Exploring RISC-V based platforms within VPSim simulation tool for High Performance Computing

Ayoub MOUHAGIR, Mohamed BENAZOUZ, Lilia ZAOURAR

Paris-Saclay University, CEA, List, F-91120 Palaiseau, France (firstname.lastname@cea.fr)

RISC-V Summit Europe Barcelona
RISC-V
Monday-Friday 5-9 June 2023

Context

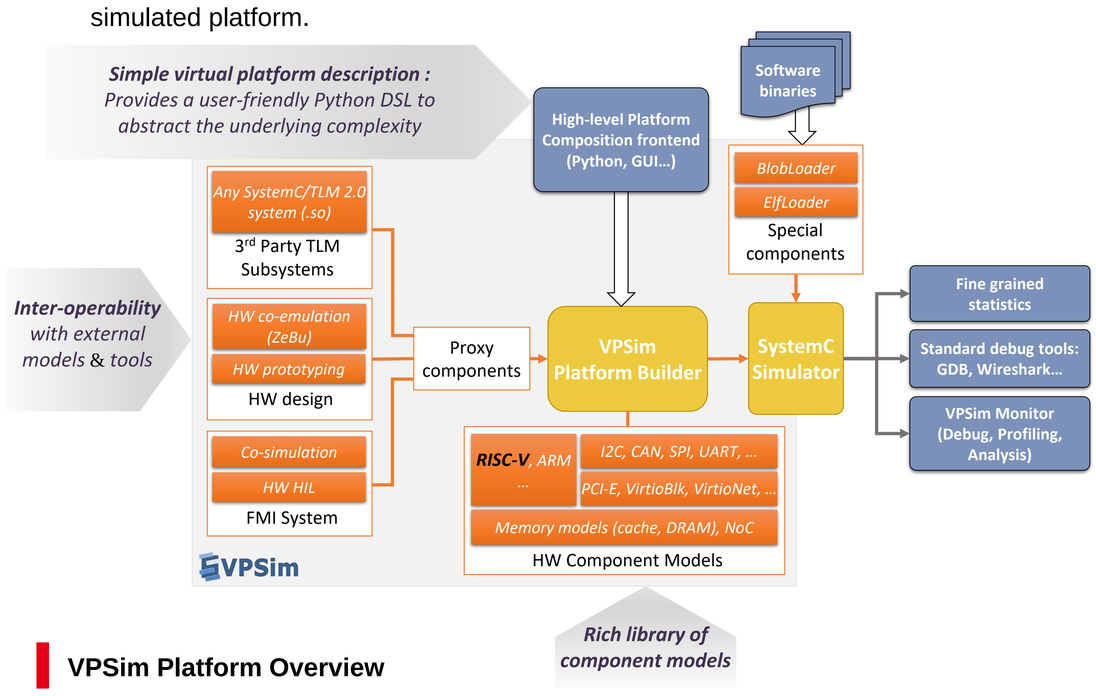
- The RISC-V Instruction Set Architecture is gaining attention for its open-source nature, flexibility, and greater customization.
- RISC-V based platforms are increasingly popular in the embedded systems industry. Designing and implementing RISC-V systems can be time-consuming and effort-intensive. To address these challenges, virtual prototyping tools have been widely adopted.
- Virtual prototyping enables modeling, simulation, testing, and optimization of complex systems in early design phases.
- **VPSim is a virtual prototyping tool specifically developed for SW/HW co-validation of computer architectures, including RISC-V based systems.**

VPSim: Virtual Prototyping Simulator

VPSim overview

The VPSim framework was designed to facilitate early-stage computer architecture design by providing support for SW/HW co-design [1]. It is a modular and highly configurable framework for:

- **Architectural exploration:** by providing configurable models to evaluate the performance of different platform configurations.
- **Software design:** by providing an enhanced user space to run, profile, and debug complete software stacks (e.g. BIOS, hypervisor, user space workloads) on the simulated platform.

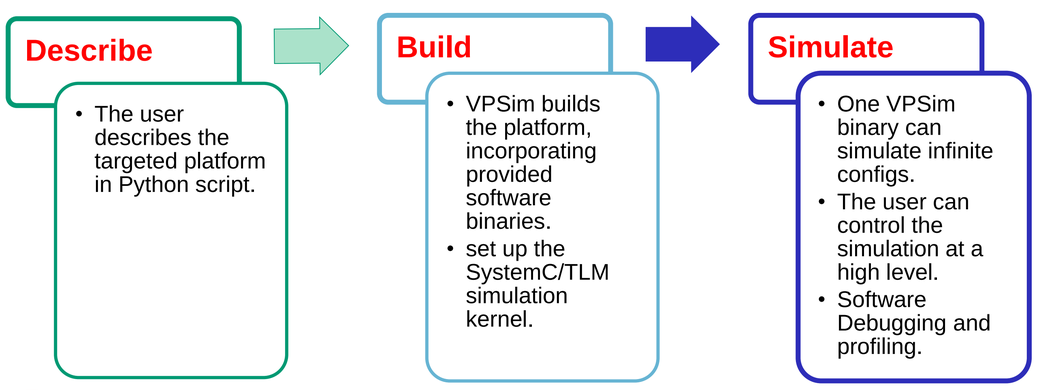


VPSim Platform Overview

- VPSim includes a large library of component models: CPU models (RISC-V, ARM, …), peripherals and memory models.
- The main supported model provider in VPSim is the open source system emulator QEMU [2], which allows unmatched simulation speed.
- VPSim integrates QEMU by running its CPU and peripheral models within SystemC threads. These QEMU models are enriched with performance information [3].
- VPSim stands out for its capacity to accommodate external subsystems through different standard and non-standard interfaces like SystemC, FMI, Python, and HW designs [4, 5].

Platform composition and simulation

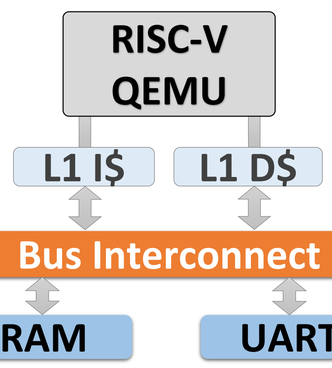
- VPSim provides a user-friendly interface to compose and build virtual platforms using an in-house Domain Specific Language (DSL) based on Python.



VPSim Workflow

RISC-V based platforms in VPSim

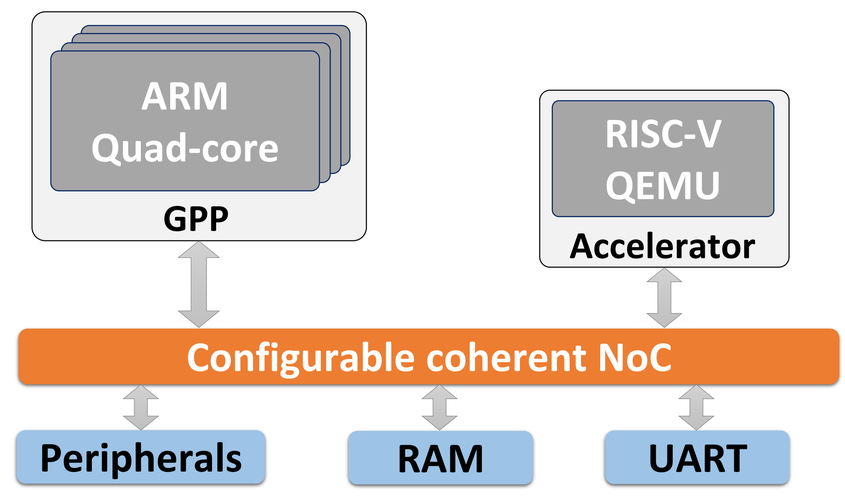
RISC-V single core platform




```python
from riscv_platform import FullSystem
config = {
    'platform_name': 'RISCV_VP',
    'cpu': {
        'cores': 1,
        'cores_per_cluster': 1,
    },

    'ram': [
        {
            'base': 0x40000000,
            'size': 0x100000000
        }
    ],

    'uarts': [
        {
            'type': 'NS16550Uart',
            'name': 'uart0',
            'base': 0x8000000,
            'irq': 11
        }
    ],
    # Next
    'software': {
        ...
    },
    'memory_subsystem': {
        'simulate': True,
        'cache': {
            'l1-data': {
                'size': 64*1024,
                'line-size': 64,
                'associativity': 4,
                'latency-ns': 0,
            },
            'l1-instructions': {
                ...
            },
        },
        'noc': {
            ...
        },
    },
}
if __name__ == '__main__':
 sys = FullSystem(config)

 stats = sys.build(simulate=True, silent=False,)
```

Snippet of a basic single RISC-V core platform composition in VPSim

RISC-V core as an accelerator



The retrieved performance counters and statistics after simulation can be classified into:

- **Functional counters:** focus on instruction counts, including the number of instructions, cache performance and memory accesses.
- **Time-related counters:** involve simulated time and associated factors, such as memory bandwidth and latencies.

Conclusion & Perspectives

- VPSim presents a valuable solution for HW/SW designers to quickly evaluate and refine their designs while minimizing associated time and costs.
- It offers an extensive range of performance counters and statistics for profiling and benchmarking purposes while ensuring a commendable trade-off between precision and execution time.
- Continual improvements are being made to VPSim, introducing a host of new features that enhance its capabilities and yield superior results for evaluation and benchmarking purposes.

ACKNOWLEDGMENT

This work has been performed in the context of the European Processor Initiative (EPI) project, which has received funding from the European Union's Horizon 2020 research and innovation program under Grant Agreement № 826647 and Specific Grant Agreement № 101036168 (EPI SGA2).

References

[1] Amir Charif, Gabriel Busnot, Rania Mameesh, Tanguy Sassolas, and Nicolas Ventroux. 2019. Fast Virtual Prototyping for Embedded Computing Systems Design and Exploration. In Proceedings of the Rapid Simulation and Performance Evaluation: Methods and Tools (RAPIDO '19).

[2] F. Bellard. 2005. QEMU, a Fast and Portable Dynamic Translator. In Proceedings of the FREENIX Track: 2005 USENIX Annual Technical Conference, Anaheim, CA, USA.

[3] Fatma Jebali, Oumaima Matoussi, Arief Wicaksana, Amir Charif, and Lilia Zaourar. 2022. Decoupling processor and memory hierarchy simulators for efficient design space exploration. In System Engineering for constrained embedded systems (DroneSE and RAPIDO).

[4] Salah Eddine Saidi, et al., 2019. Fast Virtual Prototyping of Cyber-Physical Systems using SystemC and FMI: ADAS Use Case. In Proceedings of the 30th International Workshop on Rapid System Prototyping (RSP '19).

[5] Cinzia Bernardeschi, Pierpaolo Dini, Andrea Domenici, Ayoub Mouhagir, Maurizio Palmieri, et al.. Co-simulation of a Model Predictive Control System for Automotive Application. CoSim-CPS 2021