eProcessor - European, Extendable, Energy-Efficient, Extreme-Scale, Extensible, Processor Ecosystem

Invited Paper

Lluc Alvarez[*]
Abraham Ruiz
Arnau Bigas-Soldevilla
Pavel Kuroedov
Alberto Gonzalez
Hamsika Mahale
Noe Bustamante
Albert Aguilera
Francesco Minervini
Javier Salamero
Oscar Palomar
Barcelona Supercomputing Center
Barcelona, Spain

Vassilis Papaefstathiou[†]
Antonis Psathakis
Nikolaos Dimou
Michalis Giaourtas
Iasonas Mastorakis
Georgios Ieronymakis
Georgios-Michail Matzouranis
Vasilis Flouris
Nick Kossifidis
Manolis Marazakis
Institute of Computer Science, FORTH
Heraklion, Crete, Greece

Bhavishya Goel
Madhavan Manivannan
Ahsen Ejaz
Panagiotis Strikos
Mateo Vázquez
Ioannis Sourdis
Pedro Trancoso
Per Stenström
Chalmers University of Technology
Gothenburg, Sweden

Jens Hagemeyer
Lennart Tigges
Nils Kucza
Bielefeld University
Bielefeld, Germany

Jean-Marc Philippe
Thales Research & Technology
Palaiseau, France

Ioannis Papaefstathiou
Exascale Performance Systems
EXAPSYS Plc
Thessaloniki, Greece

ABSTRACT

The eProcessor project aims at creating a RISC-V full stack ecosystem. The eProcessor architecture combines a high-performance out-of-order core with energy-efficient accelerators for vector processing and artificial intelligence with reduced-precision functional units. The design of this architecture follows a hardware/software co-design approach with relevant application use cases from the high-performance computing, bioinformatics and artificial intelligence domains. Two eProcessor prototypes will be developed based on two fabricated eProcessor ASICs integrated into a computer-on-module.

CCS CONCEPTS

• **Computer systems organization → Multicore architectures**; **Single instruction, multiple data**; **Systolic arrays**.

KEYWORDS

Multicore architecture, RISC-V, European research project

[*]Corresponding author: Lluc Alvarez, e-mail: lluc.alvarez@bsc.es
[†]Also with Computer Science Department, University of Crete, Greece.

ACM Reference Format:
Alvarez et al. 2023. eProcessor - European, Extendable, Energy-Efficient, Extreme-Scale, Extensible, Processor Ecosystem: Invited Paper. In *20th ACM International Conference on Computing Frontiers (CF '23), May 9–11, 2023, Bologna, Italy*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3587135.3592178

1 INTRODUCTION

eProcessor is an ambitious European project that aims at creating a full stack high performance processor ecosystem, including both software and hardware. The eProcessor technology is based on the RISC-V open source Instruction Set Architecture (ISA) and features high performance computing and deep learning accelerators coupled to a high performance, low energy out-of-order processor.

The project follows a hardware/software co-design approach for improved application performance and system energy efficiency. eProcessor co-designs solutions to provide high performance, low-power, and fault tolerance for both traditional and emerging High-Performance Computing (HPC) applications. Uniquely, the project specializes all components of the system in the context of a broad application domain: a combination of energy efficient accelerators, adaptive on-chip memory structures, a flexible and high performance energy-efficient CPU, and the corresponding software stack.

The project contemplates a set of application use cases of interest to be optimized for the eProcessor ecosystem. These consist of a diverse set of applications in the domains of HPC, artificial intelligence, deep learning, machine learning and bioinformatics. The application use cases are used in the project to drive the design of

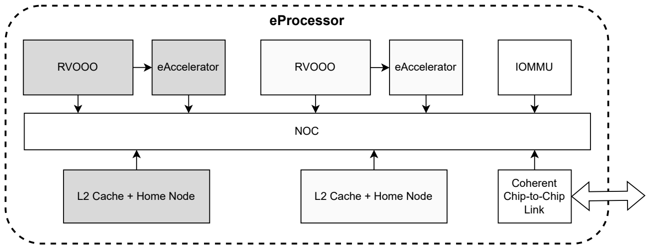

Figure 1: eProcessor architecture overview. Modules shaded in gray are included only in the second tape-out.

the overall system and, at the same time, the project extends these applications and their software frameworks to support the RISC-V ISA. In addition, instead of focusing on the peak performance of dense computations, some of the applications use sparse data sets and/or low/mixed-precision, so one of the goals of the project is to develop a system that offers sustained application performance.

To achieve these goals, the eProcessor project is developing the eProcessor architecture, which is explained in Section 2. Then, Section 3 explains the two eProcessor prototypes, one single core and one multi-core, that will be fabricated in the project. Section 4 explains the application use cases of intereset for the project and their optimizations. Finally, Section 5 draws the main conclusions of this work.

2 EPROCESSOR ARCHITECTURE

Figure 1 shows a high-level overview of the eProcessor architecture, which consists of the RISC-V out-of-order core (RVOOO), the eAccelerator, the L2 cache, the Network-on-Chip (NoC), the IOMMU and the coherent chip-to-chip link. The first and second tape-outs include one and two RVOOO cores and eAccelerators, respectively.

2.1 RVOOO Core

The RVOOO core is a 4-way out-of-order scalar RISC-V core that supports the RV64GCV ISA.

2.1.1 *CPU Pipeline.*

The RVOOO CPU pipeline has an instruction fetch width of 6 instructions per cycle. The fetch stage contains an instruction FIFO, a gshare/TAGE branch predictor, a Branch Target Buffer (BTB), and a Return Address Stack (RAS). The BTB has a size of 512 entries, while the branch predictor Global History Register and Pattern History Table have a size of 22 and 8192 entries, respectively.

The register renaming stage contains a physical register file with 128 registers that are shared between floating point and integer units. The physical register file holds both speculative and committed state.

The Re-Order Buffer (ROB) size 128 is entries, and the total number of reservation station entries is 80 with 5 ports: 2 scalar units, FPU, MULT/DIV, and LD/ST. The maximum number of entries per port is 16. The reservation stations issue the oldest ready instructions to the functional units. The issue width is 1 instruction per cycle for each reservation station port, except for 2 instructions per cycle for the LD/ST port, adding up to a total of 6 instructions per cycle. The LD/ST queue has 16 entries, it supports out-of-order execution of loads and stores, and the loads can also be speculative.

2.1.2 *Private Caches.*

The RVOOO core includes private separate L1 instruction and data (L1I and L1D) caches of 16KB each and 64B lines. Both caches are virtually indexed and physically tagged.

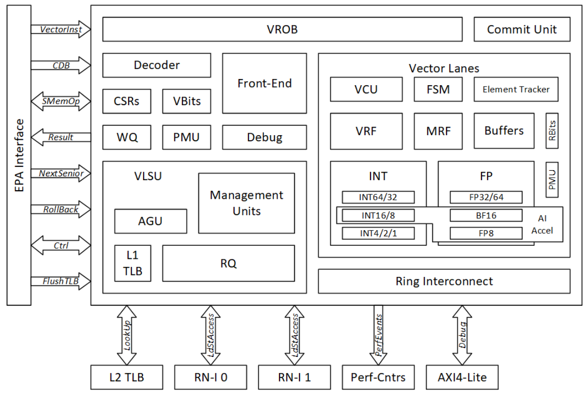

Figure 2: eAccelerator overview

The L1D supports two load or store operations (in any mix) per cycle. The latency of both operations is a single cycle. The L1D is write back and it allocates and loads on write access. A configurable number of hits under misses is also supported.

The L1D uses the MESI cache coherence protocol, supported via the AMBA5 Coherent Hub Interface (CHI) to the L2 cache. The L1D implements the RN-F functionality (fully coherent request node, which supports snoop transactions). The CHI data width is a 64B cache line, so the L1D tracks the dirty/clean state per line basis.

The L1I interface with the rest of the system is CHI RN-I with a data width of 64B. The L1I does not support hardware managed coherence.

2.1.3 *TLB Hierarchy.*

For virtual memory, the RVOOO implements the Sv39 and Sv48 address translation modes. The RVOOO includes two L1 TLBs and two L2 TLBs (one of each for instructions and for data), plus a single Page Table Walker (PTW). The data L2 TLB is shared with the eAccelerator. All the TLBs support flushing entries by virtual page number, by ASID, by both, and flushing the whole TLB.

Both instruction and data L1 TLBs are fully associative caches with 32 entries, and each entry can hold a leaf Page Table Entry (PTE) for any page size (4KB, 2MB, 1GB and 512GB in Sv48). Upon a miss, the L1 TLBs forwards the translation request to its corresponding L2 TLB.

Both instruction and data L2 TLBs share the same structure to store PTE lines as follows (note that a PTE line contains 8 contiguous PTEs coming from the same 512 bit memory word):

- 4 PTE lines for 512GB pages (fully associative, Sv48 only)
- 4 PTE lines for 1GB pages (fully associative)
- 4 PTE lines for 2MB pages (fully associative)
- 64/256 PTE lines (instruction/data) for 4KB pages (16-way set-associative)

Upon misses, both L2 TLBs forward the translation request to the PTW, which translates any virtual page number into its corresponding physical page number by walking the page table in memory. The PTW fetches an entire PTE line in a single cycle.

2.2 eAccelerator

The eAccelerator is composed by a Vector Processing Unit (VPU), reduced and mixed-precision functional units, and an Artificial Intelligence (AI) accelerator. Figure 2 shows a high-level overview of it.

The vector processing unit is a loosely-coupled vector accelerator featuring a long-vector design (*MAX_VLEN = 128 elements of 64b*)

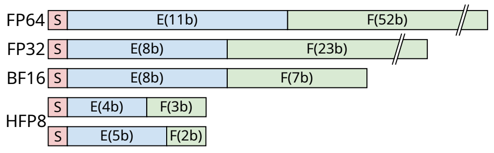

Figure 3: Supported FP formats in the eAccelerator

compliant with the RISC-V vector extension. It receives vector instructions at decoding stage from the RVOOO and returns the result including any generated scalar value or exception in a slight-out-of-order fashion. The vector accelerator has direct access to the L2 cache, it features a partial ordering for the execution of the vector loads and stores (out-of-order loads, in-order stores), and handles the memory disambiguation between scalar-vector and vector-vector memory operations. The design of the VPU is an extension of the Vitruvius [5] vector processing unit, which has been adapted to the eAccelerator architecture requirements, and a new vector load store unit has been developed from scratch. In addition, we use a novel specification for the interface between the RVOOO and the VPU, called the eProcessor Accelerator (EPA) interface. The EPA interface is divided into multiple sub-groups for different operations: Vector Instruction Dispatch, Common Data Bus, CPU MemOp (divided in Issue, Physical Address, Done and Overlap), Result, Commit (or Next-Senior), Roll-Back, Control and Flush TLB. In this implementation, the VPU has direct management of the CSRs.

The eAccelerator is composed of 4 vector lanes, each of them having a slice of the vector register file and the corresponding arithmetic units. The maximum vector length fits up to 128 64-bit elements per vector register. For narrower elements, a packed SIMD approach is used to increase the vector register utilization. In particular, it supports 64-, 32-, 16- and 8-bit integer and float operations, as well as narrow integer types (4-, 2- and 1- bits). In addition to the standard IEEE FP64 and FP32, it also supports Brain Float 16 (BF16) and two types of hybrid 8-bit Floating Point (HFP8), shown in Figure 3. The non-standard formats are aimed for more efficient training and inference of AI applications. In addition, all FP formats have support for Fused Multiply-Accumulate (FMA) operations, which achieve 2x FLOPS compared to individual operations. In particular, the functional units for HFP8 are highly optimized employing ad-hoc logic structures for the small bit-width operations. For achieving further efficiency in AI applications, the VPU is extended with a systolic array that reuses the available arithmetic units with minimal hardware overhead [6].

2.3 L2 Cache and Home Node

The eProcessor L2 Cache and Home Node (HN) are tightly coupled and operate together in conjunction to form the eProcessor Shared Last Level Cache. This is compliant with the AMBA5 CHI HN-F specification. The L2 Cache is an eight way-associative non-blocking write-back cache with Pseudo-LRU replacement policy that can handle a large number of outstanding transactions, misses and evictions. The HN is a fully mapped directory cache coherence controller responsible for tracking sharers at the cache line granularity and uses a MESI protocol to ensure coherence among the private L1 caches. The designs are modular and can be replicated and instantiated in multiple nodes of a CHI NoC to create systems with several distributed L2-HN slices, each responsible for a subset of the coherent system address space. The design of the L2-HN is fully-pipelined, it can serve up to one cache line per cycle (per slice) and it is optimized for throughput to efficiently serve long vector memory accesses, which can generate bursts of dozens of cache line requests.

The eProcessor L2-HN design is also augmented with two novel mechanisms: (i) dead-block management and (ii) runtime configurable scratchpad. The dead-block management mechanism facilitates the eviction of cache blocks that have been deemed dead, i.e. cache blocks that will not be reused before eviction as determined by the software runtime. In eProcessor, the OpenMP runtime detects address regions that will no longer be accessed based on task-dependency information and instructs the cache to evict them. The run-time configurable cache/scratchpad mechanism enables software to optimize data locality, to quickly adapt to the workload demands, and to offer predictable access latency for critical portions of the application dataset. The L2 cache allows parts of the cache data arrays to be configured as a scratchpad at run-time and at page granularity (4KB). A Linux kernel module handles the bookkeeping and supports the mapping to the application virtual address space.

2.4 Network-on-Chip

The eProcessor NoC, which is based on the FastTrackNoC [3], supports the AMBA5 CHI interface and uses five physical channels to accommodate the different CHI message classes and avoid protocol level deadlocks. Each physical channel has two Virtual Channels (VCs) to avoid head-of-line blocking and hence improve performance. The width of each physical channel corresponds to the size of the messages it delivers. This enables the NoC to deliver CHI messages as single flit packets. Each packet in a NoC router goes through three pipeline stages: (i) allocation, (ii) Switch Traversal (ST) and (iii) Link Traversal (LT). In the allocation stage round-robin priority arbitration is used for Switch Allocation (SA). Packets winning SA are allocated a free downstream VC. Parallel to SA, look-ahead route computation is performed to pre-compute the route the packet takes in the next hop. In the ST and LT stages, the packet traverses the router crossbar and the link, respectively, to propagate a hop.

The NoC reduces packet latency by allowing incoming packets to bypass up to two pipeline stages of a router when required conditions are met [3]. More precisely, packets can bypass the SA stage when entering the network, propagating a straight hop or exiting from the network, if (i) there is no competing traffic using the required ports of the switch and (ii) buffer space is available in the downstream VC. Additionally, incoming packets in VC-0 can also bypass the ST stage and directly proceed to LT using dedicated FastTrack paths [3], if (i) VC-0 FIFO buffer does not contain any other packet ahead of the bypassing packet, (ii) the packet is propagating a straight hop, (iii) the required output link is free, and (iv) buffer space is available in the downstream VC. Thereby, at best the packet latency per hop is as short as the delay of the link and a 2:1 multiplexer.

2.5 Coherent Chip-to-Chip Link

The Coherent Chip-to-Chip (C2C) link connects the NoC of the eProcessor to an external companion FPGA or to another eProcessor chip. The companion FPGA, which is described in Section 3.3, acts as a bridge from the ASIC to the DDR4 memory, PCIe and other I/O peripherals, as well as hosting accelerators connected to the eProcessor, such as the CNN off-chip accelerator described in Section 2.7.

The C2C link provides a AMBA5 CHI interface, and therefore extends the NoC to the companion FPGA. The connection between the eProcessor and the companion FPGA is physically based on an 8x SerDes link that provides 128 Gbps bandwidth per direction (i.e. 16 GB/sec) assuming zero link and packet overheads, and thus a theoretical aggregate bandwidth of 256 Gbps (i.e. 32 GB/sec).

To forward CHI packets to the FPGA, the C2C link wraps the packets in C2C link packets and these are forwarded by the SerDes. The link protocol uses 128-bit data granularity to simplify the encoding and decoding of link packets, which also minimizes the link latency. In addition, the C2C link includes link level re-transmission mechanism where each packet includes a CRC for error detection and, if a link receiver detects a CRC error, it requests a re-transmission of this packet and all the following ones from the opposite transmitter. The C2C link multiplexes the CHI packets from different virtual channels to the serial link and provides its own credit-based flow control optimized to sustain the provided link bandwidth. The serial link uses 64b/66b encoding as well as scrambling to provide enough transitions for the clock data recovery in the SerDes block.

2.6 IOMMU

The eProcessor architecture includes a RISC-V IOMMU IP fully compliant with the official specifications; this IP is among the first RISC-V IOMMU implementations. The IOMMU is utilized for IO devices connected to the eProcessor chip via the C2C link. The IOMMU supports multiple concurrent I/O devices assigned to multiple contexts (processes), includes IOTLBs with various caching structures with support for huge-pages, and performs hardware page table walks for standard 39-bit (Sv39) and 48-bit (Sv48) virtual addresses. The design targets high-performance and low-latency non-blocking operation with hits-under-misses and multiple concurrent hardware page table walks. The design includes the official CSRs, the Command and Fault queues (CQ, FQ) and a hardware performance monitor. The system interface is compliant with the AMBA5 CHI and the configuration interface compliant with AXI. Moreover, all the necessary Linux kernel support and IOMMU driver is developed in the project.

2.7 Off-Chip CNN Accelerator

A Convolutional Neural Network (CNN) hardware accelerator IP for FPGAs is developed in the eProcessor project to illustrate the ability of the eProcessor chip to be connected to external accelerators using the C2C link. The CNN IP can be generated for different FPGA targets and can be tuned with respect to the available computing or memory resources. The accelerator is programmable thanks to a firmware generation toolchain that takes CNN description files as input (Tensorflow files for example). Synchronization and data transfers between the accelerator and the host are managed using specific APIs that take into account the memory coherence feature provided by the C2C link. This off-chip CNN accelerator will be used to demonstrate a hybrid CPU/FPGA execution of the Border Surveillance application use case described in Section 4.5.

3 EPROCESSOR PROTOTYPES

This section describes the eProcessor prototypes that are developed in the project, including the two fabricated ASICs, the microserver architecture, and the companion FPGA.

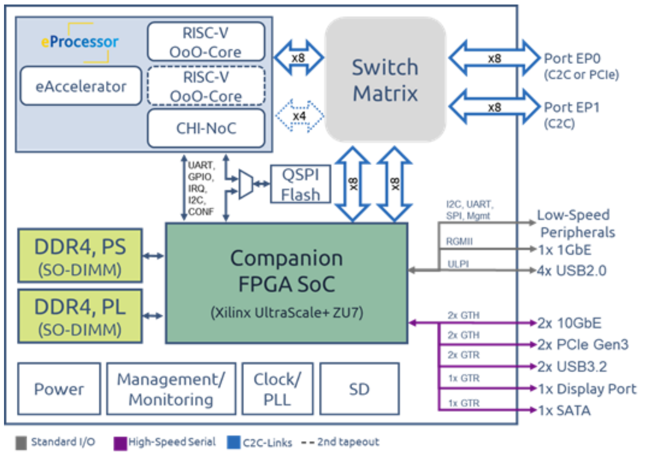

Figure 4: eProcessor Computer-on-Module overview

3.1 ASIC Implementation

The selected technology for the fabrication of the eProcessor ASICs is GLOBALFOUNDRIES 22nm FD-SOI (22FDX), which offers a good balance between performance and cost. The selection of the standard cells includes two options for high-performance (2 GHz and above): 12-track cells, which offer higher performance but have no option for body biasing, and 8-track cells, which offer high density and allow forward body biasing to increase performance. The use of either library version will be defined according to the trade off between timing and area estimations, that will be available from the first synthesis experiments. Other options such as 7.5-track and Ultra-Low Leakage (ULL) libraries usually offer worse performance, so they are not considered.

The memory compilers include register files of 1, 2 and Pseudo-2 ports for high performance and small size, and 1-, 2- and Pseudo-2-port SRAM for larger sizes, as well as a 1-port ROM. The high-performance register files support an operating frequency of up to 2.5 GHz with a size limit of 81 Kb. Larger SRAM sizes (up to 2 Mb) support only up to 1.1 GHz.

The tentative area utilization and budget for the first tape-out (the single-core ASIC) is 9 mm^2, while for the second tape-out (the multi-core ASIC) is 15 mm^2. The target frequency for the eProcessor core is 2 GHz maximum frequency at 100 °C junction temperature (max 45 °C ambient) in the slow-slow corner and a supply voltage of 0.9 V - 2%. The maximum design junction temperature will be 100 °C.

3.2 Microserver Architecture

For the bring-up and subsequent integration of the eProcessor ASICs in an operational environment, a flexible architecture is needed. To minimize costs and risks, we use a modular approach in which the ASICs are integrated into a COM (Computer-On-Module). The COM standard selected is the COM-HPC Client, which allows the integration of two large ICs (the companion FPGA and eProccesor ASIC) and also provides space for sufficient RAM. In addition, the standard features many interfaces such as 10 Gigabit Ethernet, USB 3 and PCIe. As depicted in figure 4, the targeted microserver architecture supports one eProcessor ASIC. The companion FPGA is used to accompany the eProcessor ASIC and acts as a combined north and southbridge like in classical x86 architectures (as explained in Section 3.3).

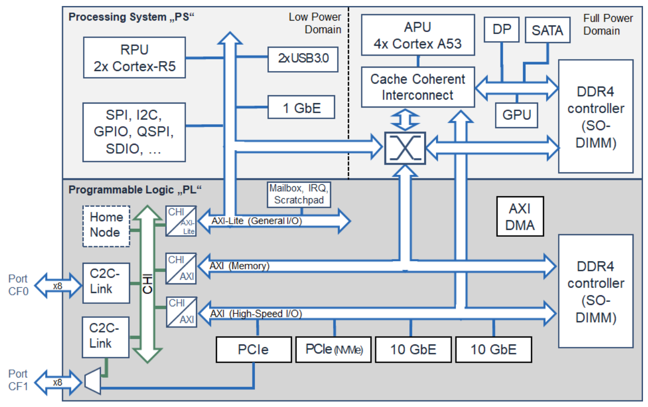

Figure 5: Companion FPGA system overview

The interconnection between the eProcessor ASIC, the companion FPGA, and external accelerators (or a dual-socket eProcessor system) is realized by a switch matrix offering a wide range of possible variations. Special care has been taken to ensure that both eProcessor tapeouts are supported by one microserver PCB.

3.3 Companion FPGA

Figure 5 shows an overview of the companion FPGA architecture. The companion FPGA provides various legacy and modern high-speed interfaces which are transparently mapped into the eProcessors address space via the C2C link. Some of the interfaces are available as hard IP core in the porcessing system, such as the USB 3 or the DisplayPort. Others are created in the programmable logic, such as the 10 Gigabit Ethernet or the UART. Beside the interfaces, a fully operational DDR4 Memory Controller is provided by the FPGA fabric.

In addition to providing interfaces, the CPU of the companion FPGA is used to manage the eProcessor module. This includes, among other, the configuration of the eProcessor ASIC or setting the bias voltage of the different substrates. Additionally, supply voltages and currents of the CPU can be managed and analyzed.

4 APPLICATION USE CASES

This section describes the application use cases that will be optimized for and demonstrated on the eProcessor prototypes.

4.1 NAS Parallel Benchmarks

The NAS Parallel Benchmarks [1] are a widely used benchmark suite to evaluate HPC systems. The benchmark suite consists of eight individual benchmark problems: EP, MG, CG, FT, IS, LU, SP and BT. These benchmarks focus on computational aerophysics, although most of the benchmarks have much broader relevance, since they are typical of many real-world scientific computing applications. The benchmark suite also includes a set of pre-defined input sets for each of the benchmarks, including small inputs that can be used in the early development stages of the architecture, up to large inputs that can be used to measure the performance of the eProcessor prototypes.

The NAS Parallel Benchmarks will be adapted to make the most of some of the features of the eProcessor architecture. In particular, in order to increase the performance of the benchmarks, the codes are going to be vectorized so they utilize the VPU of the eAccelerator. The vectorization will use the FP64 data type to fulfill the precision requirements of HPC applications. In addition, the scratchpad memory of the configurable L2 cache will also be leveraged.

4.2 Bioinformatics

Recent advances in next-generation sequencing technologies have enabled the proliferation of Bioinformatics applications. These applications have an enormous datasets and computational cost, so they have become a common workload in HPC systems and an important target for accelerators. In the project we use a set of Bioinformatics algorithms commonly found in different stages of genomic pipelines.

The Smith-Waterman-Gotoh algorithm is a dynamic programming algorithm that computes the pairwise alignment of two DNA sequences. The goal of pairwise sequence alignment is to identify similar regions between two biological sequences, which is useful for analyzing functional, structural, and evolutionary relationships between the two. Given the two sequences of lengths N and M, the time complexity of the SWG algorithm is $O(N \cdot M)$. The outputs of the algorithm are the similarity score computed using affine gap penalties and the optimal sequence of alignment operations (i.e., substitutions, insertions, and deletions) to align one sequence into the other.

The Banded Smith-Waterman-Gotoh algorithm implements a heuristic approach towards reducing the computational complexity of the original algorithm. To do so, it calculates the alignment between two sequences by considering only those residues that can be aligned within a diagonal band of width W. Using this approach, this algorithm reduces time and space complexity to $O(N \cdot W)$.

The Wavefront Alignment [4] algorithm proposes an alternative encoding of the dynamic programming matrix and an efficient algorithm to compute the alignment. To do so, it computes the cells of the dynamic programming matrix by increasing score, so it only needs to compute a minimal number of cells to find the optimal alignment. The WFA algorithm runs in $O(N \cdot S)$ time, proportional to the sequence length N and the error score S between sequences.

The FM-index is one of the most common data structures used within aligners and metagenomics classification tools. This data structure is used to identify the exact-matching locations of short sequence substrings (called seeds) within a reference genome. The algorithm is dominated by irregular memory accesses to a large memory space, being both memory-latency and memory-bandwidth bound.

4.3 DeepHealth Toolkit

The DeepHealth toolkit [2] is a software ecosystem consisting of two main open-source libraries for AI and computer vision that can be leveraged by medical applications use cases. The European Distributed Deep Learning (EDDL) library is an optimized tensor library for distributed deep learning. The library is built around the concept of tensor and offers many functionalities with a device-independent interface, enabling a strong decoupling between the network training logic and the hardware implementation. The European Computer Vision Library (ECVL) facilitates the integration and exchange of data between computer vision and image processing libraries.

The DeepHealth toolkit also provides a set of medical use cases that apply deep learning techniques for automatic medical diagnosis. One of them is the Skin Lesion Classification use case, which targets improving melanoma diagnoses and reducing melanoma mortality

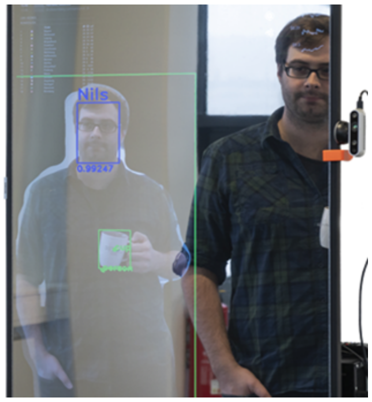

Figure 6: Smart Mirror performing face and object recognition

by facilitating the application of digital skin imaging technologies. To do so, this use case uses a VGG16 deep learning model to perform the classification of sample images from the International Skin Imaging Collaboration (ISIC) dataset across eight different categories (i.e., classes). The ISIC 2019 dataset is divided into three subsets of images: 19,330 for training, 1,000 for validation, and 5,001 for testing.

The libraries of the DeepHealth toolkit will be adapted to speed up the computations by leveraging the AI accelerator of the eProcessor architecture, including the low-precision FP formats BF16 and HFP8.

4.4 Smart Mirror

The Smart Mirror is designed as an intuitive interface to support interaction in smart home environments, focusing on local processing to protect the users data. It shows a mirror image of the user and overlays personalized information or smart home status on top of the image. For this purpose, a camera image is used to recognize faces, objects and simple hand gestures for intuitive control (see figure 6). All calculations are performed locally on the device with open-source software, which ensures the highest level of privacy, as no data is transferred to the cloud and third-party providers. The local data processing should be coupled with the highest level of energy efficiency, as excessive power consumption is not desirable in private homes. The AI accelerator of the eProcessor architecture will take over the calculation of the neural networks in a performant way, providing low latency and high throughput for a pleasant user experience.

4.5 Border Surveillance

Drone-based border surveillance consists of using drones flying at a high altitude to capture and analyze high-resolution images to detect (and sometimes recognize) specific targets, threats or illegal activities. The Border Surveillance use case targets a maritime context in which the drone is flying along maritime borders to detect boats. The drone embeds sensors such cameras to acquire colored or thermal/infrared images to be processed. Due to the speed of the drone, the varying environment and the high altitude, the detection task may be quite hard, especially if the drone flies near a coast, if there are rocks in the see, foam or wave shadows. Thus, efficient detection algorithms must be chosen to achieve the required accuracy and avoid too much false detection. In the recent years, CNN algorithms have been used with great success in detection tasks and are thus natural candidates to tackle the challenges inherent to such a use case.

One of the primary constraints on the computing subsystem is the power consumption, due to limited battery capacity of the drone. For this reason, in the embedded domain, detection algorithms with high accuracy such as CNNs rely on accelerators to reach the necessary levels of performance and energy-efficiency. Such an energy-efficient computing system is planned to be composed by the eProcessor architecture connected to an FPGA configured with the off-chip CNN accelerator described in Section 2.7. The FGPA-based off-chip CNN accelerator will be responsible for the execution of the majority of the computations, while specific parts of the CNN will be executed on the eProcessor chip if they are not relevant for hardware acceleration.

5 CONCLUSIONS

The goal of the eProcessor project is to create a full stack high performance energy-efficient ecosystem based on RISC-V. To this end, the eProcessor architecture combines a high-performance out-of-order core with energy-efficient accelerators for vector processing and artificial intelligence with low- and mixed-precision functional units, providing high performance and energy efficiency for a set of relevant application uses cases in the fields of high-performance computing, bioinformatics and artificial intelligence. In a hardware/software co-design approach, the application use cases drive the design of the system, and the project ports these applications and their software frameworks to RISC-V. The project will build two prototypes based on two fabricated eProcessor ASICs, one single core and one multi-core.

ACKNOWLEDGMENTS

The eProcessor project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 956702. The JU receives support from the European Union's Horizon 2020 research and innovation programme and the respective national research organisations from Spain (PCI2021-121991/ MCIN/AEI/10.13039/501100011033), Sweden, Greece, Italy, France (ANR-20-EHPC-0006), and Germany (BMBF). The eProcessor project is also co-funded by the UE NextGenerationEU/PRTR.

REFERENCES

[1] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, H. D. Simon, V. Venkatakrishnan, and S. K. Weeratunga. 1991. The NAS Parallel Benchmarks—Summary and Preliminary Results. In *Proceedings of the 1991 ACM/IEEE Conference on Supercomputing (Supercomputing '91)*. Association for Computing Machinery, 158–165.

[2] Michele Cancilla, Laura Canalini, Federico Bolelli, Stefano Allegretti, Salvador Carrión, Roberto Paredes, Jon A. Gómez, Simone Leo, Marco Enrico Piras, Luca Pireddu, Asaf Badouh, Santiago Marco-Sola, Lluc Alvarez, Miquel Moreto, and Costantino Grana. 2021. The DeepHealth Toolkit: A Unified Framework to Boost Biomedical Applications. In *2020 25th International Conference on Pattern Recognition (ICPR)*. 9881–9888.

[3] Ahsen Ejaz and Ioannis Sourdis. 2022. FastTrackNoC: A NoC with FastTrack Router Datapaths. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. 971–985.

[4] Santiago Marco-Sola, Juan Carlos Moure, Miquel Moreto, and Antonio Espinosa. 2020. Fast gap-affine pairwise alignment using the wavefront algorithm. *Bioinformatics* btaa777 (2020), 1–8.

[5] Francesco Minervini, Oscar Palomar, Osman Unsal, Enrico Reggiani, Josue Quiroga, Joan Marimon, Carlos Rojas, Roger Figueras, Abraham Ruiz, Alberto Gonzalez, et al. 2023. Vitruvius+: An Area-Efficient RISC-V Decoupled Vector Coprocessor for High Performance Computing Applications. *ACM Transactions on Architecture and Code Optimization* 20, 2 (2023), 1–25.

[6] Mateo Vazquez Maceiras, Muhammad Waqar Azhar, and Pedro Trancoso. 2022. VSA: A Hybrid Vector-Systolic Architecture. In *2022 IEEE 40th International Conference on Computer Design (ICCD)*. IEEE Computer Society, 368–376.