2023 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE 10 YEAR ANNIVERSARY

VPSim : Virtual Prototyping Simulator with best accuracy & execution time trade-off for High Performance Computing systems evaluation and benchmarking

Mohamed BENAZOUZ, Universite Paris-Saclay, CEA, List, Palaiseau, France (mohamed.benazouz@*cea.fr*)

Ayoub MOUHAGIR, Universite Paris-Saclay, CEA, List, Palaiseau, France (ayoub.mouhagir@*cea.fr*)

Lilia ZAOURAR, Universite Paris-Saclay, CEA, List, Palaiseau, France (lilia.zaourar@*cea.fr*)

Abstract*— Virtual prototyping has become widely adopted as a time and cost-effective solution for early hardware (HW) and software (SW) co-validation. This paper highlights VPSim, a virtual prototyping solution that addresses HW/SW designers' needs regarding performance, ease of use, rapidity and flexibility. VPSim includes a large library of hardware component models, along with its capability to accommodate external subsystems through numerous standard and non-standard interfaces. This tool features a decoupled simulation methodology that separates processor and memory hierarchy simulation, enabling more efficient design space exploration. Thanks to its modular approach, it has been used as a simulation environment in the context of HPC co-design. It allows the modeling of different computer architecture components, delivering an optimal trade-off between accuracy and execution speed. VPSim can provide detailed statistics for each system component to facilitate design choices.

Keywords—virtual Prototyping; VPSim; exploration; design; simulation

I. INTRODUCTION

To keep up with the increasing complexity and scale of High-Performance Computing (HPC) systems, digital twins or virtual prototyping tools are tremendously employed as a cost-effective and time-efficient solution for early hardware and software co-validation. The digital twin prototype allows users to adjust system parameters and reconfigure components to observe the system's response, enabling early hardware and software testing during the development cycle. This environment facilitates communication and exchanges between developers and end-users during the design phase.

In this regard, the Virtual Prototyping Simulator (VPSim) [1] offers a viable solution for HW/SW designers, providing a flexible and highly configurable framework for architectural exploration and software design. It has been used as a simulation environment in the context of HPC co-design with various design concerns and applications [6]. This paper is structured as follows: the second section presents an overview of VPSim tool. The following section unveils the extensive features of VPSim, accompanied by a brief presentation of its simulation methodology. The fourth section provides an analysis of the composition of the VPSim platform and its simulation procedures. Finally, the fifth section presents the outcomes of our simulation before concluding in the last section.

II. VPSIM OVERVIEW

With systems' increasing complexity and scale, virtual prototyping solutions should meet new and evolving requirements regarding performance, ease, rapidity, and automation of system modeling and design space exploration. In this context, VPSim addresses the challenges mentioned earlier and strives to provide maximum flexibility to its users to compose their systems. VPSim offers a powerful and versatile platform for modeling and simulating complex systems across various domains, ranging from High-Performance Computing (HPC) and Automotive applications to Electronic multi-SoC boards and heterogeneous multicore architectures.

The VPSim framework was designed to support SW/HW co-design in the early stages of computer architecture design. It is a modular and highly configurable framework for:

- **Architectural exploration:** providing configurable models to evaluate the performance of different platform configurations and identify the optimal one that best meets SW/HW requirements

- **Software design:** providing an enhanced user space to run, profile, and debug complete software stacks (e.g. BIOS, hypervisor, user space workloads) on the simulated platform. Therefore, it assists in devising software improvement strategies.

Figure 1 gives a global overview of the VPSim tool, which features a diverse collection of CPU models, such as ARM and RISC-V, as well as models of several peripherals and buses. These components are fully standard-compliant SystemC/TLM 2.0 following the Loosely-Timed paradigm with a blocking communication interface, which offers a good trade-off between execution speed and accuracy. The main supported model provider in VPSim is the open-source system emulator QEMU [2], which enables high simulation speed. Other model providers can also be imported from the ARM Fast Models, Open Virtual Platforms, etc. Users can develop models directly in SystemC or Python to extend their virtual prototypes.

Furthermore, VPSim stands out for its capacity to accommodate external subsystems through numerous standard and non-standard interfaces like SystemC, Python, QEMU devices and HW designs. Moreover, it can connect with other simulation tools and modeling software through a Functional Mockup Interface (FMI) based co-simulation, as demonstrated in [3,4].

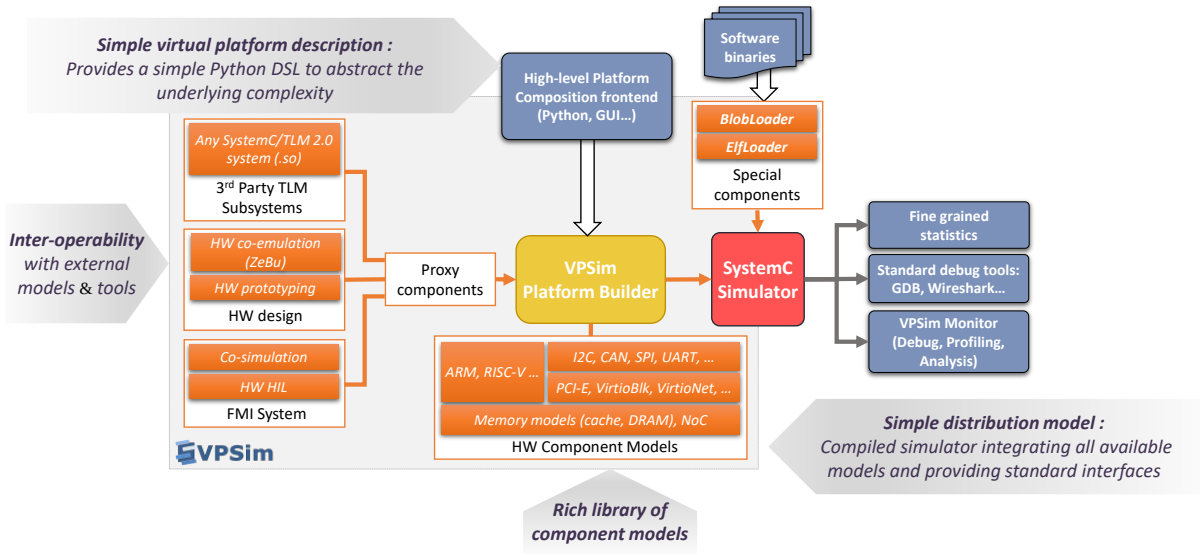

Figure 1. VPSim platform overview.

VPSim offers an abundance of performance counters and statistics for profiling and benchmarking purposes. These counters can be classified into two categories: functional and time-related counters. Functional counters focus on instruction counts, including the number of instructions and memory accesses. In comparison, Time-related counters involve simulated time and all associated factors, such as memory bandwidth and latencies.

III. VPSIM FEATURES

A. *VPSim simulation methodology*

VPSim proposes a decoupled simulation methodology that separates the processor and memory hierarchy simulation, enabling more efficient design space exploration. This approach, detailed in [5], decreases significantly the burden of simulating intricate memory hierarchies while maintaining accurate performance evaluation outcomes. The both processor and memory hierarchy simulations can progress independently and at different rates within distinct host threads and communicate asynchronously for a specific quantum period. After each period, the simulators synchronize to prevent one from outpacing the other in time.

As mentioned earlier, the main processor simulator in VPSim is QEMU, which is integrated into the full VPSim environment by running its CPU and peripheral models in the context of SystemC modules. This method can leverage QEMU's technologies, such as Dynamic Binary Translation (DBT) and para-virtualization, resulting in exceptional simulation speeds while adhering to a more deterministic and standardized SystemC environment. QEMU prioritizes fast simulation runtime over timing accuracy. VPSim supplements the QEMU time model to address these timing inaccuracies by incorporating memory access delays using SystemC components that model the memory hierarchy.

The decoupled simulation methodology [5] in VPSim relies on capturing the memory accesses in the QEMU processor emulator and feeding them to memory SystemC/TLM models. The latter is only responsible for replicating the necessary behavior for performance evaluation without managing copies of shared data.

Each memory access generates a corresponding event containing the requester CPU's identifier, access type (read/write), memory address for the operation, data size, and timestamp. These events are added to a lock-free FIFO queue, ensuring no locking mechanism is needed to add or remove elements from the queue. The memory simulator processes these queued events sequentially, ensuring a smooth and orderly consumption of each event in the queue.

However, the traditional lock-free FIFO event queue has limitations, especially in situations where multiple threads concurrently access the queue. Although lock-free mechanisms can prevent potential deadlocks, they can introduce contention and performance bottlenecks when multiple threads try to access the queue simultaneously. To address these limitations and make significant progress in the decoupling simulation methodology, VPSim has implemented a further improvement on its previous version. This improvement substitutes the conventional lock-free FIFO event queue with a TBB (Intel Threading Building Blocks) multi-threaded priority queue [7].

The TBB priority queue significantly reduces contention among threads. The handling of events is optimized, which leads to enhanced simulation performance with better coordination and synchronization between different components. Moreover, the TBB queue's dynamic memory allocation and management capabilities further optimize memory usage, ensuring smoother simulation execution. This enables VPSim to handle more complex simulation scenarios without compromising performance.

B. VPSim's main improvements

VPSim platform continues to evolve, incorporating new features to stay at the forefront of virtual prototyping. It allows both software and hardware designers to gain valuable insights into the behavior and performance of their workloads in environments that closely resemble actual hardware setups.

1) Conversion factor

Another significant enhancement in VPSim was the introduction of a Conversion Factor (CF) to overcome a limitation in the QEMU core frequency handling. Previous versions of VPSim assumed that the processors would run at a fixed frequency of 1GHz, leading to inaccurate absolute timings in the simulation. It led to observed timings that could be larger than those in the real hardware system could.

The CF is a scaling factor that adjusts the simulated timings dynamically to mimic real system behavior. By adjusting timings based on the CF value, VPSim now delivers more accurate and realistic simulations that closely reflect the actual performance of the hardware.

2) Memory and SLC n-way interleaving

Adding Memory and SLC N-way interleaving support in VPSim is a notable enhancement compared to its earlier versions. Previous versions of VPSim may have had restricted or absent capabilities for modeling and simulating memory systems using N-way interleaving, which is critical in modern computer architectures.

N-way interleaving is a memory access technique that enhances memory bandwidth and diminishes contention by distributing data across numerous memory banks or channels. The usage of this technique is widespread in high-performance computing systems and servers as it uplifts the entire system's performance. With the addition of Memory and SLC N-way interleaving support, VPSim can model and simulate memory systems with various interleaving configurations, such as 2-way, 4-way, 8-way, and so on.

3) NUMA Support

One of the main enhancements in the VPSim tool was adding the support to model and simulate NUMA-based (Non-Uniform Memory Access) architectures. NUMA is an essential memory architecture used in modern computer systems, especially in multi-core servers and high-performance computing environments. It allows processors to access different memory areas at varying speeds based on their proximity to the memory locations.

Several modifications were conducted to implement NUMA support in VPSim, including extending the QEMU machine and Linux kernel to expose NUMA systems to the guest Operating System. Besides, the memory

architecture modeling was updated to include all information about NUMA configuration, enabling the requests redirection to the right targets such as SLCs, and memory controllers.

4) Region Of Interest (ROI)

Simulating the entire model can be time-consuming and computationally burdensome in the context of complex and resource-intensive systems. Therefore, we introduced in VPSim the ability to define and focus on a Region Of Interest (ROI) within the system. Users can specify which part of the system they want to simulate in greater detail while simplifying or abstracting the rest.
This feature was dynamically incorporated into VPSim without requiring the simulator to be rerun. For example, the user can skip all memory hierarchy modeling simulations during the OS boot phase and enable it when running the application after boot. It allows a Linux minimal OS or even a full Debian boot to be started as quickly as possible with QEMU.

IV. VPSim 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. The tool takes as input a high-level platform description together with the software binaries to be executed on the virtual platform. It can also be automatically generated from a graphical user interface (GUI). It is worth mentioning that, unlike other tools that rely on high-level description languages, e.g. GEM5, VPSim is not closely tied to its Python interface and needs to be made aware of its existence. Instead, it is predominantly built on an XML data exchange format. Therefore, integrating VPSim with other XML-compatible tools is significantly easier.

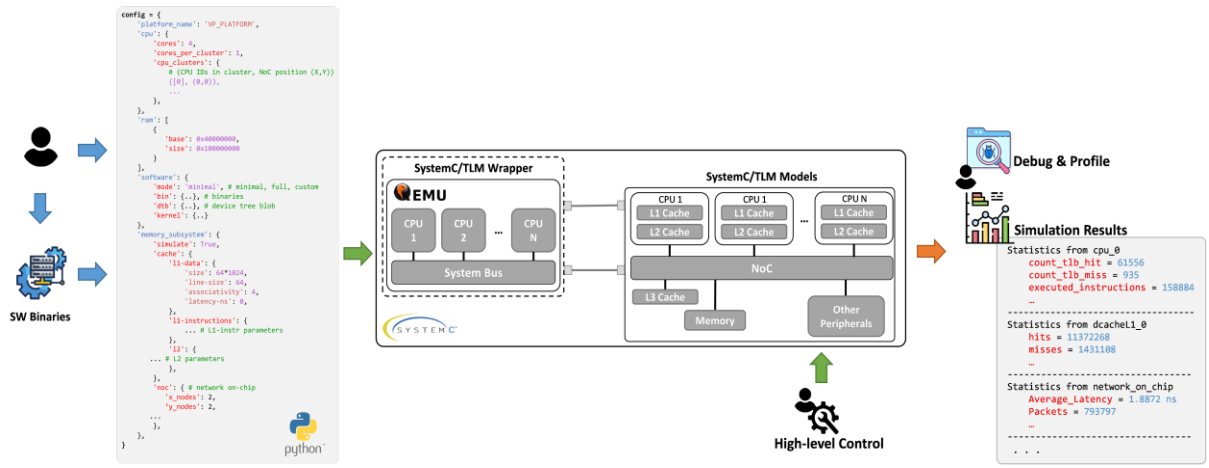

Figure 2. VPSim Workflow.

Once the platform is described, VPSim takes over building the platform, incorporating provided software binaries (e.g. kernel, file system, application, etc.), and setting up the SystemC/TLM simulation kernel with two different threads: QEMU processor emulator and memory hierarchy simulator. This process capitalizes on the decoupled simulation approach between QEMU and SystemC/TLM-based models, allowing them to operate independently and interact through well-defined interfaces. One remarkable advantage of VPSim is its versatility, as a single VPSim binary can be used to simulate an infinite number of configurations.

During the simulation, users have full control over the platform, including the ability to stop, restart, debug, and profile the software. In addition, the user can access a wide range of runtime statistics useful for detailed performance evaluation and design space exploration. For example, Table 1 presents a non-exhaustive list of performance counters provided by VPSim at the end of each simulation. Some of the new instrumented counters, which result from the latest version of VPSim, are highlighted in blue for easy comparison with the previous version.

Table 1. Performance counter for NoC evaluation.

	Performance counter	Description
CPU	Executed instructions	Number of executed instructions per cpu
	Executed FP instructions	Number of executed Floating Point instructions per cpu
	Loads/stores	Number of loads and stores per cpu
Cache	Cache protocol transactions	Stats about the cache protocol transactions per cache
	Hits/Misses	Number of hits/misses per cache
	Writes/Reads	Number of reads and writes per cache
Network on-Chip	Total Packets	Number of packets that traversed the NoC
	Total distance	Total number of hops made by a packet across routers and channels from its source node to its destination node.
	Total latency	Total latency of packets through the NoC
	Average latency	Average network latency
	Packets per router	Number of packets that traversed each router
	Contention delay per router	Contention delay of packets at each router level
	Memory Reads Crossing Numa Nodes	Number of memory reads initiated by a CPU in one NUMA node that access memory located in another NUMA node.
	Memory Writes Crossing Numa Nodes	Number of memory writes initiated by a CPU in one NUMA node that access memory located in another NUMA node.
	Packets Crossing Numa Nodes	Number of NoC packets crossing one NUMA node to another NUMA node.

V. RESULTS

In this study, our experiments are directed towards exploring parameters associated with the memory hierarchy and Network on Chip (NoC), as these elements constitute pivotal facets of HPC systems. We modeled and simulated an architecture composed of ARMv8 cores with two-level private caches. The cores are all clustered in a NoC with last-level caches. Figure 3 gives an overview of the system. In addition, we tested a diverse set of representative High-Performance Computing (HPC) benchmark applications, including PARSEC & SPLASH suites, DGEMM and waLBerla that are detailed below.

- STREAM [8] is designed to measure computer systems' memory bandwidth and latency. It consists of simple vectorized kernel instructions used to evaluate the performance of memory-intensive operations on a given architecture. It helps to assess the memory subsystem's efficiency and identify potential bottlenecks in data transfer rates and memory access patterns.
- The PARSEC [9] Benchmark Suite is a widely recognized benchmark suite designed for evaluating and analyzing the performance of multi-core processors and systems, especially in the context of high-performance computing (HPC) and parallel processing. PARSEC stands for "Parallel Applications for Scalable Computing." It includes a set of representative parallel applications covering different workloads and computational patterns commonly found in HPC and multi-core systems. These applications stress various aspects of a system's architecture, such as memory bandwidth, cache efficiency, and parallel scalability.
- The SPLASH-2 (Scalable Parallel Applications for Shared-Memory) [10] is another well-known suite of multithreaded benchmarks that focuses on High-Performance Computing.
- DGEMM [11] stands for Double-precision General Matrix Multiply. It is a fundamental linear algebra operation used in many scientific and engineering applications. It computes the product of two double-precision matrices, which involve many floating-point operations. This operation is computationally intensive and is commonly found in numerical simulations: weather forecasting, molecular dynamics, and other scientific simulations. Optimizing DGEMM execution is crucial for enhancing the performance of many HPC applications that rely on linear algebra computations.
- waLBerla [12] is an open-source software framework designed to simulate fluid dynamics and multi-physics problems on HPC architecture efficiently. It is specifically tailored for large-scale simulations that require the utilization of multiple compute nodes in parallel. Thus, it is well suited for the design of massively parallel supercomputers.

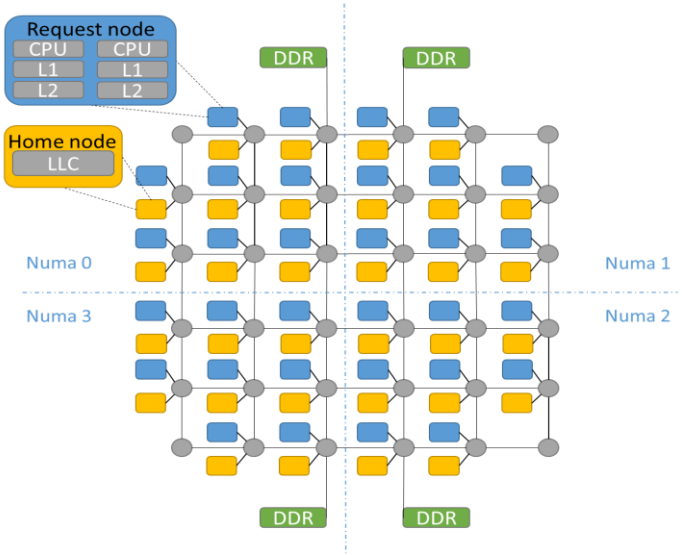

Figure 3. Simulated ARMv8 based architecture (6x6 mesh NoC, 64 cores and 32 SLC slices). If NUMA support is enabled, 4 nodes are exposed.

Throughout all experiments, the new ROI feature was enabled which allowed us to reduce system booting time by 10 minutes (resp. 1 hour) each time we had to run our minimal (resp. full Debian) Linux distribution.

In the first set of experiments, we explored the impact of SLC interleaving while having NUMA support disabled. When interleaving was not used, only a portion of the SLC slices was actively utilized. This utilization could be as low as having only 3 out of 32 slices in use, as Table 2 illustrates for DGEMM. This phenomenon occurs because the Linux kernel allocates memory for user space at higher physical addresses. Consequently, this inefficient utilization of the available SLC capacity also leads to elevated miss rates.

Table 2. Solicited SLC slices for DGEMM application (misses+hits).

Y\X	0	1	2	3	4	5
0		65922	0	0	0	
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5		0	0	30397841	1454476	

On the other hand, when SLC interleaving is enabled, all slices are nearly equally utilized. This, in turn, leads to reduced miss rates, as depicted in Figure 4.

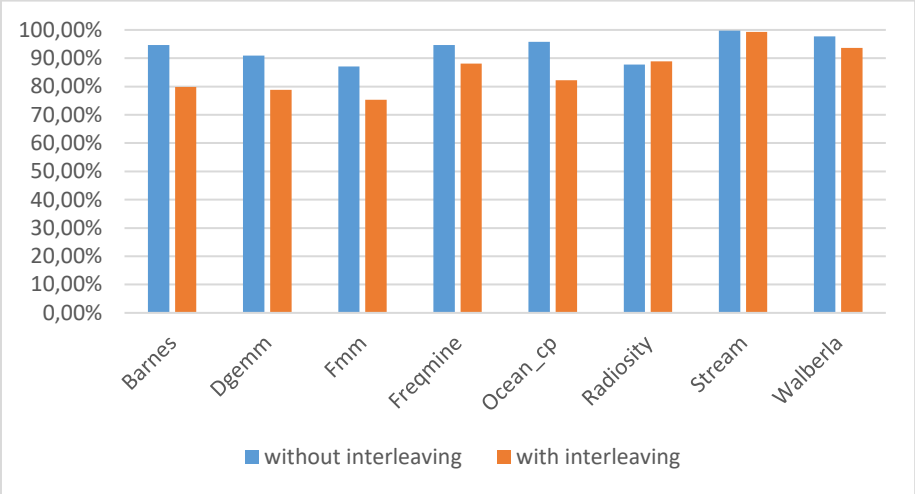

Figure 4. Impact of SLC interleaving on SLC miss ratio.

Applications such as STREAM are designed to measure sustainable memory bandwidth and handle large datasets to eliminate cache effects at all levels, which explain their high miss ratios. More generally, the observed high miss ratios can also be attributed to the adopted exclusive cache policy, which may not be adapted to the tested HPC applications. We are currently planning to implement more efficient caching policies.

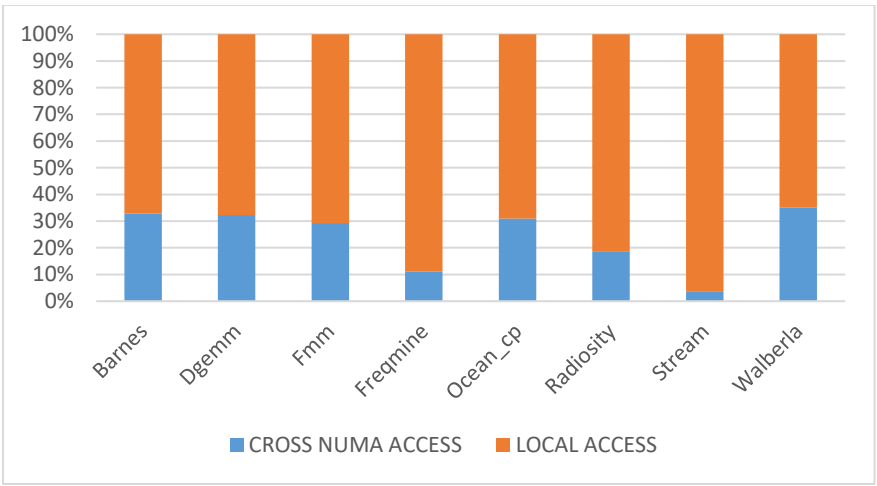

Figure 5. DDR Memory Read Activity.

In another register, cross numa counters assist us in analyzing application behavior when NUMA support is enabled. For instance, Figure 5 depicts the ratio of local and cross numa memory read operations. In 5 out of the 8 applications, at least 30% of read memory accesses are not local, i.e. accesses are the result of requests coming from other NUMA nodes. In applications such as STREAM, the majority of accessed data is generated locally by their respective threads, resulting in minimal data transfers across NUMA boundaries.

VI. CONCLUSION

As HPC systems grow in complexity and scale, virtual prototyping tools have become crucial for HW/SW designers to quickly evaluate and refine their designs while reducing associated time and costs. Besides, virtual prototypes should support reconfigurability, feature-rich models, and high simulation speed. In this work, VPSim tool is introduced to tackle the challenges above by leveraging QEMU's high-performance emulation technologies in conjunction with a rich library of models and many other standardized interfaces. It offers an abundance of performance counters and statistics (functional and time-related counters) for profiling and benchmarking purposes with a good tradeoff between precision and execution time. By keeping pace with the latest advancements in the field, VPSim continues to be improved with more features, enhancing its capabilities and delivering better results for evaluation and benchmarking.

ACKNOWLEDGMENT

This work has been funded by European Processor Initiative Specific Grant Agreement No 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). Association for Computing Machinery, New York, NY, USA, Article 3, 1–8. https://doi.org/10.1145/3300189.3300192

[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. USENIX, 41–46.

[3] Salah Eddine Saidi, Amir Charif, Tanguy Sassolas, Pierre-Guillaume Le Guay, Henrique Vicente Souza, and Nicolas Ventroux. 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). Association for Computing Machinery, New York, NY, USA, 43–49. https://doi.org/10.1145/3339985.3358488

[4] 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 : 5th Workshop on Formal Co-Simulation of Cyber-Physical Systems, Dec 2021, Virtual, France. paper 3.

[5] 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). Association for Computing Machinery, New York, NY, USA, 47–52. https://doi.org/10.1145/3522784.3522796

[6] Lilia Zaourar, et al., Multilevel simulation-based co-design of next generation HPC microprocessors. PMBS 2021: 18-29

[7] Reinders, J. Intel Threading Building Blocks: Outfitting C++ for Multi-core Processor Parallelism, O'Reilly Media, Inc., 2007.

[8] J. McCalpin, "Memory bandwidth and machine balance in high performance computers," IEEE Technical Committee on Computer Architecture Newsletter, pp. 19–25, Dec. 1995

[9] C. Bienia, S. Kumar, J. Pal Singh, and K. Li. The parsec benchmark suite: Characterization and architectural implications. In Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques, page 72–81, 2008

[10] S.Cameron Woo, M.Ohara, E.Torrie, J.Pal Singh, and A.Gupta. The splash-2 programs: Characterization and methodological considerations. SIGARCH Comput. Archit. News, 23(2):24–36, may 1995.

[11] J. J. Dongarra, J. Du Croz, S. Hammarling, and I. S. Duff, "A set of level 3 basic linear algebra subprograms," ACM Transactions on Mathematical Software, vol. 16, no. 1, pp. 1–17, Mar. 1990.

[12] C. Feichtinger, S. Donath, H. K¨ostler, J. G¨otz, and U. R¨ude, "WaLBerla: HPC software design for computational engineering simulations," Journal of Computational Science, vol. 2, no. 2, pp. 105–112, May 2011.