What is RISC-V and why should we care?

Nick Brown
n.brown@epcc.ed.ac.uk
EPCC at the University of Edinburgh
Edinburgh, UK

Abstract

RISC-V is an open Instruction Set Architecture (ISA) standard which enables the open development of CPUs and a shared common software ecosystem. With billions of RISC-V cores already produced, and this is accelerating rapidly, we are seeing a revolution driven by open hardware. Nonetheless, for all the successes that RISC-V has enjoyed, it is yet to become mainstream in HPC. This comes at a time when HPC is facing new challenges especially around performance and sustainability of operations, and recent advances in RISC-V such as data centre RISC-V hardware make this technology a more realistic proposition with potential to address these. In this survey paper we explore the current state of art of RISC-V for HPC, identifying areas where RISC-V can benefit the HPC community, the level of maturity of the hardware and software ecosystem for HPC , and identify areas where the HPC community can contribute. The outcome is a set of recommendations around where the HPC and RISC-V communities can can come together and focus on high priority action points to help increase adoption.

Keywords

RISC-V, HPC, Supercomputing

ACM Reference Format:
Nick Brown. 2025. What is RISC-V and why should we care?. In *Proceedings of Cray User Group (CUG '25).* ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

RISC-V is an open Instruction Set Architecture (ISA) that, since it was first introduced in 2010 by the University of California at Berkeley, has grown phenomenally quickly. Within the next few years it is estimated that around 60 billion RISC-V cores will have been manufactured [17] and in late 2024 Nvidia announced that there are between 10 and 40 RISC-V cores in each of their GPUs [18], indeed, Nvidia shipped over a billion RISC-V cores in 2024 alone [18].

However whilst RISC-V is already ubiquitous in computing, Nvidia's use of this technology in their GPUs is a prime example of RISC-V undertaking the underlying management and control which is hidden from the end user. Nevertheless, for all its success in the embedded computing domain, RISC-V is yet to enjoy widespread success in compute solutions that are directly leveraged by users, and High Performance Computing (HPC) is no exception. This comes at the same time that the HPC community are facing new challenges especially around moving towards more sustainable operations, and it is worth considering alternatives to the entrenched technologies of x86, Nvidia and AMD GPUs.

However, the RISC-V ecosystem is evolving very rapidly and in conjunction with the availability of more powerful hardware there are also significant efforts around the RISC-V software ecosystem for HPC. Indeed, in early 2025 the first RISC-V International ecosystem lab for HPC workloads was certified at EPCC [10], providing free access to a range of RISC-V hardware for the HPC community to experiment with for their workloads. It is our experiences in setting up and managing this testbed system that has fed into some of the aspects identified in this paper.

In this survey paper we explore the current state of the art in RISC-V for HPC. Section 2 describes why RISC-V is of importance to the HPC community and the latest status of the RISC-V ecosystem for HPC is explored across hardware, in Section 3, and software, in Section 4. These investigations also highlight areas that require further development and maturity for the HPC community to adopt this technology. Section 5 then surveys the most prominent wider community aspects, before drawing conclusions and identifying a call to action where the HPC community should engage with RISC-V to contribute in Section 6.

2 Why RISC-V for HPC?

The RISC-V standard is overseen by RISC-V International which is a global non-profit organisation responsible for administering the standard itself, related specifications and the stakeholder community. A common misconception amongst the wider computing community is what RISC-V actually is, where some find it surprising that RISC-V itself is not physical hardware and indeed RISC-V International produce no hardware Intellectual Property (IP). Anyone is able to take the RISC-V standard and build hardware around this, and indeed the fact that RISC-V is *just* a standard has been a major argument around why RISC-V should not subject to technology export restrictions. Furthermore, whilst RISC-V is a well designed, clean, ISA there is nothing inherently special in the standard itself which intrinsically delivers improved performance or reduced energy usage alone, compared with other technologies.

In our opinion there are three main reasons why RISC-V is important for the HPC community and these are in-fact based around the way in which the standard and community are organised:

(1) **Openness** means that anyone is able to freely take the standard and build compatible, but specialised, solutions because there are no restrictions on how the standard is actually used or combined with other hardware IP. Ultimately this

CUG '25, New York, NY

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

means that hardware can be driven by HPC requirements and market demand.
(2) **Community driven** means that participants ranging from hardware vendors to software developers are able to shape the standard. The HPC community are able to get involved and by having a say in the evolution of RISC-V we can ensure that it meets our needs.
(3) **Modularity** provided by RISC-V means that it is possible to leverage the standard in a flexible manner enabling the development of bespoke hardware that is targetted at a specific range of applications, with this specialisation ultimately capable of delivering improved performance and energy efficiency.

The first of these is important because it ensures that we as the HPC community are not dependent on a single vendor, such as Nvidia, and then suffer when the price increases, supply decreases or support for aspects such as FP64 is weakened. Instead, there are no legal barriers to entry or cost implications in using the ISA. Anyone is able to take the standard and use it how they want to build technologies around it, indeed whilst many companies leverage RISC-V, a thriving research and individual contributor community has also grown up around it too. As we will highlight in Section 3.2, vendors are using RISC-V in different ways to build high performance and energy efficient accelerators.

Addressing the second point, at the time of writing RISC-V International comprises over 4000 members which involves over 60,000 individuals working across 75 technical working groups. This community driven nature of the standard not only benefits from a wide range of contributing expertise and experience, but furthermore ensures that the standard suits a wide range of workloads due to the ability to easily propose a new extension, a modification to an existing extension, or to get involved in the current development of an extension. Consequently, there are no barriers to members of the HPC community becoming involved in determining the future of RISC-V, and indeed many already have.

The third point, modularity, means that RISC-V can apply across a wide range of computing platforms from embedded computing to high performance. The base RISC-V integer instruction set, RV32I for 32-bit and RV64I for 64-bit is very simple and, for example, does not even include integer multiplication and division. There are over 100 ratified extensions that vendors can pick and choose from when designing their hardware. However with some caveats and care, software, and especially tooling, developed for one set of extensions can also be leveraged with hardware built around a different set. There is an increased focus in HPC around energy efficiency and the sustainability of operations, hardware that is customised for a specific range of applications has the potential to better deliver this compared to general purpose solutions.

3 Hardware

As highlighted in Section 2, over 100 RISC-V extensions have been ratified and one of the challenges with this is that there are many different components which may or may not be leveraged by a piece of hardware. This can make it difficult to identify the most appropriate hardware to purchase, and for developers to know what technology they are developing against. To this end, RISC-V International have standardised a set of architecture profiles that contain a range of mandatory and optional extensions. The RISC-V Application (RVA) profile is most applicable for HPC, although there is also the RVB profile designed for inexpensive highly customisable application processors, and RVM for microcontrollers.

Three RVA standards have been ratified, with RVA23 being the most recent. These are intended to specify 64-bit application processors that support binary software ecosystems relying on a set of guaranteed extensions and a small number of discoverable coarse-grain additional options.

Table 1 describes the most important extensions for HPC that are provided by the RVA23 profile, with the full profile definition at [9]. It can be seen that the profile initially defines the basic CPU support such as integer multiplication and division, and then specifies how the processor will interact with the cache and main memory, along with other aspects that are important for HPC such as hardware performance counters and atomic instructions (although the set of events that these count is platform specific).

RVA23 is the first application profile to make vector extensions, the *V* extension, mandatory, and providing Vector Length Agnostic (VLA), similarly to Arm's Scalar Vector Extensions (SVE), it is possible to target a range of vector lengths using the RISC-V Vector extension (RVV). Due to the demand for vectorisation, a draft version of the RISC-V vectorisation, RVV v0.7.1, became an unofficial de facto standard. Whilst it was explicitly stated that RVV v0.7.1 was subject to change and should not be relied upon, hardware manufacturers started making CPUs which implemented RVV v0.7.1. Due to the much longer lead times in hardware development, compared to software, we still see many RVV v0.7.1 RISC-V CPUs in the market place, and indeed these will around for some time to come. The challenge is that the ratified RVV v1.0 is not backwards compatible with v0.7.1, and compiler and library support now focusses on the ratified version only. Therefore, in order to exploit hardware built around RVV v0.7.1, one needs to use vendor specific forks of the GCC compiler which lag mainline. Whilst this is an early teething issue, and will be inconsequential in years to come as hardware matures, it demonstrates the perils that the community can face when working with early technologies that are still progressing towards full maturity.

Two important extensions to highlight in Table 1 are *Sv48* and *Sv57* which extend the default, mandatory, 39-bit virtual memory address space to 48 and 57 bits respectively. We highlight these because they were proposed and championed by members of the RISC-V International HPC Special Interest Group (SIG) who are responsible for driving the adoption of RISC-V in HPC. This is an example of the community nature of RISC-V, where members with specific areas of expertise can contribute and shape the standard to meet their needs, in this case to ensure that RISC-V is future proof in the medium term for large-scale workloads.

Driven largely by machine learning workloads, and following Intel's Advanced Matrix Extensions (AMX) and Arm's Scalable Matrix Extension (SME), at the time of writing in early 2025, the RISC-V community are exploring the standardisation of matrix extensions. Indeed, numerous RISC-V hardware vendors have already developed their own custom matrix instructions as add-ons, and-so a logical step is to combine and standardise these as an officially ratified extension. Because BLAS operations are fundamental to

Table 1: Summary of most applicable extensions provided by RVA23, with the full list to be found at [9]

Extension	Description	Type
I,M,A,C,B	Base instructions for integer, bit, compressed and atomic operations	Usermode
F,D	Single and double precision floating-point	Usermode
Zicntr,Zihpm	Basic and hardware performance counters and timers	Usermode
Ziccif,Ziccrse,Ziccamoa	Defines the required support with main memory, ensuring cache coherence, support for atomics and misaligned memory accesses	Usermode
Zic64b,Zicbom,Zicbop	Defines cache support, with blocks of 64 bytes size aligned to address space with management and prefetch instructions	Usermode
V	Vector instructions	Usermode
Zvkng, Zvksg	Vector crypto algorithms	Usermode, optional
Zfh, Zvfh	Scalar and vector half-precision support	Usermode, optional
Zfbfmin, Zvfbfmin	Scalar and vector BF16 conversion support	Usermode, optional
Sv39,Svade,Svpbmt	Page-based 39-bit virtual memory system and associated support	Supervisor mode
H	Hypervisor	Supervisor mode
Sv48,Sv57	Page-based 48-bit and 57-bit virtual memory system	Supervisor mode, optional

many scientific applications, this is also very topical for the HPC community. In additional to mathematical operations performed upon entire matrices and vectors, these extensions also support individual element wise operations, data reordering such as transposition, and flexible data loading and storage between the registers and main memory. The RISC-V community are currently working on developing two separate matrix extension standards, Integrated Matrix Extensions (IME) and Attached Matrix Extensions (AME). The major difference between these is that IME leverages RVV's vector registers to hold the matrices, whereas AME introduces dedicated 2D, tiled, matrix registers. IME is more of an incremental approach, providing an extension that will leverage much of RVV's existing hardware support ultimately requiring less additional hardware complexity. By contrast AME, which follows the approach adopted by AMX and SME, requires additional dedicated hardware and potentially increased support from compilers and libraries. Whilst AME is more of a heavy weight approach, these specialised hardware structures provide the potential to deliver greater performance than IME.

Current activities around defining RISC-V matrix extensions demonstrates both the potential benefits of an open community driven standard, but also the pitfalls. It is difficult to imagine with other ISAs that there would be different versions of an extension standard in development. It might be that one of these falls by the wayside or ultimately that both are standardised and co-exist. Indeed, as already described, much of this has been driven by RISC-V vendors such as Andes, SiFive, Stream Computing and Xuantie who, due to the openness of RISC-V, have already been able to develop their own bespoke matrix extensions and couple these with RISC-V CPUs as part of a product line. Consequently, the RISC-V community already contains knowledge and experience upon which to base their decisions when moving towards standardisation. However, a potential pitfall is that RISC-V could become fragmented and additional burdens could be placed upon developers who must support a range of potential hardware configurations. This is potentially where profiles, which mandate specific extensions, can help, but furthermore it is likely that some of the specific details can be abstracted away from HPC developers by the compilers or maths libraries lower down the stack.

3.1 CPUs

Whilst there has been some progress, as of early 2025, the availability of RISC-V based CPU that are suited to high performance workloads is a challenge. Based upon our own experiences, when we first set up a RISC-V testbed in 2022 hardware choices were very limited and only CPUs designed for embedded style computing were available. Indeed, in 2022 the best option was the HiFive Unmatched which contains the SiFive Freedom U740 CPU. This is described as a System on a Chip (SoC) including a high performance quad core, 64-bit dual issue, superscalar RISC-V processor with 16GB of DDR4 memory [8]. The U740 implements RV64I with MAFDC extensions (base integer instruction set with integer multiplication and division, atomic, single and double precision scalar floating point, and compressed instruction extensions). However, through benchmarking, it was found that the performance of this hardware was lacklustre and the single RISC-V core in the Allwinner D1, which was around ten times cheaper, delivered largely comparable performance for the RAJAPerf benchmark suite [11].

Until late 2023 the highest performing commodity available RISC-V CPU was the quad core JH7110 SoC provided in the VisionFive V2. Whilst this was still designed for embedded workloads, it was able to deliver higher performance than the U740 and Allwinner D1, which was a step forwards. However, a very important development towards RISC-V in HPC was when SOPHGO released their 64-core Sophon SG2042 CPU. Comprising the XuanTie C920 core which implements the RV64IMACV instruction set, the C920 has three decode, four rename/dispatch, eight issue/execute and two load/store execution units along with 64KB of L1 instruction (I) and data (D) cache. Furthermore, there is 1MB of L2 cache shared between a cluster of four cores in the SG2042, and an additional 64MB of L3 system cache which is shared by all cores in the VisionFive V1, SiFive U740 and All Winner D1.

Table 2 provides a single core performance comparison across RISC-V technologies for the NPB benchmark kernels [1] (running

Table 2: Single core comparison between RISC-V technologies with performance reported in Mops/s (Higher is better) across NPB kernels running at class B.

Hardware	Mop/s	% performance of SG2042
Sophon SG2042	472.97	-
VisionFive V2	121.62	25.7%
VisionFive V1	39.31	8.3%
SiFive U740	49.03	10.4%
All Winner D1	47.71	10.1%
Banana Pi F3	146.83	31.0%
Milk-V Jupyter	158.64	33.5%

at class B) and compiled with GCC 13.2. From this it can be seen that a single core of the SG2042 delivers approximately four times the performance of a single core in the VisionFive V2's JH7200, and around ten times the performance of cores in the the other hardware.

Another advantage of the SG2042 is that it is a stand-alone CPU rather than an SoC, providing more flexibility around integrating with memory and other aspects such as PCIe, as well as more generally supporting the building of data centre systems around this part. In addition to Milk-V producing the Pioneer workstations that combines this CPU with 128GB of DDR4, E4 Computer Engineering have also developed an experimental dual socket system based around this CPU.

Previous work [4] undertook a performance comparison of the SG2042 against other architectures using NASA's NPB benchmark suite and measuring against a 64-core AMD EPYC 7742 which is in the ARCHER2 Cray EX supercomputer, 26-core Intel Xeon Platinum (Skylake) 8170, and a 32-core Marvell ThunderX2 CN9980. Figure 1 reports performance for the Embarrassingly Parallel (EP) benchmark, which is heavily compute bound, across these architectures. It can be seen that, until the ThunderX2's 32 cores are exhausted, the SG2042 and ThunderX2 deliver very similar performance with the increased core count in the SG2042 resulting in a performance improvement when running across all CPU cores. The AMD and Intel CPUs deliver better performance then the SG2042 at comparable core counts, however as the Xeon only contains 26 cores and at 32 cores performance delivered by the SG2042 is comparable, that at 64 cores the SG2042 outperforms the Xeon.

The SG2042 implements RVV v0.7.1, with both the ThunderX2 and SG2042 providing 128-bit wide vectorisation, albeit with the ThunderX2 having two FPUs per core compared to one on the SG2042. By contrast, the Intel Xeon and AMD EPYC CPUs provide 512-bit and 256-bit wide vectorisation respectively. This vector configuration in part helps explain the performance difference between these CPUs.

Whilst the EP benchmark is compute bound, by contrast the Multi Grid (MG) benchmark is memory bandwidth bound. Performance across these architectures for the MG benchmark is reported in Figure 2, and it can be seen that the SG2042 significantly lags behind the other three technologies. The SG2042 comprises four memory controllers, each with a single memory channel connected to DDR4-3200. The AMD EPYC has eight memory controllers and eight memory channels also connected to DDR4-3200, whereas the

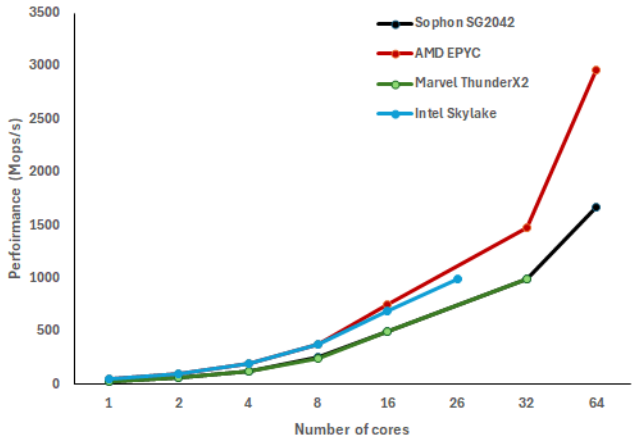

Figure 1: EP benchmark performance (higher is better) parallelised via OpenMP from [4]

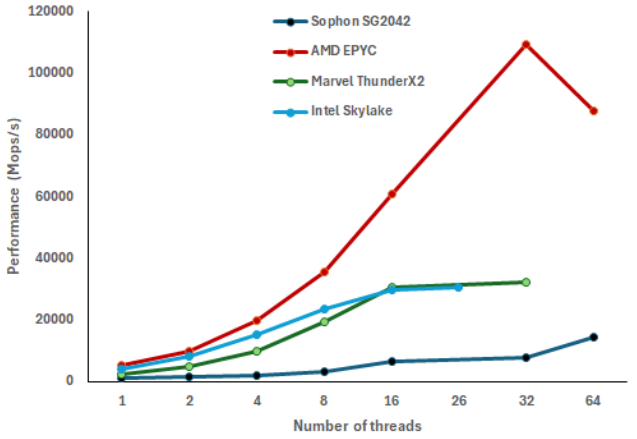

Figure 2: MG benchmark performance (higher is better) parallelised via OpenMP from [4]

Intel Xeon and Marvel ThunderX2 both have only two memory controllers connected to DDR4-2666, with the ThunderX2 having a total of 8 memory channels and there are 6 memory channels in the Xeon. Indeed, other benchmarks [3] have observed a similar pattern of behaviour and from this we conclude that the weakness of the SG2042 for HPC workloads is that the hardware is limited by memory performance. Whilst having only four memory channels might partly explain these differences, clearly there are other factors that underlie this behaviour, potentially around the design of the SG2042's memory controllers but documentation on this is sparse.

In early 2024 SOPHGO announced several next generation RISC-V CPUs including the SG2044 which provides RVV v1.0 and is claimed to have around three times the memory performance of the SG2042 [20]. Whilst this CPU has not yet been publicly released, Geekbench 6.3 benchmarking results from a development sample have been made available [7]. From these results it can be seen that single core performance between the SG2042 and SG2044 is

comparable, however for multithreaded workloads on average the SG2044 delivers 138% the performance of the SG2042, with a maximum of a 413% performance improvement for the *horizon detection* benchmark.

However it is unclear whether the SG2044, and other SOPHGO products, will ever be mass produced and available as commodity parts. This is in large part due to challenges SOPHGO has recently faced with manufacture since TSMC terminated their relationship because of SOPHGO being added to the US sanctions list [12]. Consequently, whilst SOPHGO's roadmap holds significant potential for RISC-V HPC hardware, whether they will be able to mass produce these CPUs is unclear.

In late 2024 the SpacemiT K1 RISC-V SoC became available in the Banana Pi F3 and Milk-V Jupyter boards. Whilst this SoC is targetted at embedded workloads, it is interesting because it is the first mass produced RISC-V CPU to support both the RVA22 profile (the application profile preceding the latest, RVA23) and RVV v1.0. Performance for the NPB benchmark kernels on a single core of the SpacemiT K1 in both the Banana Pi F3 and Milk-V Jupyter is reported in Table 2, where it can be seen that this is an improvement over other SoC based RISC-V technologies but performance still falls short of a single core in the SG2042.

As a demonstration of binary compatibility provided by RISC-V, and some of the related nuances, it is possible to run the same compiled binary executable executable across all hardware technologies considered in this section. However, to be compatible with the VisionFive, SiFive and All Winner CPUs this binary can not contain vector instructions. Consequently, more performance will be gained by recompiling for the SG2042 with vectorisation enabled to generate RVV v0.7.1 instructions where appropriate. However, the SG2042 target binary will not be compatible with the SpacemiT K1 because this implements RVV v1.0, which is not backwards compatible with RVV v0.7.1. The binary that has been compiled for the SpacemiT with RVV v1.0 enabled will itself not be compatible with the other hardware as these do not support RVV v1.0.

3.1.1 Networking. An important consideration when scaling out HPC workloads is support for high performance networking technologies to enable distributed memory parallelism. Experimental support for Omnipath and Infiniband has been developed by the RISC-V community with, for example, RISC-V support has been implemented for Infiniband but not IB verbs as these are proprietary to Nvidia. Whilst, to the best of our knowledge, Cray's networking technology Slingshot has not yet been demonstrated on RISC-V, OpenFabrics Interfaces (OFI), has been ported to RISC-V with the *libfabrics* driver available. OFI exports high performance communication fabric services to applications and consequently, in theory at least, this should significantly ease integration with high performance networking technologies. For example, HPE Slingshot Host Software (SHS) provides *libfabric* libraries for the HPE Slingshot NIC which is then used by Cray's MPI library.

Whilst there has been some progress around high performance network support in RISC-V, it should be highlighted that this would benefit from additional effort. The HPC community, and HPC vendors such as HPE and Nvidia, should look to support mature RISC-V networking solutions that can be leveraged in production supercomputers.

3.2 Accelerators

Thus far we have focussed on RISC-V CPUs, however the major challenge with these from the perspective of a supercomputer operator is that the entire machine must be built around the CPU, requiring dedicated motherboard support, device drivers and a mature software ecosystem. A more incremental approach to RISC-V adoption is that of supercomputing centres fitting RISC-V PCI express (PCIe) accelerators into existing systems. Indeed, there are several vendors who have announced, or are shipping, RISC-V based accelerators for AI and HPC workloads which we survey in this section.

Esperanto are shipping the ET-SoC-1 which is an accelerator designed for AI workloads and indeed was on the Penguin Computing SC24 exhibition booth who act as an integrator. Esperanto initially used their own bespoke ISA but then switched over to RISC-V as it gained in popularity because it is the only free and open ISA with a critical mass of adoption [5]. The view of their founder and CTO Dave Ditzel, who wrote the famous *The case for the reduced instruction set computer* paper [15] in the 1980s, was that RISC-V is where innovation in ISAs will occur because it enables and encourages advancements to be made with both private and official extensions, as well as the standard providing a clean ISA that can be used to build more energy efficient hardware. Ditzel highlights that, ultimately, leveraging RISC-V not only the *best* choice, but in fact the *only* choice [5].

Figure 3 sketches Esperanto's architecture. The ET-Minion, a 64-bit RISC-V CPU with two hardware threads and 4KB L1 data cache/scratchpad, is the lowest level and this is coupled with a bespoke vector/tensor unit which has been designed for AI workloads. The vector/tensor unit is capable of undertaking sixteen single precision floating point operations per cycle, or thirty two half precision, and the tensor component enables operation of up to 512 cycles with a single tensor operation to avoid the reduce the overhead on instruction fetch and reduce power. The transcendental unit provides support for transcendental functions (e.g. exponentials, logs and trigonometry) via a ROM which issues micro operations for each of these functions and saves area compared to having dedicated hardware support. The ET-Minion is an example of the Integrated Matrix Extension (IME), where the same vector registers and compute unit are reused to undertake more complex matrix operations. Whilst the RISC-V core follows the RISC-V ISA standard, the ET-Minion's vector unit has it's own simple instruction set. Eight ET-Minions are grouped together into a *neighbourhood* and share a 32KB L1 instruction cache/scratchpad memory. Four neighbourhoods are then grouped into a *shire* which also contains 4MB of SRAM memory and a router connects the shire to the Network on Chip (NoC). The ET-SoC-1 contains 34 shires, comprising 136 neighbourhoods and a total 1088 ET-Minions, draws between 15 and 60 Watts and also provides external DDR4 memory connected to the NoC. The approach is to build up from simple yet compute capable cores that have access to high speed local memory and high performance interconnect. The current generation ET-SoC-1 supports a maximum of 32-bit, single, precision floating point and is quoted as delivering 12.5 TFlops FP32. However, to increase suitability of their accelerator for traditional HPC workloads, Esperanto have announced that the next generation, with a target release date

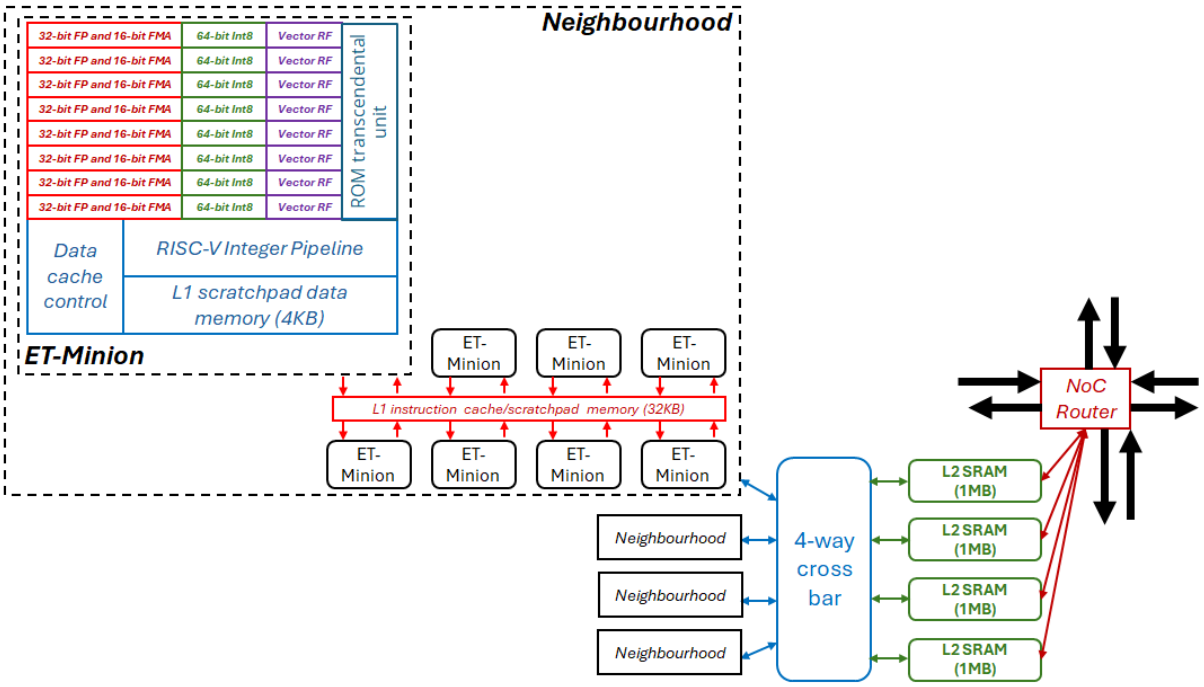

Figure 3: Architecture sketch of a Minion shire which contains four neighbourhoods, each of which contains eight ET-Minions. In an Esperanto ET-SoC-1

of 2026, will support double precision floating point and is project to deliver 16 TFlops FP64 per chip [5].

Arguably, Tenstorrent ship the most popular RISC-V based accelerators that are currently available on the market. They sell two generations of PCIe accelerator card, Grayskull and Wormhole, with the next generation Blackhole available in the coming months. Their architecture is built around a *Tensix* core which is illustrated in Figure 4 and comprises five RISC-V CPUs with a matrix and vector unit along with 1.3MB of fast SRAM. The concept behind this architecture is to decouple the movement of data from compute, where one RISC-V core is used to move data from the NoC (e.g. data held in DDR or another Tensix core) to internal SRAM, this is then consumed by three RISC-V cores that drive a bespoke matrix and vector unit, and lastly a fifth RISC-V core write results held in SRAM back onto the NoC for instance to DDR. The Tensix units are then scaled up and communicate via a NoC, with 120 Tensix units in the n300 Wormhole PCIe accelerator. The Tenstorret and Esperanto philosophies differ, as Esperanto favours a large number of identical simple cores all with their own moderately sized vector/tensor unit, whereas Tenstorrent go with a smaller overall number of matrix/vector units but each of these is more capable and furthermore in Tensix the RISC-V cores each have their own dedicated roles.

It can be seen in Figure 4 that *Circular Buffers* are used to communicate between the RISC-V cores in a producer-consumer manner by allocating pages of memory in SRAM and then making these available to the consumer, with the intention that the RISC-V cores are all running concurrently and pipelined across data read, compute and data writing. Whilst the RISC-V cores conform to the RV32IM standard (32-bit RISC-V with base integer instruction set and integer multiplication and division), the Tensix compute engine itself uses a bespoke instruction set. This is similar to Esperanto, where both vendors developed vectorisation before RVV was standardised. Tenstorrent have followed the Attached Matrix Extension (AME) approach, where separate 2D tiled matrix registers (*SrcA*, *SrcB* and *Dest* in Figure 4) are consumed by a dedicated matrix unit which is capable of performing 2048 multiplications and 2048 additions per cycle. There is also a separate vector unit which contains eight 1024-bit wide registers. The vector unit shares the *Dest* register with the matrix unit, both consuming from and writing results to this register. Whilst the first generation Grayskull provided a maximum of BF16 precision, the next generation Wormhole provides FP32 (although the matrix units relaxes IEEE standard compliance which can result in reduced accuracy). However, unlike Esperanto, to the best of our knowledge there is no current Tenstorrent technology in their roadmap with FP64 support.

In [2] the authors explored the use of the Tenstorrent Grayskull for running a Jacobi stencil application solving Poisson's equation for diffusion. This was the first published study leveraging a RISC-V based PCIe accelerator for scientific computing workloads, and both performance and energy results for the Grayskull e150 are reported in Table 3, comparing against a Xeon Platinum Cascade Lake 8260M. *e150 initial* was the initial performance obtained on the Grayskull on one Tensix core, and it can be seen that this was 217 times slower than a single CPU core. Through optimisations of the code to suit the Tenstorrent architecture, it was possible to increase

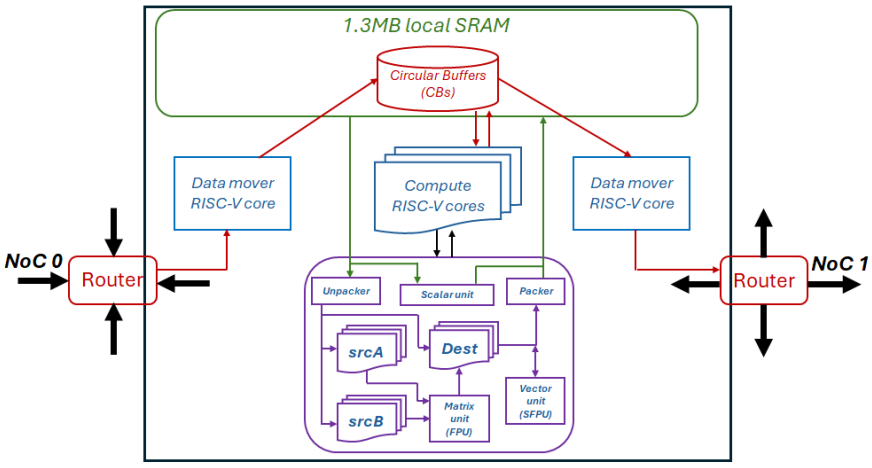

Figure 4: Architecture sketch of a single Tensix core in the Wormhole Tenstorrent accelerator

single Tensix performance 163 times and then by scaling up across the 108 Tensix cores in the Grayskull comparable performance to all 24-cores of the Xeon Platinum was obtained but at around five times less energy. Moreover, costing around $700 it should be noted that the Grayskull RISC-V accelerator is significantly cheaper than this Intel Xeon CPU. Work is early here, but demonstrates some promise and the next generation of Tenstorrent's technology, the Blackhole, will also contain sixteen application RISC-V cores (RV64) that are capable of running Linux, and Tenstorrent are also developing the high performance RISC-V Ascalon CPU [19] which is intended for the HPC market.

Table 3: Performance and energy usage comparison for a Jacobi iteration on the e150 Grayskull accelerator solving LaPlace's equation for diffusion with a problem size of 1024 by 9216 (9.4 million) BF16 elements, over 5000 iterations from [2]

Type	Total cores	Performance (GPt/s)	Energy (Joules)
Xeon Platinum CPU	1	1.41	1657
Xeon Platinum CPU	24	21.61	588
e150 initial	1	0.0065	335412
e150	1	1.06	2094
e150	2	2.48	893
e150	4	2.92	744
e150	8	7.99	276
e150	32	9.20	240
e150	64	12.96	170
e150	72	17.26	128
e150	108	22.06	110

Across both Esperanto and Tenstorrent the ability to scale up is a major focus, where the chips and accelerator cards can be connected together via one large, high bandwidth low latency, network via the NoC. Indeed, Esperanto have already built a machine with eighty interconnected ET-SoC-1 cards which provides over 80,000 RISC-V cores. Tenstorrent sell a 6U data centre server called the *Galaxy* which comprises 32 interconnected accelerators together. Whilst Esperanto and Tenstorrent first targetted the AI market and have now started to explore other markets including HPC, Inspire Semiconductor have from the start focussed on scientific computing workloads and plan on providing double precision FP64 support from their first generation onwards. At the time of writing their RISC-V Thunderbird accelerator is yet to reach the market, but they adopt a similar approach of *a supercomputer cluster on a chip* by coupling many powerful (superscalar) 64-bit RISC-V cores containing vector units, a high performance NoC and tightly integrated high speed SRAM with the intention of delivering high performance and energy efficiency across a range of HPC workloads. The Thunderbird PCIe accelerator will contain 6144 RISC-V cores and InspireSemi's energy efficiency target is 75 GFLOPS/Watt (FP64), although this will need to be validated by the community once the technology becomes available.

Regardless of the specific vendor, it can be observed that they are generally adopting an approach of combining a large number of relatively simple, energy efficient, RISC-V CPU cores integrated with dedicated compute. These are then coupled with a relatively large local DRAM memory and interconnected via a high performance NoC. This specialisation of the hardware designed with a specific set of workloads in mind and focussing on key aspects of compute, local memory and interconnect is crucial in delivering the performance and energy efficiency that these vendors claim and we observe. For HPC specifically this design approach is potentially very applicable for workloads that are memory or communication bound, and Fourier Transforms are an example where the transposition required for 2D or 3D FFTs requires all-to-all communications resulting in significant overhead at scale on existing architectures, but the focus of these accelerators on high performance communications via a NoC could be beneficial here.

The accelerators described thus far used the RISC-V CPU primary for driving the compute which is built around a custom instruction set. As part of the European Processor Initiative (EPI), the EPAC

architecture includes RISC-V vector tiles which consist of a Semidynamics Avispado RV64 CPU core coupled with Semidynamics's Vitrivius vector processor (VPU). This is a different approach because the CPU core and VPU are tightly coupled together, with vector instructions issued to the CPU core then forwarded onto the VPU. Vitrivius implements a 16384 bit wide vector unit conforming to RVV v0.7.1, ultimately providing the capability to handle a maximum of 256 double-precision, FP64, floating point numbers. Unlike the other accelerators surveyed in this section, a major advantage to this approach is that the instruction set is unified across the CPU and VPU, consequently from the perspective of the end programmer their code can remain unchanged as the VPU appears to them as a (very) wide CPU-based vector unit with the compiler mapping loop iterations to vector lanes.

The EPAC vector tile uses Semidynamics's custom Open-Vector Interface (OVI) technology to link the CPU cores and the VPU. Indeed, combining RISC-V CPU cores with additional Intellectual Property (IP) is very common and numerous vendors have developed their own approaches to managing this. Recognising that all these custom approaches were not optimal, RISC-V International began a task group in 2024 to develop a composable extensions standard that will enable a range of RISC-V extensions to be flexibly combined with RISC-V cores and driven by software in a standard way. Whilst work is still early, it can be seen from the technologies surveyed in this paper how common it is for vendors to couple RISC-V with their own bespoke hardware ranging from wide vector units to matrix engines. There is potential here for vendors to develop even more specialised hardware components for HPC workloads, for example to accelerate certain MPI activities, and a standard way to integrate these together will be beneficial for both hardware vendors but also library developers who expose them into software.

4 Software ecosystem

The Instruction Set Architecture (ISA) provides a contract between the hardware and software, and whilst hardware and the performance that it delivers (as discussed in Section 3) is important, a range of software to actually run on RISC-V is crucial if this technology will become a realistic proposition for HPC. Given the standard nature of the RISC-V ISA, a potential benefit is that work done porting and optimising software for one RISC-V technology should be able to run on other RISC-V hardware with minimal changes at the code level. Indeed, much work has been done by the wider community to port a very wide range of tools over to RISC-V and prebuilt binary packages exist for many of the popular libraries and tools.

Indeed, major Linux distributions including Debian, Fedora, Ubuntu and openSUSE all provide mature support for RISC-V and when we set up our own RISC-V testbed we found that all the major building blocks including Network File System (NFS) and Slurm were already available. Whilst NFS is currently the most popular networked filesystem to run with RISC-V, BeeGFS by ThinkParQ is also supported and has been used by E4 Computer Engineering's Monte Cimone RISC-V cluster [13]. At the time of writing there are no active efforts to port Lustre to RISC-V and in order to provide a mature HPC solution this should identified as a priority. As Lustre is open source this is something that the community can address, and work that started in 2017 to provide Arm supprt in Lustre will likely be instructive and help identify the key activities that are required.

Table 4: Summary of key library, tooling and infrastructure support of importance to the HPC community that has been ported to RISC-V

Software	Type
BLIS	Library
OpenBLAS	Library
FFTW	Library
PETSc	Library
NetCDF	Library
HDF5	Library
MPI	Library
GCC	Tooling
LLVM	Tooling
Extrae	Tooling
Slurm	Infrastructure
NFS	Infrastructure
BeeGFS	Infrastructure
libfabric	Infrastructure

Common HPC libraries, tools and build support technologies such as MPI, compiler support for OpenMP, *make* and *cmake* are all provided for RISC-V both as prebuilt binaries and are also trivial to build from source for the architecture. Table 4 summarises key libraries, tooling and infrastructure software package support provided by RISC-V that is of important to HPC. At the tooling level support for RISC-V in both mainline GCC and LLVM is mature and well supported by the community. However, the caveat here is compiler support for long vectors. Much of the work on compiler vectorisation has focussed on short, 512-bit wide or less, vectors but as highlighted in Section 3.2 computed with other architectures we are seeing RISC-V hardware, especially accelerators, with much wider vector units. This demonstrates how the flexibility provided by RISC-V challenges some of the classical assumptions in the tooling, and whilst there is on-going work by the RISC-V LLVM community to address this it is yet to be merged into mainline [6]. MLIR is a composable compiler ecosystem that, since it was developed by Google in 2019 and merged into the main LLVM codebase around five years ago, has grown very significantly. Indeed, several compilers and frameworks including Tensorflow, LLVM's Flang and Tenstorrent's ML compiler are based upon MLIR. Built around Intermediate Representation (IR) dialects, that can be mixed, and transformations between these, dialects for x86 and Arm (SVE and Neon) vectorisation are provided by MLIR. However, whilst some preliminary work has explored a RISC-V vectorisation, RVV, dialect to date this is immature and there is no such dialect as part of MLIR itself. Based upon the importance of vectorisation for HPC workloads, it would likely be beneficial for the community to enhance the underlying tooling in this manner.

It can be seen from Table 4 that a major weakness for HPC is in performance profiling tooling, with these not yet mature or widely available. Whilst BSC's Extrae performance analysis suite

Table 5: Summary of key applications on the ARCHER2 Cray-EX supercomputer and whether they support running on RISC-V

Application	Percentage ARCHER2 usage	Number of users	Supports RISC-V?
VASP	18.4%	153	No
Met Office UM	8.9%	37	No
GROMACS	6.8%	115	Yes
Python	4.2%	64	Yes
LAMMPS	4.1%	45	Yes
OpenFOAM	4.1%	49	Yes
CP2K	4.0%	48	Yes
Nektar++	2.7%	14	No
CASTEP	2.5%	46	Yes
NEMO	1.8%	19	No
SENGA	1.3%	5	No
CESM	0.7%	10	Yes
Quantum Espresso	0.7%	46	Yes
NAMD	0.5%	3	Yes
WRF	0.1%	4	Yes
Other	12.4%	121	No
Unknown	26.8%	-	N/A

is available, this is not as commonly used as other tools such as CrayPat, Intel's Vtune, or Linaro's MAP. There are no mainstream performance tools that support RISC-V which is due to a mixture of the software side by the tool developers but also the hardware events that are made available by the RISC-V CPUs. In order to drive adoption of RISC-V into HPC, performance tooling should be considered as a priority.

Table 5 summarises the key applications that run on the Cray-EX ARCHER2 UK national supercomputer, their percentage runtime usage of the machine, the number of unique users in a month and whether these applications support RISC-V or not. The *unknown* category represents jobs where the application name has been unable to be retrieved and *other* represents a large number of small usage percentage jobs none of which support RISC-V. Out of the jobs running on ARCHER2, it can be seen that 27.7% (35% when ignoring the unknown category) of the machine runtime is by applications that have been ported to RISC-V with efforts actively maintaining these for the architecture. Furthermore, 55% of ARCHER2 users would be able to run their applications on a RISC-V machine. Whilst this list is somewhat country specific, for example elsewhere the usage of WRF is likely much greater and the Met Office UM lower, clearly a high priority for the community is to support RISC-V in VASP.

In 2023 a consortia of RISC-V vendors formed the RISC-V Software Ecosystem (RISE) project, which aims to strategically address some of the shortcomings in the RISC-V software landscape. Whilst much of this effort is driven by embedded computing concerns, the HPC community can still benefit, for example RISE have undertaken vectorisation of FFTW for RISC-V (although this work has focussed exclusively on RVV v1.0 and is not compatible with CPUs implementing RVV v0.7.1). In 2024 RISE released a RISC-V software optimisation guide [16] which details best practice around porting codes to, and optimising for, RISC-V. Whilst there are some aspects of this guide that are useful for HPC developers, such as being able to trivially detect which RISC-V extensions are supported by target hardware, in the main it is rather low level and aimed heavily at those working at the assembly level. Nevertheless it is interesting to observe that the standardisation of RISC-V results in a single guide then applying across a range of different hardware, however a higher level optimisation guide that would more suit the level that HPC code developers work on would be beneficial.

As an aside, one of the potential pitfalls in the RISC-V software ecosystem is in application and library level support for cross compiling. Due to some of the current performance limitations of RISC-V hardware, as highlighted in Section 3.1, it is common for developers to cross compile RISC-V binaries on other systems, such as x86. However, at the time of writing this can cause challenges with more complex applications especially in our experience those leveraging MPI. Whilst the solution is to compile on a native RISC-V system, this can increase build times.

4.1 Fortran

The HPC community is somewhat unusual in it's reliance on Fortran, indeed on the Cray-EX ARCHER2 supercomputer on average over 60% of workloads are written in Fortran and these account for approximately 65% of the machine runtime. The wider RISC-V community has focussed heavily on support for languages such as Python, Java, C and C++, and there has been less of a focus on Fortran due to the smaller community who are interested in it.

Whilst both the GNU Fortran compiler, gfortran, and LLVM's flang, are both supported for RISC-V there isa common belief in the community that Fortran performance lags that of other languages on RISC-V. To explore this we undertook an experiment comparing the performance of Fortran and C implementations of kernels from the NASA's parallel benchmark suite (NPB). With the exception of the IS benchmark, benchmarks in the official NPB release are all written in Fortran, however an unofficial C version has also been developed [14] by the University of Versailles Saint Quentin en Yvelines. Comparing the performance delivered by the Fortran and C versions provides an indication around whether one language delivers better performance than the other. There will be some inherent performance differences between the Fortran and C versions, and-so undertaking this comparison on x86 as a base line and then RISC-V is instructive. If the hypothesis that Fortran support on RISC-V is poor then we would expect the comparable performance between Fortran and C on RISC-V to be lower than the x86 baseline.

Table 6 reports results from this experiment when running on a single core of AMD EPYC (Rome) 7742 in the Cray-EX ARCHER2 supercomputer and a single core of the SG2042 RISC-V CPU. The metric reported for each benchmark is the number of times faster the Fortran implementation is compared to the C version, where CG, EP and FT are kernels, and BT and LU are more complex pseudo applications, all compiled with GCC v13. For the simpler kernels it can be seen that performance between Fortran and C is fairly comparable between x86 and RISC-V, however there is a difference for the more complicated pseudo applications where the relative performance of Fortran compared to C on RISC-V is

Table 6: Speed up of Fortran vs C for three NPB kernels and two pseudo applications (class C) for GCC v13 on single core of x86 baseline and RISC-V SG2042.

Benchmark	x86 baseline (AMD EPYC 7742)	RISC-V (SG2042)
CG	0.95	1.01
EP	2.95	2.45
FT	1.13	1.47
BT	1.30	3.35
LU	1.38	2.15

significantly better than that of x86. These results do not support the commonly held belief that performance delivered by Fortran on RISC-V is significantly worse than other architectures due to lack of compiler support. It should be highlighted that these experiments were undertaken with GCC where the gfortran compiler is more mature than LLVM's Flang. As LLVM is extensively used on RISC-V it might be that this is part of the explanation to this misconception, where people are experiencing the general immaturity of Flang rather than support for one specific architecture.

4.2 Accelerator programming

The discussions in this software focussed section have thus far concentrated on support for RISC-V CPUs, however, as highlighted in Section 3.2 there are also several RISC-V based accelerators. Whilst each of these accelerators is based around RISC-V, there are enough differences in their architecture and general paradigm to require a vendor specific API and SDK. Esperanto provide an ML framework compiler that supports machine learning models via PyTorch and TensorFlow, and also GP-SDK which supports general purpose programming via C++. Similarly, Tenstorrent provide their TT-NN ML compiler and the TT-Metal SDK which supports low level direct programming of their technology. Tenstorrent, unlike other vendors, have open sourced their entire software ecosystem and development is done in the open via Github. This is highly beneficial, especially with technologies that are still evolving, as it enables advanced developers to dig into the internals to understand the entire flow and potentially debug underlying issues. Indeed, one of the ways in which the authors of [2] managed to optimise their stencil implementation on the Grayskull was to make changes to TT-Metal to reduce the need for explicit data copying, which was then fed back to the vendor.

However, requiring HPC developers to move their codes, many of which are legacy with many years of development, to a new programming model, paradigm and potentially new language is not a realistic proposition. Research is being undertaken around how best to drive these accelerators using familiar technologies, for example via Fortran and OpenMP. Built upon the MLIR composable compiler ecosystem, the Flang compiler offers opportunities here because vendors such as Tenstorrent have developed MLIR support. Based upon this ecosystem the community have developed a prototype flow to target Tenstorrent accelerators from Fortran. Listing 1 provides an example OpenMP decorated Fortran code that is supported by this flow, where the *simd* directive maps loop iterations to the SIMD lanes within the vector unit a Tensix core, and *parallel do* will map loop iterations across Tensix cores. In this code example *num_threads(20)* will decompose the loop across twenty Tensix cores, each of which handles a subset of the loop iterations which are mapped to their to SIMD lanes.

```
subroutine saxpy(a, x, y, n)
  ...
  !$omp omp target distribute parallel &
  !$omp& do simd num_threads(20) &
  !$omp& map(to:x) map(tofrom:y)
  do i=1, n
    y(i) = a * x(i) + y(i)
  end do
  !$omp end target teams distribute &
  !$omp& parallel do simd
end subroutine
```

Listing 1: Example Fortran code running Single-precision A times X Plus Y (SAXPY) on the Tenstorrent accelerator card (argument declarations omitted for brevity)

Whilst this work is still early, it demonstrates that recent developments elsewhere in the computing community around MLIR deliver potential opportunities to aid in the adoption of RISC-V in HPC. With additional involvement from the HPC community, compiler developers and vendors, there is the potential to make these programming technologies more mature, accessible and to help support accelerating large and complex workloads on RISC-V accelerators.

5 The wider community

The community aspect of RISC-V underlies many of the achievements and much of the work that is going on. The RISC-V International HPC Special Interest Group (SIG) meets monthly and comprises around 250 subscribers with over 30 active members. A major focus of the SIG over the past couple of years has been to raise awareness of RISC-V in the HPC community. This has been driven since 2023 by annual workshops at three major annual HPC conferences, Supercomputing (SC) which is based in the USA and the largest global HPC conference, ISC which is Europe's largest HPC conference, and HPC Asia which is a significant conference in the Asia region. All of these workshops have been well attended and involved research papers from the community around the use of RISC-V for HPC, invited talks and vendor showcases. The purpose of the invited talks and vendor showcases has been to demonstrate that there are real-world activities, such as RISE, companies and hardware vendors behind RISC-V. This is important to highlight because it demonstrates that whilst there is a significant research component involved in RISC-V, there are also a large number of commercial interests and serious products.

When it comes to government funding for RISC-V development and research the European Commission has invested heavily in RISC-V, not just through the umbrella of the European Processor Initiative (EPI), but other activities too. These including DARE, which aims to build a supercomputing compute stack, featuring high-performance and energy-efficient RISC-V-based processors and accelerators designed and developed in Europe, along with

TRISTAN whose overarching aim is to expand, mature and industrialize the European RISC-V ecosystem. Elsewhere, NASA has selected RISC-V to power their next generation High-Performance Spaceflight computer.

Whilst we have focussed on HPC specific vendors in this paper whose products are public, there are many other vendors including Rivos, Axelera AI, Calligo Technologies, Kneron and Openchip who are developing RISC-V based products and worth keeping an eye on into the future.

6 Conclusions and a call to action

In this survey paper we have explored the current state of the art of RISC-V in High Performance Computing (HPC), identifying reasons why RISC-V should be of interest to the HPC community, the existing state of hardware and software along with areas that need to be addressed.

Now is the time for the HPC community to involve itself in RISC-V and help shape the standard and associated technologies to suit our needs. Based upon the work identified in this paper we highlight the following high priority areas for the HPC and RISC-V communities to work together on:

- To support scaling of RISC-V workloads it would be beneficial to engage with HPC vendors to mature high performance networking support for RISC-V, including Slingshot. Furthermore, the community should prioritise a port of Lustre to RISC-V.
- The lack of mature RISC-V profiling tooling is a significant weakness for HPC and we should encourage developers of these profiling tools to support RISC-V. Furthermore it is important that the HPC community influences the range of hardware events that are made available by vendors, ensuring that those which are crucial for understanding code performance are present.
- Whilst numerous HPC applications have already been ported to RISC-V, the community should prioritise codes such as VASP and further tune these applications and underlying libraries for the architecture.
- Working with compiler developers to ensure these better support long vectors and developing an MLIR RVV dialect, as well as ensuring that frontends such as Fortran are able to drive these effectively, will help ensure that the full potential of RVV is realised.
- There are several nascent RISC-V standardisation activities such as the matrix extensions and 128-bit address space support that members of the HPC community should look to engage with and ensure that they meet the needs of scientific computing.

In summary, as highlighted in this paper RISC-V is moving quickly with many activities going on across the community. When it comes to adoption the devil is in the detail, but at a time where supercomputing centres are being challenged to continue delivering high performance but in a more sustainable manner, as we are seeing the potential for customisation offered by RISC-V provides additional opportunities over and above incumbent technologies and the capability to help the HPC community address the challenges that we face.

References

[1] NAS Parallel Benchmarks. 2006. NAS Parallel Benchmarks. *CG and IS* (2006).

[2] Nick Brown and Ryan Barton. 2024. Accelerating stencils on the Tenstorrent Grayskull RISC-V accelerator. In *SC24-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 1690–1700.

[3] Nick Brown and Christopher Day. 2025. Investigations of multi-socket high core count RISC-V for HPC workloads. *arXiv preprint arXiv:2502.10320* (2025).

[4] Nick Brown and Maurice Jamieson. 2025. Performance characterisation of the 64-core SG2042 RISC-V CPU for HPC. In *International Conference on High Performance Computing*. Springer, 354–367.

[5] Dave Ditzel. 2024. Why Esperanto picked RISC-V for HPC Computing. Talk. Retrieved March 21, 2008 from https://riscv.epcc.ed.ac.uk/community/workshops/sc24-workshop/

[6] Roger Ferrer. 2025. Vectorization for long vectors in LLVM on RISC-V. Talk. Retrieved March 21, 2008 from https://lists.riscv.org/g/sig-hpc/files/Meeting%20Minutes/Ferrer-20-03-25.pdf

[7] Geekbench. 2025. *SOPHGO SG2044 RISC-V vs Milk-V Pioneer.* Retrieved April 8, 2025 from https://browser.geekbench.com/v6/cpu/compare/8661173?baseline=8910646

[8] HiFive. 2021. *HF105 Datasheet.* Retrieved April 8, 2025 from https://www.sifive.com/document-file/hifive-unmatched-datasheet

[9] RISC-V International. 2024. *RISC-V Profiles.* Retrieved April 8, 2025 from https://github.com/riscv/riscv-profiles

[10] RISC-V International. 2025. *RISC-V Ecosystem Labs.* Retrieved April 8, 2025 from https://riscv.org/developers/labs/

[11] Joseph KL Lee, Maurice Jamieson, Nick Brown, and Ricardo Jesus. 2023. Test-driving RISC-V Vector hardware for HPC. In *International Conference on High Performance Computing*. Springer, 419–432.

[12] Ryan McMorrow. 2025. *Zhan Ketuan: Chinese crypto and AI mogul becomes US target.* Retrieved April 8, 2025 from https://www.ft.com/content/46c48898-c468-45e8-bbaf-b655939c941a

[13] Gianluca Mittone, Nicoló Tonci, Robert Birke, Iacopo Colonnelli, Doriana Medić, Andrea Bartolini, Roberto Esposito, Emanuele Parisi, Francesco Beneventi, Mirko Polato, et al. 2023. Experimenting with emerging RISC-V systems for decentralised machine learning. In *Proceedings of the 20th ACM international conference on computing frontiers*. 73–83.

[14] University of Versailles Saint Quentin en Yvelines. 2014. *NAS Parallel Benchmarks 3.0: Unofficial OpenMP C Version.* Retrieved April 8, 2025 from https://github.com/benchmark-subsetting/NPB3.0-omp-C

[15] David A Patterson and David R Ditzel. 1980. The case for the reduced instruction set computer. *ACM SIGARCH Computer Architecture News* 8, 6 (1980), 25–33.

[16] RISE project. 2024. *RISC-V Optimization Guide.* Retrieved April 8, 2025 from https://riscv-optimization-guide.riseproject.dev/

[17] Semico. 2019. RISC-V market analysis. Report. Retrieved March 21, 2008 from https://semico.com/sites/default/files/TOC_CC315-19.pdf

[18] Frans Sijstermans. 2025. *How NVIDIA Shipped One Billion RISC-V Cores In 2024.* Retrieved April 8, 2025 from https://riscv.org/blog/2025/02/how-nvidia-shipped-one-billion-risc-v-cores-in-2024/

[19] Tenstorrent. 2025. *TT-Ascalon.* Retrieved April 8, 2025 from https://tenstorrent.com/ip/risc-v-cpu

[20] Wang Zihan. 2024. *SG2042 Empowering RISC-V in High-Performance Computing.* Retrieved April 8, 2025 from https://github.com/RISCVtestbed/riscvtestbed.github.io/blob/main/assets/files/hpcasia24/hpc_asia_wang.pdf