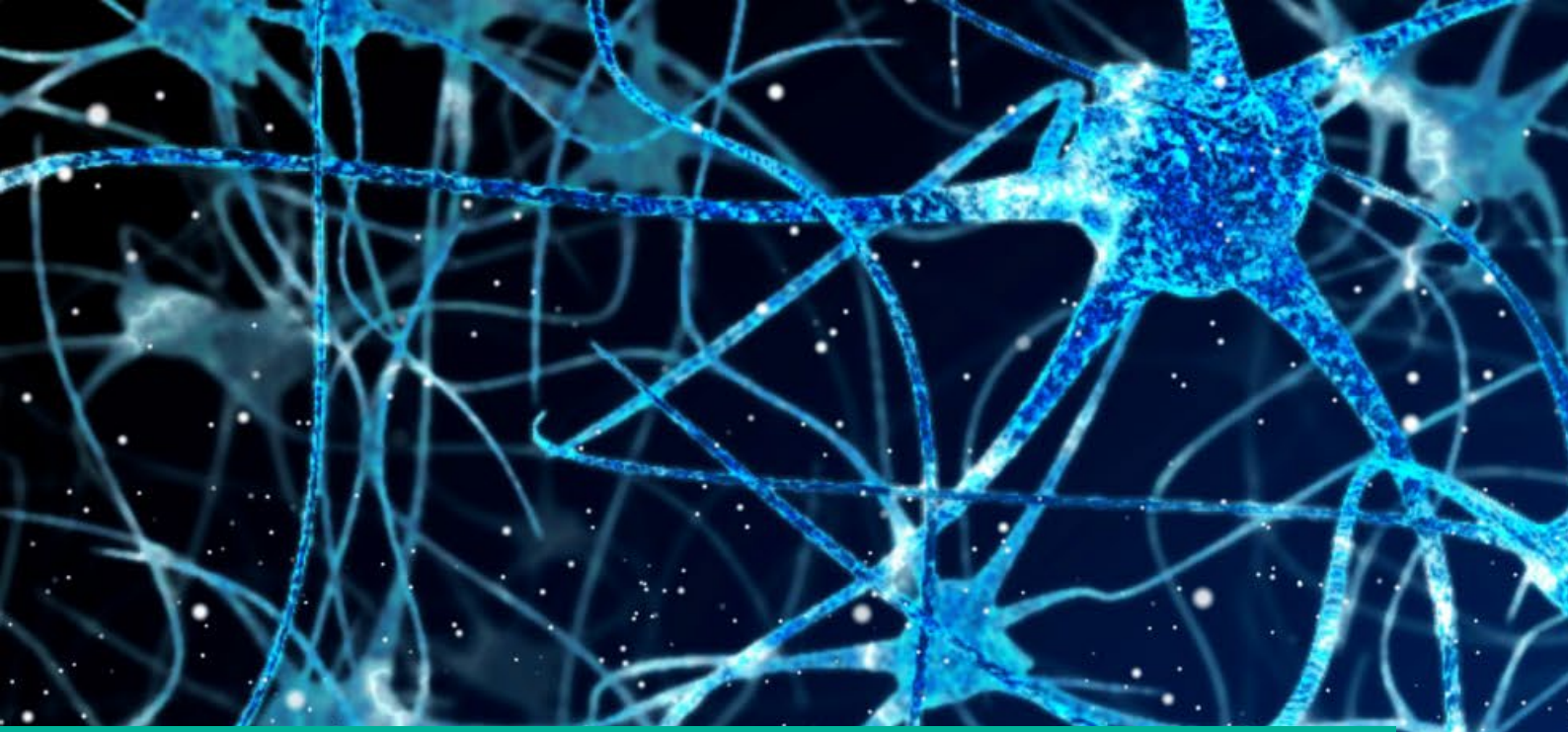


Unconventional HPC Architectures

As CMOS scaling is facing increasing challenges to deliver performance gains by conventional means, a range of new unconventional architectures emerge

White Paper

Lead authors: Tobias Becker (Maxeler),
Robert Haas (IBM),
Johannes Schemmel (Heidelberg University),
Steve Furber (University of Manchester),
Sagar Dolas (SURF)

25/04/2022

etp4hpc.eu

@etp4hpc

ETP 4 HPC

EUROPEAN TECHNOLOGY PLATFORM FOR HIGH PERFORMANCE COMPUTING

Introduction

Moore's Law, which stated that "*the complexity for minimum component costs has increased at a rate of roughly a factor of two per year* ", is slowing down due to the enormous cost of developing new process generations along with feature sizes approximating silicon interatomic distances. With the end of Dennard scaling (i.e; the rather constant power density across technology nodes, and the increase of operating frequency with each technology node) more than 15 years ago, using more transistors to build increasingly parallel architectures was a key focus. Now, other ways need to be found to deliver further advances in performance. This can be achieved by a combination of innovations at all levels: technology (3D stacking, using physics carry out computations, etc.), architecture (e.g., specialization, computing in/near memory, dataflow), software, algorithms and new ways to represent information (e.g., neuromorphic – coding information "in time" with "spikes", quantum, mixed precision). A closely related challenge is the energy required to move data, which can be orders of magnitude higher than the energy of the computation itself.

These challenges give rise to new unconventional HPC architectures, which, through some form of specialisation, achieve more efficient and faster computations. This paper covers a range of new unconventional HPC architectures which are currently emerging. While these architectures and their underlying requirements are naturally diverse, AI emerges as a technology that drives the development of both novel architectures and computational techniques due to its dissemination and computing requirements. The ever-increasing demand of AI applications requires more efficient computation of data centric algorithms. Furthermore, minimising data movement and improving data locality plays an important role in achieving high performance while limiting power dissipation and this is reflected in both architectures and programming models. We also cover models of computation that differ from those used in conventional CPU architectures, or models that are purely focussed on achieving performance through parallelisation. Finally, we address the challenges of developing, porting and maintaining applications for these new architectures.

Table of Contents

Key insights

- While a lot of efforts in the last decade have focussed on delivering more performance through parallelism, unconventional HPC architectures now seek to improve both performance and efficiency through specialisation, e.g., adapting the compute architecture to an underlying model of computation, leading to challenges of handling heterogeneity on top of parallelism;
- A key aspect of efficiency improvements is a focus on reduction of data movements and improving the locality of data, e.g., through dataflow or in/near-memory computing techniques;
- AI is an important driver for both novel architectures and computational techniques;
- There is an emergence of novel non-CMOS technologies. Novel devices will be an important enabler for the success of new in-memory analog computing concepts;
- The trend to unconventional HPC architectures is accompanied by novel computational models and techniques, e.g., using lower precision computations or accommodating for unreliable computations;
- CMOS-based analog computing can significantly reduce silicon cost for applications like AI inference.

Key recommendations

- In order to effectively leverage unconventional HPC architectures, novel tools and methods are needed for dealing with code porting, optimisation, and maintainability. It also requires a community effort and a focus on training and education;
- Novel co-design approaches are needed for bridging the gap between classical HPC applications, AI chips and novel AI-inspired formulations of HPC algorithms;
- Leveraging variable precision and custom numerics can be very beneficial in unconventional architectures but requires novel tools and approaches to code development and optimisation;
- Further research is needed on novel devices and their integration in the mainstream CMOS process technologies and infrastructure for neural and AI accelerators, e.g., resistive (memristors etc.), magnetic and phase change devices;
- Further research should be pursued into accelerators for event-based neuromorphic computing as these will be required for future neuroscience, such as full brain emulation, and for energy-efficient AI;
- Open standard approaches should be promoted that support future HPC architectures, including standardized fabrics, to enable multi-vendor composable systems from heterogeneous resources.

Novel HPC Architectures

The concepts of standardisation and specialisation have long been recognised as competing forces driving the development of electronics and computers. Standardisation has the benefit of making the technology easier to use and deploy, while specialisation can offer greater efficiency. The concept of a general-purpose processor in itself is arguably highly generic and standardised, and in the last decade we observed a dominance of x86 architectures in desktop computers and data centres, and ARM architectures in mobile devices.

However, in recent years one could observe a trend towards more diverse and specialised architectures. Many desktop CPUs now include GPUs and mobile CPUs move towards completely integrated system-on-chip architectures. They offer different cores for light and more intense workloads as is the case in the Arm big.LITTLE architecture. Other examples include neural accelerators, such as the ones used in Apple's mobile and desktop processors. Embedded systems include specialized accelerators as well, for data manipulation (DSPs, FFT, encoders/decoders, encryption, etc). While these examples are specific to desktop and mobile, we also observe a focus on new architectures in HPC and data centres.

For example, GPUs are now frequently used for a range of non-graphics tasks (General-Purpose GPU or GPGPU), and they are the most common form of accelerator used in HPC. They are commonly used to accelerate compute intensive workloads which exhibit regular parallelism and require floating-point computations and high throughput. In the following we describe some of the most prominent examples of emerging unconventional HPC architectures.

FPGA-based implementation of unconventional HPC architectures

One very promising approach for accelerating a wide range of compute-intensive applications is to use reconfigurable hardware such as field-programmable gate arrays (FPGAs). FPGAs operate at much lower clock frequencies than most CPUs but they allow to create accelerators with massive amounts of parallelism. The accelerator is not programmed using serialized instructions from a fixed and limited instruction-set-architecture (ISA); instead, it is created by spatially mapping the components of the algorithm to a large number of reconfigurable logic, memory and interconnect resources inside the FPGA. Therefore, the generated hardware is highly customised to the unique nature of the algorithm that needs to be accelerated. Often, dataflow models are used to program FPGA accelerators, which avoid the costly movement of data in and out of the multi-tier memory hierarchy of a CPU. In a dataflow model, computations are executed based on data availability. For suitable algorithms, this results in higher performance than multi-core CPUs or GPUs but also much higher energy efficiency.

As a result, FPGAs have started to become competitors to many-core accelerators, such as GPUs and vector processors, in the HPC ecosystem. For more than a decade, there has been extensive research in using FPGAs for application acceleration, and in particular in applications with irregular memory access patterns, and applications that may not need full double-precision floating point formats and instead benefit from reduced precision without loss of accuracy, where significant improvements over GPUs can be realised. For example, AI inference with low precision (even down to 1 bit) can be carried out very efficiently on FPGAs. However, a major obstacle to the widespread commercial adoption of FPGAs in HPC is the higher programming effort and the limited maturity of the development tools, which discourage application developers from utilizing them. Historically, FPGAs are used as reprogrammable hardware blocks for special purpose applications and logic emulation, and more recently, as standalone SoCs (e.g. in mobile communication), so the tools and programming environments are usually optimized for building a single monolithic application for the FPGA by following an electrical engineering approach. These programming tools are very difficult to use for software developers and the challenges of using these tools are compounded by the fact that a developed solution is typically limited to a particular device type from one FPGA vendor and porting to another device requires special skills and reengineering efforts.

The lack of software-oriented development tools and the limitations in application portability have been a major challenge for the wider uptake of FPGAs in the HPC field. Another factor has been the lack of commercial, off-the-shelf FPGA accelerator cards with surrounding software infrastructure and management tools which is expected for an HPC environment and the higher price of FPGA cards compared to GPUs. Finally, FPGAs are typically used as accelerators on the PCIe bus, where it is inefficient to accelerate fine-grain tasks. Acceleration of larger tasks or batches can be highly efficient but requires a streaming model of computation. All of these challenges have often relegated FPGAs to more niche applications in HPC.

More recently, we saw the introduction of several commercial FPGA solutions oriented for HPC. In 2017, Amazon Web Services (AWS) introduced EC2 F1 FPGA instances, which made FPGAs available to a wide community without the need to purchase special purpose hardware. Xilinx launched its Alveo accelerator cards in 2018, the first datacentre-oriented FPGA solution backed by a major vendor. Several hyperscalers, such as Alibaba, Baidu, Microsoft, Huawei, Nimbix and Tencent also started recently to expose FPGAs to application developers in their datacenter infrastructure or to provide accelerated application-as-a-service, relieving users from the complex FPGA development processes. FPGA programming tools have improved and diversified with various concepts to make them more appealing to traditional software developers. Both Intel and Xilinx have backed OpenCL based programming models which provide developers a commonly used programming model and enables some degree of code portability between different accelerator technologies.

However, the overall suitability of the OpenCL-based programming model for FPGAs remains contested and highly optimised code will no-longer be portable. Other tools have focused on High-Level Synthesis (HLS) where developers can build FPGA applications from commonly used software languages such as C and C++. While this is a more appealing model for software developers, it is not suitable for simply importing existing code and it requires careful code optimisation to create a high-performing FPGA application. Carrying out such optimisations also requires a hardware-oriented mindset and consideration of chip resources. Domain Specific Languages or languages with a specific programming paradigm can help using FPGA in this context. For example, a dataflow oriented model to programming FPGAs can be used, such as introduced in Maxeler's MaxCompiler [1] , leading to a dataflow model that is conceptually closer to an optimal FPGA accelerator architecture and lets developers reason about important design aspects such as bandwidth and parallelism while delegating lower-level optimisations to the compiler. The underlying FPGA technology has also advanced in recent years to include ultra-density on-chip memory, enable dramatic improvements in frequency, adopt 2.5D/3D stacking technology to integrate High Bandwidth Memory (HBM) and to integrate AI-oriented processing units in a network-on-a-chip architecture (e.g. in Xilinx Versal ACAP).

Another approach to optimizing FPGA architectures for HPC applications is the move towards more coarse-grain reconfigurable architectures (CGRAs) [2]. While CGRAs have not been commercially successful within the original concept of changing the device architecture to more coarse grain programmable units such as byte-wide interconnect and programmable ALUs and memory tiles, one could argue that modern FPGAs include various coarser grain blocks in addition to the conventional fine-grain fabric. One example is the introduction of a mesh-like CGRA structure of a programmable processing array in Xilinx Versal ACAP devices. While the elements themselves are fairly general-purpose, they are mostly promoted by Xilinx as targeting deep learning and telecommunication applications. However, an adequate software stack is needed to leverage the algorithm-hardware co-design of such a heterogenous fabric with fundamentally different programming models. One could also interpret Groq's AI-oriented Tensor Streaming Processor (TSP), which uses a static dataflow model, as an evolution of the FPGA concept with much coarser grain vector and matrix processing units and wider interconnect (more in section AI chips).

While significant advances towards using FPGAs in HPC have been made, the programming challenge of using productive programming models that are high-level, maintain efficiency of the developed application and support code portability remains one of the central aspects. On the system level, further work is required to better manage FPGAs through an Operating System, execute an application in-parallel on multiple FPGAs, run multiple applications on the same FPGA, and provide support for resource management in a multi-user environment and virtualisation support.

While significant advances towards using FPGAs in HPC have been made, the programming challenge of using productive programming models that are high-level, maintain efficiency of the developed application and support code portability remains one of the central aspects

Neuromorphic computing architectures

Neuromorphic or brain-inspired computing describes a field of unconventional computing research that takes certain aspects of what is known about the brain (where knowledge is far from complete) and applies it to novel computing models using various combinations of novel device technologies, analogue circuit approaches, and digital technologies. A common feature of all these approaches is the use of some form of event-based processing, inspired by the use of spikes, which are purely events, as a primary means of representing and communicating information in the brain. The information is essentially coded in time (frequency, coincidence, timestamping) and not in space, like bits. Some

neuromorphic research also considers neural models without event-based processing. Neuromorphic systems can be seen as having one or several of the following characteristics:

- Using information encoded in time;
- Using a sparse representation of information (i.e. variation/derivative of signals, event-based);
- Using physical phenomenon to carry computation (e.g., Ohm's law for product, Kirchhoff's law or accumulation of charges for adding, optics for non-linear functions, etc...);
- Using new materials for storing or simultaneously storing and computing (see just above), like the ones used in non-volatile memories (MRAM, RRAM, Spintronic, …).

While a strict definition of neuromorphic computing is still debated [3], ranging from "very strict high-fidelity mimicking of neuroscience principles" to "very vague high-level loosely brain-inspired principles", there is a "wide consensus that neuromorphic computing should at least encompass some time-, event-, or data-driven computation. In this sense, systems like spiking neural networks (SNN), sometimes referred to as the third generation of neural networks, are strongly representative."

Neuromorphic systems will operate as accelerators for specific classes of applications alongside a general-purpose host system. The host system will be responsible for configuring the neuromorphic subsystem - setting up the network topology and parameters, and for converting the data representations and handling data I/O streams. The neuromorphic system will handle the event-based computation itself.

Current developments in neuromorphic computing include architectures based upon analogue circuits, in sub-threshold (as per the original work by Carver Mead) or above threshold (e.g., the Heidelberg BrainScaleS) modes of operation, parameterised digital engines (e.g., IBM TrueNorth, Intel Loihi) and many-core programmable systems (e.g., the Manchester SpiNNaker). There are also developments in novel device technologies, such as memristors, for neuromorphic systems which, when they are mature, will transform the density and energy efficiency of those systems. It should be noted these neuromorphic accelerators chips differ from the new emerging class of AI accelerator chips that carry out neural network computation on architectures that are optimised for this class of applications (efficient sparse matrix operations, tensor operations, convolutions, etc.). These architectures are described in section (AI chips).

In many respects neuromorphic accelerators will play a similar role to accelerators for artificial neural networks (ANNs) but offer improved energy-efficiency due to their sparse event-based model of computation. Compared with conventional ANNs, spiking neural networks (SNNs) have greater capacity to process spatio-temporal data streams, though algorithms for such processing are less developed than those for conventional classification.

Neuromorphic integration with HPC

The next stage of efforts will be channelised at integrating neuromorphic architectures into the HPC ecosystem or "computing continuum" with use cases e.g. in health, agriculture, digital twins & sustainable energy. With the emergence of new architectures, scientific computing workflows are also going through radical transformations. In these transformations Neuromorphic architectures could be the main driver for energy efficient AI/ML processing at edge and at data centers. The importance and need for specialised computational architectures to cater to emerging hybrid workloads and achieve optimal energy efficiency have been highlighted in [4].

The following scenarios are envisioned for future of neuromorphic architectures:

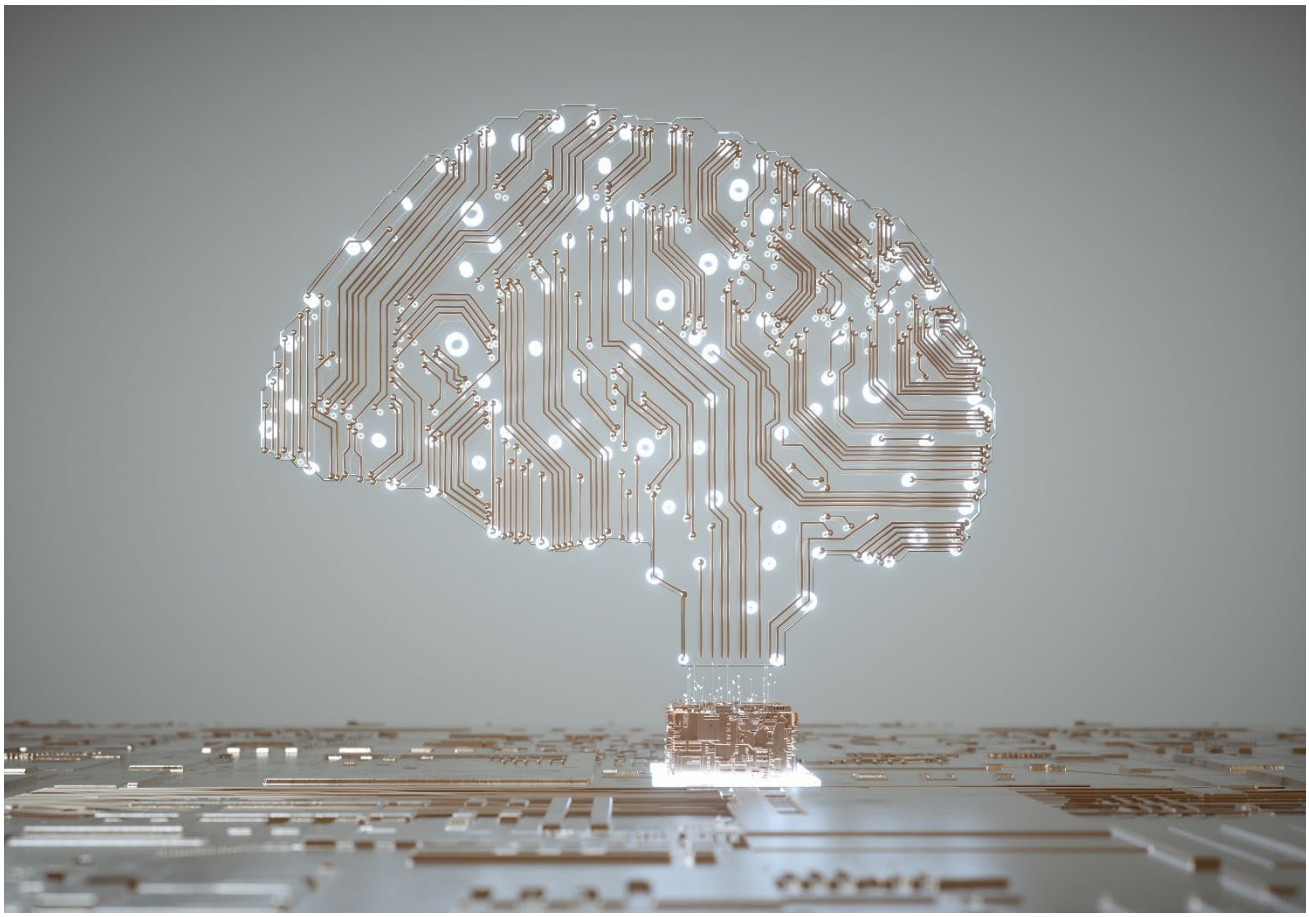
1. Intelligent edge co-processors for distributed cross-platform edge-cloud-HPC workflows
    a. AI inference & training at edge. The data movement is minimised.
2. Datacenter co-processors / accelerators for machine learning training & inference
    a. Energy efficient AI / ML training & inference at scale
        i. Inference for HPC-AI hybrid workloads
        ii. Training for AI algorithm (Spiking neural network, backpropagation)

Programmability is still a challenge for these chips, for which the tuning to the application is not done by programming, but by a "training" phase, done on device or off device (on a GPU for example). In terms of computability, there is an equivalent between space and time coding and there are tools that allow to convert Neural Networks trained "with bytes"

to “spiking” representation. “Classical” training methods such as back propagation can also be performed with time encoded information (“Spikes”). University level training programs and courses must take the leading role to introduce the concept of neuromorphic computing. Also at the same time HPC centers have to facilitate workshops / training sessions to map current algorithms onto neuromorphic devices.

To facilitate development of neuromorphic applications, various frameworks have been introduced. For instance, Intel provides a software framework called “Lava” [5]. Lava is a modular, extensible, open-source software framework for developing neuro-inspired applications and mapping them to neuromorphic hardware. Lava is platform-agnostic so that applications can be prototyped on conventional CPUs/GPUs and deployed to heterogeneous system architectures spanning both conventional processors as well as a range of neuromorphic chips such as Intel's Loihi [6].

There are several other software frameworks for neuromorphic applications, for example NEST [7], Nengo [8], EBRAINS [9], N2D2 [10], or others like NEURON, pyGeNN and the meta-framework pyNN.



Analog computing for AI and neuromorphic applications

In the quest for reduced power consumption of AI workloads, a renaissance of alternative computing technologies can be observed. Computing based directly on the analog quantities of electronic circuits, like electric charges and currents, could in principle deliver the same AI performance at a lower energy and price point compared to the usage of binary numbers, since they require far less transistors to perform the same operation [4] [11].

In addition, digital implementations depend on recent process nodes to achieve state-of-the-art power efficiency. Thereby, they incur high NRE costs. Analog realizations can be as efficient in much older technologies, thus allowing for a significant cost reduction in certain applications.

The most common analog architectures accelerate inference in deep convolutional neural networks. They implement the matrix-vector multiplication (MVM), which is the most costly operation in a deep neural network, by a predominantly

analog circuit. By utilizing a processing-in-memory structure, the number of data movements is also reduced. They operate by directly multiplying their memory content with the incoming activations, usually by pulse-width modulation. The results of these multiplications are summed-up as currents or charges on shared output lines.

Analog-to-digital converters are used at the output of the analog processing block to restore the partial sums to the digital domain for further processing and the subsequent transfer to the next layers. Analog computing can be combined with novel memory technologies based on memristive, magnetic or phase change materials to store the weight matrix in a dense, power-efficient and persistent way [12]. A first demonstration has already been made of a fully-integrated analog in-memory computing core with phase-change memory on 14nm CMOS [13]. There are also novel implementations of floating-gate and ferro-electric transistors as memory cells [14].

Recently, a second analog computing model has been demonstrated successfully, using light instead of electrical signals, for instance as shown by LightOn or LightMatter. Photonic computing uses a substrate similar to microelectronics with integrated waveguides and optical modulators [15], [16]. Although at the current state of the technology only small networks can be implemented, their operational speed in the Gigahertz range makes them unique for applications requiring extremely low latencies.

AI chips

In the last few years, a range of new architectures have emerged that are targeted specifically at accelerating AI-related computations. Demand for AI computations is driven by many applications including search, advertisement placement, personalization and data analytics. The widespread introduction of services with natural language processing or image recognition have significantly increased the performance requirements for AI computations where conventional CPUs or GPUs were no longer cost effective. Further applications of AI include cyber security and fraud detection, self-driving vehicles, industrial, healthcare and HPC applications.

The focus of AI architectures is typically on inference; training is only carried out once and can be done with very high effort. Inference is run at a very large scale, often with strict performance requirements, therefore requiring special architectures that can carry out the computations quickly and efficiently. Nevertheless, the acceleration of training is also gaining attention recently. Broadly speaking, AI chips target tensor operations (in particular matrix multiply) in combination with mixed or lower precision operations. Both of these aspects are relevant in AI (in particular inference) applications and focusing on lower precision matrix multiplication makes these chips more efficient for AI applications than GPUs which are more vector oriented.

In 2016, Google introduced the Tensor Processing Unit (TPU). TPUs are focused on matrix multiplication operations and support lower precision operations. The first version TPU only supported Int8/Int 16 and were used for inference, but since the second version, they are also optimized for learning and support a new 16 bit floating point format. TPUs are very efficient for computing models with very large batch sizes and they are programmed through Google's TensorFlow framework. TPUs are mostly used to support Google's internal services with limited access for other vendors to the TPU hardware. Groq recently launched an AI chip called the Tensor Streaming Processor (TSP), a massively parallel architecture of matrix and vector units interleaved with memory units that execute in a static dataflow model [17]. This provides abundant data parallelism and the static execution model eliminates all caches, branch predictors and arbiters. As a result, the chip offers very efficient processing with deterministic behaviour. The TSP is also very efficient at small batch sizes and computes with low latency which is important for real time applications. Furthermore, the TSP offers fast chip-to-chip interconnect which enables the efficient scaling of computational models. Graphcore is offering the Intelligence Processing Unit (IPU), a massively parallel architecture with many small processor cores and local memory, which is efficient at executing fine-grain threads with irregular behaviour [18]. The IPU is conceptually closer to a conventional multi-core processor but provides more parallelism than CPUs and is more efficient at irregular workloads than GPUs. Cerebras has launched the Wafer Scale Engine (WSE), a single, wafer-scale processor that includes compute, memory and interconnect [19]. Processor cores are optimised for sparse linear algebra. The entire architecture is optimised on system level rather than adding PCIe form factor cards to conventional servers. They released the CS-2 system featuring a very large number of cores, on-chip SRAM, and bandwidth.

To address the pervasive need for efficient AI computations in many applications, several conventional architectures have also been extended to enable efficient inference. Intel Xeon Ice Lake processors contain an instruction set extension

called DL Boost that supports efficient multiply-accumulate operations and lower precision 16-bit floating point operations. IBM Power10 includes the new matrix-multiply assist instructions executed by four engines per core to accelerate inferencing operations. Nvidia Ampere GPUs have been extended with tensor cores that also support 16-bit floating point operations. Xilinx Versal FPGAs contain AI engines that provide simple vector processors in a VLIW architecture.

AI processors are typically programmed through established machine learning frameworks such as PyTorch or TensorFlow. Some AI processors also offer lower-level programming models which can be used to develop non-AI applications on these platforms. In a similar way to how the emergence of GPGPU architectures has led to non-graphics applications being considered for GPU acceleration, there is now interest to port applications from HPC and other fields onto AI chips, leveraging the efficient matrix-compute capabilities of these architectures. This typically requires the algorithms to be restructured such that they can be expressed in a machine learning framework, use the arithmetic precision of the accelerators or some other tensor-level API, and therefore presents similar challenges to other unconventional architectures in that developers need to restructure code and learn new development frameworks.

Processing in memory - PIM

Processing in memory (PIM) is a generic term that covers any form of integration of processing capabilities into the memory system, of which there are many approaches; e.g. computation in memory arrays, peripheral circuits, and logic layers and dies integrated into the memory devices. PIM has been explored by a large number of academic studies and it has already been demonstrated in various industrial prototypes and products, initially targeting near-memory computing, with DRAM and SRAM, and extending to non-volatile memory technologies of PCM, RRAM, and Flash. Analog in-memory computing can be seen as a form of PIM, where some computations are performed in the memory arrays themselves.

The interest in PIM has grown dramatically over the last years. In 2020, the EuroLab4HPC Vision [20]already included a survey of the approaches for near- and in-memory processing and it recommended funding in this direction. In the same year, ETP4HPC's Strategic Research Agenda [21] called near and in-memory architectures the "ultimate option" to improve energy efficiency, but correctly saw the approach as not mature enough for short-term adoption in production systems. A recent white paper from ETP4HPC [22]advises that wide acceptance of PIM in high-performance computing depends on our ability to create an ecosystem in which a number of PIM approaches can be designed and evaluated, leading to the selection of potential winners and adoption by system architects and end users. The study identifies the main PIM challenges and provides guidelines to prioritise the research effort and funding, encompassing development and evaluation platforms, system software to manage the hardware, APIs and programming models to program it, profiling and analysis tools to analyse it, and benchmarks, simulators and platforms to evaluate the overall benefits. Overall, it can be concluded that only coordinated innovations and co-design across the whole stack can make PIM a reality in production.

Only coordinated innovations and co-design across the whole stack can make PIM a reality in production.

Architectures, storage, and networking

HPC system architecture is evolving from traditional monolithic servers towards servers that can be dynamically composed from a precise set of resources to meet workload requirements. Such resources are envisioned to be disaggregated into pools of various types of CPUs, accelerators, (persistent) memory, storage, etc, accessible via high-performance fabric and orchestrated by an intelligent infrastructure. This will allow maximization of usage of such resources while powering down unused ones.

This evolution in HPC architecture is needed as a result of the more complex workflows that are increasingly implemented in HPC applications. As an example, they can include large knowledge graphs (memory capacity and bandwidth intensive), training of and inference on AI models (large datasets and compute intensity), and the execution of classic simulation codes (memory or compute bound) [23]. The technologies that will enable such an evolution include the introduction of high-performance coherent fabrics to flexibly wire such composable systems together. Promising steps are the introduction of standards such as CXL, with fast signaling protocols such as PCIe Gen 6 that will provide the latency and bandwidth needed for disaggregation of system resources. In terms of I/O technologies, the continued increase in speed is expected to come from the adoption of new modulation techniques (e.g. frequency domain) and the pairing of short low-latency electrical links together with co-packaged optics.

New challenges will arise to ensure that pools of resources such as persistent memory offer a similar level of reliability and availability as with traditional fabric attached storage. This will necessitate the exploration of new approaches to perform certain types of computing tasks close to memory, or in the fabric itself. Similarly, this will apply to storage to offload the fabric, exploiting techniques such as active storage or processing in storage.

From an ecosystem point-of-view, such composable systems based on a standardized fabric will enable the creation of multi-vendor systems, avoiding lock-in to proprietary fabrics, as well as allow independent updates of components without the need to synchronize all release dates. Furthermore, these more complex HPC workflows are expected to seamlessly take advantage of resources available on-premises as well as across various locations from external cloud providers, for instance to benefit from temporary spill-over capacity or access to highly-specialized and/or bleeding-edge accelerators that aren't available elsewhere. Again, orchestration by an intelligent infrastructure will need to extend to such scenarios.

Unconventional models of computation

Approximate computing

Approximate computing encompasses diverse approaches such as variable precision models, novel number representations and stochastic rounding. These techniques are enabled by the unique features of novel architectures (e.g., low precision and integer modes in AI chips and completely flexible arithmetic on FPGAs). The vast majority of current HPC applications use either single or double-precision floating point formats but the actual numerical requirements and properties of algorithms are often not explored in greater detail as conventional CPU architectures simply offer a binary choice between single- and double-precision formats. Double-precision arithmetic is often seen as "correct" even though this number representation is also an approximation in a strict sense and not immune from introducing numerical problems. New unconventional architectures offer a much wider choice of number formats such as low-precision floating point, integer or fixed-point numbers, and using them effectively requires a careful analysis of the numerical behaviour in the application development and porting process. It requires detailed knowledge of how much error is acceptable at the output of an algorithm or solver, and then tracing backwards through the computation what value ranges and precision are needed at the various stages in order to produce suitably accurate output numbers. There is typically no general and analytic solution available to this problem. Instead, the optimisation process often requires experimentation and detailed instrumentation of the code, and the derived solution can be limited to the context of a particular input dataset type or algorithm configuration. Figure x shows an example of numerical application profiling.

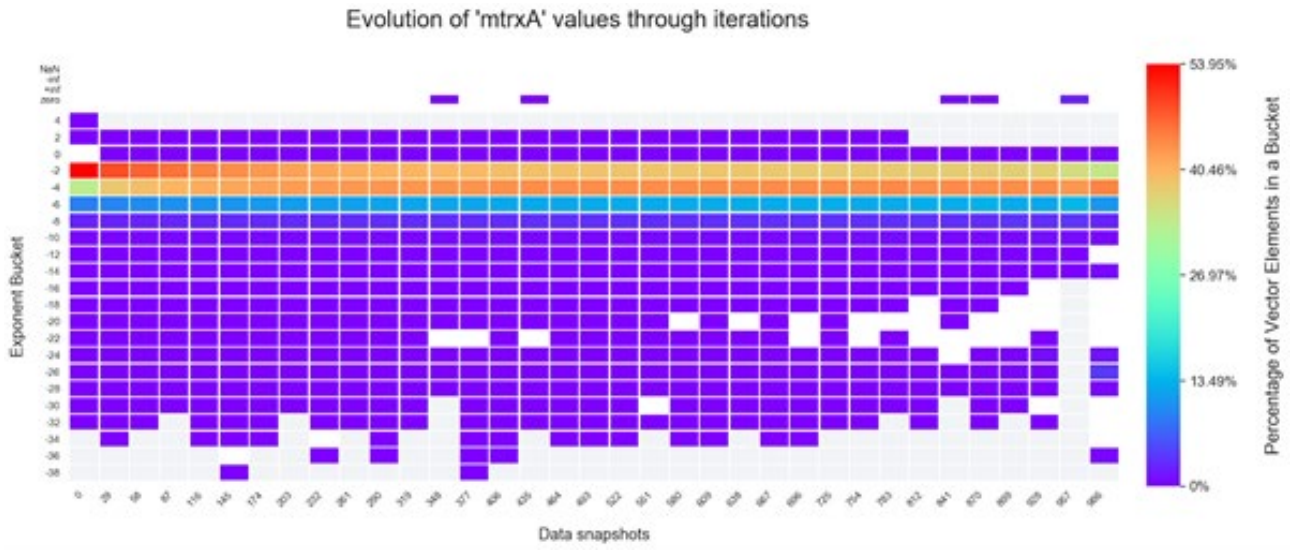


Figure 1 - Numerical analysis with Maxeler's value profiling tool

Stochastic rounding describes a rounding process where the result is randomly rounded up or down with a probability that is a function of the residue being rounded. It has been shown to yield more accurate results over long computational sequences than any of the conventional (deterministic) rounding algorithms such as round to nearest, and has the potential to be embedded at low cost into fixed- and floating-point arithmetic units. If the random process is generated by a high-quality pseudo-random number generator (PRNG) the computation can be repeatable by using the standard technique of controlling the PRNG seed. The improved accuracy can be traded against the use of reduced precision operands, leading to reduced energy and memory requirements.

Another aspect of approximate computing is to use approximation circuits instead of conventional arithmetic units, or to use arithmetic units that are somewhat faulty or operating at lower voltages thus producing a certain number of bit errors. Individual results may be wrong, but if the application is based on statistical models, then occasional incorrect results may be acceptable as long as statistical properties of the algorithm are maintained.

AI-inspired models for HPC

We also observe a convergence of AI and HPC technologies. AI is very compute intensive and requires sophisticated HPC infrastructure while HPC applications can also benefit from AI algorithms. Fundamentally, HPC applications attempt to model some form of real word process and in many cases (e.g. image or time series analysis) this model can be obtained by feeding training data into an AI algorithm. Practical examples of this are the identification of cancer from a medical imaging process or the discovery of events in a high-energy physics experiment. However, at the moment there is a strong experimental component in the application of AI algorithms to HPC algorithms. There is no common approach to pick the appropriate model and hyper parameters that can achieve fast convergence during training, and new mathematical frameworks will be needed to guide HPC developers in integrating AI into their applications. Another challenge lies in the black box behaviour of AI where the algorithm is known to work with a certain statistical level of accuracy but the behaviour of specific outputs (in particular bad outputs) cannot be understood in the context of the deployed AI model. This can be solved through new explainable AI techniques that allow the behaviour of an output to be traced back and understood through the model. Explainable AI is of particular importance for the greater acceptance of AI in the medical field. Finally, AI and HPC applications are typically built around different programming frameworks and data scientists and HPC developers often have very different backgrounds and skill sets. The convergence of AI and HPC will require efforts to bring together the different technologies and communities. At the same time, AI may also be able to address some of the development and technological challenges by applying AI to the coding process.

The convergence of AI and HPC will require efforts to bring together the different technologies and communities.

Dataflow computing

Conventional processors function by carrying out a sequence of instructions. This model is very flexible but also inherently sequential. Over the decade, many architectural innovations have been developed to boost performance of a sequential CPU such as superscalar execution, out-of-order execution, branch prediction, Single Instruction Multiple Data (SIMD) extensions, and finally multi-core CPUs. All of these make modern CPUs very complex where a lot of silicon real estate is needed to keep a small execution unit running at high speeds.

Dataflow architectures fundamentally differ in that they do not rely on control flow or program counters for execution; instead operations are carried out by making input arguments available to instructions. A whole range of different architectures have been developed under the umbrella term of dataflow and they can be typically classified as dynamic or static dataflow. Dynamic dataflow architectures result in non-deterministic behaviour and require large content addressable memories (CAMs). To date, dynamic dataflow machines have seen limited applications in practice although out-of-order execution can be seen as a very limited form of dataflow where sequential execution can be broken and rearranged depending on the availability of input data. Event-based processing as used in neuromorphic computing can also be seen as a special case of dataflow. Static dataflow is often applied in the context of a systolic array or processing network architecture. Here, data flows from memory over an array of operators where computations are performed when data arrives at the inputs. The outputs are sent forward to the next operator; hence computations are performed by data flowing through the system. This model of computation is deterministic and it eliminates the need for caches as data arrives at the inputs of an operator at a predetermined time. A significant advantage of a deterministic model of computation in static dataflow is that performance and system behaviour is completely predictable, simplifying the design and optimisation process [24]. Furthermore, it minimises data transfers to mostly small local movements and mostly avoids inefficient transfers to off-chip memory which is one the major bottlenecks in conventional CPU architectures. Maxeler Technologies has applied the model of static dataflow to programming FPGAs where a Java-like dataflow language called MaxJ is used to build deep pipelines that are then mapped to the FPGA. The dataflow model is also used in the Groq TSP where tensors flow over the execution units of the chip, resulting in deterministic and low latency operations.

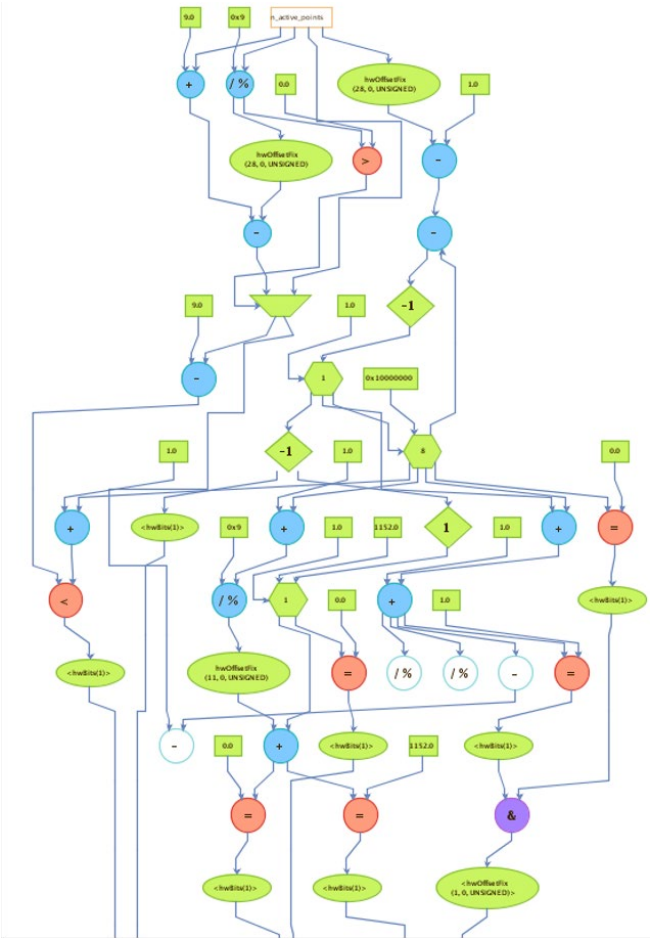


Figure 2 - Example of a static dataflow graph. Computations are carried out whenever input data is available and partial results flow through the graph

Programming models, porting and code maintenance

Unconventional architectures typically require specific programming models and tool chains. The introduction of new tool chains often results in training needs for developers who have to become proficient in using a new programming language or model, in particular if the model leverages a different style of execution which requires different thinking when developing applications. For example, using a multi-threading library is relatively easy for code that naturally presents various blocks that can operate independently, but tweaking complex code for efficient parallel execution can be very challenging and requires the developer to fundamentally re-examine the existing code. Similar challenges are faced and are even increasing when programming unconventional architectures. In particular, porting and maintaining large existing code bases is challenging. HPC applications are often developed over many years by a team of developers, resulting in large and complex code bases. A need to port this code to a new architecture is often seen as a significant obstacle by the developer team. Furthermore, the porting may not be just a one-time effort as highly specialised architectures have limited code portability to newer generation devices. The same code may still be usable but may require re-optimisation in order to fully leverage the performance of the new device, as opposed to legacy x86 code which typically does not need constant re-optimisation and can simply benefit from faster processors. This creates a significant challenge in terms of code porting and maintenance which is not sufficiently addressed at the moment.

Developing applications for systems with unconventional architecture can be far from trivial and significantly more complex than conventional HPC programming. This is mainly due to the high degree of specialisation where a set of very specific application and data properties have to be captured in efficient computational solutions. To prevent repeating the errors from the past, e.g., the initial lack of software and tools for the Cell processor, the failure of, for example, Larabee, Tabula and other products, we have to invest early into creating better software tools as well building vivid designer communities around the new, unconventional acceleration technologies. These designer communities can be organised by computing technology, by application domains or ideally by the right combination of the above two. Repositories with design examples, template architectures, application centric methodologies and active discussion boards will serve as the driving vehicle of this process. Considering the challenges around porting and developing full system applications for unconventional architectures, this is going to be crucial. Universities and research centres will play a leading role by involving their undergraduate and graduate students who will contribute at different levels. There will be plenty of opportunities for high-impact research and engineering contributions that should naturally attract many European universities to actively participate.

Moreover, these communities will also streamline the discussions between the domain experts, who are the final users of the developed systems, and the computer engineers and researchers working on specific challenges at all different system levels. If organized correctly, such vertical cross-disciplinary discussions will also provide valuable insights for fine tuning of the underlying hardware technologies, computational models and abstractions. The repositories with e.g., template designs, Q&A discussions and various documentation, should be open source.

A complementary approach could be to better structure the code into modules that have defined computing characteristics (that are exposed in a systematic way), and therefore only these parts should be optimized to benefit from the new accelerators that fit with the characteristics. From a system point of view, the work of orchestrating those different accelerators should be further explored and optimized, both at compile time and at run-time.

These challenges of code porting and development are examples that can also be addressed by the introduction of AI-based approaches, such as AI for code. Emerging benchmark datasets such as Project Codenet, and tools such as OpenAi Codex or Deepmind's AlphaCode are a first step in this direction. More generally, methodologies to develop software-hardware platforms must shift towards AI. A leading example is already showing the benefit of ML for the IC EDA tool chain [25]. Developing AI-based tool chains to address the problem of finding the best HPC architecture and associated code bases is clearly one of the most promising approaches for unconventional HPC systems.

Conclusions

With the end of Moore's Law and the end of "free" performance improvements in sight, the industry has shifted from following a parallelism-oriented approach of pursuing further performance gains to embracing specialised and unconventional architectures as a paradigm to improve both performance and energy efficiency. A specialisation trend has already been observed in mainstream CPU architectures from mobile and desktop to data center. FPGA accelerators are now emerging in the data centre and offer an extreme amount of specialisation in terms of compute architecture and numerical choices, but are burdened by non-standard programming models and code porting challenges. New AI chips, developed to address the processing needs of data-driven applications, are highly efficient at lower-precision, matrix-oriented computations and are now also being considered for HPC applications. Neuromorphic architectures, based on spiking models, can play a similar role in AI but could potentially be more energy efficient because of their inherent sparsity and potential simpler non-CMOS implementations. Analogue implementations could also help to drive down development cost and power consumption. Processing in memory architectures address the IO bottleneck problem of conventional von-Neuman architecture and enable simple operations close to the data. An overarching theme across all of these architectures is that they reduce data movements because these (in particular off-chip IO) are increasingly becoming a limiting factor from both a performance and energy perspective.

These new architectures are accompanied by new models of computation such as computing with custom-precision and different number formats which can deliver good-enough results at higher efficiency. Dataflow computing models benefit from minimal data movements as well as deterministic execution. AI algorithms can help to derive models for HPC applications from large amounts of data. All of these technologies provide unique benefits but they can also pose challenges because they are non-standard. In order to harness these technologies effectively we need support for better tools for software development and porting as well as increased efforts for community building, training and education.

References

[1] R. Dimond, M. J. Flynn, O. Mencer and O. Pell, "MAXware: Acceleration in HPC," in *2008 IEEE Hot Chips 20 Symposium (HCS)*, 2008.

[2] A. Podobas, K. Sano and S. Matsuoka, "A Survey on Coarse-Grained Reconfigurable Architectures From a Performance Perspective," *IEEE Access,* pp. 146719-146743, 2020.

[3] D. V. Christensen, R. Dittmann, B. Linares-Barranco, A. Sebastian, M. L. Gallo, A. Redaelli, S. Slesazeck, T. Mikolajick, S. Spiga, S. Menzel, I. Valov, G. Milano, C. Ricciardi, S.-J. Liang, F. Miao, M. Lanza, T. J. Quill, S. T. Keene, A. Salleo, J. Grollier, D. Markovic, A. Mizrahi, P. Yao, J. J. Yang, G. Indiveri, J. P. Strachan, S. Datta, E. Vianello, A. Valentian, J. Feldmann, X. Li, W. H. Pernice, H. Bhaskaran and S. Furber, "2022 roadmap on neuromorphic computing and engineering," *Journal of Neuromorphic Computing and Engineering,* 2022.

[4] S. Yu, H. Jiang, S. Huang, X. Peng and A. Lu, "Compute-in-Memory Chips for Deep Learning: Recent Trends and Prospects," *IEEE Circuits and Systems Magazine,* vol. 21, no. 3, p. 31–56, 2021.

[5] [Online]. Available: https://github.com/lava-nc/lava.

[6] [Online]. Available: https://www.hpcwire.com/2021/09/30/intel-unveils-loihi-2-its-second-generation-neuromorphic-chip/.

[7] [Online]. Available: https://www.nest-simulator.org/.

[8] [Online]. Available: https://www.nengo.ai/.

[9] [Online]. Available: https://ebrains.eu/.

[10] [Online]. Available: https://github.com/CEA-LIST/N2D2.

[11] A. Sebastian, M. Le-Gallo, R. Khaddam-Aljameh and E. Eleftheriou, "Memory devices and applications for in-memory computing," *Nature Nanotechnology,* vol. 15, p. 529–544, 2020.

[12] Y. Xi, B. Gao, J. Tang, A. Chen, M.-F. Chang, X. S. Hu, J. Van-Der-Spiegel, H. Qian and H. Wu, "In-memory Learning with Analog Resistive Switching Memory: A Review and Perspective," *Proceedings of the IEEE,* vol. 109, no. 1, pp. 14-42, January 2021.

[13] R. Khaddam-Aljameh, M. Stanisavljevic, J. F. Mas, G. Karunaratne, M. Braendli, F. Liu, A. Singh, S. M. Müller, U. Egger, A. Petropoulos, T. Antonakopoulos, K. Brew, S. Choi, I. Ok, F. L. Lie, N. Saulnier, V. Chan, I. Ahsan, V. Narayanan, S. R. Nandakumar, M. Le Gallo, P. A. Francese, A. Sebastian and E. Eleftheriou, "HERMES-Core—A 1.59-TOPS/mm2 PCM on 14-nm CMOS In-Memory Compute Core Using 300-ps/LSB Linearized CCO-Based ADCs," *IEEE Journal of Solid-State Circuits,* vol. 57, no. 4, pp. 1027-1038, April 2022.

[14] M. Halter, L. Bégon-Lours, V. Bragaglia, M. Sousa, B. J. Offrein, S. Abel, M. Luisier and J. Fompeyrine, "Back-End, CMOS-Compatible Ferroelectric Field-Effect Transistor for Synaptic Weights," *ACS Applied Materials & Interfaces,* vol. 12, no. 15, pp. 17725-17732, 2020.

[15] G. Wetzstein, A. Ozcan, S. Gigan, S. Fan, D. Englund, M. Soljačić, C. Denz, D. A. B. Miller and D. Psaltis, "Inference in artificial intelligence with deep optics and photonics," *Nature,* vol. 588, no. 7836, pp. 39-47, 2020.

[16] B. J. Shastri, A. N. Tait, T. F. d. Lima, W. H. P. Pernice, H. Bhaskaran, C. D. Wright and P. R. Prucnal, "Photonics for artificial intelligence and neuromorphic computing," *Nature Photonics,* vol. 15, no. 2, pp. 102-114, January 2021.

[17] D. Abts, J. Ross, J. Sparling, M. Wong-VanHaren, M. Baker, T. Hawkins, A. Bell, J. Thompson, T. Kahsai, G. Kimmell, J. Hwang, R. Leslie-Hurd, M. Bye, E. Creswick, M. Boyd, M. Venigalla, E. Laforge, J. Purdy, P. Kamath, D. Maheshwari, M. Beidler, G. Rosseel, O. Ahmad, G. Gagarin, R. Czekalski, A. Rane, S. Parmar, J. Werner, J. Sproch, A. Macias and B. Kurtz, "Think Fast: A Tensor Streaming Processor (TSP) for Accelerating Deep Learning Workloads," in *ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, Valencia, Spain, 2020 .

[18] Z. Jia, B. Tillman, M. Maggioni and D. P. Scarpazza, *Dissecting the Graphcore IPU Architecture via Microbenchmarking,* arXiv, 2019.

[19] G. Lauterbach, "The Path to Successful Wafer-Scale Integration: The Cerebras Story," *IEEE Micro,* vol. 41, no. 6, pp. 52-57, 2021.

[20] “Eurolab4HPC Long-Term Vision on High-Performance Computing (2nd edition),” January 2020. [Online]. Available: https://www.eurolab4hpc.eu/media/public/vision/vision_final.pdf.

[21] M. Malms, M. Ostasz, M. Gilliot, P. Bernier-Bruna, L. Cargemel, E. Suarez, H. Cornelius, M. Duranton, B. Koren, P. Rosse-Laurent, M. S. Pérez-Hernández, M. Marazakis, G. Lonsdale, P. Carpenter, G. Antoniu, S. Narasimhamurthy, A. Brinkman, D. Pleiter, A. Tate, J. Kueger, J. Krueger, H.-C. Hoppe, E. Laure and A. Wierse, “ETP4HPC's Strategic Research Agenda for High-Performance Computing in Europe 4,” Zenodo, 2020.

[22] P. Radojković, P. Carpenter, P. Esmaili-Dokht, R. Cimadomo, H.-P. Charles, A. Sebastian and P. Amato, “Processing in Memory: The Tipping Point,” Zenodo, 2021.

[23] C. Hagleitner, D. Diamantopoulos, B. Ringlein, C. Evangelinos, C. Johns, R. N. Chang, B. D'Amora, J. A. Kahle, J. Sexton, M. Johnston, E. Pyzer-Knapp and C. Ward, “Heterogeneous Computing Systems for Complex Scientific Discovery Workflows,” in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021.

[24] N. Voss, B. Kwaadgras, O. Mencer, W. Luk and G. Gaydadjiev, “On Predictable Reconfigurable System Design,” *ACM Transactions on Architecture and Code Optimization,* vol. 18, no. 2, pp. 1-28, June 2021.

[25] A. Mirhoseini, A. Goldie, M. Yazgan, J. W. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, A. Nazi, J. Pak, A. Tong, K. Srinivasa, W. Hang, E. Tuncer, Q. V. Le, J. Laudon, R. Ho, R. Carpenter and J. Dean, “A graph placement methodology for fast chip design,” *Nature,* vol. 594, pp. 207-212, 2021.

[26] [Online]. Available: https://www.taylorfrancis.com/chapters/edit/10.1201/9781351036863-9/modular-supercomputing-architecture-idea-production-estela-suarez-norbert-eicker-thomas-lippert.

Lead authors:

Tobias Becker is the VP of research and development at Maxeler Technologies.

Robert Haas is the department head for Cloud and AI Systems Research at the IBM Zurich Research Laboratory, Switzerland.

Johannes Schemmel is the head of the Electronic Visions research group at Heidelberg University.

Steve Furber CBE FRS FREng is ICL Professor of Computer Engineering at the University of Manchester, UK and he leads the SpiNNaker neuromorphic computing project.

Sagar Dolas is the program manager within SURF, Netherlands with focus on trends & innovation in computing technology.

Additional authors:

Petar Radojkovic is leader of the Memory Systems team at the Barcelona Supercomputing Center (BSC).

Yannis Papaefstathiou is the CEO of EXAPSYS Plc and a professor at ECE School at Aristotle University of Thessaloniki.

Georgi Gaydadjiev is a full professor in Innovative Computer Architectures at the University of Groningen and a honorary visiting professor at Imperial College.

Ken O'Brien is a senior scientist at AMD working on HPC applications and communication technologies on reconfigurable hardware.

Cite as: T. Becker et al., Unconventional HPC Architectures, ETP4HPC White Paper, 2022, doi 10.5281/zenodo.6470840

DOI: 10.5281/zenodo.6470840