Coyote: An Open Source Simulation Tool to Enable RISC-V in HPC

Borja Perez
Barcelona Supercomputing Center
Barcelona, Spain
borja.perez1@bsc.es

Alexander Fell
Barcelona Supercomputing Center
Barcelona, Spain
alexander.fell@bsc.es

John D. Davis
Barcelona Supercomputing Center
Barcelona, Spain
john.davis@bsc.es

***Abstract*—The confluence of technology trends and economics has reincarnated computer architecture and specifically, software-hardware co-design. We are entering a new era of a completely open ecosystem, from applications to chips and everything in between. The software-hardware co-design of supercomputers for tomorrow requires flexible tools today that will take us to the Exascale and beyond. The MareNostrum Experimental Exascale Platform (MEEP) addresses this by proposing a flexible FPGA-based emulation platform, designed to explore hardware-software co-designs for future RISC-V supercomputers. This platform is part of an open ecosystem, allowing its infrastructure to be reused in other projects. MEEP's inaugural emulated system will be a RISC-V based self-hosted HPC vector and systolic array accelerator, with a special aim at efficient data movement. Early development stages for such an architecture require fast, scalable and easy to modify simulation tools, with the right granularity and fidelity, enabling rapid design space exploration. Being a part of MEEP, this paper introduces Coyote, a new open source, execution-driven simulator based on the open source RISC-V ISA and which can provide detailed insights at various levels and granularities. Coyote focuses on data movement and the modelling of the memory hierarchy of the system, which is one of the main hurdles for high performance sparse workloads, while omitting lower level details. As a result, performance evaluation shows that Coyote achieves an aggregate simulation of up to 6 MIPS when modelling up to 128 cores. This enables the fast comparison of different designs for future RISC-V based HPC architectures.**

***Index Terms*—high performance computing, simulation, accelerator, processor architecture**

I. Introduction

The MareNostrum Experimental Exascale Platform (MEEP) is a flexible FPGA-based emulation platform designed to explore hardware/software co-designs for Exascale Supercomputers serving two main purposes: 1) The evaluation/validation platform for pre-silicon Intellectual Property (IP) and 2) a software development vehicle (SDV) to enable software development and readiness for new hardware.

Traditionally, highly parallized workloads in the domain of High Performance Computing (HPC) and High Performance Data Analytics (HPDA) are offloaded to dedicated accelerators such as General-Purpose Graphics Processing Units (GPGPU). However, this execution paradigm suffers from a communication bottleneck as the data and results need to be constantly moved between the host and the dedicated processors resulting in multiple inefficiencies: Multiple copies of data must be managed in the system, related energy overheads of data marshalling, and limiting the applications that can be accelerated due to the communication overhead and limited memory capacity of the accelerators. On the other hand, the architecture proposed by MEEP, the Accelerated Compute and Memory Engine (ACME), will mitigate these problems by executing most of the application on the accelerator, removing the need for a powerful host Central Processing Unit (CPU). In this case, the host processor provides limited services, i.e., storage and name services, daemons and other support infrastructure. This is achieved by a traditional vector architecture using the RISC-V Instruction Set Architecture (ISA) and associated vector extensions [1]. This architecture combines a normal RISC-V scalar processor with co-processors optimized for parallel execution.

The ACME architecture pushes beyond the traditional vector architecture by also incorporating additional accelerators for video processing and neural networks using systolic arrays. The combination of processor and co-processors supports traditional HPC and emerging HPDA applications. The goal of the MEEP platform is to demonstrate an accelerator, ACME, that is able to run a full featured operating system including container support, as well as parallel code targeting the traditional accelerators, and is hence classified as a self-hosted accelerator.

A. MEEP Hardware

Figure 1 provides a high-level overview of the envisioned ACME hardware architecture. The central component of this architecture are the Vector and Systolic Accelerator (VAS) tiles. One tile consists of eight cores including a Vector Processing Unit (VPU) with 16 lanes, each can be further divided or fused and assigned to a single thread depending on the workload.

The Memory Controller CPUs (MCPU) shown in Figure 1 are RISC-V CPUs that manage the memory requests, both scalar and vector from the VAS tiles. Our accelerator wants to leverage the additional application semantic knowledge that vectors provide. Thus, we can move beyond single loads and stores to vectors or other data structures that can be managed in aggregate at the memory controller. This provides a broader view of future global memory access patterns and other properties that can be provided by the programmer, compiler, runtime or combination of all of the above. In MEEP's case, processing this additional information is the job of the MCPUs. These MCPUs are processors that reside near the memory controllers

The MEEP project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 946002. The JU receives support from the European Union's Horizon 2020 research and innovation program and Spain, Croatia, and Turkey.

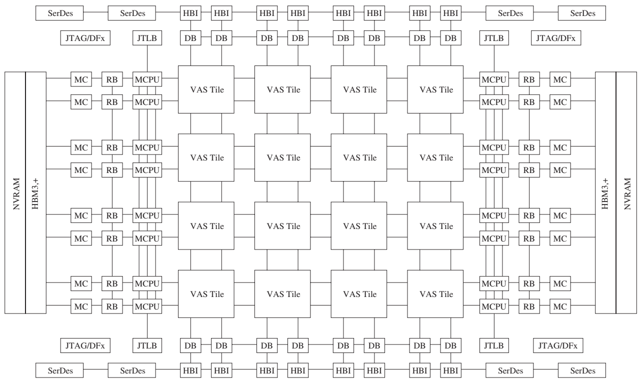

Figure 1. A block diagram about the components of ACME

(MC) for the High Bandwidth Memory (HBM). As such, they operate on vectors, both dense (unit stride) and sparse with the help of vector index registers for scatter/gather operations, and perform memory operations like atomics and simple, sequential arithmetic operations. Essentially, the MCPUs are memory co-processors and are over-provisioned to have enough resources available to run the local operating system including its daemons and other resource scheduling and accounting software.

As a first prototype, ACME is to be emulated on a set of Field-Programmable Gate Arrays (FPGA). MEEP provides a common infrastructure for all hardware that is mapped to it called the FPGA shell, which provides interfaces to the resources available such as Peripheral Component Interconnect Express (PCIe), Double Data Rate Random-Access Memory (DDR RAM), Ethernet and HBM. Representative portions of the ACME architecture will be mapped to a single FPGA because current FPGA generations such as Xilinx' Virtex UltraScale+ platforms [2] will not be able to hold all resources depicted in Figure 1. It is a fruitful area of research to determine other mapping strategies that provide high performance (100-300 MHz) and the right level of fidelity to enable software development [3]–[5]. Distributing a single design across multiple FPGAs will slow down the design [6]–[9], but provide higher capacity and fidelity. In the case of the SDV, we can relax the capacity and gate-level fidelity to gain performance to enable software development. However, the final MEEP emulator will provide tight coupling of eight FPGAs with associated high-speed Serializer/Deserializers (SerDes), providing the flexibility to aggregate FPGAs or simply use those links for FPGA-to-FPGA (F2F) communication for emulated coherence or other features.

The design decisions related to the F2F communication, the selection of the MCPU cores and the arrangement of the VAS tiles, are complex and require preliminary investigations before committing to the work-intensive task of an Application-Specific Integrated Circuit (ASIC) and/or FPGA implementation. To account for this, a common approach in industry is going multiscale: having several simulation tools and platforms, with different levels of detail, that serve different purposes and complement each other. This is exemplified by the ARM cycle models and fast models [10]. For the purpose of design space exploration prior to FPGA implementation, a new open source simulator called *Coyote* [11] has been developed, targeting RISC-V in an HPC environment. The core requirements are support for the RISC-V vector ISA extensions, multicore support, and high performance. Coyote is based on pre-existing open source tools and has been developed with a focus on the modelling of the movement of data within the system, which is one of the key factors that limit performance and energy efficiency. This captures the right amount of detail to enable significant comparisons of different design points within feasible simulation times.

Coyote is described in section III after providing some background on the emerging efforts of RISC-V in HPC and RISC-V compatible simulation tools in section II. In section IV, an example is given demonstrating the purpose of Coyote from the hardware as well as software perspective and section V concludes the paper.

MEEP makes the following contributions:

- A large scale emulation platform for SDV and pre-silicon validation
- The ACME disaggregated architecture targeting dense and *sparse* workloads and hence improving the performance of HPC and HPDA applications using a self-hosted accelerator.
- Provision of a hardware/software co-design simulator targeting RISC-V compatible CPU in the domain of HPC that serves as a testing field for hardware as well as software developers to examine the impact of new hardware and algorithmic optimizations.
- MEEP will be licensed as open source, whenever it is permitted. Therefore it will improve the RISC-V ecosystem by optimizing the hardware as well as software toolchain to include HPC and HPDA applications.

II. Background

RISC-V is an open source ISA defined by the RISC-V International Foundation [12], which rapidly managed to attract hundreds of international institutions, including universities, research institutions and companies worldwide, that consistently contribute to its design, implementation, and worldwide adoption. It is foreseen that the market will consume a total of 62.4 billion RISC-V CPU cores by 2025, with the industrial sector forecasted to be the largest segment with 16.7 billion cores [13]. Given the large number of collaborating institutions, the amount of interest in this open source ISA, and the quantity of recently developed RISC-V based novel IPs and software, Europe has an opportunity to drive the adoption of this rapidly growing ecosystem. This could include a variety of RISC-V cores, extensions, application-specific accelerators and software stacks. These would span across a spectrum of performance targets for different markets.

In the following sections, we discuss the importance of RISC-V and its ability to create an entirely open ecosystem on the hardware and software side. Significant effort is required to build the MEEP platform. We will outline how we can reuse MEEP in the context of other projects. Likewise, as mentioned earlier, one tool cannot solve all problems. We describe some of the support simulation tools that we have built and used to do early design exploration. This provides a powerful suite of tools to rapidly evaluate and develop software for future systems.

A. RISC-V in HPC

In this section, we elaborate on why RISC-V is important for HPC. The combination of the RISC-V ISA and Open Source Software (OSS) is enabling a future where the entire ecosystem will be based on open source. In this new world, system optimizations can be done in the layer that is most appropriate, resulting in an overall optimized co-designed solution. Prior to the RISC-V ISA, options were very limited and building prototype systems was left for the few. Today, there are over 80 instances of RISC-V cores and System-on-Chips (SoC) available for use as free Open Source Hardware (OSH) or as licensed cores [14].

The genesis of RISC-V corresponds with the demise of Moore's Law, the engine powering exponential transistor growth, and the resulting performance of Complementary Metal-Oxide-Semiconductor (CMOS) technology. This ushers in a transition to specialization via accelerators and co-designed software and hardware systems. By specializing, we can achieve the power, performance and area requirements of the system, both big and small. RISC-V has started at the low-end of computer systems in the embedded, microcontroller and Internet-of-Things (IoT) space. However, the trend of specialization exists across all verticals and we want MEEP to be the infrastructure that enables specialization of future high performance accelerators and CPUs. Accelerators provide an opportunity to maximize performance gain with minimal software effort, generally only focused on one application or class of applications or algorithms. Thus, creating a complete demonstration can be done much faster. On the other hand, we need full ecosystem support from applications to software libraries, down to the toolchain, for CPUs to be truly useful. This is a much larger software effort and starting early with tools like MEEP can enable faster adoption for new features and capabilities.

Like many other research groups [15], the Barcelona Supercomputing Center (BSC) supports Open ISAs in general and the RISC-V ISA, in particular. We see this as an important main ingredient to build the foundation for a future open HPC ecosystem. This started in 2004, when the BSC was the first supercomputing center to deploy Linux as the operating system. We have participated in several projects, like Mont-Blanc [16], that were limited by the proprietary hardware platform which prevented research and development at the hardware level.

The situation has changed in HPC with the start of the European Processor Initiative (EPI) project. The EPI Accelerator (EPAC) is a collection of accelerators, all based on the RISC-V ISA, including vector, variable precision processors, stencil/tensor accelerators, coherent shared L2 cache and Network-on-Chip (NoC) configurations [17]. The MEEP project is an outgrowth of the EPAC in that it too will provide an advanced HPC accelerator, ACME (see figure 1), targeting fabrication in the 2025 time frame, a few generations beyond the current EPAC target [18]. However, MEEP and ACME are demonstrators of the larger goal of building a complete infrastructure that supports the hardware and software development of the RISC-V ecosystem, including, but not limited to accelerators and CPUs. While ACME targets traditional HPC, emerging HPDA, video and neural network applications, from the software perspective, MEEP will enable system level exploration and development using multiple FPGAs, tightly coupled, to emulate ACME or other systems. The MEEP FPGA infrastructure or FPGA shell will be reusable across other projects, ideally continuing and extending work from the EPAC.

The BSC is not only focused on a variety of accelerators, but also RISC-V CPUs for HPC. The new *e*Processor project [19] targets creating an open source out-of-order RISC-V single and dual core CPU. The processor will have both on- and off-chip coherence, enabling dual socket configurations, including connections to accelerators and FPGAs that comply with the coherent external interface. We will use MEEP to emulate these processor configurations and enable the software development targeting new bioinformatics workloads.

B. Simulation tools

RISC-V enjoys a vibrant community that has gone to great lengths to develop high quality simulation tools, some already established and in wide use. Several tools are available, that cover different purposes and degrees of modelling detail and fidelity. This section provides a brief overview about some of the most salient tools and frameworks that can be used for RISC-V simulation and their suitability for HPC design space exploration. The main aspects for a tool to be deemed suitable are:

- High simulation throughput to compare very different design points within a reasonable time.
- Significant modelling capabilities to capture the main characteristics of HPC systems.
- Easy extensibility to simplify the addition and integration of new features.

The QEMU [20] RISC-V implementation, being based on binary translation to the host architecture, offers an excellent simulation throughput that would enable design space exploration. However, it does not support the RISC-V vector extension, which is one of the likely sources of extra performance and efficiency in HPC systems. Gem5 [21] is the *de facto* standard open source tool for execution-driven simulation in academia. It offers great modelling capabilities, but suffers from a low simulation throughput and is a complex and hard to extend tool. The Renode simulator [22], developed by Antmicro, is a development tool targeting multinode embedded networks, which means that it does not support features related to HPC like vector instructions. Imperas offers riscvOVPsim [23], which is a reference simulator with support for vector instructions. However, this simulator is not fully open source. MUSA [24] is a multi-level simulation infrastructure developed by BSC. It uses traces of vector instructions generated by a real program execution of a RISC-V deployed on a development platform, to model the behavior when accessing the memory hierarchy. Unfortunately, it does not support scalar instructions. Spike [25] is the golden standard RISC-V Instruction Set Simulator (ISS). It is capable of fast simulation of multicore systems and supports vector instructions. However, it has very limited modelling capabilities regarding the memory hierarchy. The Sail RISCV Model [26] is a new ISS adopted by the RISC-V Foundation. It shows promising features, such as booting Linux, but currently has no modelling capabilities. The SST framework [27] offers a wide array of modules that provide great modelling capabilities. An integration of SST and Spike is available, named Stake [28]. It enables RISC-V simulation using Spike but can only model a single CPU core. Sparta [29] is a toolkit to build CPU, GPU, and platform simulations, currently developed by SiFive. This framework provides structure and organization as well as helper classes and interfaces allowing models built with Sparta to work together for functional and performance simulation. However, it has no RISC-V support.

III. Hardware/Software Co-Design

As an evaluation platform of pre-silicon IP, MEEP needs tools to test new ideas at different stages of the development of a new hardware target. Different stages have different simulation requirements. Earlier ones require fast tools, which enable the comparison of disparate design points within reasonable time, even if that means that the fidelity is lower. These tools are usually easy to modify and extend. Therefore, getting a first-order notion of how two architectures compare does not involve excessive modelling effort. Only once the general flavor of the design has been decided using this kind of tools, low level details start to become essential. This is when higher fidelity tools, such as FPGA-based emulation platforms, which require significantly more effort, come into play. In sum, the development of new architectures requires a multiscale approach: late design tools are impractical for design space exploration, while those for earlier stages lack the fidelity to capture low level details.

Section II-B shows that, rich as it may be, the RISC-V simulation ecosystem does not provide the necessary tools for HPC architecture design space exploration. The available tools either do not model the capabilities that are the staple of HPC (high core counts, complex memory hierarchies, vector architectures, etc.) or are too slow and hard to extend to represent feasible options. For this reason, we set out to build a fast and flexible tool for HPC design space exploration.

A. *Coyote*

HPC architectures often feature high core counts and complex memory hierarchies. Therefore, when design space exploration is the goal, choosing the level of detail to capture in your simulation tools is the key. Too much detail and the simulation throughput will not be enough, too few and comparisons will not be meaningful. Coyote[1] is a new open source, execution-driven simulation tool, based on the canonical RISC-V ISA, that has a focus on the movement of data throughout the memory hierarchy. This provides the right amount of detail to perform first-order comparisons between different design points, as the behavior of memory accesses and the well-known memory-wall are the main barrier to efficient computing.

[1]MEEP is a reference to the roadrunner for its speed and efficiency. In the well-known cartoon, Wile E. Coyote aspires to match the roadrunner using his wits. This is precisely the goal of our simulator: matching the MEEP Hardware as closely as possible through approximations.

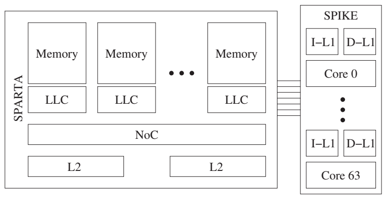

Figure 2. An approximate simulatable 64-core slice with elements simulated by each tool.

Figure 2 shows an example of an HPC architecture that would be reasonable to model as part of an exploration. Note that it neither represents the current capabilities of the tools, nor the actual MEEP Hardware design, but a sample system that could be considered for simulation. Three levels of cache and 64 cores are depicted. We consider these numbers sufficiently big to draw significant conclusions and to reason about the even bigger architectures that the MEEP project targets, which would be unfeasible to simulate. Cache coherent systems might also be considered in the future, but the modelling of coherence protocols and traffic is currently beyond the scope of Coyote.

In order to develop a tool that is useful not just for MEEP, but to the whole RISC-V community, Coyote was not conceived as a completely new simulator, but as an integration of existing RISC-V tools. This is a desirable feature. It both leverages previous community efforts and also produces a tool that users find familiar and easy to adopt.

Coyote is based on two pre-existing tools: Spike and Sparta. Spike, as the golden standard for RISC-V simulation, is a widespread tool within the community. It offers many of the required capabilities for HPC design space exploration, since it has a significant simulation throughput, supports vector instructions and the simulation of multicore systems. However, it lacks regarding memory modelling features. To account for this, Sparta was used to build a flexible memory model based on a modular design, in which the functionality of each element (e.g. an L2 Bank) is encapsulated as an independent component. The result is high flexibility and easy extensibility. Adding new functionality to a component is just a matter of modifying the corresponding module accordingly. Evaluating systems of different scale just requires connecting fewer or more modules.

The purpose of Spike in this integration is providing its functional execution capabilities and also modelling the L1 caches. This is to reduce the number of interactions between Spike and Sparta, with the intent of improving the simulation throughput. Sparta is in charge of modelling the rest of the memory hierarchy. The pieces of the simulated system that are managed by each of the tools are shown in Figure 2.

Spike and Sparta are slaves to an Orchestrator that handles the simulation, keeping track of timing, and synchronizing both parts. Every cycle, the Orchestrator first tries to simulate an instruction on each of the active cores using Spike. Spike runs in baremetal mode to leverage its capability to simulate multicore systems. This entails some limitations on the applications that can be run, due to the very limited availability of syscalls. An attempt to simulate an instruction has two possible effects:

- If a Read-After-Write (RAW) dependency is detected on one of the instruction operands and a pending memory access, the core is marked as inactive. No further instructions will be simulated on this core until the dependency is satisfied.

- The simulation of an instruction might generate an L1 miss (or more). In this case, the information on the missing access(es) is enqueued into Sparta, which will generate the necessary events to target the modelled memory hierarchy.

Once an instruction has been simulated in each of the active cores, the Orchestrator checks, if Sparta has any in-flight events for the current cycle. If this is the case, the Sparta model is advanced to keep it in sync with the rest of the simulation. This also effectively handles the events themselves. Once an L1 miss is serviced, the registers that it writes to are made available, and any pending dependency is satisfied, while stalled cores are set as active once again.

The memory modelling and simulation orchestration capabilities have been achieved using base Sparta classes. Spike has been minimally modified/customized to add the means to simulate instruction by instruction and track L1 misses and RAW dependencies.

Coyote models tiled systems that resemble the ACME architecture. Each tile holds a number of cores and L2 cache banks. Memory accesses first exercise the core-private instruction and L1 data caches, modelled in Spike, and access the Sparta modelled memory hierarchy on a miss. The L2 can be configured as fully-shared across the system or private to the cores of each tile. Its configuration can be set using input parameters, including its size, associativity and line size, the number of banks it is split into, the maximum number of in-flight misses, and the hit/miss latencies. Two different well-known data mapping policies have been implemented, that use different bits of the address to identify the L2 bank that holds a certain memory block: page-to-bank and set-interleaving. When a miss occurs in L2, a request is sent to a memory controller. The tiles and memory controllers are interconnected by a NoC, currently modelled as a highly idealized crossbar, that uses fixed, configurable latencies. A more realistic modelling of this component and the memory controllers is currently work in progress. Deeper memory hierarchies or more heterogeneous systems can currently be modelled, but require knowledge of the internals of the simulator to adequately connect each of the components.

Simulation outputs statistics about memory accesses (miss rates, number of stalls due to dependencies, etc.), the execution time of the simulated application and a trace of L1 misses. This trace can be analyzed using the Paraver Visualization Tools [30] to truly understand the behavior of applications, by identifying access patterns or analyzing how and when the L2 banks, NoC, or memory are stressed.

Performance evaluation has been carried out simulating up to 128 cores, obtaining an aggregate simulation throughput of up to 6 million instructions per second (MIPS), which enables design space exploration. Figure 3 shows the evolution of the simulation throughput as the number of simulated cores grows, for scalar implementations of matrix and sparse matrix vector multiplications (SpMV). The results identify a bottleneck when the number of cores is low, which can be traced to Spike: to simulate one instruction per cycle in each core and correctly exercising the memory hierarchy as Coyote does, interleaving had to be disabled in Spike. Interleaving speeds up simulation in the original Spike implementation by executing several instructions on the same core back to back, before switching to the next core. This feature enables optimizations in ways the simulated instructions are handled. However, as the number of cores grows, reuse increases and so does performance, as the impact of disabling interleaving decreases. This kind of effects will be further analyzed, as Spike currently takes the largest part of a Coyote simulation.

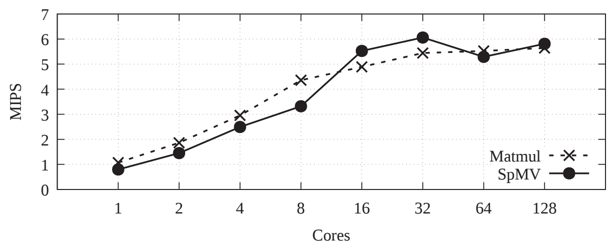

Figure 3. Evolution of the simulation throughput with the number of simulated cores

Four different kernels have been adapted to baremetal simulation in Spike and can be executed using Coyote. They can be compiled using the standard GNU toolchain for RISC-V. The kernels are: scalar matrix multiplication, vector matrix multiplication, vector SpMV (three different implementations of the algorithm) and vector stencil. These form a basis to study the behavior of memory accesses under dense and sparse workloads. More kernels will be adapted in the future to cover the applications of interest to MEEP. These will include FFT, AI and other representative HPC and HPDA kernels.

Our work so far has set a solid foundation for a fast and flexible tool for HPC architecture design space exploration. It will enable designers to make informed decisions early in the development of new architectures, based on the movement of the data, a key limiting factor of performance and efficiency. Next steps are to extend the modelling capabilities and flexibility of Coyote even further, including the modelling of the memory controllers, the NoC and different data management policies such as prefetching, streaming, etc.

IV. The RISC-V Multiscale Environment

Leveraging Coyote, a software developer can quickly obtain an overview, if the changes in algorithms or data exhibit the promising impact on the overall system performance. An MC model which is adapted to support a particular compression method addressing the physical limitations of memory interfaces, is one playground the simulator supports. Likewise, hardware engineers will also benefit from the results, since the computational performance also depends on the underlying memory architecture such as the number of interfaces, their width, bank composition and the overall size of the available space to name a few. Hence, there is ample opportunity to tweak and optimize just this one module with a global effect on an entire system.

Combining Coyote with the capabilities of FPGA emulation provide a high fidelity environment to validate the design choices made from software simulation. As an example, the FPGA MC can be extended considering the memory access patterns of SpMV, a scenario often faced in the HPC domain. Extensive research has been published to accelerate the computational aspects as well as the memory interfaces for SpMV. For instance, to avoid the memory bandwidth limitations, authors in [31] replaced non-zero values by indices in a look-up table to compress the matrix, while [32] consider the same compression scheme for FPGAs. Since the decompression algorithm is placed on the FPGA, less data is to be transferred between the memory and the computing units effectively increasing the bandwidth utilization.

While memory bandwidth limitation is a problem that developers face as of now, newly emerging applications in machine learning and video processing will intensify the problem. The Xilinx FPGA [33] used in MEEP has both a DDR RAM and HBM. HBM, a stacked DDR RAM, brings high bandwidth close to the computational resources in an FPGA. The HBM is connected to FPGA chiplets through an interposer, reducing the wire length drastically [34]. Therefore, we can trade off latency and capacity for bandwidth in the FPGA emulation system.

The FPGA enables the developers to customize the IP to suit their needs through the IP parameters provided as well as enhancements with additional logic in the FPGA fabric. Some of the parameters include the number of HBM interfaces, availability of total memory space, and support for global addressing [35]. In addition, settings such as reordering of data, refresh cycles and look-ahead pre-charge for the MCs are in the hands of the developer. Any architectural change will potentially have a tremendous impact on the development, optimization and performance of algorithms. The effectiveness and costs of the myriad of available settings like re-ordering or read-ahead of data or how high-level observations such as how the clustering of non-zero values in sparse matrices can be exploited, are some of the questions that need to be answered by Coyote before MEEP emulates the ACME accelerator. Later, these features can be implemented in the FPGA and demonstrated on real applications.

V. Conclusion

With the advent of open ISAs, the time for a fully open ecosystem is now, from applications to chips, including everything in between. MEEP, being a flexible FPGA-based emulation platform, will provide the RISC-V community with a multi-level infrastructure for HPC, which will serve as both a software and pre-silicon validation vehicle. To cover the whole development process, MEEP also introduces Coyote, a scalable open source simulator with a focus on data movement, that targets early design decisions. Coyote offers the speed and flexibility to enable the comparison of different configurations within feasible simulation times and implementation efforts, before committing to the endeavor of FPGA emulation.

MEEP provides the necessary hardware-software co-design tools to propel the construction of a fully open HPC stack in the near future, and poses definite reuse opportunities across different RISC-V projects, both within BSC and beyond. Thus, it will be central to the development of Exascale architectures, while guaranteeing their software readiness.

References

[1] RISC-V. (2020, Sep.) The RISC-V Vector Extension. https://github.com/riscv/riscv-v-spec.

[2] Xilinx. (2020, Nov.) Virtex UltraScale+. https://www.xilinx.com/products/silicon-devices/fpga/virtex-ultrascale-plus.html.

[3] E. S. Chung *et al.*, "PROToFLEX: FPGA-accelerated Hybrid Functional Simulator," in *IEEE IPDPS*, 2007, pp. 1–6.

[4] J. D. Ellithorpe *et al.*, "Internet-in-a-Box: Emulating Datacenter Network Architectures Using FPGAs," in *DAC*. New York, NY, USA: Association for Computing Machinery, 2009, p. 880–883.

[5] D. Chiou *et al.*, "FPGA-Accelerated Simulation Technologies (FAST): Fast, Full-System, Cycle-Accurate Simulators," in *MICRO*, 2007, pp. 249–261.

[6] Cadence Design Systems, Inc. (2020, Sep.) Cadence Palladium Z1 Enterprise Emulation Platform. https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/system-design-verification/palladium-z1-ds.pdf.

[7] V. Chobisa and C. Selvidge, "The Innovation Behind Veloce: Time for a Closer Look," *Horizons Newsletter*, Oct. 2008.

[8] X. Li and O. Hammami, "Multi-FPGA emulation of a 48-cores multiprocessor with NOC," in *IDT*, 2008, pp. 205–208.

[9] K. E. Fleming *et al.*, "Leveraging Latency-Insensitivity to Ease Multiple FPGA Design," in *FPGA*. ACM, Feb. 2012, pp. 175–184.

[10] ARM. (2020, Sep.) The ARM Cycle Models and Fast Models. https://developer.arm.com/tools-and-software/simulation-models.

[11] B. Perez. (2020, Dec.) Coyote. https://github.com/borja-perez/Coyote.

[12] RISC-V. (2020, Sep.) The RISC-V International Foundation. https://riscv.org/.

[13] Semico. (2020, Sep.) RISC-V Market Analysis. The New Kid on the Block. https://semico.com/sites/default/files/TOC_CC315-19.pdf.

[14] RISC-V. (2020, Sep.) RISC-V Exchange: Cores & SoCs. https://riscv.org/exchange/cores-socs/.

[15] International Conference on Supercomputing. (2020, Jun.) Workshop on RISC-V and OpenPOWER.

[16] (2020, Sep.) The Mont-Blanc Project. https://www.montblanc-project.eu/.

[17] (2020, Sep.) The European Processor Initiative. https://www.european-processor-initiative.eu/accelerator/.

[18] (2020, Sep.) The MEEP Project. https://meep-project.eu/about/objectives.

[19] BSC. (2020, Dec.) eProcessor: European, extendable, energy-efficient, energetic, embedded, extensible, Processor Ecosystem. https://www.bsc.es/research-and-development/projects/eprocessor-european-extendable-energy-efficient-energetic-embedded.

[20] F. Bellard, "QEMU, a Fast and Portable Dynamic Translator," in *USENIX Annual Technical Conference*. USA: USENIX Association, 2005, p. 41.

[21] N. Binkert *et al.*, "The Gem5 Simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, p. 1–7, Aug. 2011.

[22] Antmicro. (2020, Sep.) The Renode Simulator. https://github.com/renode/renode.

[23] Imperas. (2020, Sep.) The riscvOVPsim Simulator. https://github.com/riscv/riscv-ovpsim.

[24] T. Grass *et al.*, "MUSA: A Multi-Level Simulation Approach for next-Generation HPC Machines," in *SC*, 2016.

[25] RISCV. (2020, Sep.) The Spike RISC-V ISA Simulator. https://github.com/riscv/riscv-isa-sim.

[26] RISC-V. (2020, Sep.) The SAIL RISC-V Model. https://github.com/rems-project/sail-riscv.

[27] A. Rodrigues *et al.*, "The structural Simulation Toolkit," *SIGMETRICS Perform. Eval. Rev.*, vol. 38, no. 4, pp. 37–42, March 2011.

[28] J. D. Leidel, "Stake: A Coupled Simulation Environment for RISC-V Memory Experiments," in *MEMSYS*. New York, NY, USA: Association for Computing Machinery, 2018, p. 365–376.

[29] Si-Five. (2020, Sep.) The Sparta Framework. https://github.com/sparcians/map/tree/master/sparta.

[30] D. Computadors *et al.*, "PARAVER: A Tool to visualize and analyze parallel Code," *WoTUG-18*, vol. 44, 03 1995.

[31] J. Willcock and A. Lumsdaine, "Accelerating sparse Matrix Computations via Data Compression," in *ICS*. ACM, Jul. 2006, pp. 307–316.

[32] P. Grigoras *et al.*, "Accelerating SpMV on FPGAs by Compressing Nonzero Values," in *FCCM*. IEEE CS, 2015, pp. 64–67.

[33] Xilinx. (2020, Dec.) Alveo U280 Data Center Accelerator Card. https://www.xilinx.com/products/boards-and-kits/alveo/u280.html.

[34] M. Wissolik *et al.*, "Virtex UltraScale+ HBM FPGA: A Revolutionary Increase in Memory Performance," 2019.

[35] Xilinx, "AXI High Bandwidth Memory Controller v1.0 LogiCORE IP Product Guide," 2019.