Multilevel simulation-based co-design of next generation HPC microprocessors

Lilia Zaourar, Mohamed Benazouz, Ayoub Mouhagir, Fatma Jebali, Tanguy Sassolas, Jean-Christophe Weill, Carlos Falquez, Nam Ho, Dirk Pleiter, Antoni Portero, et al.

Multilevel simulation-based co-design of next generation HPC microprocessors

Lilia Zaourar, Mohamed Benazouz, Ayoub Mouhagir, Fatma Jebali, Tanguy Sassolas, Jean-Christophe Weill*
*Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France *CEA, DAM, DIF, F-91297 Arpajon, France*
{lilia.zaourar, mohamed.benazouz, ayoub.mouhagir, fatma.jebali, tanguy.sassloas, Jean-christophe.weill}@cea.fr

Carlos Falquez, Nam Ho, Dirk Pleiter, Antoni Portero, Estela Suarez
Jülich Supercomputing Centre, Institute for Advanced Simulation, Forschungszentrum Jülich GmbH, Jülich, Germany
{c.falquez, n.ho, d.pleiter, a.portero, e.suarez}@fz-juelich.de

Polydoros Petrakis, Vassilis Papaefstathiou, Manolis Marazakis
Institute of Computer Science, Foundation for Research and Technology - Hellas (FORTH), Heraklion, Greece
{ppetrak,papaef,maraz}@ics.forth.gr

Milan Radulovic, Francesc Martinez, Adrià Armejach, Marc Casas
Barcelona Supercomputing Center (BSC), Barcelona, Spain
{milan.radulovic,francesc.martinez,adria.armejach,marc.casas}@bsc.es

Alejandro Nocua
ATOS, Les Clayes-sous-Bois, France
alejandro.nocua@atos.net

Romain Dolbeau
SiPearl, Rennes, France
romain.dolbeau@sipearl.com

***Abstract*—This paper demonstrates the combined use of three simulation tools in support of a co-design methodology for an HPC-focused System-on-a-Chip (SoC) design. The simulation tools make different trade-offs between simulation speed, accuracy and model abstraction level, and are shown to be complementary. We apply the MUSA trace-based simulator for the initial sizing of vector register length, system-level cache (SLC) size and memory bandwidth. It has proven to be very efficient at pruning the design space, as its models enable sufficient accuracy without having to resort to highly detailed simulations. Then we apply gem5, a cycle-accurate micro-architecture simulator, for a more refined analysis of the performance potential of our reference SoC architecture, with models able to capture detailed hardware behavior at the cost of simulation speed. Furthermore, we study the network-on-chip (NoC) topology and IP placements using both gem5 for representative small- to medium-scale configurations and SESAM/VPSim, a transaction-level emulator for larger scale systems with good simulation speed and sufficient architectural details. Overall, we consider several system design concerns, such as processor subsystem sizing and NoC settings. We apply the selected simulation tools, focusing on different levels of abstraction, to study several configurations with various design concerns and evaluate them to guide architectural design and optimization decisions. Performance analysis is carried out with a number of representative benchmarks. The obtained numerical results provide guidance and hints to designers regarding SIMD instruction width, SLC sizing, memory bandwidth as well as the best placement of memory controllers and NoC form factor. Thus, we provide critical insights for efficient design of future HPC microprocessors.**

***Index Terms*—Co-design, Simulation, Emulation, Benchmarking, HPC**

I. Introduction

Designing high-performance, multi-core processors to satisfy the needs of a given set of end users is a highly complex process that requires bringing together vastly different expertise and settle on complex architectural trade-offs. The European Commission and the EuroHPC Joint Undertaking have set up a strategy to support the development of European technologies for high performance computing (HPC). One cornerstone of this strategy is the European Processor Initiative (EPI), which aims at creating a new family of high-performance processor and accelerator technologies designed in Europe [26]. Its two main elements are a general purpose processor based on the Arm architecture and an accelerator implementing the RISC-V instruction set architecture. With this portfolio, the EPI processing units address the requirements of HPC while keeping larger market sectors in mind, including the automotive, cryptography, artificial intelligence industries, and trusted IT infrastructures, among others.

In EPI, processor development is driven via *co-design*, a bi-directional and iterative interaction process between application owners, hardware- and system-software developers. The success of the co-design strategy relies on establishing an efficient methodology to tightly connect application and benchmark experts with the hardware and software development team and across the various members of the project. This interaction is achieved by using common tools and methods, naturally establishing a common language between all involved parties. Co-design considers the entire system stack from underlying hardware technologies to software frameworks and applications.

The EPI co-design methodology is based on a multi-level hierarchy of models and simulators providing growing levels of precision at an increasing evaluation time, including reference platforms, analytical models, trace-based simulators, functional simulators, FPGA emulators, and RTL. The former (higher level ones) provides fast response to design questions while more precise insights are obtained from detailed simulation tools. Quantitative user-requirements are determined running a selection of benchmarks and applications on this set of simulation and modelling tools, in order to study the impact of specific design parameters onto application's performance, energy efficiency, and cost. This paper describes and exemplifies the architecture analysis and insights that can be obtained in the context of chip design by combining different simulation tools to gain insight into the impact of alternative design parameters onto application performance and system efficiency. For brevity, energy efficiency and chip area are out of the scope of this paper.

Given the trade-offs between simulation speed, accuracy and model abstraction level, we have selected three complementary simulation tools to support a full co-design methodology: MUSA, SESAM/VPSim and gem5. We consider several design concerns, such as processor SIMD instruction width, NoC topology, as well as placement and sizing (processor, on-chip memory and memory controllers). We exploit the selected tools, focusing at different levels of abstraction, to study several configurations with various design concerns and guide architectural design and optimization decisions.

The main contributions of this work are (i) a multilevel simulation-based co-design methodology, enabling (ii) a thorough study of key design concerns of modern HPC processors based on the Arm Neoverse V1 Reference Design. In this work we assume support of Arm's new Scalable Vector Extension (SVE) instructions, which allows SIMD instructions having a width between 128 and 2048 bits. Section II puts the current paper in the context of other work performed in the same area. Section III presents the methodology applied, and the three used modelling tools: MUSA, SESAM/VPSim and gem5. Section IV introduces the HPC processor under study and raises design concerns addressed thanks to experimental results provided by the co-design methodology. Lastly, conclusions and perspectives are presented in Section VI.

II. Related work

Co-design and simulation of multiprocessors are both important and challenging steps for computer architects. This is due to the increasing complexity of design and the need for early prototypes as a way of providing feedback and validation of requirements and design specifications. This section provides an overview of the different approaches in this context without being exhaustive.

Many existing solutions take advantage of functional Instruction Set Simulators (ISS), which execute program instructions while maintaining the internal processor registers to emulate the behavior of microprocessors. *gem5* [11] is one of the most popular simulators, providing detailed simulation of CPU models with full pipelining description. It also offers the flexibility to evaluate systems at various abstraction levels that cover a wide range of speed/accuracy trade-offs [12]. Other solutions [7], [47] use instruction-based ISS solutions to model MPSoC environments, but they lack flexibility. While offering a high degree of accuracy, slow simulation speed is expected when addressing a detailed multiprocessor system, such as complex systems embedding multiple CPUs and running full-fledged OSs.

To address this complexity without hampering the simulation speed, other approaches use Dynamic Binary Translation (DBT) in an event-driven simulation to provide high simulation speed while being able to maintain instruction accuracy. Among the open source virtual platforms, OVP shows promising performance reaching hundreds of MIPS [31]. It uses a DBT-engine called OVPSim, which is dedicated to various processor, peripheral and platform models that are available as open source software. OVPSim supports parallel simulation for even higher performance, but this is limited to non-deterministic simulation. Another DBT-based solution is Graphite, whose functional core model relies on Pin [32]. In addition to parallelizing the functional and timing aspects of the simulation, Graphite is also a distributed simulator designed to enable the study of large-scale multicore architectures, which would be otherwise unachievable with sequential and cycle-accurate simulators.

Many full system simulation approaches take advantage of the open source QEMU emulator [8]. QEMU is the state-of-the-art DBT-based machine emulator, known for its good compromise between simulation performance and porting complexity. It supports multiple hosts and target processors including x86, Arm, RISC-V, MIPS, SPARC and Alpha. In [15], [20], [35], QEMU models are integrated into an event-driven simulation environment such as SystemC while in [27] they are coupled with an analytical timing model. Approaches based on the integration of QEMU and SystemC differ mainly in the way they interface the two simulation environments.

Many industrial solutions, such as Virtualizer [46] or Fast Models [41] are available but have a significant cost and little degree of customization for the needs of co-design activities.

A number of frameworks that target simulations with thousands of nodes have been proposed in the past [16], [23], [52], however, these frameworks focus mainly on network events, and they do not model CPU components or system software interactions in detail. More recently, Wang et al. [48] presented a multi-level simulation framework capable of modeling in detail many parts of the system architecture, including power estimations; but is limited to single node applications. Similarly, SST [40] is a multi-scale simulator often used in combination with other simulators to model distributed applications. In BE-SST [39], authors combine SST

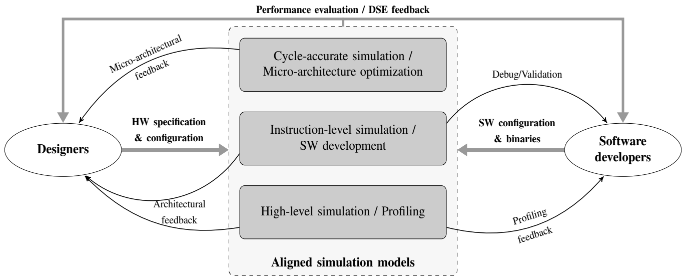

Performance evaluation / DSE feedback
Micro-architectural feedback
Designers
HW specification & configuration
Architectural feedback
Cycle-accurate simulation / Micro-architecture optimization
Instruction-level simulation / SW development
High-level simulation / Profiling
Aligned simulation models
Debug/Validation
SW configuration & binaries
Software developers
Profiling feedback

Fig. 1: Overview of the co-design methodology

with coarse-grained behavioral emulation models abstracting from microarchitectural details in favor of simulation speed. Other implementations integrate SST with a highly accurate simulator but require too costly full system simulations to produce a wide set of experiments [25].

The EPI itself uses multi-level simulation for purposes others than co-design. For instance, software development and tuning is done using a combination of full-system simulation (such as QEMU), instruction set architecture (ISA) extension simulation (Arm Instruction Emulator [2], Vehave [13]) and RTL-based simulation of the Neoverse V1 core. Simulators like gem5 have recently also been used to co-design another Arm-based processor, namely the A64FX processor where Fujitsu and RIKEN entered into a co-design process [29], [44]. Their evaluation, in particular, considered the SVE width [50].

III. Co-design Methodology

Virtual prototype tools are essential to perform Design Space Exploration (DSE) of future HPC microprocessors. In fact, we need to simulate several configurations with various design concerns and evaluate them to make the best architectural design decisions. For efficient exploration, the tools must be reconfigurable and fast enough to run multiple iterations, as is required for architectural exploration as well as software debugging and optimization. Meeting the above requirements is challenging, as the simulation speed slows down when the architectural complexity and the number of processors increase. Hence, a continuous set of trade-offs between simulation speed and model abstraction levels must be considered. Therefore, we selected three simulation tools: MUSA, SESAM/VPSim and gem5, which are used for different purposes to obtain a full co-design methodology as depicted in Fig. 1. Each of them is at a different level, ranging from architectural exploration and analysis to SW development with different runtimes. The following sections present details and features of each one.

A. MUSA

The MUlti-level Simulation Approach (MUSA) is an end-to-end methodology that uses traces to enable large-scale simulations with different communication networks, numbers of cores per node, and micro-architectural parameters in a comprehensive HPC environment that considers the effects of system software [21], [22]. To this end, MUSA employs two components: (i) a tracing infrastructure that captures communication, computation and runtime system events; and (ii) a simulation infrastructure that leverages these traces for simulation at multiple levels. Fig. 2 illustrates its modular methodology that provides a streamlined workflow from tracing to the final simulation output. MUSA's simulation infrastructure is able to change the level of simulation detail, from detailed microarchitectural simulations to high-level analytical models.

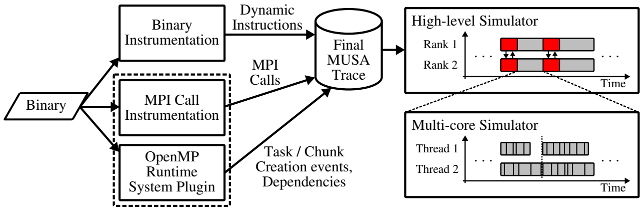


Fig. 2: Overview of the MUSA simulation infrastructure

Therefore, the MUSA methodology allows combining detailed (higher computational cost) and high-level (higher simulation speed) simulations, with simulation speeds of several MIPS. This enables large design space exploration studies both at the socket level and of large-scale machines with thousands of cores, in a reasonable amount of computational time; while guaranteeing a high degree of accuracy. MUSA supports instruction tracing both for Arm binaries, with a DynamoRIO [18] plugin, and RISC-V binaries, using the Vehave emulator. MUSA can simulate these detailed instruction traces using a parameterized architectural simulator that models an out-of-order multicore and a full memory hierarchy.

The out-of-order model considers dependencies between the most fundamental vector memory instructions: load, store, gather, and scatter. For loads and stores, dependencies are considered when an in-flight memory operation is modifying a subset of the addresses of the incoming memory operation. For the case of gather and scatter instructions, these are split into N loads or stores, respectively. For arithmetic instructions, MUSA does not explicitly take into account their dependencies. Instead, we consider the dependencies of arithmetic instructions to be solved once the previous memory operation becomes ready to be executed. This simplified dependency model allows MUSA to simulate complex workloads in a short amount of time. Regarding the memory hierarchy, MUSA can be configured with multiple cache levels, each with its size, associativity, miss status holding registers, latency, etc. Main memory can be simulated via a simple model that considers a fixed latency per access and allows a maximum number of outstanding accesses; or it is also possible to hook detailed simulators like DRAMSim2 [42] or Ramulator [28].

MUSA enables understanding application performance while tuning multiple architectural parameters at large-scale. However, it is not a cycle-accurate simulator and it lacks detailed models for certain system components, e.g. NoC. While MUSA is able to model key system software events such as task scheduling, it does not enable booting a fully fledged Operating System (OS), as supported by SESAM/VPSim.

B. SESAM/VPSim

SESAM/VPSim [14], depicted in Fig. 3, is designed for SW/HW co-validation at early design stages. It provides flexible and highly-configurable framework for SW design and architectural exploration. The core of the simulation is *VPSim Platform Builder*, which takes as input a high-level platform specification written in a Python-based Domain Specific Language (DSL). From this high-level specification, a platform can be built using an internal hardware library as well as *proxy* components to interface with external subsystems, including SystemC, Python, and Functional Mockup Interface. One of the main advantages of SESAM/VPSim is its ability to dynamically configure the modeled platforms, by instantiating virtual platforms at runtime without recompiling the simulator. This offers a high flexibility in constructing the platform and making adjustments during iterative HW and SW validation.

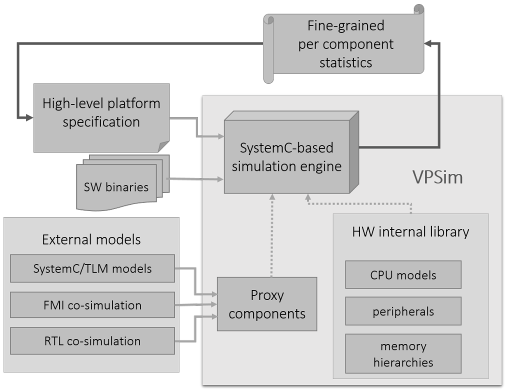


Fig. 3: Overview of the SESAM/VPSim platform

SESAM/VPSim has a rich library of CPU models, including Arm and RISC-V, which are mainly provided by QEMU, enabling very high simulation speed to be reached. QEMU models are encapsulated in loosely-timed SystemC/TLM 2.0 modules and executed in the context of SystemC threads. For performance estimation, QEMU models are complemented with a SystemC library. It models various peripherals, memory hierarchy components and implements HW counters.

Simulation models are highly-configurable, enabling to evaluate the performance of different platform configurations and find a suitable one that best meets SW/HW requirements. On top of the constructed platform, full software stacks (e.g. BIOS, hypervisor, user space workloads) can be run, profiled and debugged on the simulated platform. Since SESAM/VPsim is not a cycle-accurate simulator, it must be completed with more detailed micro-architectural parameters study thanks to gem5.

C. gem5

The gem5 simulator [11], [30] is a cycle-accurate computer architecture simulator, capable of modeling a variety of hardware platforms. It provides models of varying complexity for CPU cores, memory devices, coherent caches and on-chip networks, which can be combined in a modular fashion. An important component of gem5 is Ruby, which provides a comprehensive model for the memory subsystem, and enables the exploration of alternative cache organizations, interconnection networks, as well as cache coherency protocols (with the SLICC DSL [30]). In particular, the Arm AMBA CHI protocol [3] has been recently implemented for gem5 Ruby [36]. The protocol implementation defines two basic components: cache and memory controllers, implemented as state machines. The cache controller can function as private L1/L2 cache or as System Level Cache (SLC). When modeling a private cache, the CHI cache controller acts as a Request Node (RN), while functioning as a Home Node (HN) when modeling an SLC. Most CHI transactions are implemented, and the block allocation and replacement policy can be configured.

Once instantiated, CHI nodes communicate over a simulated on-chip network. The gem5 on-chip network interconnect model is provided by the Garnet subsystem [1]. Garnet supports detailed simulation of network traffic and timing effects by modeling network routers at the micro-architectural level.

The main feature of gem5 is the ability to collect highly detailed statistics for applications running on the simulated hardware platform. This allows studying the effect of different hardware design parameters on performance, and makes gem5 ideally suited for co-design investigations. In this paper, we use gem5 in full-system mode, so that the simulation supports the execution of both unmodified binaries (libraries and applications) and operating system (Linux).

The accuracy of the simulations, however, results in relatively long simulation times. The gem5 simulator is therefore suited for benchmarking, e.g., mini-applications [24].

IV. Experimental methodology

A. Reference Architecture

We have selected as a reference architecture the Arm Neoverse V1 Reference Design (RDV1) [6]. RDV1 uses the latest Neoverse V1 core [5] that targets high-performance and exascale computing markets. Fig. 4, shows the top level RDV1 architecture. The processor element contains high-performance Armv8.4-A Neoverse V1 cores, which include micro-architectural improvements over previous Arm architectures, and is the first implementation of the Scalable Vector Extension (SVE) [45] by Arm, with two 256 bits wide units (16 double-precision Flop/cycle). The core has separate 64 KiB L1 data and instruction caches, and a private unified data and instruction 1 MiB L2 cache.

The interconnect element contains the Arm CoreLink Coherent-Mesh-Network 650 (CMN-650), which is a high bandwidth and low-latency system interconnect that supports a wide range of applications. It is a highly scalable mesh optimized for Armv8-A processors that can be customized to meet system performance and area chip requirements. The SLC provides CPUs and I/O requesters with a distributed, scalable, low-latency, high-bandwidth SRAM for evicted or stashed data. CMN-650 provides the option to connect either CHI-based memory controllers or AXI-based memory controllers using a AMBA5 to ACE5-lite bridge. This enables

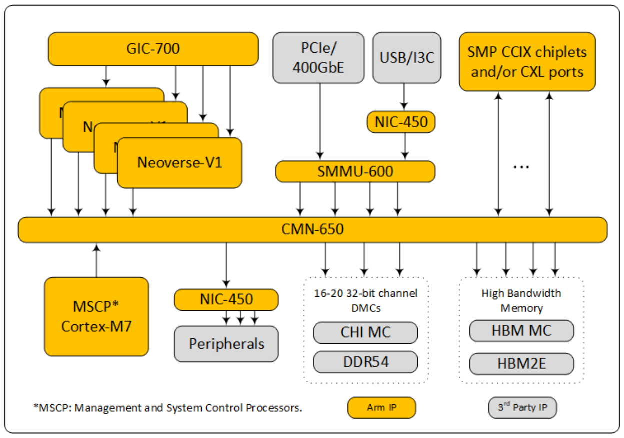


Fig. 4: Top Level RDV1 Architecture

memory elements such as HBM memory stacks and DDR5/4 memories to be directly integrated into the system.

The RDV1 has three main configurations based on the mesh size and the total number of connected processing elements. Config-M presents a 3 × 5 size mesh that can interconnect up to 16 cores. It interconnects 16 Neoverse V1 cores running at 1.6 GHz. It includes 1 MiB of SLC per core, distributed across the mesh and DDR memory controllers. Config-L is a scaled version with a 6 × 6 mesh that can interconnect up to 32 cores, with a 32 MiB SLC. It also includes several bridges to interconnect HBM and/or DDR memory controllers. Figure 5 shows a detailed view of the right-bottom quadrant mesh.

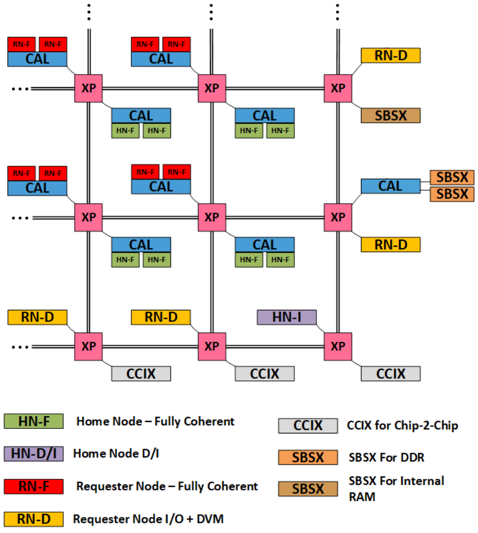

RN-F
CAL
XP
HN-F
RN-D
SBSX
HN-I
CCIX
HN-F Home Node – Fully Coherent
HN-D/I Home Node D/I
RN-F Requester Node – Fully Coherent
RN-D Requester Node I/O + DVM
CCIX CCIX for Chip-2-Chip
SBSX SBSX For DDR
SBSX SBSX For Internal RAM

Fig. 5: Quadrant detail: RDV1 Config-L

B. Design Concerns

The definition of a large-scale multiprocessor architecture, such as RDV1, raises several architectural questions that our work addresses. In this paper, we highlight three areas where our co-design methodology produced actionable insights for the design of our target HPC-optimized SOC: (i) vector processing resources; (ii) cache and main memory system performance; (iii) NoC topology and IP block placement. First, computing resources must be correctly sized according to application needs. Specifically, the impact of SVE register length of Neoverse V1 cores on performance is one key focus of our study. Then, memory access performance needs to be thoroughly evaluated. We consider required on-chip memory size dimensioning and external I/O bandwidth requirements to settle on the number of memory controllers. Finally, HPC designs also raise the question of best NoC topology and IP placement, while considering various design scales.

The initial dimensioning of SVE register length, SLC size and memory bandwidth is done first, thanks to MUSA which is very efficient at pruning the design space for certain architectural components, as its models enable good accuracy without the need to resort to highly detailed simulations. Having selected the most relevant design parameters, refined analysis is conducted with gem5, whose models capture the real hardware behavior at the cost of simulation speed. NoC topology and IP placements are studied using both gem5 for maximal accuracy and SESAM/VPSim for larger scale systems with good simulation speed and sufficient architectural detail. This speed also allows the usage of real-case parallel applications to increase confidence in observed behavior.

C. Benchmarks

To study the behavior of the architecture for its various configurations, a set of HPC benchmarks were selected to specifically stress each design aspect of the target architecture. We use DGEMM (Double-precision, GEneral Matrix-Matrix multiplication [17]), developed within the BLIS framework [51], as a compute-bound benchmark for assessing CPU performance. The binaries were compiled with fixed SVE vector length. We use STREAM (TRIAD) [34] as a representative benchmark for HPC applications sensitive to the available system bandwidth. We choose WaLBerla [19] as an example of a stencil kernel. For the STREAM benchmark, the percentage of memory bus serves as our performance measure. For WaLBerla, we use the attained number of Million Lattice Updates Per Second (MLUPS). These benchmarks are complemented with more HPC-targeted applications that mimic large-scale emerging programs. In this category, the PARSEC [10] and SPLASH-2 [49] benchmark suites are selected for multiprocessor architectures with shared memory.

D. Multi-level consistency Validation

For the proposed multilevel methodology to be valid, simulators used through the different design stages must deliver comparable values on basic performance hardware counters for the core and memory hierarchy, such as executed instructions and committed instructions, operations, reads, writes, SIMD, floats and integers. Counters also include the number of accesses for L1D, L1I, L2 and SLC accesses, misses, and bandwidths from the memory layers. Comparison study was performed in two steps. First, performance counters reported by the simulators were compared through microbenchmarks:

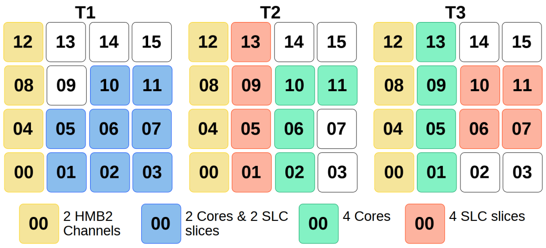


Fig. 6: NoC topologies for gem5 exploration

FMLA-Bench using 30 concurrent vector instructions, and a simplified version of STREAM (TRIAD). Second, results from gem5 were proof-checked against an equivalent real machine (i.e. Neoverse N1) [4] using *libfmp4* library. The main counters of the microbenchmarks between the simulated and real architecture are identical, with errors lower than one per cent. This paper does not dwell on experiments performed to ensure good simulators alignment but will now exploit complementarity of the tools to address specific design concerns raised by RDV1.

E. Architectural exploration and setup

All relevant architectural simulation parameters are summarized in Table I. Fixed parameters for the study are shown in regular font, while **explored** parameters are shown in bold. All simulators used in this study reproduce the target architecture at different levels of abstraction, so that the model parametrization used by one simulator does not necessarily correspond to the model parametrization of another. The gem5 simulator in particular takes as cache model parameter the number of *transaction buffer entries* (TBEs[1]), which serve as generalized version of the more familiar miss status holding registers (MSHRs).

The gem5 network interconnect is simulated using a recently improved version of Garnet [9], which supports the simulation of multiple physical links between routers, and allows the investigation of increased NoC bandwidth. The cache coherency protocol used in gem5 is an implementation of the AMBA CHI specification, as discussed above. The gem5 memory controller models an HBM2 device taken from the gem5-X project [37], [38]. The simulated HBM2 controller delivers a peak bandwidth of up to 38.4 GB/s per channel.

The NoC topology layout, i.e. the placement of different components on the NoC, is of critical importance for system performance. Using gem5, we examine the effect of CPU and SLC placement by evaluating three distinct 4×4 NoC layouts, corresponding to the North West quadrant of a larger 8×8 NoC, labeled T1, T2 and T3, shown in Fig. 6. In layout T1, CPU cores and SLC slices are distributed homogeneously over 8 routers. Whereas in layout T2, the SLC slices are placed closer to the memory controllers, and T3 places the cores closer to the memory controllers. Further architecture parameters such as private cache latencies and sizes are derived from Arm Neoverse V1 reference design.

The reference model of SESAM/VPSim explores the behavior of larger systems ranging from 32 to 64 cores, modelled using QEMU Arm V8.4-A ISA. The existing memory hierarchy was extended to evaluate consistency traffic with an MSI protocol and included a hybrid NoC model [33] able to account for network congestions based on simulated traffic. The model was validated against Garnet for its accuracy. Based on this setup, three configurations for the placement of four DDR controllers were evaluated as shown in Fig. 7. The first configuration puts DDR blocks around corners. In the second one, DDR controllers are placed in the middle of NoC sides, forming a rhombus shape. Finally, the third configuration groups them two-by-two on the longest opposite sides (for NoCs of rectangle form). Besides, the impact of the NoC size (*i.e.* form factor: square vs rectangle) on the NoC global performance is also studied. Thanks to the simulation speed reached by the model, for these experiments the set of benchmarks was extended with nine more applications from the PARSEC and SPLASH-2 set. Running these applications on a single configuration generates around 11.4 Billion packets on the NoC, increasing the confidence of the results for real-case applications.

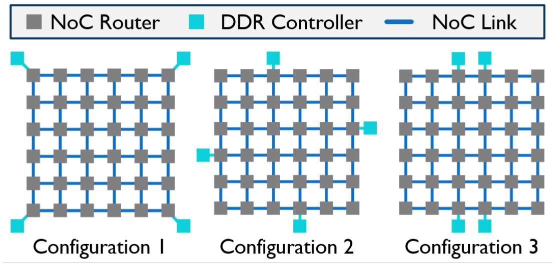


Fig. 7: DDR controllers NoC positions for 32 cores

We detailed MUSA's out-of-order pipeline and cache hierarchy models in Section III-A, and simulations use the experimental parameters shown in Table I. Additionally, the main memory system in MUSA is handled by Ramulator [28], which is set to model HBM2 memory controllers and devices. Therefore, Ramulator is interfaced with MUSA; all main memory requests from MUSA are forwarded to Ramulator memory controllers as they are generated.

V. Numerical Results and Analysis

To answer the design questions raised by the RDV1, this section summarizes key results obtained using the proposed co-design methodology. The studies cover a range of different SVE register lengths, memory sizes, memory bandwidths, NoC topologies, and IP placements.

A. SVE register length

Longer vectors can provide large performance benefits if exploited efficiently. This largely depends on (i) workload characteristics, i.e., the amount of code that is vectorized; and

[1] Within gem5 Ruby, TBEs are used to keep track of protocol transactions, see [36] and the gem5 Ruby documentation for a more detailed description.

	Fixed/**explored** architectural parameters		
	gem5	MUSA	SESAM/VPSim
Clocks	System: 1.6 GHz; CPU: 2.4 GHz; NoC: 2.0 GHz;	2.0 GHz	CPU: 1.0 GHz (one instruction per cycle, per core)

CPU	#Cores: 16; Adjusted A76; Branch Pred.: BiMode;
	Vector Unit: 2xSVE; **SVE length:{256, 512}**
		#Cores: 32; ROB: 256 entries; 4-wide issue/commit
		Vector Unit: 2xSVE; **SVE length:{128, 256, 512, 1024}**
			#Cores: 32, 64
			Vector Unit: 1xSVE; SVE length: 256

L1	Line size: 64B; Size: 64 KiB; Associativity: 4-way;
	Inclusion policy: strict inclusive ; TBEs: 256;
	Hit latency: 2-cycles (L1-D), 1-cycles (L1-I);
		Line size: 64B; Size: 64 KiB; Associativity: 4-way;
		Inclusion policy: strict inclusive ; MSHRs: 64;
		Hit latency: 2-cycles (L1-D)
			Line size: 64B; Size: 64 KiB; Associativity: 4-way;
			Inclusion policy: NINE
			Hit latency: 2-cycles (L1-D);

L2	Unified cache; Line size: 64B; Size: 1 MiB;
	Associativity: 8-way; Hit latency: 4-cycles;
	Inclusion policy: strict inclusive ; TBEs: 256
		Unified cache; Line size: 64B; Size: 1 MiB;
		Associativity: 8-way; Hit latency: 8-cycles;
		Inclusion policy: strict inclusive ; MSHRs: 64
			Unified cache; Line size: 64B; Size: 1 MiB;
			Associativity: 8-way; Hit latency: 4-cycles;
			Inclusion policy: NINE

SLCs	Shared SLC cache; #Slices: 16; Line size: 64B;
	Associativity: 16-way; Hit latency: 20-cycles;
	Inclusion policy: Exclusive; TBEs: 256 per slice;
	Size per slice: {1 MiB, 3 MiB}
		Shared SLC cache; One slice per core; Line size: 64B;
		Associativity: 16-way; Hit latency: 22-cycles;
		Inclusion policy: Exclusive; TBEs: 64 per slice;
		Size per slice: {256 KiB, 512 KiB, 1 MiB, 2 MiB, 4 MiB}
			Shared SLC cache; One slice per core; Line size: 64B;
			Associativity: 16-way; Hit latency: 10-cycles;
			Inclusion policy: Exclusive;
			Size per slice: 2 MiB

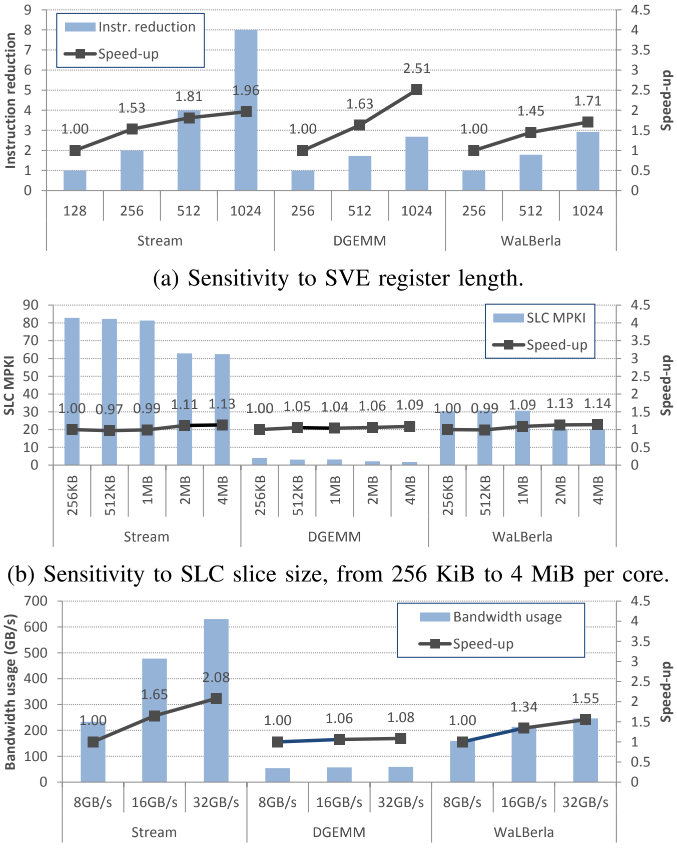
NoC	Model: Garnet 3.0; Protocol: AMBA-CHI; Mesh: 4x4;
	Flit width: 64B; Router latency: 1-cycle;
	Link latency: 1-cycle; #VNETs: 4; XY Routing
	Topology: {T1, T2, T3};
	Link configuration: {single, multiple}
		Simple crossbar model with fixed 6-cycle latency;
		SLC slices connected to memory controllers
			Hybrid NoC model [33]
			Buffer size flits: 1; Router latency: 1-cycle
			Link latency: 1-cycle; #VNETs: 1;
			Topology: {Mesh 6x6, 4x8, 8x8, 6x12};

Memory	Model: HBM2; #Channel: 8; Size: 2x8 GiB Bandwidth per channel: 38.4 GB/s	Model: HBM2; Bandwidth per channel: 32 GB/s; **Bandwidth per core (GB/s): 8, 16, 32**	Model: DDR; #Channel: 8; Size: 4x1 GiB **Memory blocks XY position in the NoC**

TABLE I: Details of fixed and **explored** parameters setup with the methodology.

(ii) vector unit usage, i.e., by providing them with a constant stream of instructions to process. The latter requires the core to expose enough instruction-level parallelism, and the memory hierarchy has to provide sufficient bandwidth to bring in the demanded data.

1) Early SVE register length: As shown in Table I we perform experiments with MUSA for SVE lengths of 128, 256, 512 and 1024 bits. Fig. 8a shows the instruction reduction achieved as the SVE vector length increases, as well as the obtained speed-up. For STREAM we obtain an almost linear instruction reduction, as the main loop is composed of just a few SVE instructions and there is no scalar code. However, increasing the vector length has diminishing returns for lengths above 256 bits, as the benchmark is memory bound. DGEMM and WaLBerla present smaller instruction reductions: 1.72× and 1.78× when going from 256 to 512 bit SVE, and 2.67× and 2.91× when going from 512 to 1024 bits, respectively. In contrast, the obtained speed-ups are larger than in STREAM. DGEMM scales remarkably well up to 1024 bit SVE, and WaLBerla obtains significant benefits from 512 bit vectors. While scaling to 1024 bits SVE for many benchmarks is challenging, SVE units of 256 or 512 bits are good design points. Making a more informed decision between these two requires further analysis, as described below.



(a) Sensitivity to SVE register length.

(b) Sensitivity to SLC slice size, from 256 KiB to 4 MiB per core.

(c) Sensitivity to available memory bandwidth per core.

Fig. 8: Evaluation of different key architectural parameters on 32 cores using MUSA with the parameters shown in Table I.

2) Refined Impact of SVE register length: A further study of the impact of SVE vector lengths is also performed with gem5. Fig. 9 shows instruction reduction and speedup when evaluating 256- and 512-bit SVE vector lengths for the topology T1. Fig. 9 shows that the breakdown of committed instructions indicates high vectorization achieved with memory and SIMD operations. When scaling vector length from 256 to 512 bits, the total committed instructions of DGEMM are reduced to a half. Meanwhile, the two memory-bound kernels STREAM (TRIAD) and WaLBerla present around 40% reduction. The impact of scaling to larger vector lengths for the two memory-bound kernels is twofold: the effect of the Write allocation mechanism (loading data into cache before updating), and the benefits of instruction reduction. In the former, due to fitting 64B cache line size on write miss when using 512-bit vector length, Write allocation is not necessary. For the STREAM (TRIAD) kernel, this translates into a reduction in memory operation usage from 3 LDs and 1 ST, to 2 LDs and 1 ST, when scaling vector length from 256 to 512 bits. In the latter, reducing instruction when scaling vector length could lead to reducing throughput on register file allocation, and thus fewer stalls in the pipeline execution. To this end, speedups for DGEMM, STREAM (TRIAD), and WaLBerla are 1.63×, 1.34×, and 1.24× respectively.

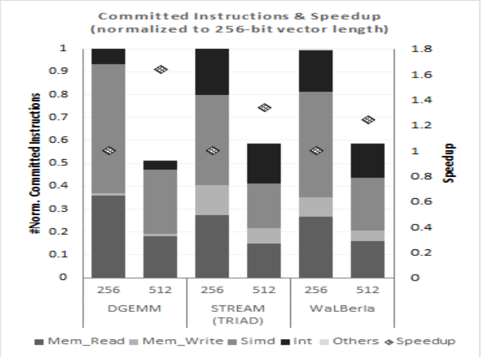


Fig. 9: Impacts of 256- and 512-bit SVE lengths evaluated for Topology T1 (16 threads)

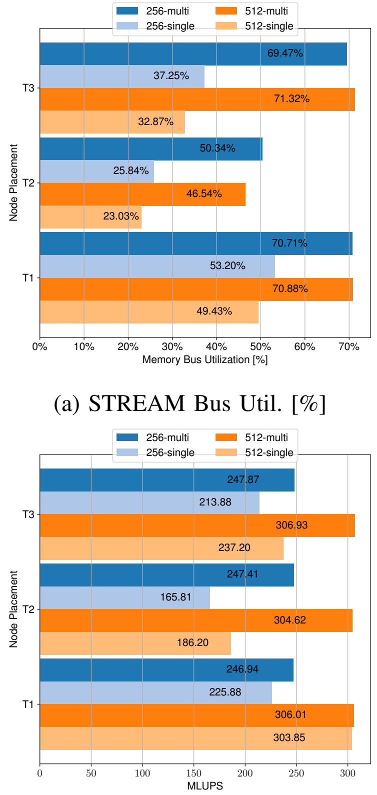
B. On-chip memory

Careful sizing of the SLC is paramount to design a processor that exhibits a good balance between storage and compute capabilities. An over-provisioned SLC wastes resources that can be used to increase computational throughput by, for example, adding more cores. Therefore, finding the appropriate SLC size for the target workloads, i.e., a size that enables capturing most of the performance of larger caches, is a key design goal that can make a significant difference in the overall chip performance. As shown in Table I we perform experiments with SLC slices of 256 KiB, 512 KiB, 1 MiB, 2 MiB and 4 MiB per core using MUSA.

Fig. 8b shows the misses per kilo instruction (MPKI) and performance speed-up for different SLC slice sizes. In STREAM the 2 MiB SLC slice configuration attains a 1.12× speed-up over the 1 MiB slice. While on DGEMM and WaLBerla the 512 KiB and 1 MiB slice sizes already capture most of the benefits, respectively. Therefore, sizes between 512 KiB and 1 MiB are likely to provide a good balance.

C. Memory bandwidth

1) Bandwidth requirements: Memory bound applications are common in the HPC domain. Therefore, providing the cores with sufficient memory bandwidth to feed the execution units is one of the main bottlenecks in most systems. This is exacerbated by the fact that as the length of the vector units increases, workloads become even more memory bound. As shown in Table I, we perform experiments with MUSA by setting the amount of memory bandwidth available per core to 8, 16 and 32 GB/s. Fig. 8c shows the memory bandwidth usage when changing the available bandwidth per core between 8, 16 and 32 GB/s. That is, peak bandwidths of 256, 512 and 1024 GB/s. On memory bound benchmarks like STREAM, increasing the available bandwidth leads to significant performance improvements. With 16 GB/s per core we reach a bandwidth usage of 93%, while with 32 GB/s per core it drops to 61%; as cores are not able to produce enough requests to saturate it. In WaLBerla, using 16 GB/s per core also provides significant improvements, of 1.34×. DGEMM is computed bound and 8 GB/s per core is sufficient to feed the functional units. Given that memory bound applications are common, more bandwidth is desirable. Having a bandwidth of around 20 GB/s per core can likely capture most of the performance benefits seen on the 32 GB/s configurations, as that is the maximum amount of bandwidth used on STREAM per core.

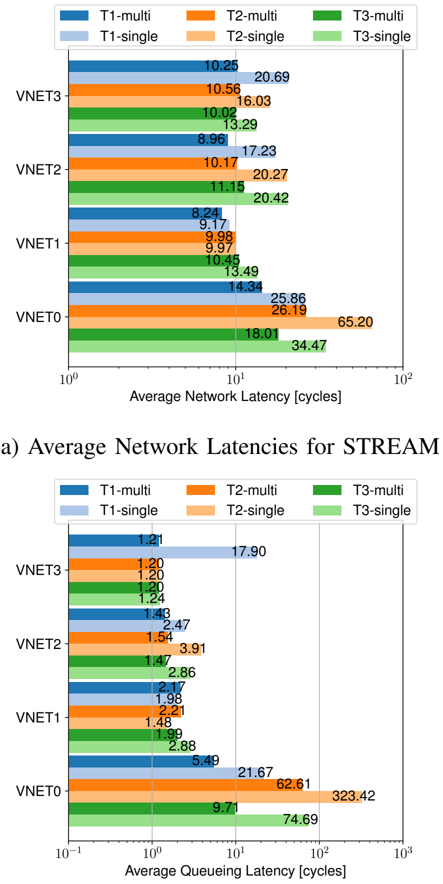


(a) STREAM Bus Util. [%]

(b) WaLBerla MLUPS

Fig. 10: STREAM (TRIAD) memory bus utilization and WaLBerla (16 threads, 3MiB SLC) MLUPS.

2) Refined bandwidth evaluation: To achieve the necessary memory bandwidth as evaluated with MUSA, NoC bandwidth is crucial. To shed light on this area, gem5 has been used to assess NoC routers performance. Fig. 10 shows performance of STREAM-TRIAD and WaLBerla (16 threads) for each of the topology layouts T1, T2 and T3, using single- and multi-channel VNET support, with 256 and 512 SVE vector length, and 3 MiB SLC size. The performance effects of node placement will be discussed below, here we focus on the effect of additional network links. From Fig. 10, it is evident that both STREAM and WaLBerla benchmarks benefit from increased NoC bandwidth. For the T1 topology, STREAM goes from around 50% bus usage to around 71%, for both 256 and 512 SVE register length. T2 shows similar improvement, as does



(a) Average Network Latencies for STREAM

(b) Average Queueing Latencies for STREAM

Fig. 11: STREAM (TRIAD) NoC latencies (VL=256, 16 threads, 3MiB SLC).

T3, although the T2 percentage bus usage only reaches around 50% of maximum when using multiple links, and both T2 and T3 show lower bandwidth for the single link configurations. Fig. 11a and Fig. 11b show the average network and queueing latencies for STREAM-TRIAD running on the T1, T2 and T3 topology layouts. Queueing latency is defined by gem5 as the time a NoC packet has to wait in the egress NoC Network Interface until it gets injected in the NoC. NoC Network latency, on the other hand, is defined as the NoC traversal time, measured by gem5 from the moment a NoC packet is injected in the NoC by the source Network Interface, until the moment it reaches the destination Network Interface. As expected, all topologies benefit from increased bandwidth. For all topologies, the multiple link usage significantly reduces the network latencies of the Request, Response and Data VNETs, and the queueing latencies for the Request VNET. For T1, the increased link bandwidth also drastically reduces the Data (VNET3) queueing latency.

The WaLBerla benchmark also shows a performance improvement when using multiple links, although the performance improvement depends strongly on the CPU/SLC placement, and the SVE register length. For the T1 layout and a vector length of 256 bit, the additional link bandwidth improves the performance from $225.88 \cdot 10^6$ to $246.94 \cdot 10^6$ lattice updates per second (LUPS), which represents a speedup of around 10%. On the other hand, for T1 with a 512-bit vector length, the performance improvement is less than 1%. Layouts T2 and T3 show greater sensitivity for reduced bandwidth. Performance for multi-link configurations are similar for all topologies, but the single-link configurations always perform worse than T1 single-link.

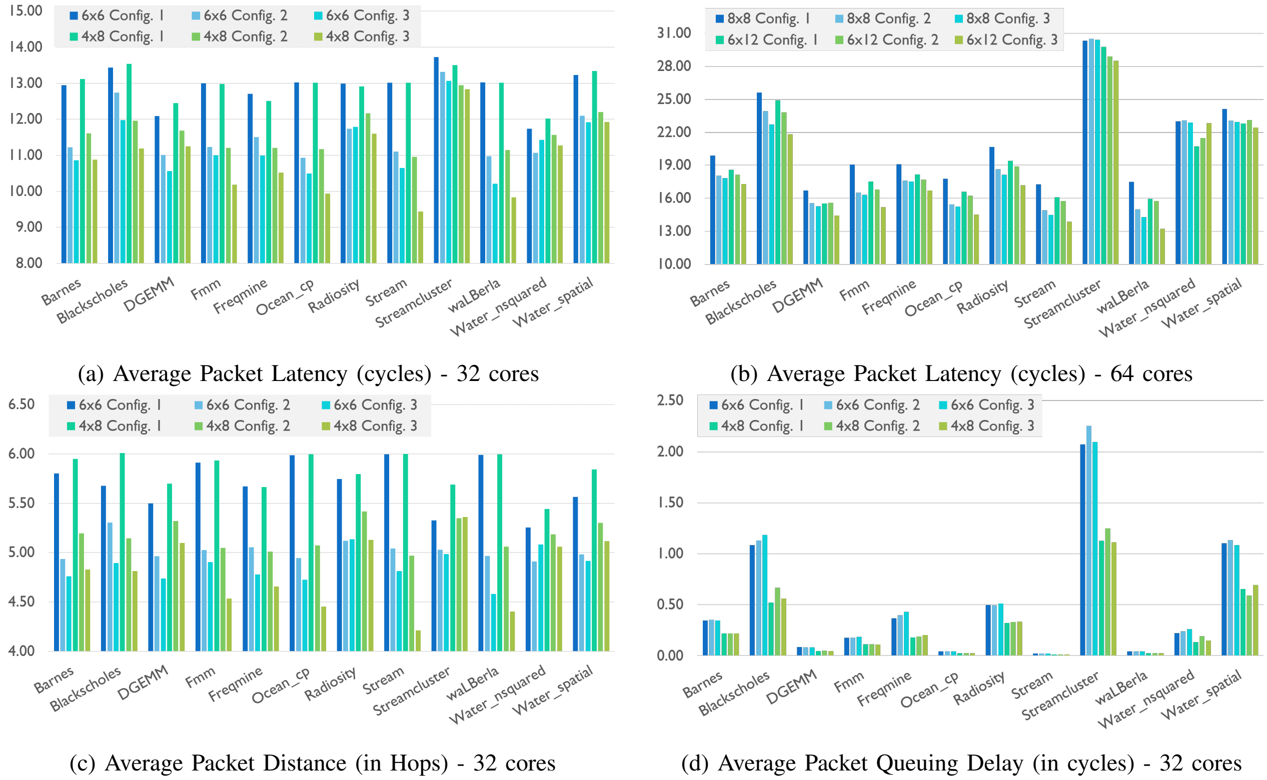
As expected, the increased NoC bandwidth translates to increased benchmark performance for memory sensitive kernels. The magnitude of the speedup depends strongly on NoC placement and SVE register length.

D. NoC dimensioning and IP placement

1) Node placement: Using gem5, we have evaluated the effect of node placement on the performance of STREAM-TRIAD (SVE-256, 16 threads, 3MiB SLC, multi-link), measured by percentage of Memory Bus usage (Fig.10a). As we can see in Fig.11, T1 has the lowest NoC queueing and network latencies, and is the best performing Topology (reaching 71% memory bus utilization (in %)). Topology T3 presents slightly higher NoC Queueing and Network latencies than Topology T1, and achieves comparable bandwidth performance. On the other hand, Topology T2, the only layout with two 4-Core Routers in the same Mesh row, shows the worst performance with 50% Memory Bus. Having that many RNFs, in the same row, in combination with the XY NoC Routing algorithm, causes the VNET-0 (Requests) Queueing latency to increase more than 10-fold, when compared to T1. Also, the Network latency for the same VNET is doubled.

The average number of NoC hops for T1, T2, and T3 is 2.3, 2.5 and 2.7, respectively. The average hops per cycle for the three topologies are 19.6, 15.2 and 22.7. Although T3 requires higher NoC hops for STREAM TRIAD benchmark, it does not get congested and maintains a high hops/cycle rate. These factors contribute to T3 performing similarly to T1. Having defined the best processing and memory IP placements, we then focus on evaluating memory controllers positions and NoC placements.

2) DDR memory controllers placement: NoC performance is primarily evaluated using packets average latency with more HPC-oriented applications, thanks to the faster SESAM/VPSim simulation speed. As shown in Fig.12a and Fig. 12b, no matter the form factor of the NoC, Configuration 3 performs better than Configuration 2, which in turn performs better that Configuration 1 (see Configuration definitions in Fig. 12). Indeed, the rhombus shape improves NoC performance in routing packets but not as much as Configuration 3. Detailed analysis of the source of the performance gain shows that it is mostly contributed by the *average packet distance* (Fig. 12c) which clearly improves with Configuration 3. DDR controllers should be placed far from corners in order to reduce the total distance of packets. On the other hand, DDR controllers placement does not seem to have a significant



(a) Average Packet Latency (cycles) - 32 cores

(b) Average Packet Latency (cycles) - 64 cores

(c) Average Packet Distance (in Hops) - 32 cores

(d) Average Packet Queuing Delay (in cycles) - 32 cores

Fig. 12: Evaluation of DDR placement and NoC form factor on data exchange performance with SESAM/VPSim

impact on the *average packet queuing delay* (Fig.12d); no configuration performs better on all benchmarks.

Besides, no conclusive behavior could be extracted from the water_nsquared benchmark. This application suffers from non-determinism [43]. Indeed, while the number of memory reads/writes remains almost the same across multiple runs, the number of generated packets varies from simple to fourteenfold due to coherency protocols. This prevents comparison between configurations on the same basis.

3) NoC form factor: For each number of cores, two form factors were compared: square (6×6 or 8×8) vs rectangle (4×8 or 6×12). Comparison is made on the basis of Configuration 3 that gives the best performance for both form factors using SESAM/VPSim. On 32-core architectures, only DGEMM and some small benchmarks (around 200 million generated packets in total) give a slight advantage to the 6×6 mesh NoC. All the remaining applications, especially those that generate more than 85% of the total traffic (*i.e.* ocean_cp, stream_c and WaLBerla), reach higher performance with the rectangular form factor 4×8. The *average packet distance* analysis shows that the square shape induces an increase in the number of routers traversed by packets. The same behavior is generally observed on 64-core architectures (Fig. 12b).

In terms of congestion, unlike the placement of DDR controllers, the NoC form factor can have an impact on the average queuing latency. Congestion is managed better by the 4×8 NoC than by the 6×6 (Fig. 12d). However, results on 64-core architectures (not included in Fig. 12) do not confirm that rectangle NoCs always perform better than square ones; very comparable average queuing latency values were obtained independently of the DDR memory placement or the NoC form factor.

VI. Conclusion and Perspectives

This paper describes and exemplifies the architecture analysis and insights that can be obtained by combining different simulation tools to better understand the impact of alternative chip design parameters onto application performance and system efficiency. Given the trade-offs between simulation speed, accuracy and model abstraction level, three complementary simulation tools were selected to support a full co-design methodology: MUSA (a trace-based simulator), SESAM/VPSim (a transaction-level simulator/emulator) and gem5 (a cycle-accurate microarchitecture simulator). We consider here several system design concerns.

Thanks to MUSA, a first dimensioning of SVE register length, on-chip memory requirements and external memory bandwidth was identified, reducing the design space for more detailed simulators. Using SESAM/VPSim, large scale HPC designs were studied with respect to NoC topology and memory controller positioning. More detailed exploration was conducted on smaller designs with gem5 providing a full insight on the impact of SVE register length, NoC bandwidth, and components placement strategies on RDV1 performance.

Our co-design methodology led us to actionable insights about our reference HPC SoC architecture: (a) SVE register length above 512 bits can significantly increase performance for compute-bound benchmark such as DGEMM. However, memory-bound benchmarks do not exhibit the same scaling. This observation motivates the adjustment of trade-offs when considering chip area and power consumption; (b) The benefit of increasing the SLC size varies highly for different benchmarks. Improvements on miss rate do not always induce a significant overall performance gain (limited to +12% from 512 KiB to 1 MiB for memory-bound applications). This observation justifies keeping the SLC at moderate size; (c) Memory bandwidth needs were determined to be between 16 GB/s/core and 20 GB/s/core, which motivates the inclusion of multiple links at the NoC level (up to 50% reduction in network latencies); (d) NoC topology studies (for 16-core system configurations) show that putting together on the same node 2 SLCs slices and 2 cores is more efficient than specializing nodes for either compute or storage; (e) Looking at larger scale systems (32 and 64 cores), we find that it is worth opting for a rectangle NoC form factor with the DDR controllers placed in the middle of the longer sides.

The co-design methodology in this paper has already proven valuable for the architectural definition and design optimization in the EPI project. Its use is expected to continue to serve us well in upcoming research development activities.

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. A special thanks to Amir Charif and Arief Wicaksana for their invaluable contributions to the SESAM/VPSim tool in the initial phases of the EPI project.

References

[1] Niket Agarwal, Tushar Krishna, Li-Shiuan Peh, and Niraj K Jha. Garnet: A detailed on-chip network model inside a full-system simulator. In *2009 IEEE international symposium on performance analysis of systems and software*, pages 33–42. IEEE, 2009.

[2] Arm. Arm instruction emulator. https://developer.arm.com/tools-and-software/server-and-hpc/compile/arm-instruction-emulator.

[3] Arm. AMBA® 5 CHI architecture specification. https://developer.arm.com/documentation/ihi0050/ea/, 2020.

[4] Arm. Arm® Neoverse™ N1 core - technical reference manual. https://developer.arm.com/documentation/100616/0401/?lang=en, 2020.

[5] Arm. Arm® Neoverse™ V1 core - technical reference manual. https://developer.arm.com/documentation/101427/0101?lang=en, 2021.

[6] Arm. Arm® Neoverse™ V1 reference design - software developer guide. https://developer.arm.com/documentation/PJDOC-1779577084-33214/RelG?lang=en, 2021.

[7] D. August, J. Chang, S. Girbal, D. Gracia-Perez, G. Mouchard, D. A Penry, O. Temam, and N. Vachharajani. UNISIM: An open simulation environment and library for complex architecture design and collaborative development. *IEEE Computer Architecture Letters*, 6(2):45–48, 2007.

[8] F. Bellard. QEMU, a fast and portable dynamic translator. In *Proceedings of the FREENIX Track: 2005 USENIX Annual Technical Conference, April 10-15, 2005, Anaheim, CA, USA*, pages 41–46. USENIX, 2005.

[9] Srikant Bharadwaj, Jieming Yin, Bradford Beckmann, and Tushar Krishna. Kite: A family of heterogeneous interposer topologies enabled via accurate interconnect modeling. In *Proceedings of the 57th ACM/EDAC/IEEE Design Automation Conference*, DAC '20. IEEE Press, 2020.

[10] 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*, pages 72–81, 2008.

[11] N. Binkert, B. Beckmann, G. Black, S. K Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R Hower, T. Krishna, S. Sardashti, et al. The gem5 simulator. *ACM SIGARCH computer architecture news*, 39(2):1–7, 2011.

[12] A. Butko, R. Garibotti, L. Ost, and G. Sassatelli. Accuracy evaluation of gem5 simulator system. In *7th International Workshop on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC)*, pages 1–7. IEEE, 2012.

[13] Barcelona Supercomputing Center. V for vector: software exploration of the vector extension of RISC-V. https://www.european-processor-initiative.eu/v-for-vector-software-exploration-of-the-vector-extension-of-risc-v/.

[14] A. Charif, G. Busnot, R. H. Mameesh, T. Sassolas, and N. Ventroux. Fast virtual prototyping for embedded computing systems design and exploration. In Daniel Chillet, editor, *Proceedings of the Rapid Simulation and Performance Evaluation: Methods and Tools, RAPIDO 2019, Valencia, Spain, January 21-23, 2019*, pages 3:1–3:8. ACM, 2019.

[15] G. Delbergue, M. Burton, F. Konrad, B. Le Gal, and C. Jego. QBox: An industrial solution for virtual platform simulation using QEMU and SystemC TLM-2.0. In *Proceedings of the 8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*, Janaury 2016.

[16] Wolfgang E. Denzel, Jian Li, Peter Walker, and Yuho Jin. A framework for end-to-end simulation of high-performance computing systems. *Simul.*, 86(5-6):331–350, 2010.

[17] J. J. Dongarra, Jeremy Du Croz, Sven Hammarling, and I. S. Duff. A set of level 3 basic linear algebra subprograms. *ACM Trans. Math. Softw.*, 16(1):1–17, March 1990.

[18] DynamoRIO. Dynamorio, dynamic instrumentation tool platform. https://dynamorio.org/.

[19] C. Feichtinger, S. Donath, H. Köstler, J. Götz, and U. Rüde. WaLBerla: HPC software design for computational engineering simulations. *Journal of Computational Science*, 2(2):105–112, 2011.

[20] M. Gligor, N. Fournel, and F. Pétrot. Using binary translation in event driven simulation for fast and flexible mpsoc simulation. In W. Rosenstiel and K. Wakabayashi, editors, *Proceedings of the 7th International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS 2009, Grenoble, France, October 11-16, 2009*, pages 71–80. ACM, 2009.

[21] Constantino Gómez, Francesc Martínez, Adrià Armejach, Miquel Moretó, Filippo Mantovani, and Marc Casas. Design space exploration of next-generation HPC machines. In *2019 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2019, Rio de Janeiro, Brazil, May 20-24, 2019*, pages 54–65. IEEE, 2019.

[22] Thomas Grass, César Allande, Adrià Armejach, Alejandro Rico, Eduard Ayguadé, Jesús Labarta, Mateo Valero, Marc Casas, and Miquel Moretó. MUSA: a multi-level simulation approach for next-generation HPC machines. In John West and Cherri M. Pancake, editors, *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2016, Salt Lake City, UT, USA, November 13-18, 2016*, pages 526–537. IEEE Computer Society, 2016.

[23] Eric Grobelny, David Bueno, Ian A. Troxel, Alan D. George, and Jeffrey S. Vetter. FASE: A framework for scalable performance prediction of HPC systems and applications. *Simul.*, 83(10):721–745, 2007.

[24] Michael A Heroux, Douglas W Doerfler, Paul S Crozier, James M Willenbring, H Carter Edwards, Alan Williams, Mahesh Rajan, Eric R Keiter, Heidi K Thornquist, and Robert W Numrich. Improving performance via mini-applications. *Sandia National Laboratories, Tech. Rep. SAND2009-5574*, 3, 2009.

[25] Ming-yu Hsieh, Kevin T. Pedretti, Jie Meng, Ayse K. Coskun, Michael Levenhagen, and Arun Rodrigues. SST + gem5 = a scalable simulation infrastructure for high performance computing. In George F. Riley, Francesco Quaglia, and Jan Himmelspach, editors, *International ICST Conference on Simulation Tools and Techniques, SIMUTOOLS '12, Sirmione-Desenzano, Italy, March 19-23, 2012*, pages 196–201. ICST/ACM, 2012.

[26] European Processor Initiative. EPI website. https://www.european-processor-initiative.eu.

[27] S.-H. Kang, D. Yoo, and S. Ha. TQSIM: A fast cycle-approximate processor simulator based on QEMU. *J. Syst. Archit.*, 66-67:33–47, 2016.

[28] Yoongu Kim, Weikun Yang, and Onur Mutlu. Ramulator: A Fast and Extensible DRAM Simulator. *IEEE Computer Architecture Letters*, 15(1):45–49, 2016.

[29] Yuetsu Kodama, Tetsuya Odajima, Akira Asato, and Mitsuhisa Sato. Evaluation of the RIKEN post-k processor simulator. *CoRR*, abs/1904.06451, 2019.

[30] Jason Lowe-Power, Abdul Mutaal Ahmad, Ayaz Akram, Mohammad Alian, Rico Amslinger, Matteo Andreozzi, Adrià Armejach, Nils Asmussen, Srikant Bharadwaj, Gabe Black, et al. The gem5 simulator: Version 20.0+. *arXiv preprint arXiv:2007.03152*, 2020.

[31] Imperas Ltd. Open Virtual Platforms. http://www.ovpworld.org/, 2020.

[32] Chi-Keung Luk, Robert Cohn, Robert Muth, Harish Patil, Artur Klauser, Geoff Lowney, Steven Wallace, Vijay Janapa Reddi, and Kim Hazelwood. Pin: Building customized program analysis tools with dynamic instrumentation. In *Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '05, page 190–200, New York, NY, USA, 2005. Association for Computing Machinery.

[33] Oumaima Matoussi. Noc performance model for efficient network latency estimation. In *2021 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 994–999, 2021.

[34] John D. McCalpin. Memory bandwidth and machine balance in current high-performance computers. *IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter, December*, 1995.

[35] M. Monton, J. Engblom, and M. Burton. Checkpointing for virtual platforms and systemc-tlm. *IEEE Trans. VLSI Syst.*, 21(1):133–141, 2013.

[36] Tiago Mück. gem5 Ruby CHI Protocol Documentation. https://www.gem5.org/documentation/general_docs/ruby/CHI/, 2021.

[37] Yasir Mahmood Qureshi, William Andrew Simon, Marina Zapater, David Atienza, and Katzalin Olcoz. Gem5-x: A gem5-based system level simulation framework to optimize many-core platforms. In *2019 Spring Simulation Conference (SpringSim)*, pages 1–12, 2019.

[38] Yasir Mahmood Qureshi, William Andrew Simon, Marina Zapater, Katzalin Olcoz, and David Atienza. Gem5-x: A many-core heterogeneous simulation platform for architectural exploration and optimization. *ACM Trans. Archit. Code Optim.*, 18(4), July 2021.

[39] Ajay Ramaswamy, Nalini Kumar, Aravind Neelakantan, Herman Lam, and Greg Stitt. Scalable behavioral emulation of extreme-scale systems using structural simulation toolkit. In *Proceedings of the 47th International Conference on Parallel Processing, ICPP 2018, Eugene, OR, USA, August 13-16, 2018*, pages 17:1–17:11. ACM, 2018.

[40] A. F. Rodrigues, K. S. Hemmert, B. W. Barrett, C. Kersey, R. Oldfield, M. Weston, R. Risen, J. Cook, P. Rosenfeld, E. Cooper-Balis, and B. Jacob. The structural simulation toolkit. *SIGMETRICS Perform. Eval. Rev.*, 38(4):37–42, March 2011.

[41] Nizar Romdan. ARM fast models-virtual platforms for embedded software development. *ARM Information Quarterly*, 7(4):33–36, 2008.

[42] Paul Rosenfeld, Elliott Cooper-Balis, and Bruce Jacob. DRAMSim2: A Cycle Accurate Memory System Simulator. *IEEE Computer Architecture Letters*, 10(1):16–19, 2011.

[43] Giordano Salvador, Siddharth Nilakantan, Baris Taskin, Mark Hempstead, and Ankit More. Effects of nondeterminism in hardware and software simulation with thread mapping. In *2015 28th International Conference on VLSI Design*, pages 129–134, 2015.

[44] Mitsuhisa Sato, Yutaka Ishikawa, Hirofumi Tomita, Yuetsu Kodama, Tetsuya Odajima, Miwako Tsuji, Hisashi Yashiro, Masaki Aoki, Naoyuki Shida, Ikuo Miyoshi, Kouichi Hirai, Atsushi Furuya, Akira Asato, Kuniki Morita, and Toshiyuki Shimizu. Co-design for A64FX manycore processor and "fugaku". In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '20. IEEE Press, 2020.

[45] Nigel Stephens, Stuart Biles, Matthias Boettcher, Jacob Eapen, Mbou Eyole, Giacomo Gabrielli, Matt Horsnell, Grigorios Magklis, Alejandro Martinez, Nathanael Premillieu, Alastair Reid, Alejandro Rico, and Paul Walker. The ARM Scalable Vector Extension. *IEEE micro*, 37(2):26–39, 2017.

[46] Synopsys. Virtualizer. https://www.synopsys.com/verification/virtual-prototyping/virtualizer.html, 2020.

[47] N. Ventroux, A. Guerre, T. Sassolas, L. Moutaoukil, G. Blanc, C. Bechara, and R. David. SESAM: An MPSoC simulation environment for dynamic application processing. In *2010 10th IEEE International Conference on Computer and Information Technology*, pages 1880–1886. IEEE, 2010.

[48] Jun Wang, Jesse G. Beu, Rishiraj A. Bheda, Tom Conte, Zhenjiang Dong, Chad D. Kersey, Mitchelle Rasquinha, George F. Riley, William J. Song, He Xiao, Peng Xu, and Sudhakar Yalamanchili. Manifold: A parallel simulation framework for multicore systems. In *2014 IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS 2014, Monterey, CA, USA, March 23-25, 2014*, pages 106–115. IEEE Computer Society, 2014.

[49] S. Cameron Woo, M. Ohara, E. Torrie, J. Pal Singh, and A. Gupta. The splash-2 programs: Characterization and methodological considerations. *ACM SIGARCH computer architecture news*, 23(2):24–36, 1995.

[50] Y. Kodama and T. Odajima and M. Matsuda and M. Tsuji and J. Lee and M. Sato. Preliminary Performance Evaluation of Application Kernels Using ARM SVE with Multiple Vector Lengths. In *2017 IEEE International Conference on Cluster Computing (CLUSTER)*, volume , pages 677–684, Sept 2017.

[51] Field G. Van Zee and Robert A. Van de Geijn. Blis: A framework for rapidly instantiating blas functionality. *ACM Transactions on Mathematical Software,Volume*, 41(3):14, June 2015.

[52] Gengbin Zheng, Gagan Gupta, Eric J. Bohm, Isaac Dooley, and Laxmikant V. Kalé. Simulating large scale parallel applications using statistical models for sequential execution blocks. In *16th IEEE International Conference on Parallel and Distributed Systems, ICPADS 2010, Shanghai, China, December 8-10, 2010*, pages 221–228. IEEE Computer Society, 2010.