PetaOps/W edge-AI μ Processors
Myth or reality?

Gomony, Manil Dev; de Putter, Floran ; Gebregiorgis, Anteneh ; Paulin, Gianna ; Mei, Linyan ; Jain, Vikram ; Hamdioui, Said; Bishnoi, Rajendra; Sanchez, Victor; More Authors

DOI
10.23919/DATE56975.2023.10136926

Publication date
2023

Document Version
Final published version

Published in
Proceedings of the 2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)

2023 Design, Automation & Test in Europe Conference (DATE 2023) – Multi-Partner Project

PetaOps/W edge-AI μProcessors: Myth or reality?

Manil Dev Gomony[1], Floran de Putter[1], Anteneh Gebregiorgis[4], Gianna Paulin[3], Linyan Mei[2], Vikram Jain[2], Said Hamdioui[4], Victor Sanchez[1], Tobias Grosser[5], Marc Geilen[1], Marian Verhelst[2], Friedemann Zenke[8], Frank Gurkaynak[3], Barry de Bruin[1], Sander Stuijk[1], Simon Davidson[9], Sayandip De[1], Mounir Ghogho[10], Alexandra Jimborean[11], Sherif Eissa[1], Luca Benini[3], Dimitrios Soudris[6], Rajendra Bishnoi[4], Sam Ainsworth[5], Federico Corradi[1], Ouassim Karrakchou[10], Tim Güneysu[7], and Henk Corporaal[1]

[1]Eindhoven University of Technology, [2]Katholieke Universiteit Leuven, [3]ETH Zurich, [4]Delft University of Technology, [5]University of Edinburgh, [6]National Technical University of Athens, [7]Ruhr-Universität Bochum, [8]Friedrich Miescher Institute, [9]The University of Manchester, [10]International University of Rabat, [11]University of Murcia

***Abstract*—With the rise of deep learning (DL), our world braces for artificial intelligence (AI) in every edge device, creating an urgent need for edge-AI SoCs. This SoC hardware needs to support high throughput, reliable and secure AI processing at ultra-low power (ULP), with a very short time to market. With its strong legacy in edge solutions and open processing platforms, the EU is well-positioned to become a leader in this SoC market. However, this requires AI edge processing to become at least 100 times more energy-efficient, while offering sufficient flexibility and scalability to deal with AI as a fast-moving target. Since the design space of these complex SoCs is huge, advanced tooling is needed to make their design tractable. The CONVOLVE project (currently in Inital stage) addresses these roadblocks. It takes a holistic approach with innovations at all levels of the design hierarchy. Starting with an overview of SOTA DL processing support and our project methodology, this paper presents 8 important design choices largely impacting the energy efficiency and flexibility of DL hardware. Finding good solutions is key to making smart-edge computing a reality.**

***Index Terms*—ULP, dynamic DL, edge-AI, SoC, memristor, approximate computing, DSE, compiler stack.**

I. Introduction

As the world braces for smart applications powered by AI in almost every edge device, there is an urgent need for ultra-low-power (ULP) edge-AI System-on-Chips (SoC) or Smart Edge Processor (SEP) that offloads the computing closer to the source of data generation to address the limitations (e.g., latency, bandwidth, privacy) of cloud computing. Based on current projections, the SEP market is expected to grow about 40% per year, beyond 70 Billion USD by 2026. In contrast to cloud computing, edge-AI hardware is far more energy-constrained. Hence, low-cost application-specific ULP SoCs are needed to make the edge intelligent. The strong ULP requirements can only be achieved by combining innovations from all levels of the design stack, from AI deep learning models, compilers, architecture, and micro-architecture, to circuits and devices, as we have proposed in our project *CONVOLVE* [1]. Innovations in CONVOLVE include ULP memristive circuits, exploiting Computing-in-Memory (CIM) and approximate computing, more advanced DL models, online learning, exploiting dynamism and reconfiguration at DL-, Architecture- and Circuit-levels, while rethinking the whole compiler stack. This results in extremely complex edge systems. Therefore, a single framework is needed that ties the innovations from the different levels together to fast design and design-space-exploration (DSE). Hence, we define the main objectives as follows: *(1) Achieve 100x improvement in energy efficiency compared to state-of-the-art COTS solutions. (2) Reduce design time by 10x to be able to quickly implement an ULP edge-AI processor combining innovations from the different levels of stack for a given application.* To understand how we can accomplish the key objectives, we present this paper with the following contributions:

- Summary of state-of-the-art low-power microprocessors for deep learning applications (Sec.II).
- CONVOLVE three-pillar design methodology that includes design-space exploration and compilation flow that reduces the design time by 10X (Sec.III).
- Key design choices to be considered for improving energy efficiency by a factor of 100X (Sec.IV).

II. State-of-the-art (SOTA) Edge-AI processing

Edge-AI applications require high-performance and flexible SoCs to efficiently map a diverse set of workloads. Heterogeneous multi-core SoCs can provide such duality by utilizing highly energy-efficient specialized hardware accelerators, possibly supporting different operand precisions. In order to judge the SotA for Edge-AI processing recent SoCs (including several from CONVOLVE partners) and ULP processing approaches are presented.

***TinyVers* - embedding MRAM**: TinyVers [2] (see Fig. 1) integrates a highly flexible-precision scalable digital accelerator, with a RISC-V core, a power management unit and an eMRAM, to provide a complete standalone edge-AI solution. The accelerator supports diverse AI layer types from Deep neural networks (NNs) (CNN, FC, TCN, GAN, AE) to traditional ML models like SVM at INT2/4/8 precisions. Fabricated in 22nm FDX, it provides 0.8-17 TOPS/W with power consumption ranging from 1.7 μW in deep sleep to sub-mW when running real AI workloads.

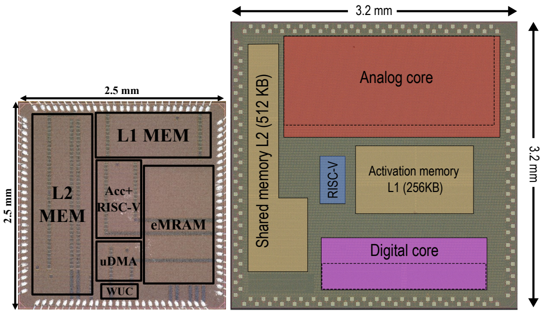

Figure 1. TinyVers (left) and DIANA (right) die micrographs.

***DIANA* - mixed-signal, mixed-precision**: DIANA [3] (Fig. 1) extends the idea of heterogeneity by combining an ULP analog in-memory core (AIMC) with a precision-scalable digital NN accelerator, an optimized shared-memory subsystem, and a RISC-V host processor to achieve SotA end-to-end inference at the edge. The SoC achieves peak energy efficiencies of 600 TOPS/W (7bit I, ternary W, 6bit

978-3-9819263-7-8/DATE23/© 2023 EDAA

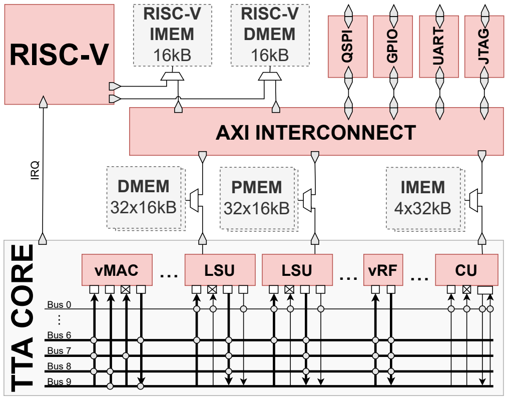

Figure 2. Block diagram of the BrainTTA SoC. The AXI-interconnect forms the border between the RISC-V host and the flexible TTA AI accelerator.

O) for the AIMC and 14 TOPS/W (8bit I/W/O) for the digital accelerator. When end-to-end ResNet20/CIFAR-10 and ResNet18/ImageNet classification workloads are mapped on the chip, 7 TOPS/W and 5.5 TOPS/W efficiencies are reported at system level respectively.

***BrainTTA* - Flexible AI support**: A popular method to achieve extremely high energy efficiency is to perform aggressive quantization i.e., reduce operand bit widths to as low as a single bit. However, this may not always be optimal in terms of accuracy. Thus, a typical edge-AI workload consists of various precision levels and layer dimensions. BrainTTA [4] is able to efficiently map various typical AI workloads, because of its inherent flexible datapath from the Transport-Triggered Architecture (TTA). As illustrated in Fig. 2, the SoC consists of a RISC-V processor and a TTA-based accelerator. The accelerator is fully-programmable and is supported by a C-compiler, which greatly simplifies mapping various AI (and other) workloads. BrainTTA, fabricated in 22nm FDX, has a peak energy efficiency of 29/15/2 TOPS/W (binary, ternary, and 8-bit precision) and a throughput of 614/307/77 GOPS.

Digital CIM, SRAM-based: Computing-in-Memory (CIM) has been proposed as a paradigm capable of overcoming the memory-wall problem of traditional computing architectures, where the input vector and weight matrix multiplication (i.e. MAC operations), is carried in the analog or digital domain within the memory sub-array. CIM can be realized using standard SRAM as well as emerging non-volatile memory. Digital MAC operation in SRAM-based CIM is performed by modifying the memory macro to add the required logic components such as a multiplier, shift logic, and accumulator unit in the periphery. A digital CIM [5], shown in Fig. 3, using 12T bitcell supporting wide-range dynamic voltage-frequency scaling (0.5V-0.9V) and flexible precision (4-b and 8-b) MAC operations has an area efficiency of 221 $TOPS/mm^2$ (4b), 55 $TOPS/mm^2$ (8b), and energy efficiency of 254 $TOPS/W$ (4b) and 63 $TOPS/W$ (8b).

Analog CIM, RRAM-based: Resistive memories store analog values in the form of resistances, however, the surrounding data communication remains digital. Quantization of analog output to digital data streams is done using an Analog-to-Digital Converter (ADC); it largely determines the overall efficiency of the architectures. NeuRRAM architecture, which is a 48-core RRAM-based CIM hardware, proposes a variable computation bit-precision while performing ADC at low power

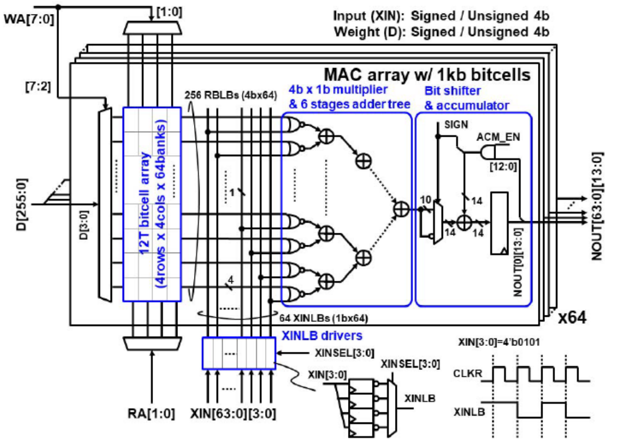

Figure 3. Overall architecture of SRAM-based Digital CIM macro. [5].

consumption and compact-area footprint, and achieves the energy efficiency of around 40 TOPS/W [6]. Furthermore, a 195.7 TOPS/W is reported using RRAM-based CIM macro supporting 8b-input and 8b-weight MAC operations as shown in Fig. 4 [7]. This architecture includes an asymmetric group-modulated input scheme to reduce the computing latency as well as a weighted current-to-voltage signal stacking converter for the MAC operations.

***Kraken*, SoC with SNN and ANN accelerators**: Kraken [8] (Fig. 5) is an example for an ultra-low-power heterogeneous SoC fabricated in $22\,\mathrm{nm}$ and combines a 32-bit RISC-V host core, $1\,\mathrm{MiB}$ of scratchpad L2 SRAM memory, and an autonomous I/O subsystem with three programmable, power-gateable accelerators: (1) A $1.8\,\mathrm{TOp/s/W}$ parallel general-purpose compute cluster with 8 RISC-V cores sharing $128\,\mathrm{KiB}$ of L1 scratchpad memory. The RISC-V cores support hardware loops, SIMD sub-byte dot-product integer operations with mixed-precision capabilities, MAC with concurrent data load (MAC-LD), and floating-point capabilities for energy-efficient digital signal processing. (2) $1.1\,\mathrm{TSyOp/s/W}$ accelerator called *Sparse Neural Engine (SNE)* targets spiking convolutional layers with 4-bit 3×3 filter and 8-bit leaky-integrate and fire (LIF) neuron states. (3) *Completely Unrolled Ternary Inference Engine (CUTIE)* [9] is a $1036\,\mathrm{TOp/s/W}$ Ternary Neural Network (TNN) accelerator.

***µBrain/THOR*, Digital SNN**: A fundamentally different approach to improving energy efficiency is one of *neuromorphic* devices, which takes inspiration from the brain and research spiking neural networks (SNNs) both at the algorithmic and the hardware implementation fronts. A key difference between artificial neural networks (ANNs) and SNNs is the stateful nature of spiking neurons, compared to the statefulness of the ReLU functions and the fact that SNNs communicate by

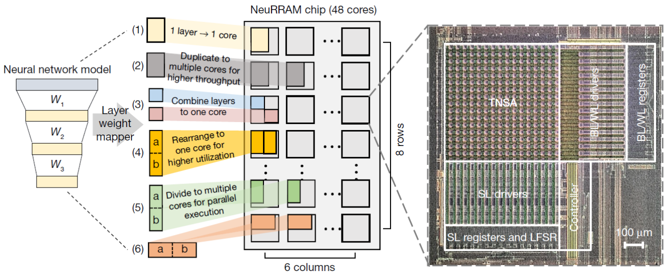

Figure 4. RRAM-based CIM chip and its NeuRRAM embedding [6].

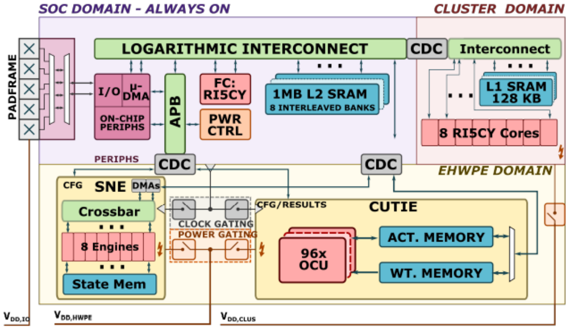

Figure 5. High-level architecture of Kraken SoC [8].

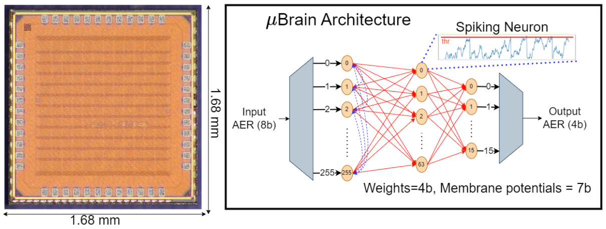

Figure 6. μBrain die micrograph (left) and network architecture (right) [10].

passing a 1-bit message or spike, thus, resulting in sparse operation. Good examples of Digital SNNs are μBrain [10] (Fig. 6 and THOR [11]. μBrain addresses extreme low power while THOR aims at higher performance.

Summary: Fig. 7 shows the energy efficiency of many neuromorphic chips, including above ones. We conclude: 1) ANN chips are more mature, approaching an energy-efficiency close to 1 fJ/Op; however this requires complete unrolling (CUTIE) or AIMC (DIANA); both have their issues (see Sec. IV); 2) Although SNNs have high potential, and our brain as 'proof-of-concept', their chips are (only) in the pJ/Op range; 3) Flexibility has its price: SoCs like BrainTTA suffer at least one-order in energy efficiency. We conclude that CONVOLVE should 1) beat the fJ/Op barrier, while being flexible enough to deal with future NNs, and 2) unleash the potential of SNNs.

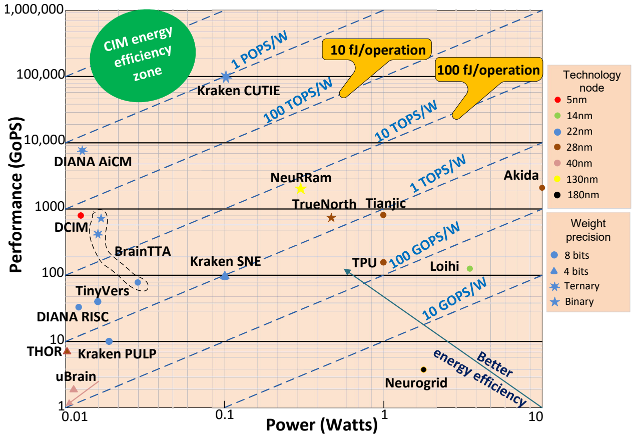

Figure 7. Energy-efficiency of SOTA edge-AI μProcessors.

III. CONVOLVE METHODOLOGY

CONVOLVE (convolve.eu) proposes a novel three-pillar design methodology, on which relies its four key objectives: *(1) Achieve 100X energy efficiency, (2) Reduce design time by 10X, (3) Guarantee hardware security and reliability and (4) Enable smart edge applications*, as shown in Fig. 8. The first pillar: *ULP blocks* focuses on exploitation of different hardware acceleration possibilities at microarchitecture, circuit and device level; the second pillar *Smart and dynamic application models* focuses on capturing the dynamic application behaviour efficiently; and the third pillar *Architecture and design tools* efficiently bridges the first two pillars by generating system architecture and mapping of application models to the vast amount of hardware acceleration possibilities. Each pillar covers different levels of the design stack leading to various design choices, discussed in Sec. IV.

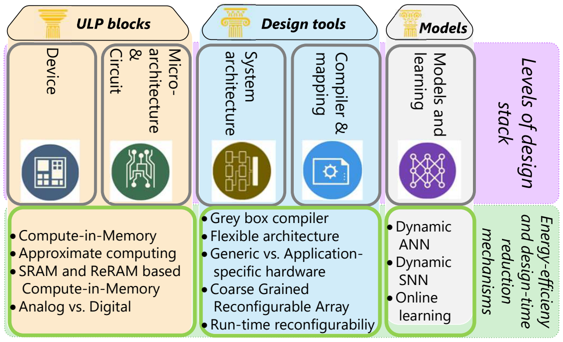

Figure 8. Three-pillar design methodology of CONVOLVE.

CONVOLVE's design flow starts from a given application suite, selects in a very short time the optimal SoC configuration, implements and verifies it, and compile algorithms for the generated hardware, as shown in Fig. 9. The goal is to automatically generate the optimal processing system for any given edge-AI application, based on ULP building blocks and their code generation, including the building blocks for hardware security. The application use-cases and scenarios are analyzed for understanding application dynamism that will define the dynamic NN model and the learning strategies needed. An efficient, transparent and security-aware compilation flow built within the MLIR [12] framework will be used to generate code for the heterogeneous set of ULP building blocks. A fully automated framework for DSE and hardware generation will be developed based on the ZigZag ML performance estimation model [13]. The DSE framework uses as input performance models at Core- and SoC-level. *(1) Core-Level Modeling:* We model each accelerator's architecture, including its run-time configurability, which enables rapid cost estimation for the design space of mapping a wide range of ML workloads onto each individual accelerator. In this way, the vast combinatorial space of hardware, algorithm, and mapping can be separately/jointly explored in a fast manner. *(2) SoC-Level Modeling:* At this level, we model and estimate the cost of end-to-end mapping one or multiple ML workloads onto the SoC system. This requires modeling not only each core/accelerator's own attributes and mapping each operator of an ML algorithm one at a time, but also the interconnection between the cores/accelerators and fine-grained data dependencies between operators. This allows early-stage DSE of graph lowering and optimization, workload-core allocation, and inter-/intra-core scheduling in parallel with workload, HW and compiler development, and give feedback to other stages.

IV. CONVOLVE DESIGN SPACE

The design space for edge-AI systems is huge; we treat 8 key design choices, partitioned over the 3 CONVOLVE pillars.

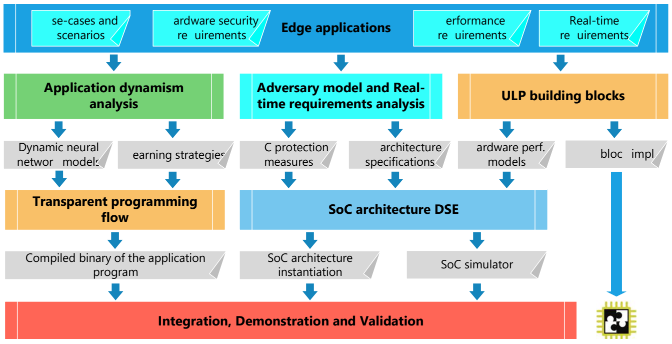
se-cases and scenarios
ardware security re uirements
Edge applications
erformance re uirements
Real-time re uirements
Application dynamism analysis
Adversary model and Real-time requirements analysis
ULP building blocks
Dynamic neural networ models
earning strategies
C protection measures
architecture specifications
ardware perf. models
bloc impl
Transparent programming flow
SoC architecture DSE
Compiled binary of the application program
SoC architecture instantiation
SoC simulator
Integration, Demonstration and Validation

Figure 9. CONVOLVE's fast design flow showing different steps in generation of fully integrated and optimized SoC from application specification.

A. Pillar: Smart and dynamic application models

ANN vs SNN: Many edge applications require constant monitoring of data streams on a tiny power budget; e.g., acoustic scene analysis, speech denoising, and keyword spotting. Current solutions typically use ANNs, namely recurrent neural networks (RNNs), that process their input frame by frame. This is very inefficient because it does not exploit input sparsity, thereby disabling edge-based computing. Use of the cloud has it's own problems, like added latency, energy cost and privacy issues; solutions are urgently needed.

Compared to RNNs, brain-inspired SNNs are still relatively immature. However, they offer compelling features. In SNNs, neurons communicate through events, so-called spikes, that are sparse in time [14]. This increases the information content of each message passing between neurons, promising more energy-efficient computation and communication: no inputs, no spike propagation. This event-driven model may be a better fit for the continuous processing of sparse real-world data.

Beyond the sparsity of network activity, the connectivity between layers of neurons can itself be sparse, further reducing the memory footprint required to store the connectivity matrix and impacting silicon area and system cost as well as the energy cost of moving data. This structural sparsity can also be the target of biologically-inspired learning rules, termed *structural plasticity*, permitting the creation and destruction of inter-neural connections. This is again a significant area of research in the project. From an implementation point of view, managing sparse connectivity matrices is challenging. Its hardware support is a major area for future innovations in the CONVOLVE project that emphasizes on the co-design of models and accelerators and seeks opportunities to combine the best of the ANN and SNN worlds.

Online vs Offline learning: Stateful networks like RNNs and SNNs require back-propagation through time (BPTT) to achieve high accuracy. However, BPTT is extremely costly. Recent innovations allow SNNs to be optimized without BPTT [15], thereby enabling end-to-end training on otherwise prohibitively long sequences and paving the way for real-time on-chip learning.

Such online continual learning is essential for edge-AI systems allowing them to seamlessly adapt to different users, environments, or task requirements. Online adaptation may even bestow self-healing by providing robustness to component or sensor drift over the system's lifetime. To address these points, within CONVOLVE we will develop new algorithms for online self-supervised learning in continuous time that dispense with or at least minimize back-propagation requirements through time and space, and are robust against catastrophic forgetting.

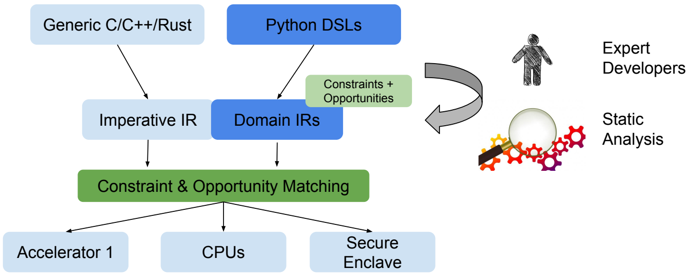

Figure 10. CONVOLVE gray-box compiler uses a semi-automatic compilation flow where static-analyses and expert developers collaborate to obtain peak-performance when running large NNs on HW accelerators.

Static vs. Dynamic NNs: Many NN models have been developed and have demonstrated excellent performance in many domains. However, they are often extremely compute demanding. Approaches to reducing their processing complexity include model compression and response approximation. They reduce the model size by injecting sparsity, adding collaborative layers or designing tiny architectures. Recently, dynamic neural networks (DynNNs) were introduced to make the processing complexity at the inference stage input-dependent. The idea behind DynNNs is borrowed from biological NNs which adapt the neural pathways to the stimulus in order to speed up decision-making.

The most straightforward implementation of DynNNs is through *early exit* [16]. It applies internal classifiers to make quick decisions for easy inputs, without using the full-fledged network. A response is returned if the internal classifier is sufficiently confident; otherwise, the example is passed to a subsequent internal classifier. Other studies made input dependence possible through *attention mechanisms* which allow focusing on the most important parts of the input data: 1) gating functions that remove the least salient components (eg. channels of an image); 2) runtime parameter adaptation that aims at altering the architecture's intrinsic characteristics (e.g., network width or depth) given the input's features; and 3) dynamic activation functions that activate neurons according to the relevance of the input, thus increasing the representation power of models. For a review of DynNNs see [16].

CONVOLVE develops efficient DynNNs for smart low-resource processors, extend the multi-criterion network architecture search to include DynNNs, and adapt recent compression techniques, like pruning-at-initialisation [17].

B. Pillar: Architecture and design tools

Black-box vs. Grey-box compiler: To effectively map complex NNs to the heterogeneous CONVOLVE hardware, our compiler must scale to large applications while ensuring a code quality that matches handwritten kernels developed by domain-expert programmers. While typical black-box compilers such as LLVM [18] offer a generic performance baseline, NNs are increasingly targeted via domain-specific frameworks such as MLIR [12] or TVM [19]. These can significantly improve performance and targetability by rigidly *adding domain knowledge within a fixed-function stack*. However, their overspecialization hurts targetability of new applications to new hardware.

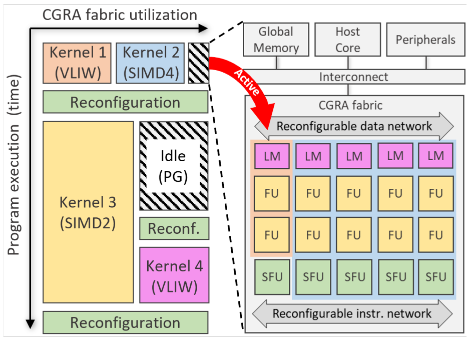

Figure 11. CONVOLVE CGRA fabric executes multiple kernels in parallel while reconfiguring the fabric on a kernel-level granularity.

The CONVOLVE compiler (Fig. 10) extends MLIR to offer a generic *grey-box* approach: a novel theory of *constraints* and *opportunities* will guide static analysers and expert developers to symbiotically work towards optimal hardware mappings in support of the fast-evolving CONVOLVE hardware ecosystem, by embedding knowledge throughout program transformation, while preserving key invariants.

Based on powerful static analysis, exploiting the *opportunities* exposed by the applications' characteristics, such as the degree of parallelism, reduced precision, code layout, algorithmic structure, or a limited input domain, and satisfying the *constraints* imposed by the execution environment, like latency requirements, timing and security guarantees, the accuracy of the target, or performance penalties of a complex control flow, the compiler customizes the application and automatically generates optimized code, tailored for the target architecture. In addition, we offer domain-expert developers efficient access to the compiler internals to specialize code optimization and application-to-accelerator mapping for each use case.

Application specific HW, Static vs. Dynamic: DL is a fast-moving field, with rapidly changing network design and optimization parameters, like pruning and quantization strategy, number and types of network layers, data reuse, and exploited network dynamism. FPGAs are very flexible, but have high area, energy and performance penalties. On top, their SoC integration is rather difficult. GPUs offer a more cost-efficient alternative, however, their somewhat hardwired datapath limits accelerator utilization and hence they deliver sub-optimal performance for certain kernels. Static architectures with fixed function accelerators offer high efficiency, but can not offer the required flexibility.

A Coarse-Grained Reconfigurable Architecture (CGRA) forms an interesting compromise. It consists of an array of processing elements or functional units (FUs) that are interconnected through a (reconfigurable) switching fabric or network-on-chip (NoC). Similar to FPGAs, there are (coarse-grained) specialized DSP blocks and local memories (LM) to increase the energy efficiency, but its reconfiguration overhead is much lower. CGRAs cover a wide range of reconfigurable architectures, as shown in [20].

CONVOLVE uses the Blocks CGRA template [21]; see Fig. 11. Its template consists of a reconfigurable instruction and data network with programmable FUs. The physical fabric can be configured to execute multiple application-specific processors (i.e., virtual cores) in parallel. These virtual processors execute multiple operations/cycle, like in VLIWs. In addtion Blocks supports flexible SIMD execution by broadcasting the same instruction to multiple FUs over the reconfigurable instruction network. Complex instructions are supported via specialized FUs (SFUs). To accommodate changing workloads, the fabric can be (partially) reconfigured on a kernel-level granularity with a tolerable reconfiguration penalty [22]. The fabric contains support for zero-overhead loops to enable *spatial computation* where the instruction stream remains static. Leakage power of unused FUs is taken care off by advanced power management.

To summarize, the use of CGRAs has several advantages: (1) they support highly parallel calculations, (2) have good area and energy efficiency, due to coarse granularity, (3) energy-efficiency is increased due to the static interconnect (configured per kernel) and spatial computation, (4) and they are flexible, supporting the fast-moving NN area. Its reconfigurability allows extension towards DynNN support.

C. Pillar: ULP blocks

Analog vs Digital computing: Current ANN/SNN compute architectures can be classified into whether they can be implemented using fully synthesizable logic using standard-cell library (i.e., all-digital) and custom-designed using analog/mixed-signal design techniques. Despite SotA analog and mixed-signal architectures offer the lowest energy consumption, digital implementations dominate the commercial solutions due severe limitations of analog/mixed-signal: 1) poor reliability as their performance is easily affected by noise induced by process, temperature, and voltage variations and hence require a complex calibration process; 2) poor technology portability as they need to be re-designed when porting the design to a different technology node; and 3) poor scalability as larger designs cannot be easily built using powerful design automation tools available for digital designs. To tackle these limitations, innovating across the entire design hierarchy is essential. As shown in [6], from a reconfigurable dataflow architecture, an energy- and area-efficient voltage-mode neuron circuit, to a series of algorithm-hardware co-optimization techniques needs to be applied to demonstrate a simultaneous improvement in efficiency, flexibility, and accuracy of analog/mixed-signal designs.

In-memory vs. Classical computing: Moore's law has enabled the traditional multi-level cache based CPUs to deliver better performance for successive generations. However, the big distance between main memory and compute units leads to high energy consumption and calls for a different approach: Computing-in-Memory.

CIM integrates computation and storage of data within the same physical location. CIM can be realized using different emerging memristive technologies such as Resistive Random Access Memory (RRAM), Phase Changing Memory (PCM) and Magnetic RAM (MRAM) as well as conventional memory technologies such as SRAM, DRAM and Ferroelectric FETs. In-memory computing using emerging memrisitve devices benefits from their non-volatile nature and their practically *zero leakage* compared to their conventional memory technology counterparts. Table I presents a qualitative comparison of traditional CPUs and CIMs using different memory technologies. CONVOLVE explores ultra-low power implementation of domain-specific analog and digital CIM flavors using different memory technologies as well as Coarse-Grained Reconfigurable Arrays (CGRA).

Table I
DESIGN METRICS FOR VON-NEUMANN AND CIM ARCHITECTURES USING VARIOUS MEMORY TECHNOLOGIES (DATA OBTAINED FROM [23])

Comparison metric	Conventional CPU	CIM digital SRAM	CIM analog SRAM	CIM analog memristive
Technology	CMOS	CMOS	CMOS	RRAM
Architecture	von-Neumann	non von-Neumann	non von-Neumann	non von-Neumann
Operation mode	Digital	Digital	Analog	Analog
Design effort	Easy	Easy	Complex	Complex
Resolution	High	High	Low	Low
Volatility	Yes	Yes	Yes	No
Endurance	High	High	High	Low
Scalability	medium	medium	medium	high
Write energy	~fJ	~fJ	~fJ	~pJ
Write latency	~1ns	~1ns	1ns	~10ns
Compute energy efficiency	~GOPS/W	10's GOPS/W	~TOPS/W	~TOPS/W

Exact vs. Approximate computation: Diminishing energy-efficiency gains from semiconductor scaling as per Moore's law and continued increase in compute requirements, as evident from the latest NNs, like GPT3 and Transformers, force researchers to look for newer computing paradigms. *Approximate Computing* (AxC), which trades off accuracy for improved energy efficiency, emerges as a potential alternative owing to the error-resilient characteristics of modern ML workloads. Most AxC techniques can be classified into three broad categories: 1) timing approximation, wherein the circuit is operated at a lower supply voltage without reducing the corresponding operational frequency, resulting in efficiency improvements for added timing errors, 2) functional approximation, wherein the logic functionality of the circuit is modified to trade off quality for added efficiency, e.g., netlist modification, boolean rewriting, precision-scaling, etc. and 3) cross-level approximation where approximation knobs at logic-, structural- and physical level are leveraged.

AxC has been widely adopted for ML. Among all the different approximations being investigated for DL, *precision-scaling* has emerged as a success story. DL models often tolerate very aggressive precision scaling. It provides very high gains by reducing both compute as well as off-chip traffic in memory. For training, different data formats like 16-bit floats (FP16), BFloat, DLFloat, etc. have been adopted for activation and weights. For inference, varying fixed-point formats are adopted, scaling down to ternary or binary bit widths. Accuracy lost due to approximations are regained using quantization-aware training (QAT); see [24] for a survey.

Another widely adopted approximation technique is *pruning* which forces weight values in a NN to zero, thereby introducing sparsity. Studies have proposed the combination of weight pruning with precision scaling to achieve higher energy efficiency for NN inference. Pruning introduces irregularities in compute and memory access patterns; hence, specialized architectures with sparsity support have been proposed. Motivated by the promising returns proposed by the close synergy of AxC and ML, CONVOLVE aims to explore novel AxC techniques to obtain extremely energy-efficient edge-AI μprocessors.

V. SUMMARY

CONVOLVE aims at 100X improvement in energy efficiency and 10X in design-time. We outlined the SotA for edge-AI processing, the CONVOLVE automated design flow, and treated 8 important design choices supporting above goals. Major conclusions are: 1) In the short term, ANNs are to be favored above SNNs; 2) Although SNNs have favorable properties, inspired by our brain, they require more research; 3) Online learning requires rethinking back-propagation through time and space; 4) Dynamic NNs will become dominant in low-power edge computing; 5) Quick and easy adaptation requires a Grey-box compiler that can deal with new accelerators comprehensively; 5) Architectures should support more dynamism, e.g., by using 2-step code generation (i.e., Code → Virtual cores → Physical architecture); 6) The ML field is moving quickly and therefore requires flexible architectures; 7) CIM is promising but requires adapting the memory periphery; 8) Analog computing has a high potential for energy savings but is too inflexible in the short term; since NNs do not fit spatially yet, time-sharing and reconfiguration flexibility are key. CONVOLVE aims at unleasing above potential by a holistic approach, rethinking the full design stack.

ACKNOWLEDGMENT

This work is funded by EU's Horizon Europe research and innovation programme under grant agreement No. 101070374.

REFERENCES

[1] M. Gomony *et al.*, "CONVOLVE: Smart and seamless design of smart edge processors," *preprint arXiv:2211.*, 2022.

[2] V. Jain *et al.*, "TinyVers: A 0.8-17 TOPS/W, 1.7 uW-20 mW, Tiny Versatile System-on-chip with State-Retentive eMRAM for Machine Learning Inference at the Extreme Edge," in *VLSI Symp.*, 2022.

[3] K. Ueyoshi *et al.*, "DIANA: An End-to-End Energy-Efficient Digital and ANAlog Hybrid Neural Network SoC," in *ISSCC*, vol. 65, 2022.

[4] M. Molendijk *et al.*, "BrainTTA: A 35 fJ/op Compiler Programmable Mixed-Precision Transport-Triggered NN SoC," *preprint arXiv:2211.11331*, 2022.

[5] H. Fujiwara *et al.*, "A 5-nm 254-TOPS/W 221-TOPS/mm2 Fully-Digital Computing-in-Memory Macro Supporting Wide-Range Dynamic-Voltage-Frequency Scaling and Simultaneous MAC and Write Operations," in *ISSCC*, vol. 65, 2022.

[6] W. Wan *et al.*, "A compute-in-memory chip based on resistive random-access memory," *Nature*, vol. 608, no. 7923, pp. 504–512, 2022.

[7] C.-X. Xue *et al.*, "16.1 a 22nm 4mb 8b-precision reram computing-in-memory macro with 11.91 to 195.7 tops/w for tiny ai edge devices," in *ISSCC*, vol. 64. IEEE, 2021.

[8] A. Di Mauro *et al.*, "Kraken: A direct event/frame-based multi-sensor fusion soc for ultra-efficient visual processing in nano-uavs," in *2022 IEEE Hot Chips 34 Symposium (HCS)*, 2022, pp. 1–19.

[9] M. Scherer *et al.*, "A 1036 TOp/s/W, 12.2 mW, 2.72 μJ/Inference All Digital TNN Accelerator in 22 nm FDX Technology for TinyML Applications," in *COOL CHIPS*. IEEE, 2022.

[10] J. Stuijt *et al.*, "μbrain: An event-driven and fully synthesizable architecture for spiking neural networks," *Frontiers in neuroscience*, 2021.

[11] M. Senapati *et al.*, "THOR - A Neuromorphic Processor with 7.29G $TSOP^2/mm^2Js$ Energy-Throughput Efficiency," *preprint arXiv:2211.*, 2022.

[12] C. Lattner *et al.*, "Mlir: A compiler infrastructure for the end of moore's law," *arXiv preprint arXiv:2002.11054*, 2020.

[13] L. Mei *et al.*, "ZigZag: Enlarging joint architecture-mapping design space exploration for dnn accelerators," *IEEE TC*, 2021.

[14] T. Hoefler *et al.*, "Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks," *arXiv:2102.00554 [cs]*, Jan. 2021.

[15] F. Zenke *et al.*, "Brain-Inspired Learning on Neuromorphic Substrates," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 935–950, May 2021.

[16] Y. Han *et al.*, "Dynamic neural networks: A survey," *IEEE TPAMI*, 2022.

[17] H. Wang *et al.*, "Recent advances on neural network pruning at initialization," in *Thirty-First International Joint Conference on Artificial Intelligence*, 2022, p. 5638–5645.

[18] C. Lattner *et al.*, "Llvm: A compilation framework for lifelong program analysis & transformation," in *Symp. on CGO*. IEEE, 2004.

[19] T. Chen *et al.*, "TVM: An automated End-to-End optimizing compiler for deep learning," in *OSDI 18*, 2018.

[20] M. Wijtvliet *et al.*, "Coarse grained reconfigurable architectures in the past 25 years: Overview and classification," in *SAMOS*, 2016.

[21] ——, "Blocks: Challenging SIMDs and VLIWs With a Reconfigurable Architecture," *IEEE TCAD*, 2022.

[22] B. de Bruin *et al.*, "Multi-level optimization of an ultra-low power brainwave system for non-convulsive seizure detection," *TBioCAS*, 2021.

[23] S. Salahuddin *et al.*, "The era of hyper-scaling in electronics," *Nature Electronics*, vol. 1, no. 8, pp. 442–450, 2018.

[24] A. Reuther *et al.*, "Ai accelerator survey and trends," in *2021 IEEE High Performance Extreme Computing Conference (HPEC)*, 2021, pp. 1–9.