Multi-Partner Project: Advancing European Semiconductor and Chiplet Innovation Through the Bavarian Chip Design Center

Hussam Amrouch[1]*, Jehaan Joseph[1], Michael Schirmer[2], Johannes Geier[2], Ulf Schlichtmann[2], Michael Meidinger[3],Thomas Wild[3], Andreas Herkersdorf[3], Jens Nöpel[4], Georg Sigl[4,7], Yicheng Zhang[5], Carsten Trinitis[5,6], Aswathy Nedumpalli Sankaranarayanan[6], Martin Schulz[6], Andreas Korb[7], Konrad Hohentanner[7]
Chairs of [1]AI Processor Design, [2]Electronic Design Automation, [3]Integrated Systems,
[4]Security in Information Technology, [5]Computer Architecture and Operating Systems, and
[6]Computer Architecture and Parallel Systems, Technical University of Munich, Munich, Germany
[7]Fraunhofer Institute for Applied and Integrated Security (AISEC), Garching, Germany

***Abstract*—Europe's semiconductor industry relies heavily on Asian and US manufacturers. The EU Chips Act seeks to strengthen Europe's capabilities across the semiconductor value chain. Aligned with this goal, the Bavarian Chip Design Center (BCDC) supports local chip design, manufacturing, and talent development, with a focus on RISC-V computing and heterogeneous integration. Within BCDC, the Technical University of Munich and Fraunhofer are developing a chiplet-based architecture optimized for low-power edge AI. The system integrates two chiplets, combining a security-enhanced RISC-V core and AI accelerators, connected via a chiplet-optimized serial interface that supports encrypted data. The chiplets are mounted on a custom interposer with low-capacitance wires for efficient data transmission. System- and component-level development is currently ongoing, with a tapeout in 22 nm FD-SOI planned for 2027. The overall goal is to deliver a proof of concept for a small-scale energy-efficient chiplet system that demonstrates Bavaria's and Europe's capability to drive innovation in novel chip design fields.**

***Index Terms*—chiplet, chiplet architecture, SoC, heterogeneous integration, edge AI, RISC-V, ASIC, interconnect design**

I. Introduction

The European semiconductor landscape remains dependent on manufacturing and packaging capacity in Asia and the United States. Recent initiatives, such as the EU Chips Act, therefore aim not only to expand domestic fabrication capabilities but also to strengthen Europe's position in semiconductor research, system design, and advanced integration strategies. Within this context, the Bavarian Chip Design Center (BCDC) aims to support local chip design, to strengthen local supply chains, and to train new semiconductor engineers. A central focus of the BCDC is the exploration of RISC-V–based computing platforms and heterogeneous system integration approaches for flexible, energy-efficient architectures. This project is a small part of a much larger-scale strategy by the government of Bavaria to further establish the state of Bavaria as a center for microelectronics within the European Union (EU) [1]. The goal is to help establish a sustainable semiconductor design ecosystem within Bavaria and Europe.

This work was funded as part of the Bavarian Chip-Design-Center by the Bavarian Ministry of Economic Affairs, Regional Development and Energy (funding numbers: RMF-SG20-3410-2-16-3 and RMF-SG20-3410-2-17-14).
*corresponding author: amrouch@tum.de

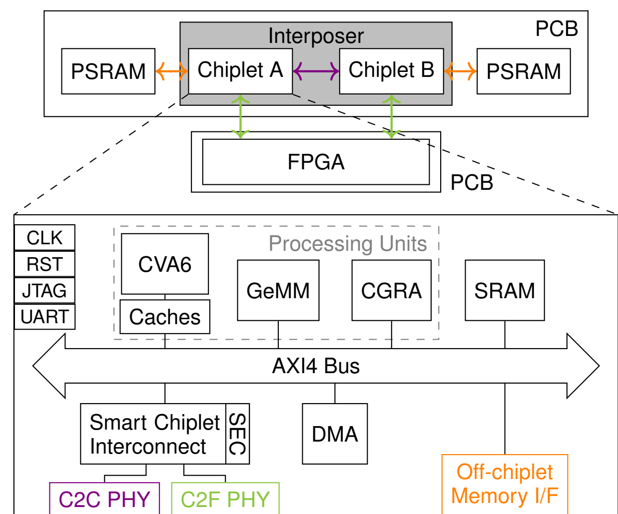


Fig. 1: The BCDC demonstrator system: Two identical chiplets connected via a fast and power-efficient Chiplet-to-Chiplet (C2C) interface (I/F) on a silicon interposer. Chiplet-to-FPGA (C2F) and off-chiplet memory I/Fs for evaluation purposes.

As part of this effort, the Technical University of Munich, in collaboration with Fraunhofer, is investigating a compact chiplet-based architecture tailored for low-power edge Artificial Intelligence (AI). Rather than relying on a large monolithic System on Chip (SoC), the system splits its functionality across two chiplets in a shared package. Each chiplet implements a SecureSoC comprised of a processing subsystem with a security-enhanced CVA6 core [2], a systolic array General Matrix Multiplication (GeMM) core, and a Coarse Grained Reconfigurable Arrays (CGRA) unit for dense and sparse compute acceleration, respectively. As depicted in Fig. 1, two equivalent SecureSoC chiplets are mounted on a custom silicon interposer with low-capacitance interconnect lanes, which is placed on

978-3-9826741-1-7/DATE26/© 2026 EDAA

a demonstrator printed circuit board (PCB) featuring off-chip Pseudo-Static Random Access Memory (PSRAM). The system includes a Field Programmable Gate Array (FPGA) for outside communication, system bring-up, and debugging. Responsible for connecting the SecureSoCs is a smart chiplet interconnect that builds on and extends the Parallel Ultra Low Power (PULP) Serial Link. In addition to basic data transfer between chiplets or a chiplet and the demonstrator host FPGA, the interconnect incorporates mechanisms for secure (encrypted) communication, as well as optional power reduction and reliability features, enabling efficient data movement with a low pin count. Furthermore, the SecureSoC features on-chip Static Random Access Memory (SRAM) for tightly-coupled instruction and data memory. Software development proceeds in parallel and includes a Machine Learning (ML) compiler and runtime tailored to the GeMM and CGRA accelerators that operate on an edge AI application.

System- and component-level development is currently underway, with RTL of the individual SoC and FPGA prototypes scheduled for Q4 2026. Tapeout of the SoC is planned for Q2 2027, followed by integration into a system of chiplets starting in Q1 2028, as depicted in Fig. 1. The resulting demonstrator aims to serve as an initial reference platform, showcasing a small-scale, energy-efficient RISC-V-based system of chiplets. We outline the early-stage development process for a large-scale prototyping effort of a chiplet-based design, spanning multiple disciplines of semiconductor design. The presented system-level design for digital computing and communication within a security-driven framework is a unique, multi-partner effort. In summary, this work has the following contributions:

1) We review the current state-of-the-art (SotA) in accelerators, compilers, and chiplet interconnects and highlight the need for a lightweight, open, and secure architecture and present a novel smart chiplet interconnect that incorporates encryption, compression, and reliability features.
2) We describe the incorporation of open-source tools and software, describing how the system integrates our own CGRA and GeMM accelerators with IREE-based compiler and runtime support to enable distributed, heterogeneous AI workloads across multiple chiplets.
3) We outline the modernization of the open-source CVA6 software development kit to simplify integration of novel RISC-V security extensions, ensuring long-term maintainability and accessibility for new developers.

II. Related Work

Compute Acceleration for Dense and Sparse Problems: Recent research in the domain of edge AI acceleration primarily focuses on optimizing dense and sparse matrix operations. In dense calculations, large systolic arrays are effective for high-throughput GeMM operations, as discussed in designs like Google's TPU [3] and NVIDIA's Tensor Cores [4]. Open-source frameworks, such as Gemmini [5] and VTA [6], develop flexible exploration of dataflows, tiling strategies, and mixed-precision arithmetic. Recent design VerSA [7] proposes advanced architectures with efficient interconnects, tiling techniques, and low-latency data flows, which help increase utilization and throughput. CGRA designs, on the other hand, increasingly integrate support to achieve a wider application range, as well as more agile compilation and mapping, in parallel with the long-respected target of performance and efficiency optimization. To that end, a variety of efforts are proposed in terms of microarchitecture, Instruction Set Architecture (ISA), and compilation toolchain. Control flow operator [8]–[11], speculative execution [12], [13], and dynamic temporal pipelining [14] are reducing the limitation of traditional Compute Unit based CGRA, which only accelerates the inner-loop kernel. Therefore, CGRAs have gained the ability to cope with irregular memory access and control flow, bridging the gap of sparse linear algebra and graph analytics execution.

ML Compilers - Retargeting for Efficient Hardware: ML compiler frameworks have evolved to support heterogeneous and distributed hardware targets. Apache TVM [15] provides an end-to-end stack, ranging from high-level graphs (Relax) to tensor-level code, TensorIR (TIR), with automated optimization and a Bring-Your-Own-Codegen (BYOC) interface for vendor-specific accelerators. Recent work also explores retargetable and distributed ML compilation beyond monolithic SoCs. MATCH [16] introduces a model-based abstraction for automatic retargeting across microcontrollers and accelerators on top of the TVM stack. While TVM offers strong auto-tuning and heterogeneous scheduling, its runtime integration remains coarse-grained for controlling device dispatch and synchronization across execution domains. The Intermediate Representation Execution Environment (IREE) [17] builds on the Multi-Level Intermediate Representation (MLIR) [18] and LLVM [19] compiler infrastructure and demonstrates a modular, end-to-end approach to heterogeneous ML deployment. Unlike TVM's optimization-centric design, IREE focuses on portability through its multi-backend architecture. TinyIREE [20] extends IREE for bare-metal systems targeting deeply embedded edge devices. Additionally, Google's most recent effort with the open-sourced Coral NPU platform [21] integrates IREE and its Hardware Abstraction Layer (HAL) to compile and deploy workloads efficiently on custom RISC-V-based NPUs. Furthermore, the MLIR framework has been used in recent studies to facilitate systematic compilation for CGRAs. ML-CGRA [22] presents a unique MLIR dialect to express CGRA operations, interconnects, and configuration semantics. [23] introduces MLIR passes for placement, routing, and configuration and specifies explicit CFG-to-CDFG transformations for CGRAs. HDCC [24] targets heterogeneous systems, including CPUs, CGRAs, and FPGAs, employing a dataflow-centric compilation model, which analyzes high-level affine loop programs to extract parallel kernels as dataflow. Multi-chip and chiplet inference systems, such as Simba [25] and Occamy [26], a RISC-V platform from the PULP initiative [27], demonstrate that die-to-die communication and partitioned workloads can deliver scalable edge performance.

Chiplet Interconnects: Commercial and open-source chiplet interconnects today are mainly designed for very high bandwidth. A leading example is Universal Chiplet Intercon-

nect Express (UCIe), which is emerging as the primary industry standard alongside proprietary solutions. UCIe targets large multi-chiplet systems and supports at least 4 GT/s per lane with a minimum of eight parallel lanes [28], enabling system scaling without the yield penalties of large monolithic dies. However, such high-performance links are excessive for small, low-power systems. In these designs, chiplets are motivated by flexibility, faster development, and cost- and power-efficiency through mixing technology nodes. To support this domain, simpler, lower-bandwidth interconnects are needed to reduce power, input/output (I/O) pin count, and interposer complexity, while still allowing interoperability through an open standard, which current high-end solutions do not provide.

The PULP Serial Link [29] addresses these requirements by being open-source and scalable. It is developed within the PULP platform [27], which also provides the CVA6 RISC-V core used in this work. The Serial Link offers configurable numbers of channels and lanes at single or double data rate frequencies, supporting both high and low bandwidths, including compact, low-power chiplet setups as targeted here. It follows a three-layer stack aligned with the Open Systems Interconnection (OSI) model. The network layer maps Advanced eXtensible Interface (AXI) to AXI-Stream, the data link layer manages channel allocation and synchronization, and the physical layer provides transmit/receive paths with parallel lanes and a forwarded clock [29]. Only the digital specification is defined; analog implementation is left to the designer. Occamy [26] utilizes this Serial Link to integrate CVA6 clusters with AI cores in a multi-chiplet architecture.

III. Chiplet Features and Interconnect

To meet the power, area, and I/O constraints of our system, the chiplets are connected through high-frequency interfaces that communicate over a few low-capacitance wires. Placing the chiplets on a shared silicon interposer, rather than using separately packaged dies, enables higher interconnect frequencies at lower power. The analog aspects of the physical layer are outside the scope of this work; the digital components, higher layers, and additional features are discussed in later sections.

A. Smart Chiplet Interconnect

Building on the PULP Serial Link and motivated by the need for lower-bandwidth chiplet interconnect standards, we introduce a smart chiplet interconnect that adds advanced capabilities inspired by standards such as UCIe. Its primary function, described below, is implemented through a custom network-layer protocol, with additional enhancements integrated across the stack. While the physical layer is split for the C2C and C2F interfaces, as shown in Fig. 1, the upper layers are unified.

a) Inter-Chiplet Memory Access: The protocol enables efficient memory transfers across chiplets with minimal changes to standard intra-chiplet AXI transactions. Processing units on one chiplet can issue AXI read and write operations targeting reserved remote address ranges. The interconnect stack performs packet assembly, metadata handling, and routing in partially connected topologies, without system bus involvement.

For the expected edge AI workload, a single bidirectional channel with eight lanes per direction (18 I/O pins total) is sufficient. Since the interconnect stack includes an AXI manager interface, processing units can perform transfers in the style of remote direct memory access (RDMA) directly, with CPU interrupts generated only when required. This allows accelerators to write processed data directly into remote memory, reducing orchestration overhead.

b) Application-Specific Enhancements: Beyond this base function, the smart interconnect integrates optional, data-selective processing features inspired by smart network interface cards (SmartNICs), applied only when beneficial to avoid unnecessary latency and power consumption. Data flows can be classified via flags or inferred characteristics to determine which hardware modules are used. A key extension is lossless compression, which reduces interconnect bandwidth demands and thereby lowers power, pin count, and complexity of transceivers and interposers. A modified streaming Lempel-Ziv 4 (LZ4) variant with preloaded dictionaries and compression ratio tracking is employed, chosen for its low latency and suitability for repetitive data patterns. Data may be decompressed upon receipt or stored in compressed form. Additional modules provide the security features detailed in section III-B1, as well as reliability features including packet- and sub-packet-level error detection and correction. Retransmissions or redundant transmission can be triggered when needed, offering more flexible reliability control compared to UCIe. Furthermore, the number of active lanes may be dynamically scaled to reduce power consumption under low load.

In summary, the smart chiplet interconnect retains the simplicity of the PULP Serial Link, while adding selected advanced features typically found in more high-end interconnects. Its modular design and open-source foundation makes it well-suited for small-scale, energy-efficient chiplet-based systems such as our BCDC edge AI system.

B. Security Features

1) Secure Interchiplet-Connect: Physical and logical interfaces connecting chiplets, especially via high-speed serial links or interposers, introduce new attack surfaces largely absent in traditional monolithic SoC designs. Disaggregating a design exposes more interconnects and interfaces that are easier to access and probe, making physical attacks more feasible. Interposer-level wiring between chiplets offers adversaries a practical surface for tampering or contactless probing [30], underscoring the need for cryptographic protections.

In this project, ensuring confidentiality and integrity across the chiplet interconnect is a key objective to achieve a secure system for edge AI applications, similarly to emerging serial link security standards such as Integrity and Data Encryption (IDE) defined by PCI-SIG and the security extensions proposed in UCIe. The BCDC architecture integrates security features directly into the interconnect stack of the PULP Serial Link, including cryptographic primitives that ensure message integrity and confidentiality between chiplets. To maintain high throughput while providing these protections, Authenticated Encryption (AE) is applied on a per-package basis.

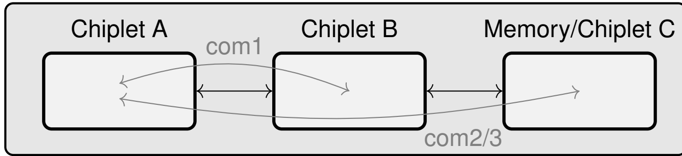


Fig. 2: Targeted communication scenarios (com1/com2/com3) protected by the extensions of the PULP Serial Link interface.

In heterogeneous chiplet systems, several communication scenarios arise that require dedicated security mechanisms to ensure data confidentiality and integrity across potentially untrusted boundaries. In the project, we focus on the following scenarios illustrated in Fig. 2:

a) Communication scenario Com1: involves the secure connection between *Chiplet A* and *Chiplet B*, where the direct communication channel, such as the interposer link, is not trusted and may be susceptible to eavesdropping or tampering.

b) Communication scenario Com2: covers secure access from *Chiplet A* to external memory attached to *Chiplet B*. Since *Chiplet B* is untrusted, data must be encrypted and authenticated to prevent unauthorized observation or modification.

c) Communication scenario Com3: represents a multi-hop path where *Chiplet A* communicates securely with *Chiplet C* by routing traffic through *Chiplet B*, which acts only as an intermediary. In this case, *Chiplet B* must not have access to plaintext data or security credentials, requiring link-layer protection schemes that preserve confidentiality and integrity.

2) RISC-V Security Extensions: Software vulnerabilities based on unsafe languages, such as C/C++ (e.g., buffer overflows, use-after-free, and double-frees), are frequently used to initiate attacks, targeting either other data or control flow, with significant security implications. To protect a heterogeneous software stack efficiently, new RISC-V security extensions have been proposed to enable effective software security methods by either mitigating the effects of such errors (e.g., through enforcement of Control-Flow Integrity (CFI)) or preventing initial memory errors entirely (through enforcement of memory safety). The extensions rely on compiler instrumentation coupled with automated hardware-based check logic, which avoids the need to rewrite existing application source code.

In our prototype design, our goal is to implement a set of extensions that enable a flexible protection scope, tailored to the specific use case. In addition to adding the new instructions, exceptions, and Control and Status Registers, we also modify the relevant components in the CVA6 pipeline.

Shadow Stack (Zicfiss): Shadow stacks are separated stack regions that are used to securely store return addresses, providing backward-edge CFI. With shadow stack enabled, instrumented instructions automatically verify the return address before each function return to prevent such attacks. To adapt the extension in CVA6, a check logic for return addresses is added, along with an isolation mechanism for the shadow stack.

Landing Pads (Zicfilp): Landing pads are compiler-generated indicators that assign metadata to jump targets, enabling forward-edge CFI. By grouping valid jump targets, the instrumentation sets up which branches are reachable from a specific location. Changes to the branch logic enforce the verification of the landing pads.

Memory Tagging (Zimt): In memory tagging, the so-called tags (4 or 7-bit values) are added to both the upper pointer bits and the memory object during allocation. Pointers are only allowed to access data where the memory tag matches the one stored in the pointer, ensuring memory safety by enforcing in-bounds and in-lifetime accesses. Memory tagging relies on a modified load-store unit to automatically verify memory tags on reads/writes and new instructions for managing tag memory. The integration of various software security extensions within the same hardware design, for the first time, enables a comprehensive evaluation of the respective resource costs and timing implications. Importantly, this will provide the computer architecture research community with a better understanding of the design trade-offs when adapting these extensions.

IV. AI Retargeting

A. Application Accelerators

The project utilizes two accelerators: systolic arrays as GeMM engines and CGRAs to accelerate irregular workloads.

1) Systolic Array-based GeMM: The core computation in DNNs is dominated by matrix multiplications, which can be efficiently accelerated using Multiply-Accumulate (MAC) systolic arrays. SotA research has demonstrated various implementations of systolic arrays, showcasing significant performance improvements in executing NN layers. However, when higher energy efficiency is targeted, a key challenge arises due to the rapid evolution of DNN architectures in which different models, and even different layers within the same model, often require varying precisions. To address this, our implemented AI accelerator employs a *multi-precision MAC systolic array* capable of supporting flexible precision configurations for both activations and weights, ranging from 2-bit up to 16-bit. This configuration can be *dynamically* adjusted at runtime, enabling a tunable balance between computational accuracy and throughput. As a result, when lower precision is selected, the systolic array can execute a greater number of MAC operations per cycle, achieving higher speedup with minimal loss in model accuracy, thereby offering an effective trade-off between performance and precision. Fig. 3 shows the accelerator's architecture.When lower precision is selected, it can fetch more data from memory, ensuring the systolic array remains fully utilized.

2) CGRA: CGRA features a Network on Chip (NoC) interconnected, ALU-like Processing Element (PE) array delivering flexible parallelism. Though similar to the systolic array, with reconfigurable PE and interconnect, CGRA is programmable by mapping various workloads in the format of Data-Flow Graph (DFG) onto the fabric. On the other hand, CGRA as a spatial architecture is able to demonstrate higher computational parallelism than the scalar CPU core by distributed functional units. Despite the additional area consumption or compilation effort required to achieve such advantages, CGRA is proven to approach ASIC-level energy efficiency [8].

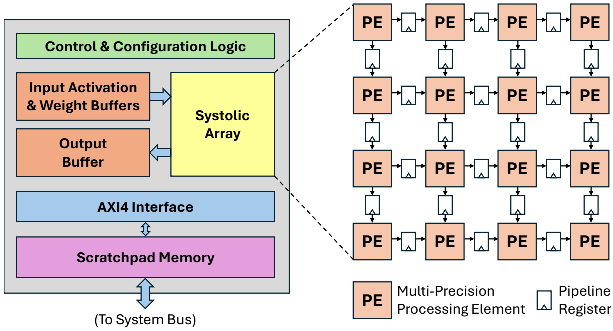


Fig. 3: Architecture of systolic array-based GeMM subsystem.

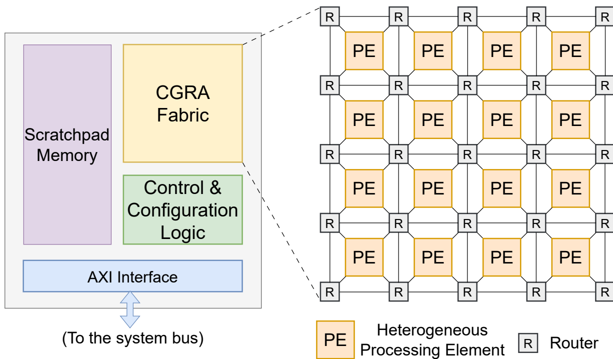


Fig. 4: Architecture of CGRA subsystem.

a) Subsystem and Integration: As in Fig. 4, CGRA fabric composed of router-connected PE array is the core of the subsystem, following the specialized Control and Configuration Logic. When the DFG to be accelerated cannot fit the fabric, it would be partitioned into sub-graphs and intermediate results are buffered on the scratchpad memory. The subsystem is attached to the system bus by its AXI interface, and managed by CPU through register configuration using bus transactions.

b) ISA: Integer and low-precision floating-point arithmetic instructions are essential to workloads like EdgeAI, fitting the limited power and memory budget on edge deployment. Moreover, control flow behavior in irregular parallelism, i.e., conditional branches and synchronization, should be extracted and attributed to a set of representative control flow instructions.

c) Microarchitecture: By heterogeneous PE, the ISA is implemented in a distributed way as each PE type supports a subset of instructions to improve area and energy efficiency; In addition, dynamic scheduling of operations by dataflow execution is suitable for irregularity as the nondeterministic memory access can be processed by the latency tolerant handshake communication protocol in hardware, rather than challenging the static scheduling over compilation.

d) Bitstream Generation: To pave the way from application code to CGRA configuration, high-level language code like C would first be converted to intermediate representation by compiler front end e.g. Clang. Afterward, middle-end compiler passes optimize and produce the DFG. The bitstream would then be generated by mapping the DFG onto CGRA fabric.

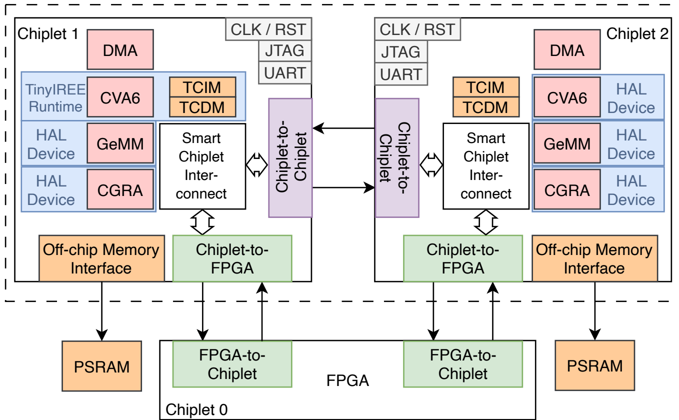
B. ML Compilers and Retargeting

AI workloads are naturally suited for distributed and heterogeneous execution, combining general-purpose cores with domain-specific accelerators such as GeMM engines or CGRAs. To enable such distributed inference across chiplets within a unified compiler-runtime stack, IREE was adopted for its flexible HAL, which abstracts memory management, command dispatch, and synchronization across diverse hardware targets. The modular HAL design enables the integration of new accelerators with minimal effort by implementing only a lightweight driver that defines device setup, buffer handling, and command execution. Together with IREE's *EmitC* backend, which generates portable C code, this supports rapid hardware-software co-design with minimal compiler-side modifications.

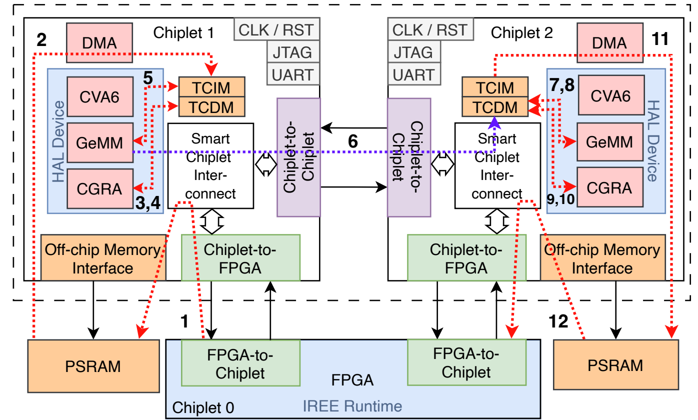
Two runtime integration variants are explored for the BCDC demonstrator (see Fig. 5): In the first configuration (Fig. 5a), a TinyIREE-inspired runtime runs directly on the CVA6 of Chiplet 1. Each hardware accelerator, namely the CGRA and GeMM engines on both chiplets, as well as the CVA6 core on Chiplet 2, is represented as an individual HAL device. The MLIR model is static, compiled through the *EmitC* backend into C code and linked into the minimal runtime binary. Workloads are therefore embedded at build time, with dispatches to other chiplets handled via the smart chiplet interconnect. This configuration minimizes runtime and Read-Only Memory (ROM) footprint and enables embedded-class execution.

In the second configuration (Fig. 5b), a full IREE runtime runs on the FPGA host. Chiplet 1 and Chiplet 2 are each exposed as composite HAL devices, simplifying orchestration since only two HAL entities are managed by the host. In this setup, the MLIR model is compiled ahead-of-time into a virtual-machine bytecode file (`.vmfb`), which can be dynamically loaded and executed by the runtime at inference time. This enables more flexible deployment and model exchange, but incurs higher overhead due to the full bytecode interpreter. The increased driver complexity inside each chiplet allows the respective HAL driver to internally distribute workloads to local compute elements and coordinate their execution. Overall, this configuration reflects a scalable and adaptable model for heterogeneous multi-chiplet AI systems, hiding system complexity behind the HAL interface. In the future, the FPGA host handling system-level orchestration will also be a chiplet.

1) Demonstrator Application - Real-time Object Detection: The demonstrator uses Mobilenet-SSD [31], [32], a compact Convolutional Neural Network (CNN) designed for real-time object detection on embedded systems. The data flow, illustrated in Fig. 5b, is represented using numbered stages to indicate the sequential movement of data through two chiplets. The CPU manages configuration tasks, and the on-chip SRAM (TCDM/TCIM) enables pipelined execution. The RAW frames from the camera captures (for example, resolution of 1280×720 pixels) are sent to the FPGA. The FPGA resizes and outputs a $300\times300\times3$ RGB frame ($\approx$270 KB), which serves as the input for Mobilenet-SSD (Flow 1). The CGRA (of Chiplet 1) divides frames into tiles and schedules tiles with a size of $32\times32\times4$ ($\approx$4 KB each) and issues Direct Memory Access (DMA)



(a) Embedded IREE runtime on Chiplet 1 with individual HAL devices for all accelerators.



(b) Full IREE runtime on FPGA host with each chiplet represented as a composite HAL device showcasing the demonstrator application.

Fig. 5: The BCDC system orchestration: Hardware abstraction and runtimes for the proposed system.

from PSRAM to Chiplet 1's SRAM (Flow 2). When DMA is complete, CGRA (of Chiplet 1) performs early convolution and the intermediate feature maps are stored in the on-chip memory (Flow 3,4). The Chiplet 1 GeMM performs depthwise and pointwise convolutions on these tiles for the mid-level layers (Flow 5). The output feature tiles of size $32\times32\times8$ (≈ 8 KB) are sent across the chiplet interconnect to the on-chip memory of Chiplet 2 (Flow 6). The GeMM (of Chiplet 2) processes the deeper layers and classification heads, producing bounding-box activations (≈ 1 KB per frame) (Flow 7,8). CGRA (of Chiplet 2) performs post-processing on these activations (Flow 9,10). CGRA produces and sends the final detections to PSRAM of Chiplet 2 (Flow 11) and then to the FPGA (Flow 12) via DMA. Both chiplets store buffers and model weights in PSRAM. The design enables real-time, low-latency, and high-throughput object detection (30 FPS at 300×300 resolution), while utilizing a limited amount of on-chip SRAM.

C. *Software and Software Development Kit (SDK) Support*

The software components that typically require adaptation to integrate a new security extension on a Linux system are (i) the compiler (gcc); (ii) OpenSBI; (iii) U-Boot; and (iv) the Linux kernel. The OpenHW Group provides a SDK to build and bundle these components to run them on the CVA6 [33]. However, the SDK is outdated, with the Linux kernel used being 5.10.7, released in January 2021, and the GCC compiler being version 10.3.0 from April 2021. This hinders the adoption and testing of new software components and RISC-V security extensions. For example, we want to add the RISC-V CFI extensions for forward and backward branch protection, namely landing pads (Zicfilp) and shadow stack (Zicfiss). However, these extensions have yet to be included in Linux, while the current patchset (v22) targets Linux 6.18-rc1 (Oct. 2025). We adapt the SDK for the following goals:

1) Sustainable maintenance: Updating components alone is not sufficient; it is also important to reduce maintenance efforts to ensure an up-to-date SDK in the future. Similarly to the original SDK, we built the setup around buildroot. Conversely, we leverage the Buildroot infrastructure whenever feasible, such as by utilising its provided interfaces to add external packages. This streamlines the build process, thus ensuring that workarounds and deviations from the buildroot workflow are avoided. CVA6-specific changes are added as minimal patches, applied by buildroot to the respective components at build-time, to enable an upstream-oriented codebase. By that, the components are automatically updated by upgrading buildroot instead of having to manually upgrade each component.

2) Easy integration of new technologies: Buildroot has a workflow for integrating patches to any of its integrated software components. This facility enables the straightforward integration of patches adding support for new security technologies. Buildroot also supports different configurations. This can be used to test new extensions or target different platforms, such as Qemu, by using different patches or components overall.

3) Reduce the training period for new workers in the RISC-V space: We expand the documentation to provide a primer for the CVA6 chip. It provides a thorough explanation of the boot process, enabling new users to swiftly understand and customize the system to their needs. As we adhere to the infrastructure and interfaces provided by Buildroot, individuals with experience will quickly become familiar with the SDK.

V. Conclusion and Future Work

This work presents a path toward secure, energy-efficient chiplet-based edge AI in Europe. By combining a security-enhanced CVA6 core with specialized accelerators and a lightweight, intelligent interconnect, we demonstrate how a multi-chiplet system may outperform monolithic SoCs in terms of flexibility and efficiency. The supporting open-source AI software stack and enhanced SDK enable seamless model deployment and enforce security from hardware to runtime. Our real-time object-detection demonstrator validates the concept and prepares a first use case for the tapeout. Overall, this approach provides an open and scalable blueprint for advancing Europe's semiconductor autonomy.

References

[1] Bavarian Ministry of Economic Affairs, Regional Development and Energy, "Press release: Bavaria sets the course for the future of microelectronics." https://www.stmwi.bayern.de/presse/pressemeldungen/230-2025/, 2025.

[2] F. Zaruba and L. Benini, "The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 11, pp. 2629–2640, 11 2019.

[3] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th annual international symposium on computer architecture*, 2017, pp. 1–12.

[4] S. Markidis, S. W. Der Chien, E. Laure, I. B. Peng, and J. S. Vetter, "Nvidia tensor core programmability, performance & precision," in *2018 IEEE international parallel and distributed processing symposium workshops (IPDPSW)*. IEEE, 2018, pp. 522–531.

[5] H. Genc, S. Kim, A. Amid, A. Haj-Ali, V. Iyer, P. Prakash, J. Zhao, D. Grubb, H. Liew, H. Mao, A. Ou, C. Schmidt, S. Steffl, J. Wright, I. Stoica, J. Ragan-Kelley, K. Asanovic, B. Nikolic, and Y. S. Shao, "Gemmini: Enabling systematic deep-learning architecture evaluation via full-stack integration," in *Proceedings of the 58th Annual Design Automation Conference (DAC)*, 2021.

[6] T. Moreau, T. Chen, L. Vega, J. Roesch, E. Yan, L. Zheng, J. Fromm, Z. Jiang, L. Ceze, C. Guestrin *et al.*, "A hardware–software blueprint for flexible deep learning specialization," *IEEE Micro*, vol. 39, no. 5, pp. 8–16, 2019.

[7] J. Seo and J. Kong, "Versa: Versatile systolic array architecture for sparse and dense matrix multiplications," *Electronics*, vol. 13, no. 8, p. 1500, 2024.

[8] G. Gobieski, S. Ghosh, M. Heule, T. Mowry, T. Nowatzki, N. Beckmann, and B. Lucia, "Riptide: A programmable, energy-minimal dataflow compiler and architecture," in *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2022, pp. 546–564.

[9] N. Serafin, S. Ghosh, H. Desai, N. Beckmann, and B. Lucia, "Pipestitch: An energy-minimal dataflow architecture with lightweight threads," in *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 1409–1422. [Online]. Available: https://doi.org/10.1145/3613424.3614283

[10] X. Kong, Y. Huang, L. Chen, J. Zhu, L. Li, X. Man, M. Gao, S. Wei, and L. Liu, "Raccoon: Lightweight support for comprehensive control flows in reconfigurable spatial architectures," *IEEE Transactions on Parallel and Distributed Systems*, vol. 36, no. 6, pp. 1294–1310, 2025.

[11] J. Qin, T. Xia, C. Tan, J. Zhang, and S. Q. Zhang, "Picachu: Plug-in cgra handling upcoming nonlinear operations in llms," in *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ser. ASPLOS '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 845–861. [Online]. Available: https://doi.org/10.1145/3676641.3716013

[12] B. Yuan, J. Zhu, X. Man, Z. Ma, S. Yin, S. Wei, and L. Liu, "Dynamic-ii pipeline: Compiling loops with irregular branches on static-scheduling cgra," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 9, pp. 2929–2942, 2022.

[13] H. Cao, Z. Wu, D. Li, P. Jing, S. H. Pun, and Y. Liu, "Accelerating control flow on cgras via speculative iteration execution," *IEEE Computer Architecture Letters*, vol. 24, no. 1, pp. 109–112, 2025.

[14] Q. M. Nguyen and D. Sanchez, "Fifer: Practical acceleration of irregular applications on reconfigurable architectures," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 1064–1077. [Online]. Available: https://doi.org/10.1145/3466752.3480048

[15] T. Chen, T. Moreau, Z. Jiang, L. Zheng, E. Yan, M. Cowan, H. Shen, L. Wang, Y. Hu, L. Ceze, C. Guestrin, and A. Krishnamurthy, "Tvm: an automated end-to-end optimizing compiler for deep learning," in *Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'18. USA: USENIX Association, 2018, p. 579–594.

[16] M. Amine Hamdi, F. Daghero, G. Maria Sarda, J. Van Delm, A. Symons, L. Benini, M. Verhelst, D. Jahier Pagliari, and A. Burrello, "Match: Model-aware tvm-based compilation for heterogeneous edge devices," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 44, no. 10, pp. 3844–3857, 2025.

[17] IREE Project, "Iree documentation and developer guides," https://iree.dev, 2024.

[18] C. Lattner, M. Amini, U. Bondhugula, A. Cohen, A. Davis, J. Pienaar, R. Riddle, T. Shpeisman, N. Vasilache, and O. Zinenko, "MLIR: Scaling compiler infrastructure for domain specific computation," in *2021 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, 2021, pp. 2–14.

[19] C. Lattner and V. Adve, "Llvm: a compilation framework for lifelong program analysis & transformation," in *International Symposium on Code Generation and Optimization, 2004. CGO 2004.*, 2004, pp. 75–86.

[20] H.-I. C. Liu, M. Brehler, M. Ravishankar, N. Vasilache, B. Vanik, and S. Laurenzo, "Tinyiree: An ml execution environment for embedded systems from compilation to deployment," *IEEE Micro*, vol. 42, no. 5, pp. 9–16, 2022.

[21] Google Coral Team, "Introducing the coral npu: A full-stack platform for edge ai," https://developers.googleblog.com/en/introducing-coral-npu-a-full-stack-platform-for-edge-ai/, 2025.

[22] Y. Luo, C. Tan, N. B. Agostini, A. Li, A. Tumeo, N. Dave, and T. Geng, "Ml-cgra: An integrated compilation framework to enable efficient machine learning acceleration on cgras," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2023, pp. 1–6.

[23] D. Atienza, "An mlir-based compilation framework for cgra application deployment," in *Applied Reconfigurable Computing. Architectures, Tools, and Applications: 21st International Symposium, ARC 2025, Seville, Spain, April 9–11, 2025, Proceedings*, vol. 15594. Springer Nature, 2025, p. 33.

[24] S. Li, M. Xing, and Y. Wu, "Hdcc: A hierarchical dataflow-oriented cgra compiler for complex applications," in *Proceedings of the 30th Asia and South Pacific Design Automation Conference*, 2025, pp. 265–271.

[25] Y. S. Shao, J. Clemons, R. Venkatesan, B. Zimmer, M. Fojtik, N. Jiang, B. Keller, A. Klinefelter, N. Pinckney, P. Raina, S. G. Tell, Y. Zhang, W. J. Dally, J. Emer, C. T. Gray, B. Khailany, and S. W. Keckler, "Simba: Scaling deep-learning inference with multi-chip-module-based architecture," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-52. New York, NY, USA: Association for Computing Machinery, 2019, p. 14–27. [Online]. Available: https://doi.org/10.1145/3352460.3358302

[26] P. Scheffler, T. Benz, V. Potocnik, T. Fischer, L. Colagrande, N. Wistoff, Y. Zhang, L. Bertaccini, G. Ottavi, M. Eggimann, M. Cavalcante, G. Paulin, F. K. Gürkaynak, D. Rossi, and L. Benini, "Occamy: A 432-core dual-chiplet dual-hbm2e 768-dp-gflop/s risc-v system for 8-to-64-bit dense and sparse computing in 12-nm finfet," *IEEE Journal of Solid-State Circuits*, vol. 60, no. 4, p. 1324–1338, Apr. 2025. [Online]. Available: http://dx.doi.org/10.1109/JSSC.2025.3529249

[27] PULP Platform, "The pulp platform: Open risc-v multi-chip research," https://pulp-platform.org, 2023.

[28] *Universal Chiplet Interconnect Express (UCIe) Specification*, UCIe Consortium, 2025, revision 3.0, version 1.0.

[29] PULP Platform, "Pulp platform: Serial link," https://github.com/pulp-platform/serial_link, 2025.

[30] A. Deric, K. Mitard, S. Tajik, and D. Holcomb, "Evaluating vulnerability of chiplet-based systems to contactless probing techniques," 2024. [Online]. Available: https://arxiv.org/abs/2405.14821

[31] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[32] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[33] OpenHW Group, "cva6-sdk," https://github.com/openhwgroup/cva6-sdk.