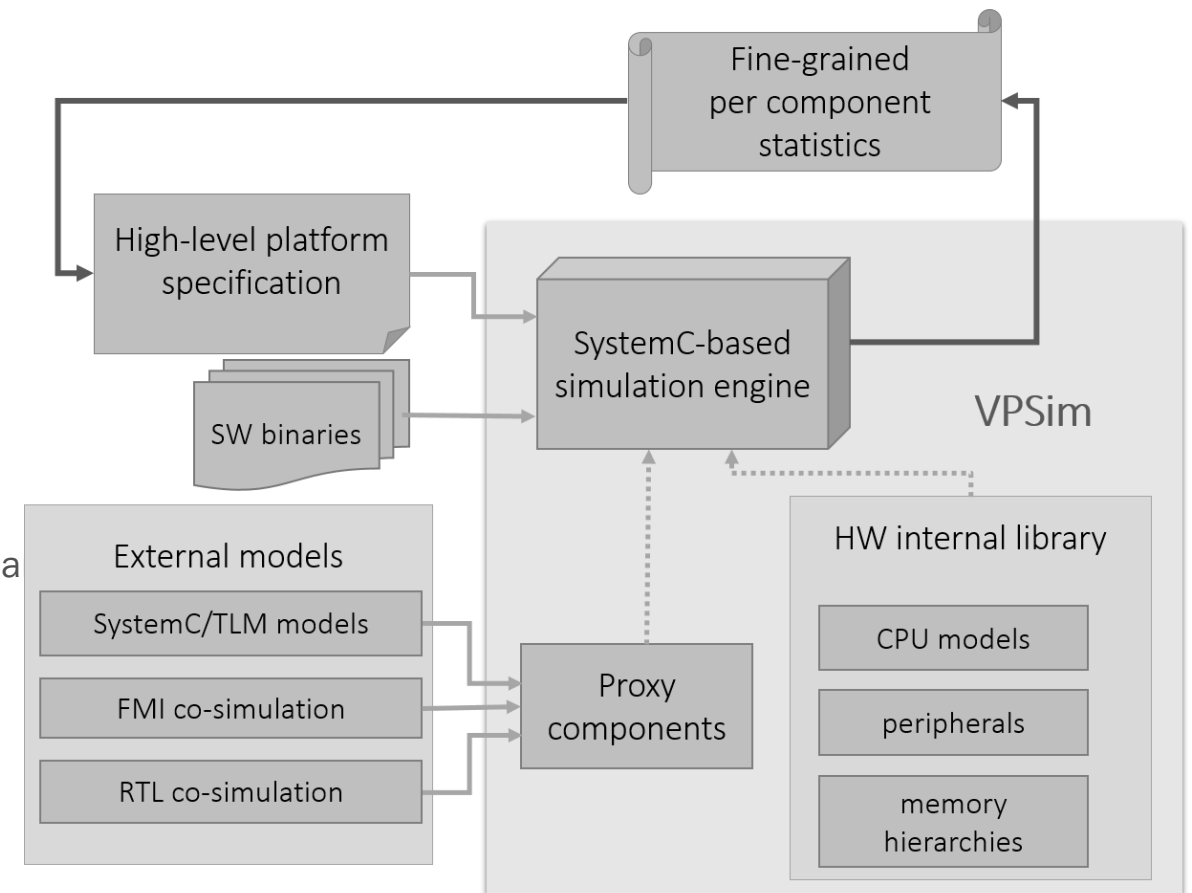
- Software development and co-design supporting complex software environments with improved timing model
 - Debian Linux with network connectivity and build environment
 - Coherent cache and NoC models
 - Custom execution counters for application profiling
 - Stats accessible online from PMU registers
- Imported latest upstream advances in QEMU
 - Performance improvement for large pages
 - PMU interrupts
- Collaboration with partners for integration of specific/needed features
 - OpenMPI for BSC, PMU for Fraunhofer & BSC, NVMe for FORTH, FMI for UNIP, etc.



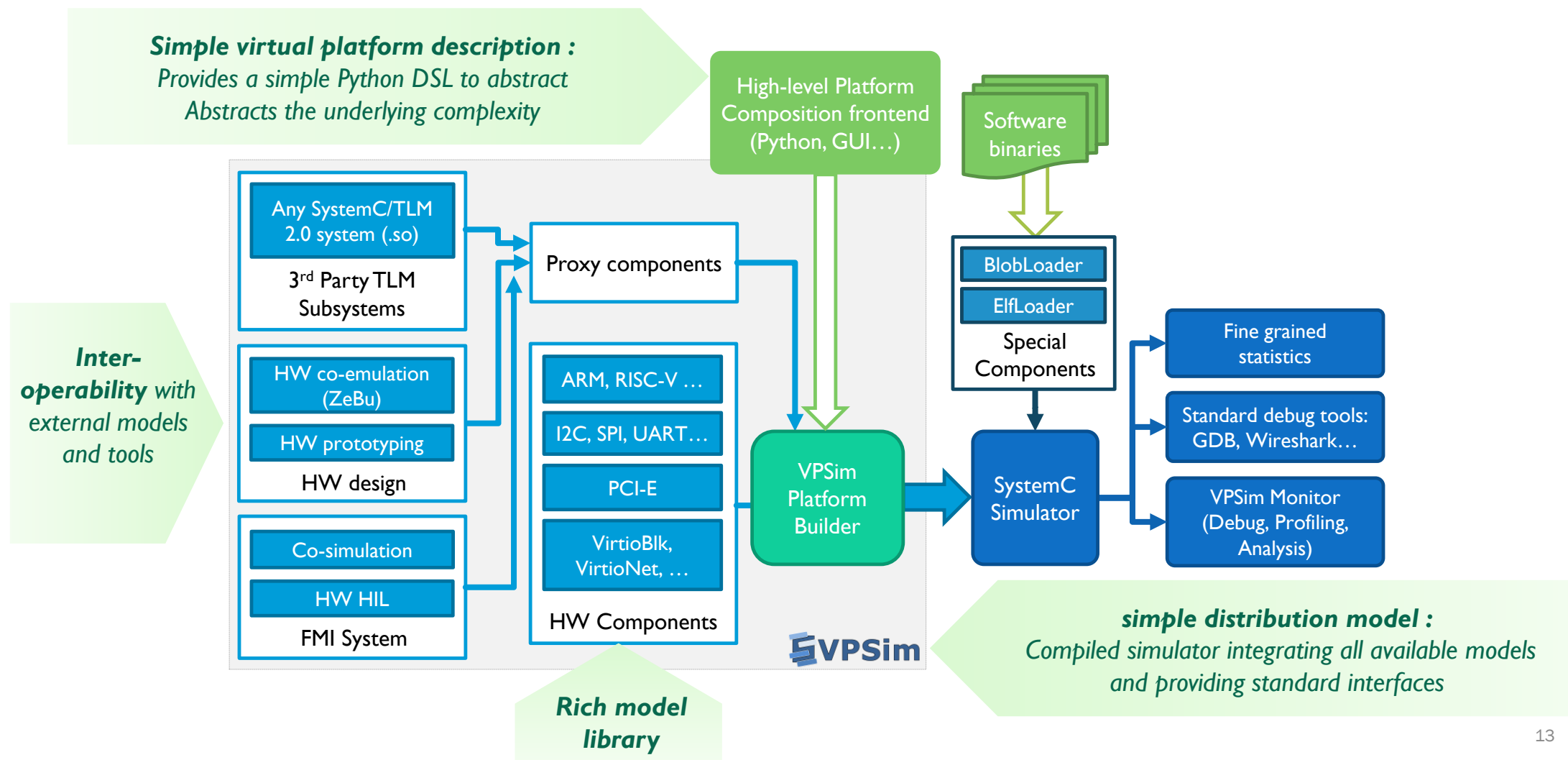
VPSim BACKGROUND OVERVIEW

VPSIM TOOL OVERVIEW

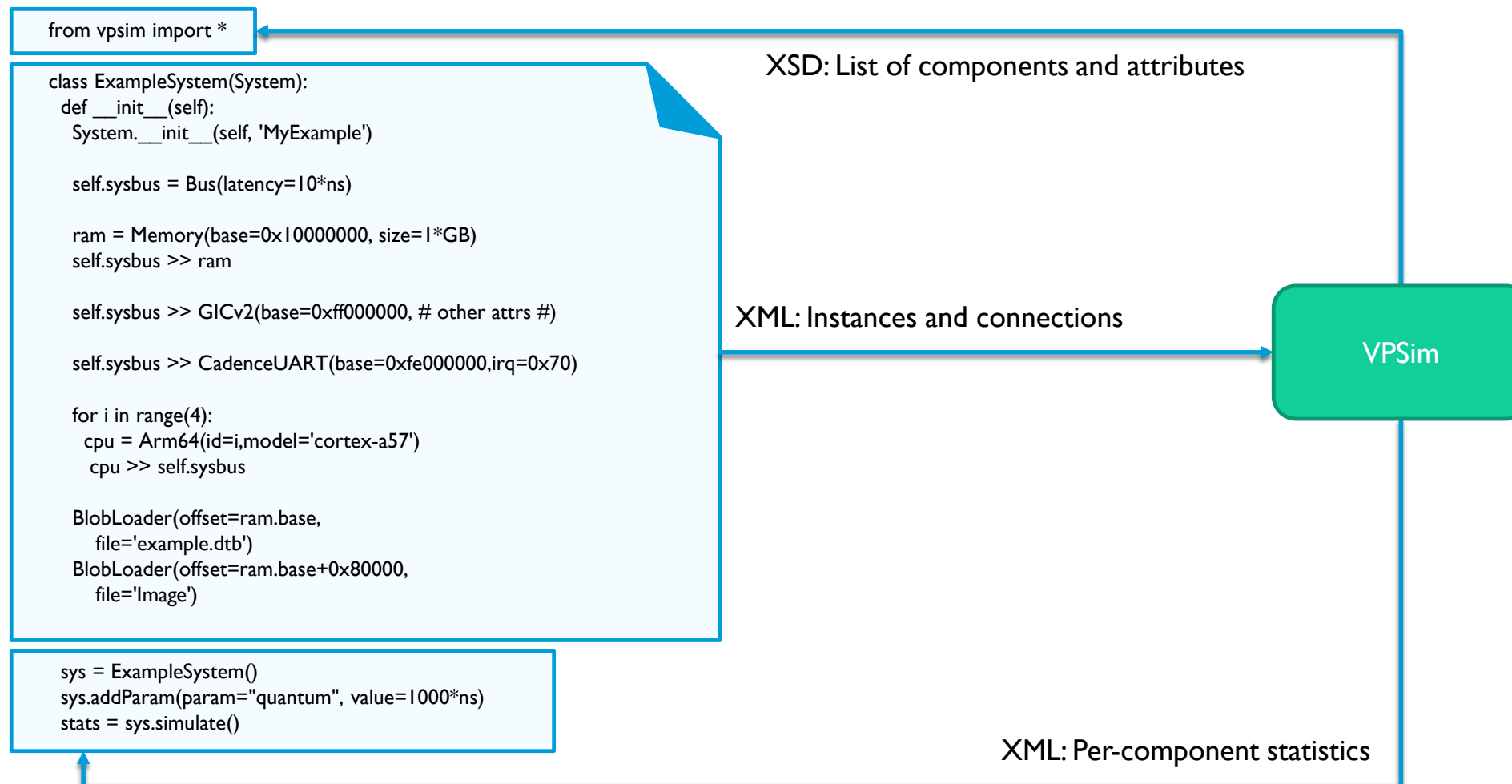
- SystemC/TLM-based virtual prototyping platform
- Internal models
 - CPU models provided by QEMU
 - Peripheral and memory hierarchy components developed from scratch
- External models
 - Third-party subsystems using many standard and non-standard interfaces
- Release v0.3.0
 - Configurable cache hierarchy with coherence
 - NoC model with contention



FAST PLATFORM SETUP

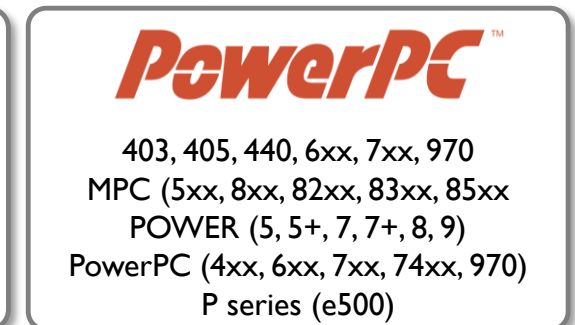
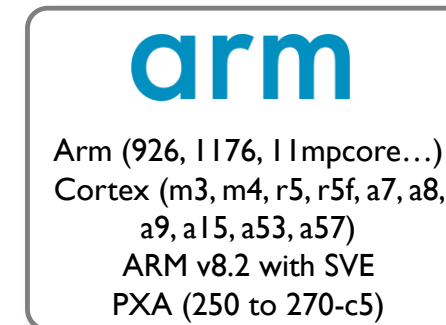


HIGH-LEVEL PLATFORM DESCRIPTION EXAMPLE



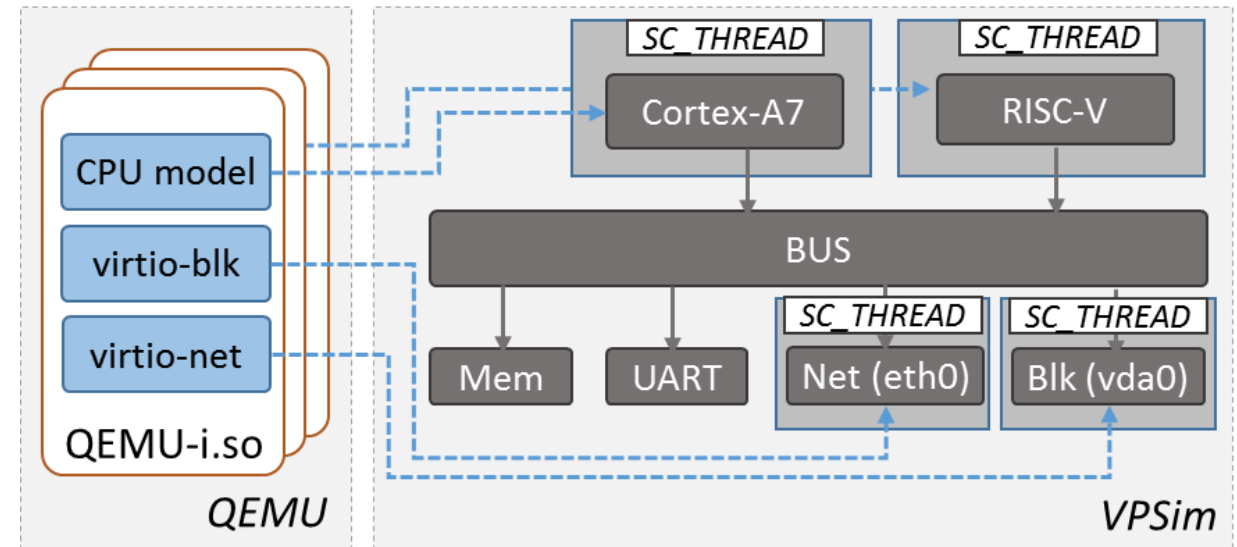
LARGE AND FLEXIBLE IP PORTFOLIO

- Built-in SystemC/TLM components
 - Memories : SRAM, DDR3 controller, Caches
 - Interconnects
 - Peripherals : UARTs, SPI, I2C, RTC, Timer...
 - Interrupt controllers
 - High-speed interfaces :PCI-Express host bridge, switches...
- Native support for QEMU as a model provider
 - CPUs
 - Everything in QEMU can be used in a SystemC environment
 - Network, block devices, GPUs, ARM GIC...
 - Always up-to-date with the latest versions
- 3rd party SystemC subsystems
 - Any SystemC/TLM subsystem can be imported
 - ARM Fast Models, Open Virtual Platforms, QBox...



QEMU INTEGRATION

- Use everything in QEMU, not just CPUs
 - Peripheral models
 - Virtualization with host systems (e.g. Ethernet)
- High-level instantiation of QEMU models
 - No need to know QEMU's internals
- Inline checking of memory access accuracy
 - No impact if DMI allowed (host ld/st instruction)
 - RAM accesses handled in SystemC otherwise



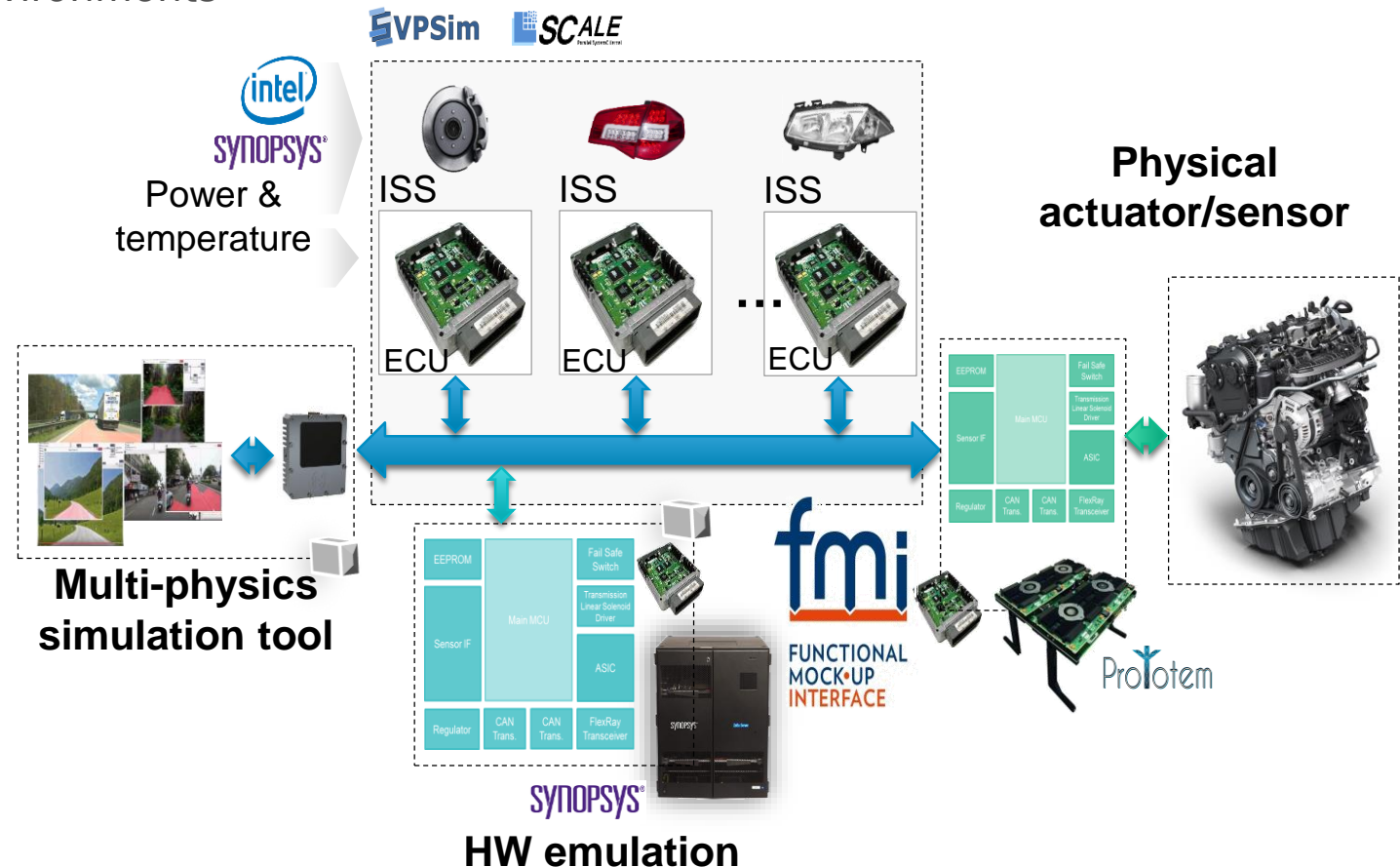
```
uart = ModelProviderDev(provider='qemu', model='pl011',
base=0xfe000000, irq=50)
```

```
# Connection to SystemC bus
sysbus >> uart
```

```
# QEMU's command line configuration
ModelProviderOpt(option='-gdb', value='tcp::2222')
```


INTERFACING WITH VARIOUS SIMULATION ENVIRONMENTS

- Co-simulation with FMI and RTL simulation environments
- Pervasive FMI 2.0 Co-simulation
 - Holistic system view encompassing any external simulator with FMI co-simulation
 - Allows integrating SESAM tools throughout design cycles in a model-driven environment
 - Allows cross domain analysis
- RTL co-simulation
 - Fast IP test environment setup
 - Significantly accelerates test cycles through adapted modelling level
 - RTL for Design Under Test (DUT)
 - VP for the rest



EXTERNAL SIMULATOR INTEGRATION

- Additional interface to co-simulate with peripherals implemented in external simulators :

```
#Python instantiation example
extsim = ExternalSimulator(
    base_address = 0x80000000,
    size = 0x40000000,
    lib_path = '/vpsim/ExternalSimulator.so',
    irq_n = 7,
    interrupt_parent = 'RiscV0' )
hh

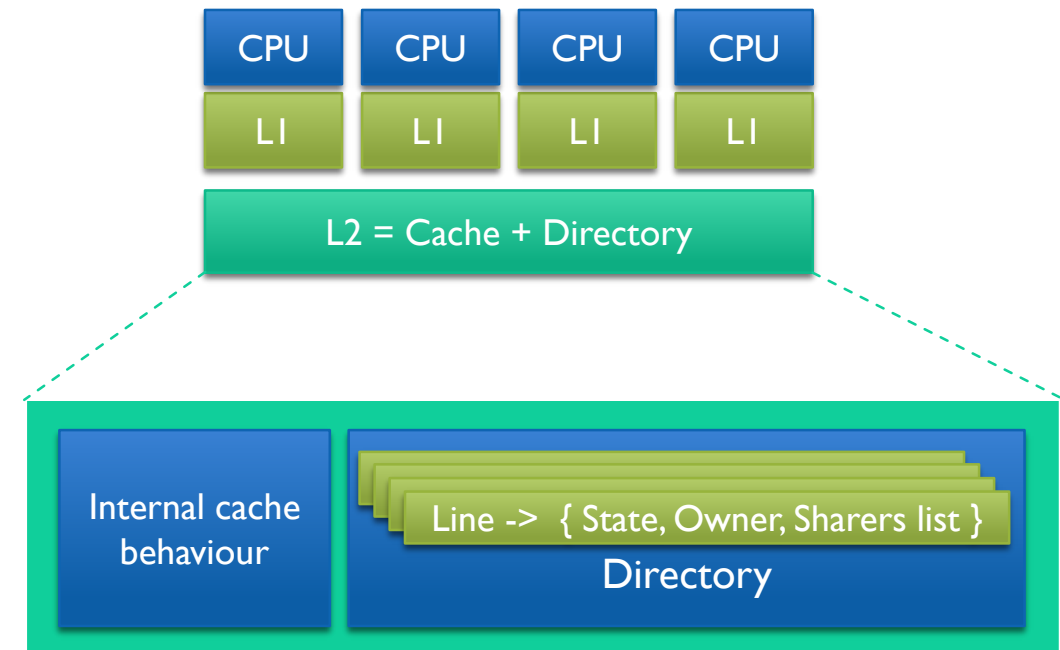
#connection to system bus
sysbus >> extsim
sysbus.n_out_ports+=1
```

- C API to wrap the external simulator :
 - Main execution thread launch :
 - void **run_simulator**(void)
 - Memory mapped data access
 - uint32_t **write_simulator** (uint64_t addr, unsigned size, uint64_t p_data)
 - uint32_t **read_simulator** (uint64_t addr, unsigned size, uint64_t p_data)
 - Provision of implemented callback functions in VPSim
 - IRQ managment :
 - void **register_irq_callback** (InterruptCb cb)
 - void **update_irq** (uint32_t line, uint32_t value)
 - Time managment
 - void **register_sync_callback** (SynchroCb cb)
 - void **sync_cb** (uint64_t executed, bool wait_for_event)

CACHE HIERARCHY

CACHE HIERARCHY

- New coherence protocol
 - Design choices
 - Directory-based
 - MSI (Modified, Shared, Invalid)
 - Specification based on [1] with:
 - Extension to 3-level cache hierarchies
 - Extension to exclusive caches
- Validation by means of assertions (around 160)
- Worst-case time model (over-approximated values expected)



[1] Nagarajan, V., Sorin, D. J., Hill, M. D., & Wood, D. A. (2020). A Primer on Memory Consistency and Cache Coherence. *Synthesis Lectures on Computer Architecture*, 15(1), 1-294.

Input configuration file



CACHE HIERARCHY – USER’S VIEW : OUTPUT

Performance counter	Description	
Coherence-related counters		
GetS	(Down Stream)	Get line in read-only mode
GetM	(Down Stream)	Get line in read-modify mode
FwdGetS	(Up Stream)	Get line ownership and put in read-only mode
FwdGetM	(Up Stream)	Get line ownership and invalidate
PutS	(Down Stream)	Line replacement in read-only mode
PutM	(Down Stream)	Line replacement in read-modify mode
PutI	(Up Stream)	Invalidate line
General-purpose counters		
Hits	(GetS and LineState in {Shared, Modified}) Or (GetM and LineState=Modified)	
Misses		
Writebacks		

Cache performance counters

NoC MODELLING

VPSIM NoC MODELS : LATENCY COMPUTATION

- A parametric NoC performance model (DATE 2021) :
 - Features an abstract router model (represented by a set of output buffers)
 - Can account for network contention at a router basis : Queuing delay & Congestion delay
- Elementary NoC model
 - Zero-load latency (assuming no contention) → lower bound on average network latency
 - Hop latency + serialization latency
- NoC with contention modeling
 - Buffer waiting time due to resource contention → realistic estimation of average network latency
 - Zero-load latency + queuing latency

$$Zlat(pkt) = P_{pkt} \times nbr\ hops + F\ T \times (L - 1)$$

$$Lat(pkt) = P_{pkt} \times nbr\ hops + F\ T \times (L - 1) + W_{pkt}$$

Notation used as in Dally, William & Towles, Brian. (2004). *Principles and Practices of Interconnection Network*

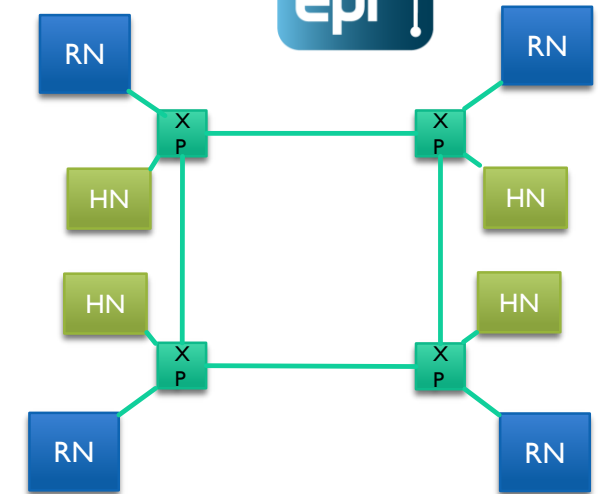
- P_{pkt} : physical delay (router + link transfer)
- FT : flit transmission time
- L : packet size (in flits)
- W_{pkt} : queuing delay

NOC MODEL – USER’S VIEW : INPUT

2D mesh with XY routing

```
'noc':{
  'x-nodes':2, 'y-nodes':2, # mesh size
  'diagnosis' : False,
  'with-contention' : True, # enable/disable contention model
  'contention-interval-ns' : 10, # contention is evaluated during this predefined period of time
  'buffer-size-flits' : 1, # buffer size in flits
  'router-latency-ns' : 1, # physical switch latency (in nanoseconds)
  'link-latency-ns' : 1, # physical link latency (in nanoseconds)
  'virtual-channels' : 1, # number of virtual channels per physical channel
},
```

Input configuration file



Components' coordinates on the mesh

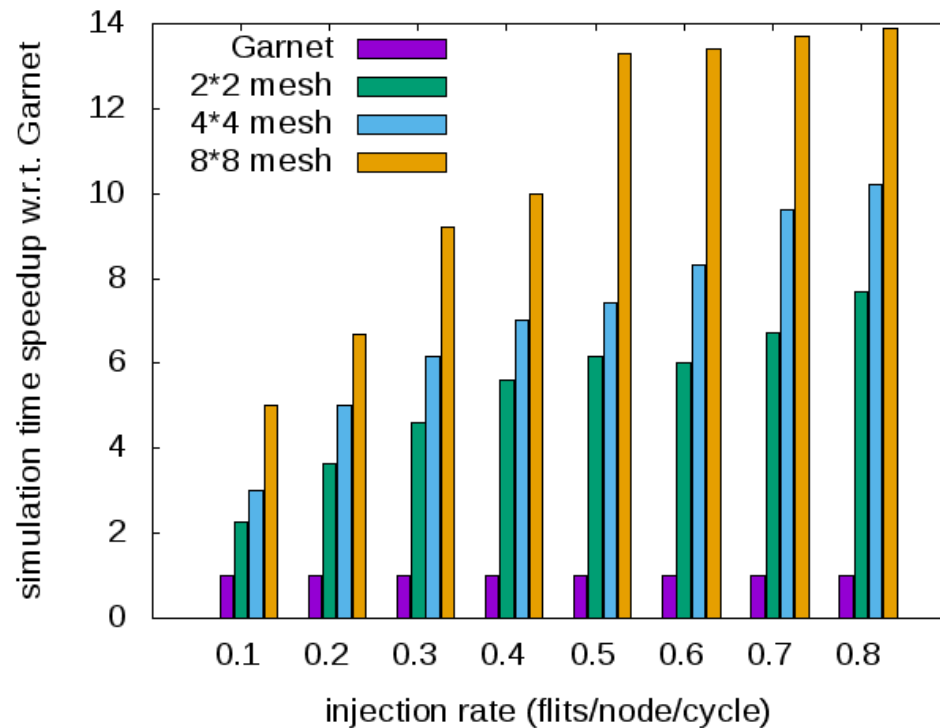
```
memory_subsystem':{
  '13': {
    'home-nodes': [ # base address, address space, noc position
      (0x40000000, 0x40000000, (0,0)),
      (0x80000000, 0x40000000, (0,1)),
      (0xC0000000, 0x40000000, (1,0)),
      (0x100000000, 0x40000000, (1,1))
    ],
    ...
  }
}
```

NOC MODEL – USER’S VIEW : OUTPUT

NoC performance counters

Performance counter	Description
Packets	Number of packets transmitted by the NoC
Total distance	Total distance crossed (in hops)
Average Network Latency	Zero-load delay + contention delay (in ns)

SIMULATION SPEED OF THE NoC MODEL



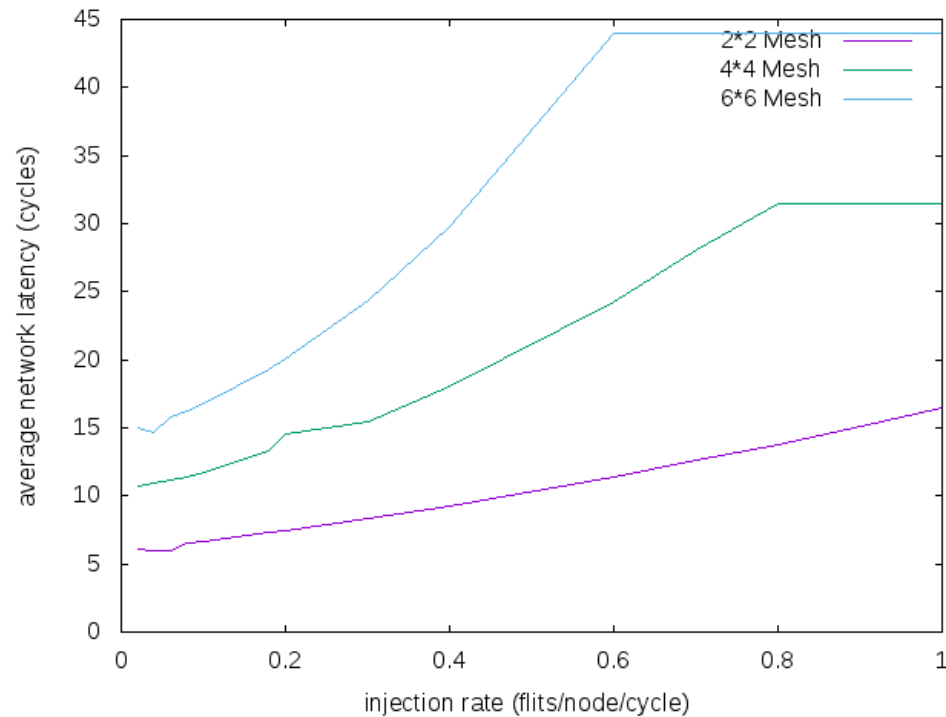
Up to x14 Speedup of the NoC model w.r.t. Garnet

Benchmark	Slowdown
Swaptions	1,6
radiosity	2,5
barnes	1,5
fmm	2,3
Black-scholes	2
Water-spatial	1,5

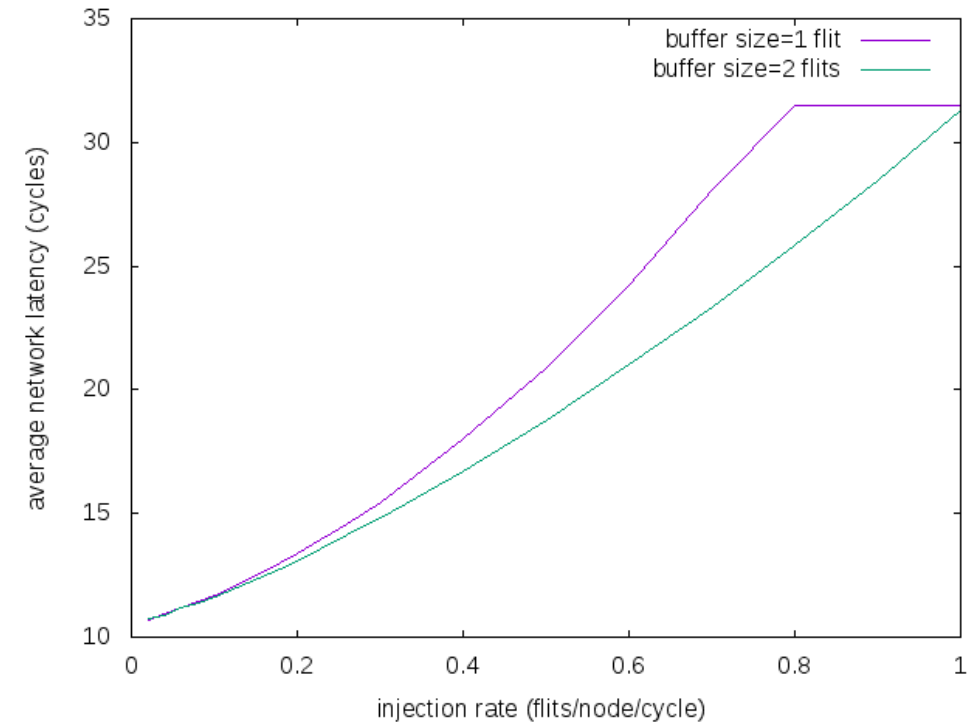
VPSim MIPS slowdown induced by the NoC model

NOC CONFIGURABILITY

- NoC parameters : NoC mesh size, buffer size, router latency (hop & serialization), VCs per Port



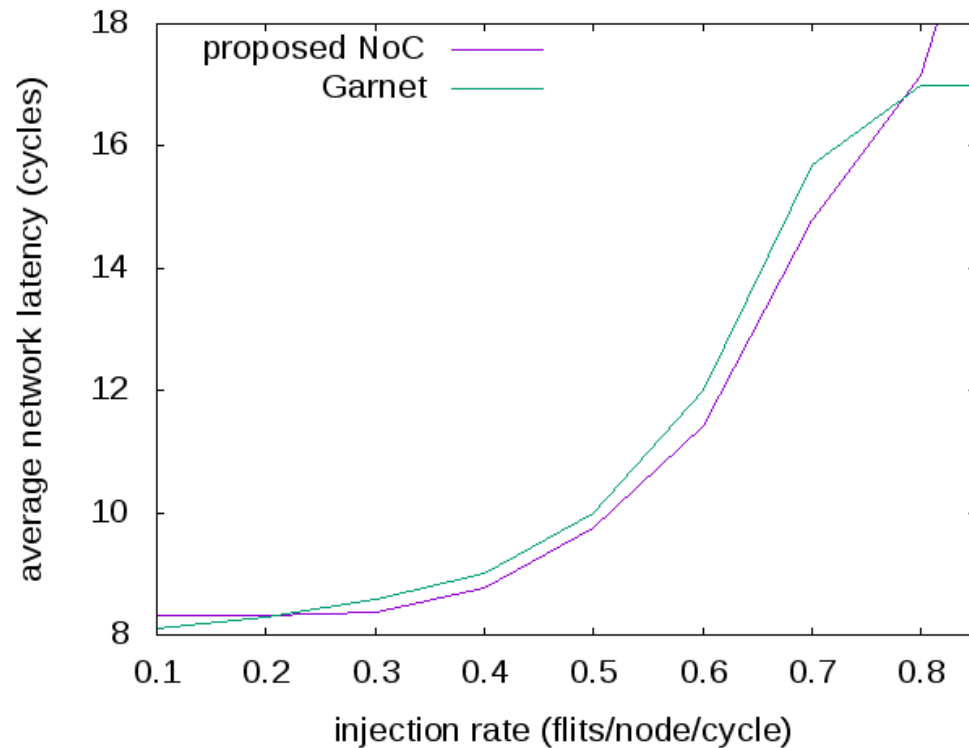
Average network latency w.r.t. mesh size



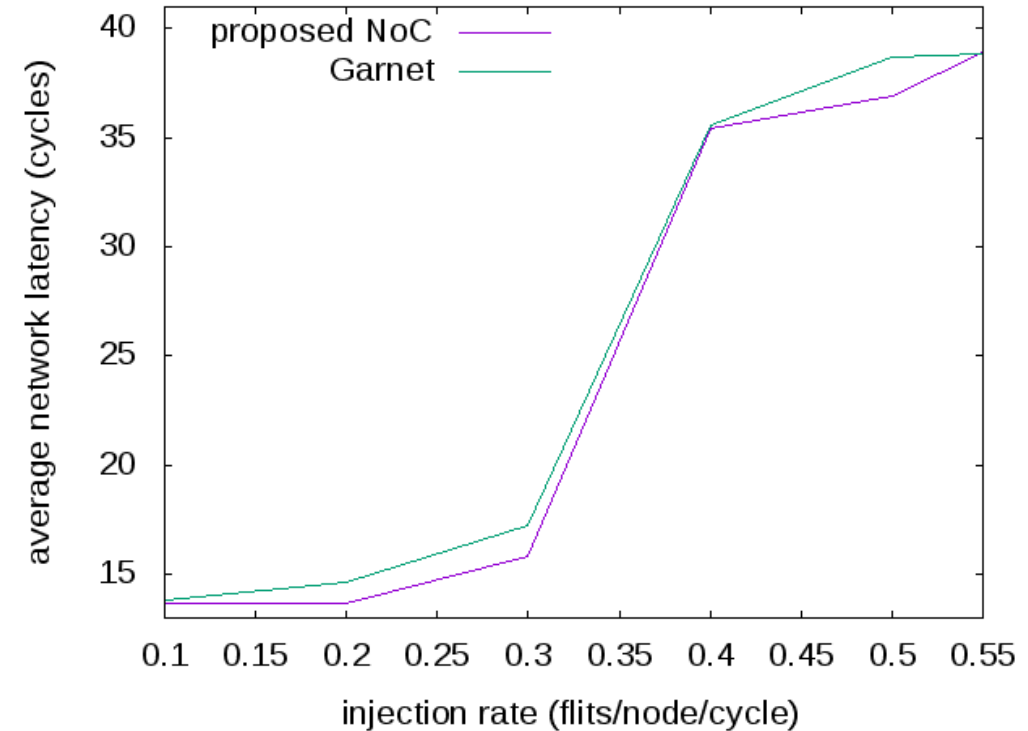
Average network latency w.r.t buffer size

COMPARISON WITH GARNET : NoC SIZE (STANDALONE MODE)

- Experiment : Random uniform traffic | 2 VC per port | 1-stage router (1 cy) + link transfer (1 cy)



Average network latency: **4*4 mesh** with XY routing

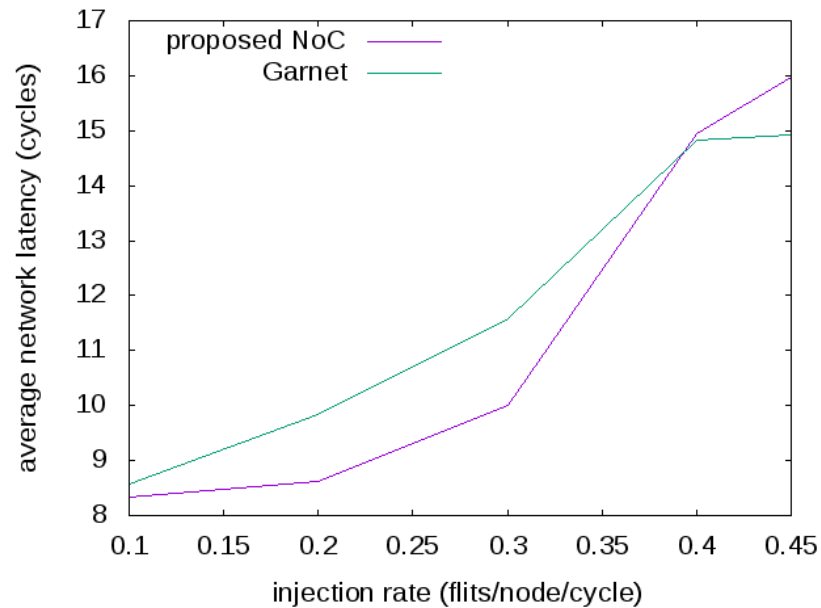


Average network latency: **8*8 mesh** with XY routing

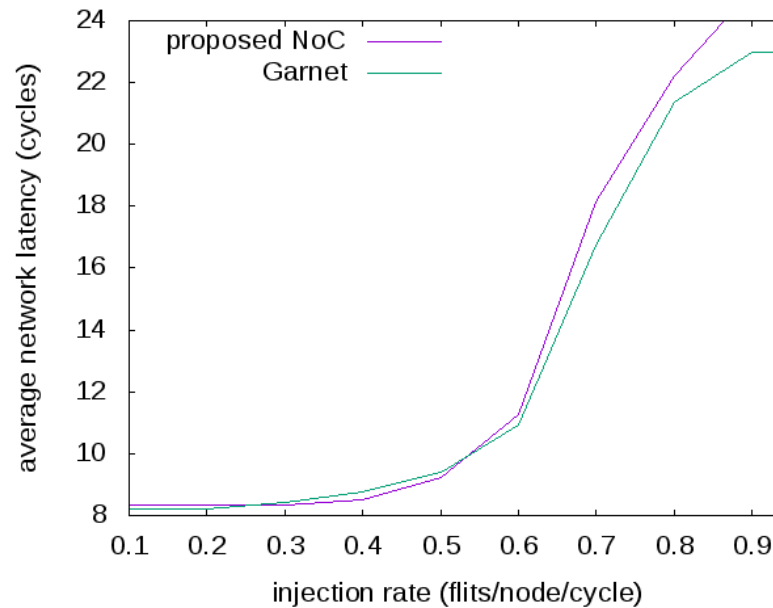
COMPARISON WITH GARNET : VIRTUAL CHANNEL (STANDALONE MODE)



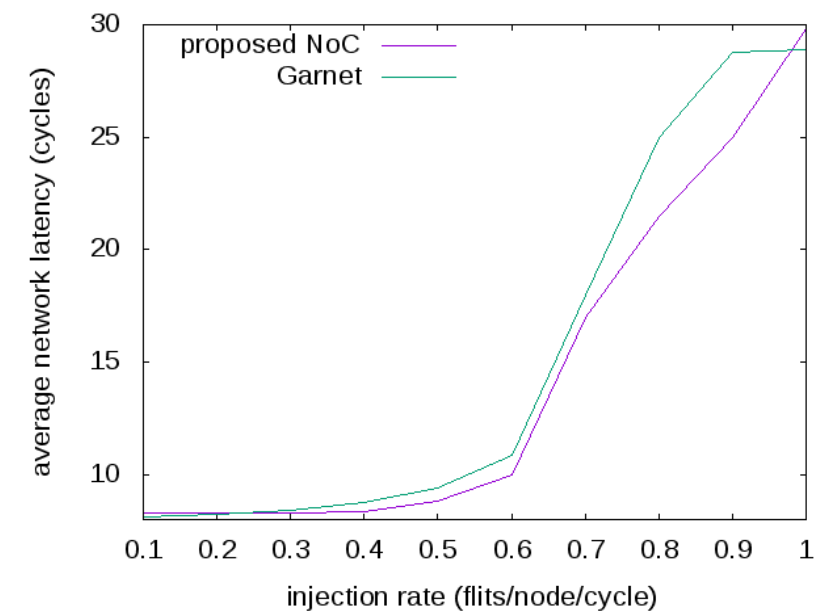
- Experiment : Random uniform traffic | 4x4 mesh XY routing | 1-stage router (1 cy) + link transfer (1 cy)



Average network latency : **1 VC** per port



Average network latency : **3 VC** per port



Average network latency : **4 VC** per port

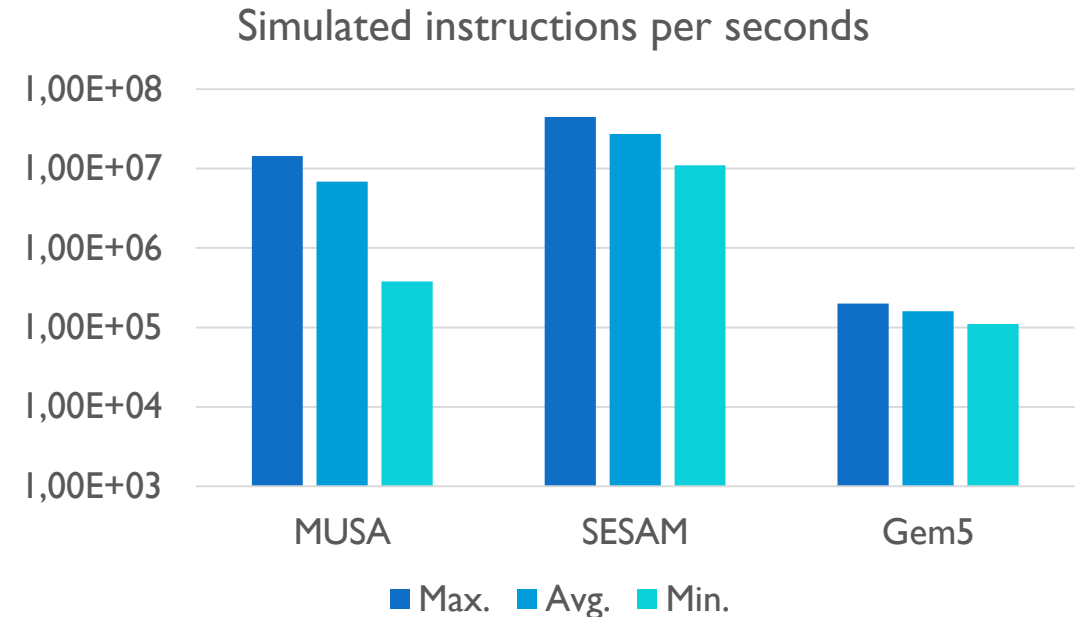
VPSim SPEED / ACCURACY TRADE-OFF

VPSIM SIMULATION SPEED

- EPI leverages several simulators to best fit all design needs from architecture exploration to SW dev
- Consistent results between simulators (MUSA, GEM5, VPSim) is paramount to make reliable design decisions

→ A thorough co-validation of simulators was conducted

- Evaluation of VPSim speed and accuracy w.r.t complementary EPI simulators
- Evaluation of achieved simulation speed in instructions per seconds
 - Total guest instructions / total simulation duration
- Used benchmarks
 - FMA-bench
 - simple STREAM kernel
 - Various test size
- Used model :
 - Single ARM core architecture
 - 3 cache levels
- VPSim achieves significant acceleration over more accurate simulators to provide faster DSE and SW development environment

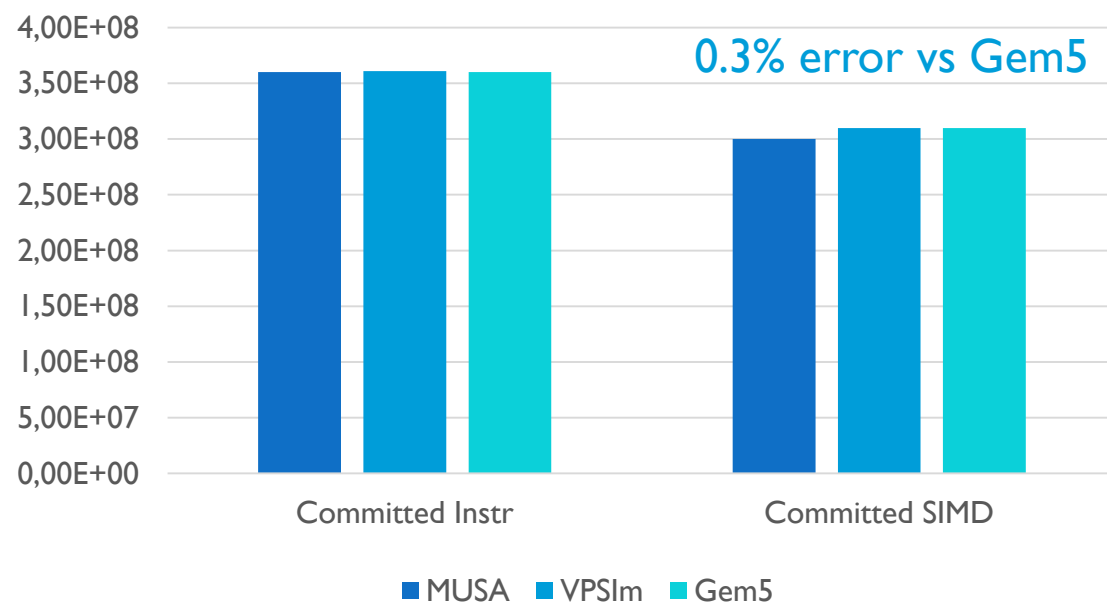


HARDWARE PERFORMANCE COUNTERS

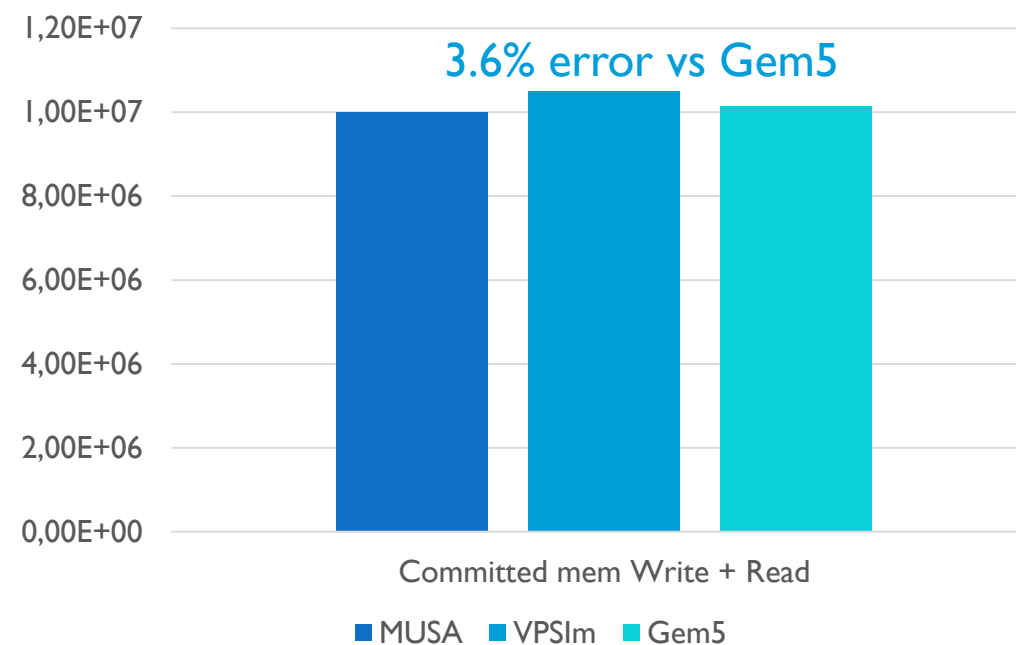
Cache counter	Description
overall access	Total number of overall (read+write) accesses on cache. The counter accumulates demand requests (hits + misses) and non-demand requests, but should not include.
overall miss	Total number of overall (read+write) misses on cache. The counter accumulation is similar to that of overall access. For non-blocking caches, an access misses on a cache block and if it then leads to a hit on the MSHR queue, it should be counted as a cache miss.
overall hit	Total number of overall (read+write) hits on cache. The counter accumulation is simulated to that of overall miss.
Overall miss rate	A ratio of overall miss over overall access
Instruction counter	Description
Simulated instruction	Total number of instructions are simulated.
Committed instruction	Total number of instructions are committed.
Executed instructions	Total number of instructions that are executed.
Operation counter	Description
Simulated Operation	Total number of operations (including micro operations)_are committed.
Committed operation	Total number of operations (including micro operations)_are committed.
Committed integer	Total number of integer operations are committed.
Committed float	Total number of floating-point operations are committed.
Committed SIMD	Total number of SIMD operations are committed. The SIMD counter should exclude Predicate operations.
Committed mem read	Total committed memory read operations (including micro operations). Note that a SIMD load instruction can be split into multiple micro operations
Committed mem write	Total committed memory write operations (including micro mem write operations). The counter accumulation is similar to that of committed mem read.
Simulated time counter	Description
Simulated second	Number of seconds is simulated.

EXAMPLE RESULTS

Instruction counters with FMA Benchmark



Memory access counters with FMA Benchmark



→ Consistent results with more detailed architecture simulators

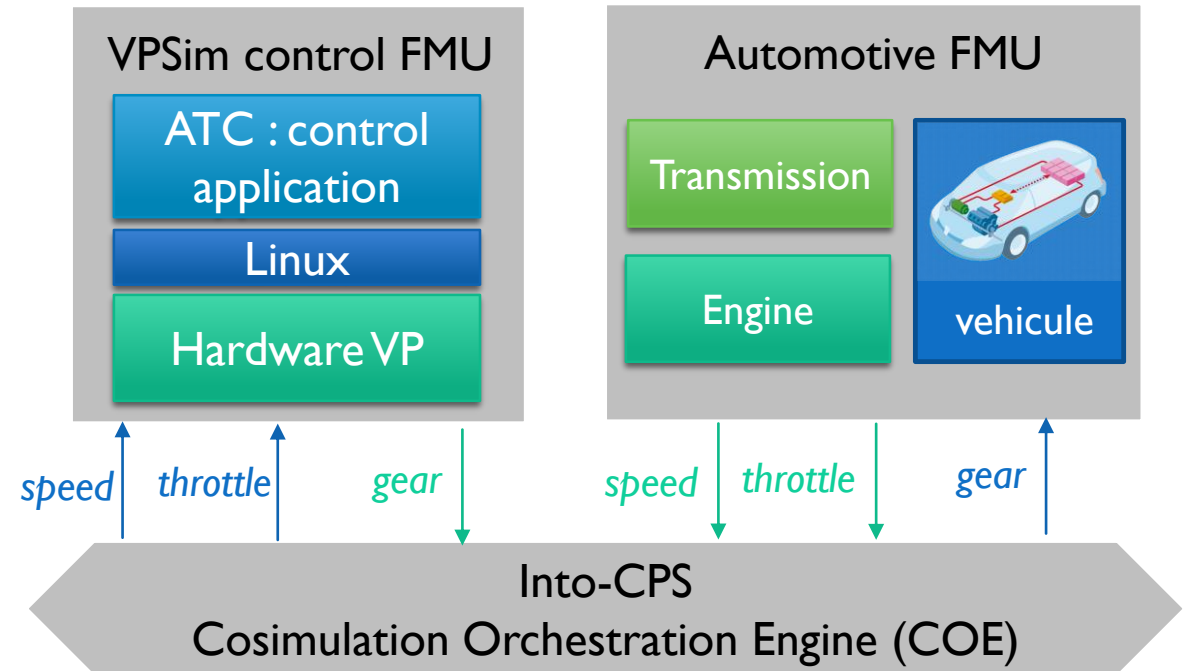
FMI CO-SIMULATION FOR SAFETY CRITICAL APPLICATIONS

CO-SIMULATION FOR VERIFICATION OF SAFETY CRITICAL APPLICATIONS IN EMBEDDED PROCESSORS



- FMU generated by VPSim to emulate a single cluster of ARMv8 64-bit architectures
 - 1-core processors with private L1 and L2 caches connected to the on-chip interconnect and peripheral devices
 - The architecture executes a Linux OS which supports the ShiftLogic application
- Performance-related properties can be assessed, possibly evaluating alternative choices of hardware components

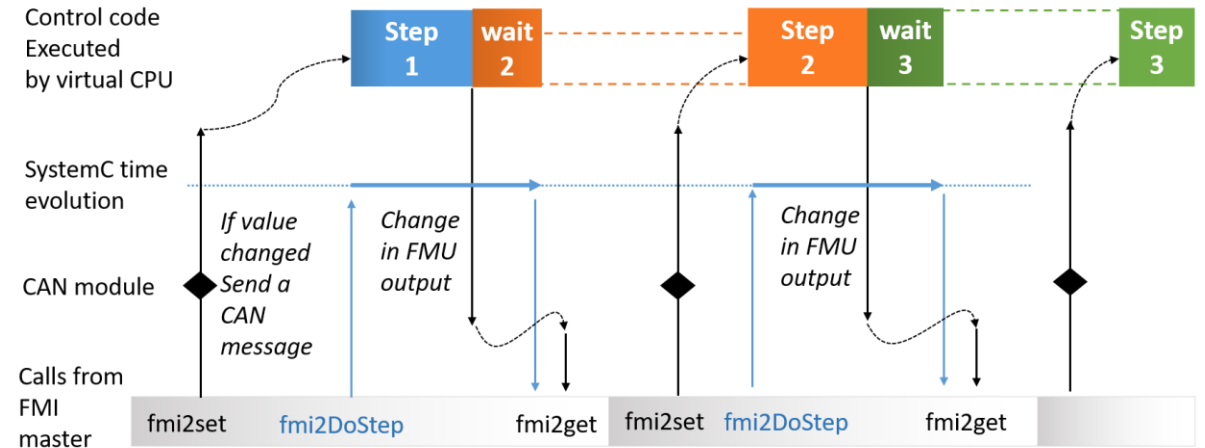
- Automatic Transmission Controller (ATC)



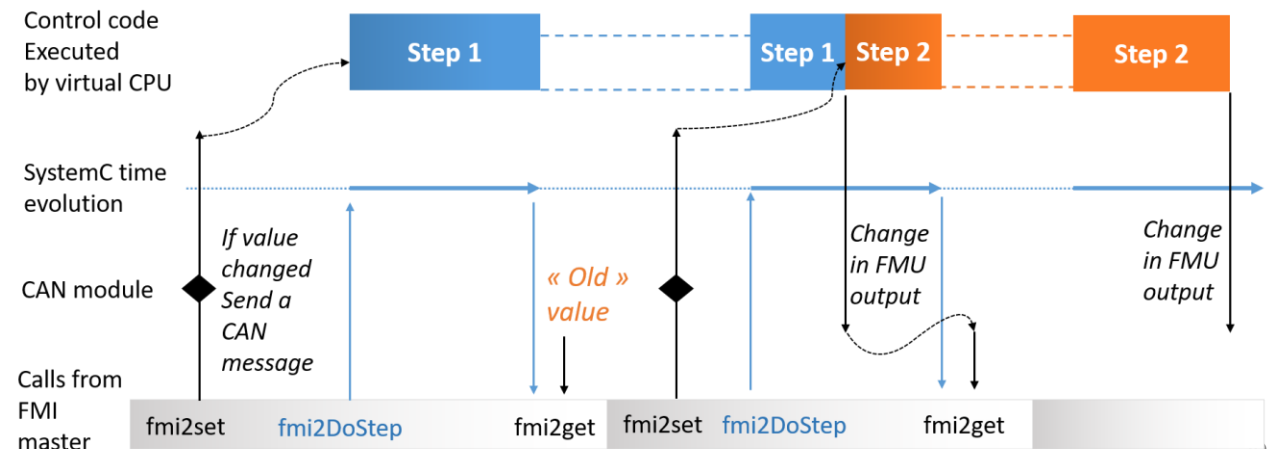
Joint work with **Universita de Pisa** published in CoSim-CPS 2020 : *Cross-level co-simulation and verification of an automatic transmission control on embedded processor*, C. Bernardeschi, A. Domenici, M. Palmieri, S. Saponara, T. Sassolas, A. Wicaksana & L. Zaourar

IMPACT OF SIMULATED CPU SPEED ON BEHAVIOR

- If the ATC execution duration is shorter than the FMI cosimulation step, output values are updated at every steps
- Consistent result with more abstract modeling of application execution

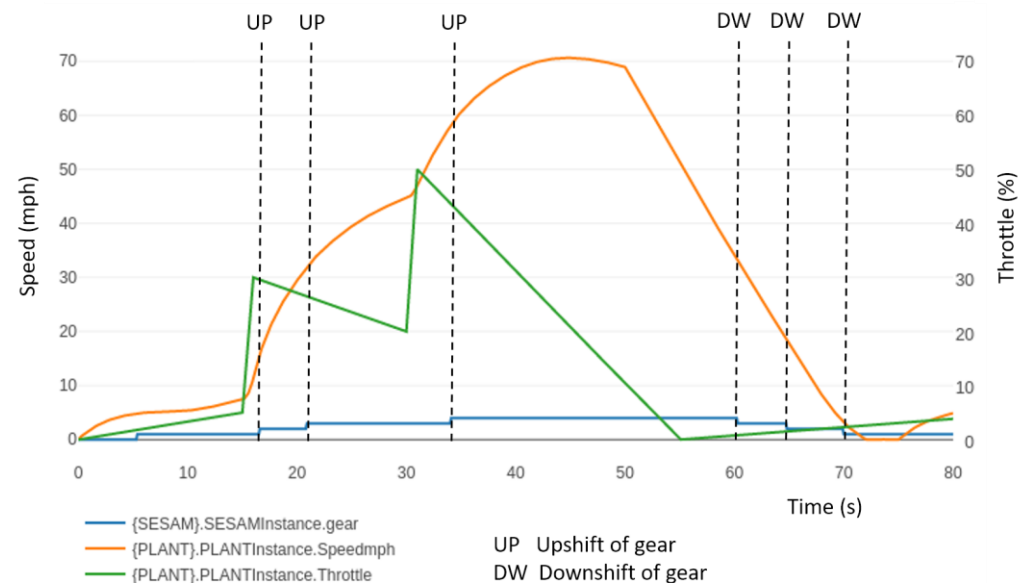


- If the ATC execution duration is longer than the FMI cosimulation step, output values are updated in later simulation steps
- (tested with added delay loops in the application code)

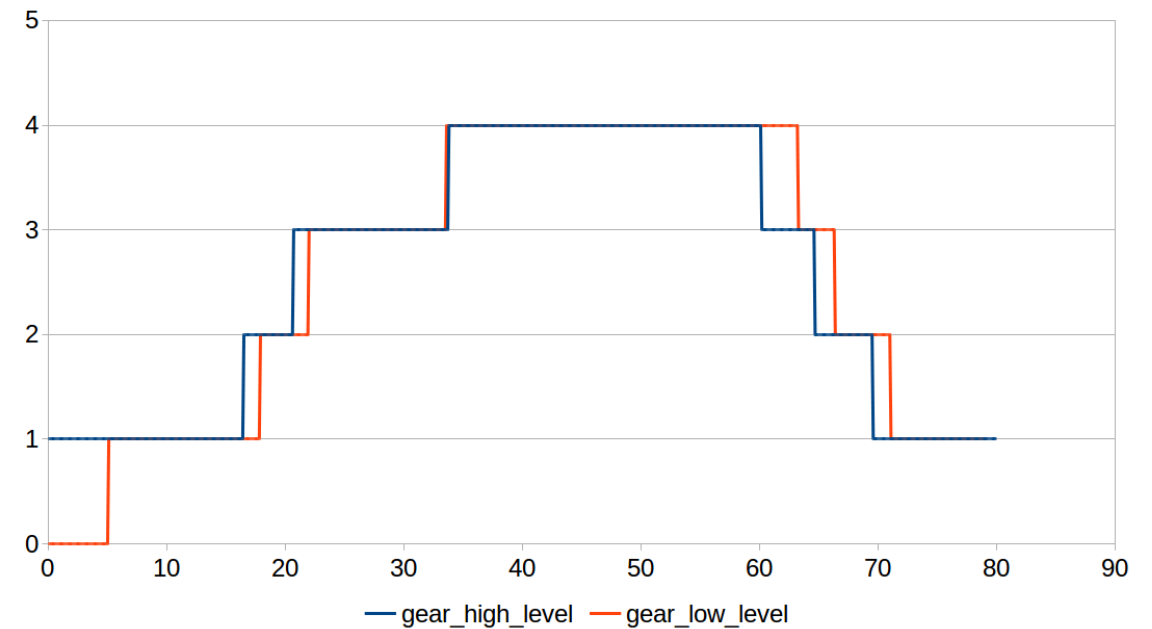


CO-SIMULATION RESULTS

- Resulting execution profile



- Difference with abstract control code modeling (when facing too long execution duration of the control code)



CONCLUSION

- Virtual prototypes
 - Abstract simulation models that enable SW development in the absence/limited access to the HW
 - Reduce the lag between HW delivery and SW availability
 - Improve time-to-market
- VPSim provides good compromise between accuracy and execution time
 - Achieves **high simulation speed** to enable the simulation of large scale HPC processors from design space exploration to SW developments
 - Though abstracting most of the modelled HW, simulation **accuracy** is kept high with error limited to 3,6% on performance counters
 - Several interfaces to enable efficient **co-simulation** at several levels of abstraction ranging from system simulation with the FMI standard, to external TLM peripherals and RTL simulation
 - By leveraging QEMU library as well as in-house models VPSim provides a **rich model library** to build a new platform in a glimpse through its user friendly python interface

PERSPECTIVES

- Further improve the timing accuracy of VPSim
 - Account for out-of-order execution, branch predictors and prefetchers
 - Automate timing modelling from more detailed models using learning techniques
- Key to enable architectural decisions through efficient automated Design Space Exploration
- Leverage VPSim co-simulation capabilities to couple with other partner's simulation tools and provide a system simulation (e.g. Synopsys Virtualizer)
- Define novel parallel methods to accelerate the simulation of memory hierarchies for ever-larger HPC models
- Keep enriching the model library (e.g. GPUs, eFPGAs)

SELECTED REFERENCES

- Ventroux, Nicolas, et al. "Highly-parallel special-purpose multicore architecture for SystemC/TLM simulations." 2014 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIV). IEEE, 2014.
- Ventroux, Nicolas, et al. "Sesam: An mp soc simulation environment for dynamic application processing." 2010 10th IEEE International Conference on Computer and Information Technology. IEEE, 2010.
- Ventroux, Nicolas, and Tanguy Sassolas. "A new parallel SystemC kernel leveraging manycore architectures." 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2016.
- Ventroux, Nicolas, Tanguy Sassolas, and Alexandre Guerre. "SESAM: AVirtual Prototyping Solution to Design Multicore Architectures Andriamisaina." *Multicore Technology*. CRC Press, 2018. 87-130.
- SESAM/Par4All: a tool for joint exploration of MPSoC architectures and dynamic dataflow code generation. N Ventroux, T Sassolas, A Guerre, B Creusillet, R Keryell. Proceedings of the 2012 Workshop on Rapid Simulation and Performance ...
- Charif, Amir, et al. "Fast virtual prototyping for embedded computing systems design and exploration." *Proceedings of the Rapid Simulation and Performance Evaluation: Methods and Tools*. 2019.
- Saidi, Salah Eddine, et al. "Fast Virtual Prototyping of Cyber-Physical Systems using SystemC and FMI: ADAS Use Case." *Proceedings of the 30th International Workshop on Rapid System Prototyping (RSP'19)*. 2019.
- Wicaksana, Arief, et al. "Hybrid Prototyping Methodology for Rapid System Validation in HW/SW Co-Design." 2019 Conference on Design and Architectures for Signal and Image Processing (DASIP). IEEE, 2019.
- Busnot, Gabriel, et al. "Standard-compliant Parallel SystemC simulation of Loosely-Timed Transaction Level Models." 2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC). IEEE, 2020.
- Cinzia Bernardeschi et al. « Cross-level co-simulation and verication of an automatic transmission control on embedded processor ». CoSim 2020
- Oumaima Matoussi, NoC Performance Model for Efficient Network Latency Estimation. *To appear IEEE DATE 2021*

SELECTED PATENTS



- Computer-implemented method of performing parallelized electronic-system level simulations - US Patent 10,789,397, 2020
- Electronic system level parallel simulation method with detection of conflicts of access to a shared memory –US 2019/0057173 A1
- Electronic system level parallel simulation method with detection of conflicts of access to a shared memory - US 2019/0057173 A1
- Computer-implemented method of performing parallelized electronic-system level simulations - US 2018/0173825 A1
- Device and method for accelerating the update phase of a simulation kernel - US 2017/0004232 A9

THANK YOU



Lilia Zaourar, PhD

Lilia.Zaourar@cea.fr

Research Engineer

CEA LIST

LECA Laboratory

IC and Digital System Division



Tanguy Sassolas

Tanguy.Sassolas@cea.fr

Head of LECA Laboratory

CEA LIST

LECA Laboratory

IC and Digital System Division



@EuProcessor



www.european-processor-initiative.eu



European Processor Initiative



European Processor Initiative