CONVOLVE: Smart and seamless design of smart edge processors

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[10], 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]The University of Manchester
[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 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 addresses these roadblocks. It takes a holistic approach with innovations at all levels of design hierarchy. Starting with an overview of SOTA DL processing support and our project methodology, this paper presents 8 important design choices largely impacting energy-efficiency and flexibility of DL hardware. Finding good solutions is key in making smart-edge computing 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) of cloud or centralized computing. Based on the 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, micro-architecture, to circuits and devices. Innovations include ULP memristive circuits, exploiting Compute-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-VI).

II. State-of-the-art 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,

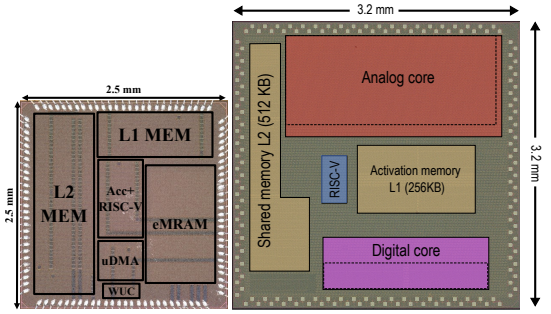

Figure 1. TinyVers (left) and DIANA (right) die micrographs.

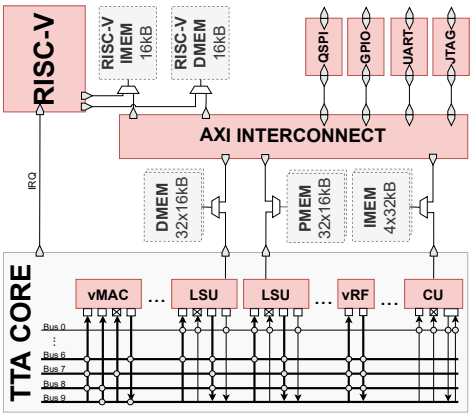

Figure 2. Block diagram of the BrainTTA SoC. The AXI-interconnect forms the border between the RISC-V host and the flexible TTA AI accelerator.

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 [1] (Fig. 1) integrates a highly flexible-precision scalable digital accelerator, with a single core RISC-V processor, a power management unit and an eMRAM, to provide a complete standalone edgeAI solution. The accelerator supports diverse AI layer types from DNN (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.

***DIANA* - mixed-signal, mixed-precision**: DIANA [2] (see Fig. 1) extends the idea of heterogeneity by combining an ULP analog in-memory core 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 O) for the AIMC core 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 [ref] 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.

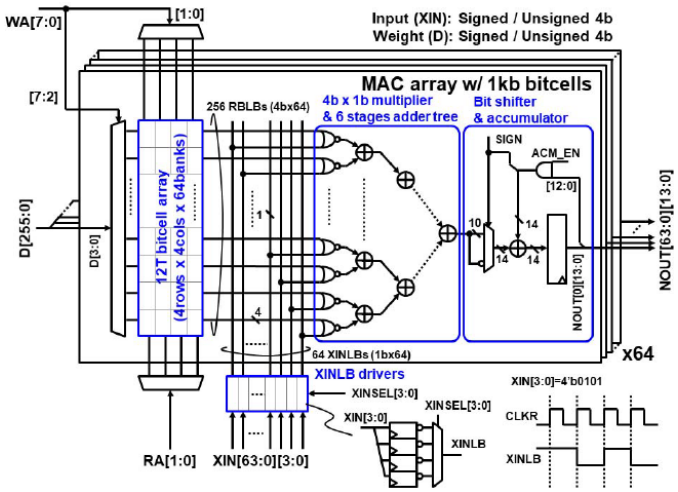

Figure 3. Overall architecture of Digital CIM Macro [4].

The input vector and weight matrix multiplication (i.e. MAC operations), is carried in the analog or digital domain within the memory sub-array. This leads to significant improvements in throughput and energy efficiency. CIM can be realized using standard SRAM as well as emerging non-volatile memory. SRAM-based CIM provides faster write speed and lower write energy with almost unlimited endurance [3]. Digital MAC operation in SRAM-based CIM is performed by modifying the memory macro to add the required logic components such as multiplier, shift logic and accumulator unit in the periphery. A digital CIM [4], 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 [5]–[9]. 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 [10], [11]. An RRAM-based CIM macro with voltage-regulating current sense topology is proposed to improve the area and energy efficiency (27 TOPS/W) of the ADC design [12]. Moreover, NeuRRAM architecture, which is a 48-core RRAM based CIM hardware, proposes a variable computation bit-precision while performing ADC at low power consumption and compact-area footprint, and achieves the energy efficiency of around 40 TOPS/W [13]. Furthermore, a 195.7 TOPS/W is reported using RRAM-based CIM macro supporting a 8b-input and 8b-weight MAC operations [14]. 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 [15] (Figure 4) is an example for an ultra-low-power heterogeneous SoC fabricated in 22 nm and combines a 32-bit RISC-V host core, 1 MiB of scratchpad L2 SRAM memory, and an autonomous I/O subsystem with three programmable, power-gateable accelerators: (1) A 1.8 TOp/s/W parallel general-purpose compute cluster with 8 RISC-V cores sharing 128 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 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)* [16] is a 1036 TOp/s/W Ternary Neural Network (TNN) accelerator.

***µBrain*, 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 both at the algorithmic and the hardware implementation fronts. A key difference between 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 passing a 1-bit message or spike, thus, resulting in sparse operation. Recent trends in neural network hardware are mainly three (µBrain die micrograph (left) and its fixed network architecture (right) [17]) 1) CMOS-based neuromorphic SNN 2) traditional ANN accelerators 3) non-volatile-memory-based accelerators.

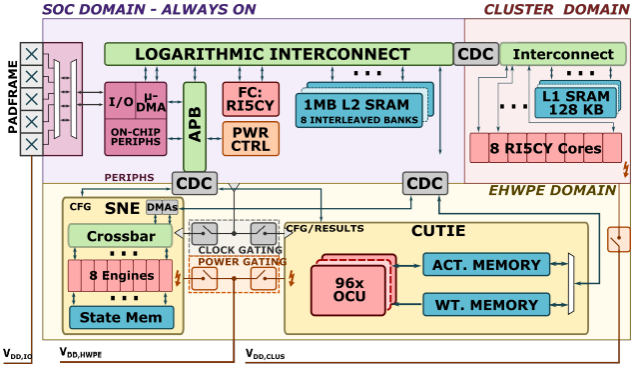

Figure 4. High-level architecture of Kraken SoC.

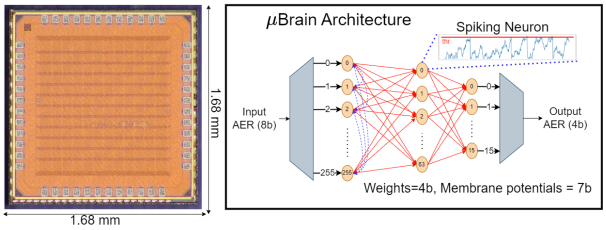

Figure 5. µBrain die micrograph (left) and its fixed network architecture (right) [17].

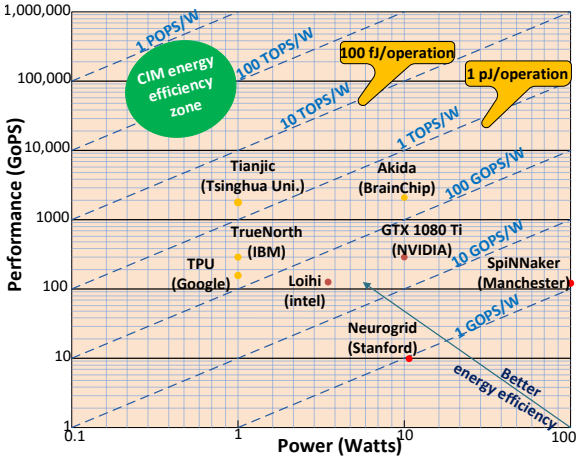

Figure 6. Energy-efficiency of various neuromorphic chips and CIM potentials [18], [19].

Summary: As shown in Figure 6 the energy-efficiency of existing neuromorphic chips is limited (few pJ/operation) [17]–[19].

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 Figure 7. The first pillar: *ULP building 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 *Compositional and fast design flow* 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.

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. 8. The goal is to automatically generate the optimal processing system for any given edge AI application, based on the 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 neural network model and the learning strategies needed. An efficient, transparent and security-aware compilation flow built within the MLIR [20] framework will be used to generate code for the heterogeneous set of ULP building blocks. MLIR is an industry-supported framework for optimizing programs by incremental lowering through a multitude of domain-specific IRs. A fully automated framework for DSE and hardware generation will be based on the ZigZag ML performance estimation model [21]. The DSE framework uses as input performance models at two different levels: Core-level and SoC-level.

Core-Level Modeling: At the single-ULP-processing-core level, 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. This is beneficial for 1) hardware designers aiming at optimizing accelerators' design-time/run-time architectural parameters, 2) algorithm developers working towards constructing hardware-friendly/compatible artificial neural networks' topologies, and 3) compiler builders aspiring to design flexible compilation flows that can easily be adapted for evolving ML algorithms, as well as new ML accelerators, optimally mapping between the two.

SoC-Level Modeling: New trends of larger and more varied ML workloads require more performant and flexible hardware platforms. Both homogeneous and heterogeneous multi-core/-accelerator SoCs for ML are becoming ever more popular in recent years. At the SoC level, we will model and estimate the cost of end-to-end mapping one or multiple

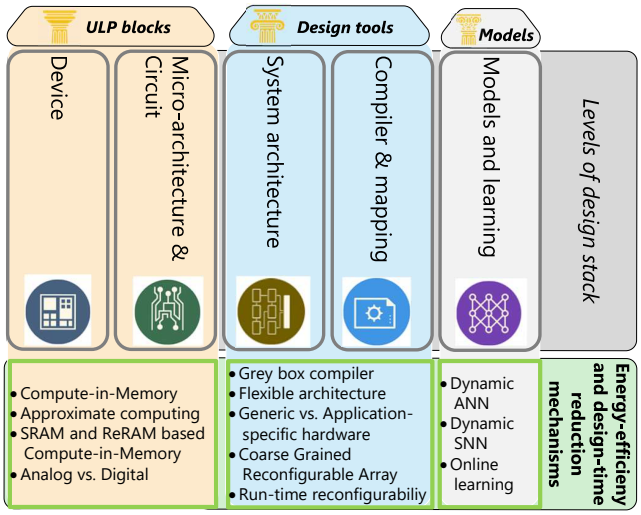

Figure 7. Three-pillar design methodology of CONVOLVE to tackle energy efficiency and design time reduction objectives.

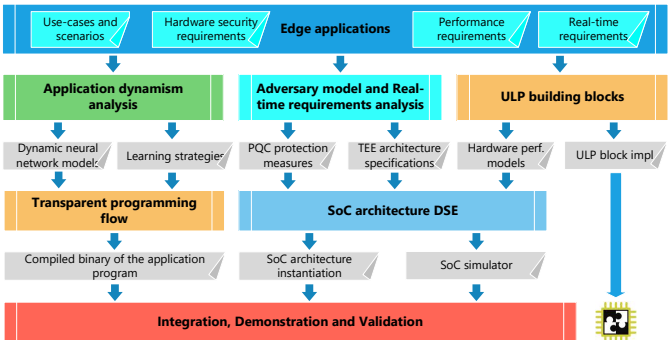

Figure 8. CONVOLVE compositional and fast design flow: The different steps for the generation of fully integrated optimized SoC architecture from the application specification.

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. In this way, early-phase design space exploration of graph lowering and optimization, workload-core allocation, and inter-/intra-core scheduling can be performed in parallel with workload, HW and compiler development, and provide early feedback to other stages.

IV. CONVOLVE DESIGN SPACE

A. Pillar: Smart and dynamic application models

ANN vs SNN + Online learning vs Offline learning Many edge applications require constant monitoring of data streams on a tiny power budget. Examples include acoustic scene analysis, speech denoising, and keyword spotting. Furthermore, most current solutions rely on recurrent neural networks (RNNs) that process their input frame by frame. This frame-based approach is inefficient for these applications because it does not use sparsity in the input. The limited computational capabilities of low-power edge devices require much of the processing in the cloud. However, non-local cloud processing results in increased energy costs and latency due to data movement. Finally, additional security and data privacy risks may be inherent to information transmission that could remain local provided sufficient compute power to process it on-device. These limitations call for innovating toward smart dynamic application models.

Compared to RNNs, brain-inspired spiking neural networks (SNNs) are still a relatively immature technology. However, they may offer compelling long-term solution to above problems. In SNNs, neurons communicate through stereotypical events, so-called spikes that are often sparse in time. This sparsity of activity is also referred to as *ephemeral sparsity* [22]. The sparseness of activity increases the information content of each message passing between neurons, promising more energy-efficient computation and communication: When there is no input, no spikes propagate through the network, reducing memory access and thus saving energy. This event-

driven processing model used by the brain may be a better fit for the continuous processing of sparse real-world data.

However, SNNs are not a mature technology, and much remains to be done. There are no well-established learning strategies to rival the success of back-propagation through time (BPTT) in conventional RNNs. Until recently, the inability to readily define a differentiable error function for spiking neurons posed a major conceptual obstacle. Today, new methods sidestep this problem by introducing practical surrogate gradients [23]. Combined with other innovations, SNNs can now be optimized without BPTT [24]–[26], thereby enabling end-to-end training on otherwise prohibitively long sequences and paving the way for real-time on-chip learning. Combined with bio-inspired neuronal mechanisms such as heterogeneous adaption [27], [28], SNN task performance is becoming increasingly competitive with RNNs. Capitalizing on the merits of this active research area will be a significant focus of CONVOLVE.

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. Much of the performance gains in recent hardware for neural networks have come from parallelism, typified by the graphics processing unit (GPU) in which multi-lane compute resources are kept occupied with contiguous data fed by wide memory buses. Sparse matrices violate many of the underlying assumptions of such architectures, which must be addressed to avoid crippling inefficiencies in the resulting hardware. To support sparse connectivity efficiently in hardware 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 artificial neural network (ANN) and SNN worlds.

Finally, a key CONVOLVE focus area is improving the ability to learn continuously during deployment. Such online continual learning is essential for edge-based systems allowing them to seamlessly adapt to different users, environments, or task requirements. Similar online adaptation may also bestow self-healing capabilities to the system by providing robustness to component or sensor drift over the system's lifetime. Solving these challenges requires real-time on-chip learning capabilities robust to catastrophic forgetting. However, on-chip learning precludes using BPTT and supervised learning, thus requiring major algorithmic innovations. To address these points, within CONVOLVE we will develop new algorithms for self-supervised learning in continuous time that dispense with or at least minimize back-propagation requirements through time and space.

Static vs Dynamic NN Several neural network models have been developed in recent years and have demonstrated excellent performance in many application domains. However, due to their computational complexity, these models may not be suitable for low-resource devices or latency-critical applications. Typical approaches to reducing the processing complexity of these models include model compression and response approximation. They aim at reducing the model size by injecting sparsity, adding collaborative layers [29] or designing tiny architectures [30]. Dynamic neural networks (DNN) were introduced to make the processing complexity at the inference stage input-dependent. The idea behind DNN is borrowed from biological neural networks which are believed to adapt the neural pathways to the stimulus in order to speed up decision-making [31].

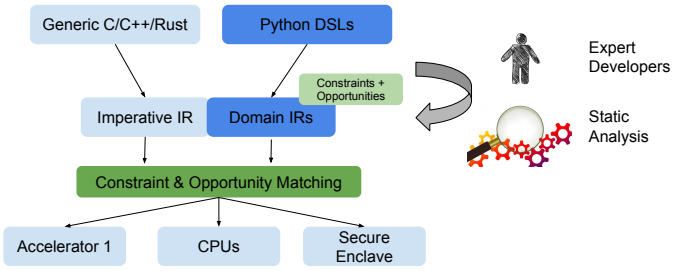

Figure 9. The CONVOLVE gray-box compiler established a semi-automatic compilation flow where static-analyses and expert developers collaborate to obtain peak-performance when running large neural networks on the CONVOLVE accelerators.

The most straightforward implementation of DNN is through Early Exit [32]. It involves using internal classifiers to make quick decisions for easy inputs, i.e. without using the full-fledged network. A response is returned if the internal classifier is sufficiently confident; otherwise, the example is passed on 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 [33]; gating functions that remove the least salient components (eg. channels of an image) [34]; parameter adaptation that aims at adaptively generating or altering the architecture's intrinsic characteristics (eg. network width or depth) given the input's features [35]; and dynamic activation functions that activate neurons according to the relevance of the input stimulus, thus increasing the representation power of models [36]. A comprehensive review of DNN can be found in [37].

We aim to develop efficient DNN for smart low-resource processors. Various constraints will be applied to the models, including those related to computational complexity, latency, energy consumption, and performance (e.g. precision, robustness). We will first review existing DNN from a resource-limited setting viewpoint, investigate the multi-criterion neural architecture search paradigm, and explore recent compression techniques (e.g. pruning-at-initialisation [38]). Both supervised and self-supervised learning approaches will be investigated to train DNN models.

B. Pillar: Architecture and design tools

Black-box vs Grey-box compiler To effectively map complex neural networks 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 [39] offer a generic performance baseline, neural networks are increasingly targeted via domain-specific frameworks such as MLIR [20] or TVM [40]. These can significantly improve performance and targetability by rigidly building in domain knowledge within a fixed-function stack, but just as the overspecialization in the frontend of LLVM hurts targetability of accelerators, the overspecialization in the backend of these newer frameworks hurts targetability of new applications, or the intended applications on accelerators with altered capabilities (e.g. lower precision). The CONVOLVE compiler (Figure 9) will extend MLIR to offer a generic *grey*-box approach, where 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, and preserving key invariants.

Powerful static analyses of the applications, feeding information throughout the intermediate-representation stack, will enable our compiler to semi-automatically target CONVOLVE accelerators. Based on the results of the static analysis, the *opportunities* exposed by the applications' characteristics, and the *constraints* imposed by the execution environment, the compiler will attempt to customize the application and automatically generate efficient code, optimized and tailored for the target architecture. In addition, we will offer domain-expert developers efficient access to the compiler internals to specialize code optimization and application-to-accelerator mapping for each use case.

The compiler first analyzes the *opportunities* exposed by the application, such as the degree of parallelism, reduced precision, code layout, algorithmic structure, or a limited input domain. Then it considers the *constraints* of the execution context: latency requirements, timing and security guarantees, the accuracy of the target, performance penalties of a complex control flow, etc. These will serve as a description of the applications and available hardware platforms, preserved under transformation, to guide the compiler in optimizing the applications. We envision optimizations such as remapping data in memory (e.g., data transfer reduction), targeting fixed-function accelerators via algorithmic matching, and reorganizing code layout to alleviate bandwidth bottlenecks (for new custom hardware and for providing real-time constraint guarantees). CONVOLVE will complement static analysis with the manual and test-oriented generation of opportunities, verified dynamically or with profile guidance, that offers domain experts the opportunity to guide our gray-box compiler and attain peak performance.

Static vs dynamic architecture - Design hierarchy level: Architecture Dynamic Neural Networks based on ANN and SNN are a fast-moving field of research in terms of network design and optimization. Optimization knobs include the quantization strategy and granularity, the number and types of network layers, and the supported types of network dynamism. As such, the traditional ways of designing heterogeneous multi-core SoCs with different accelerators to maximize energy-efficiency might be less effective when the number of different configurations/accelerators explodes or the computation/network changes in the future.

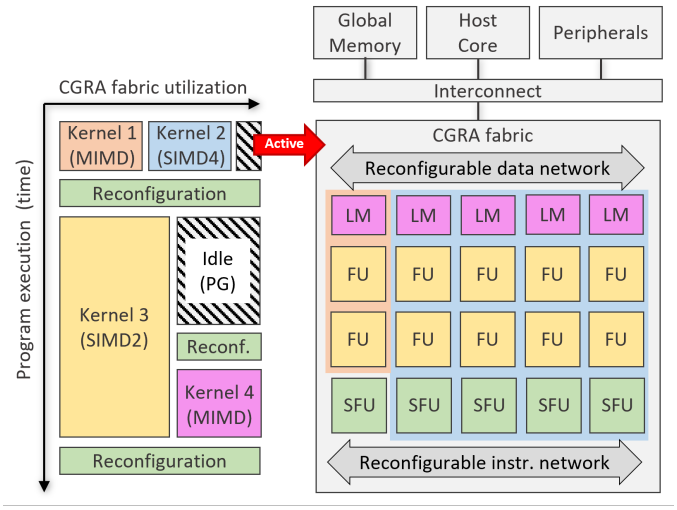

Figure 10. CONVOLVE CGRA fabric that executes multiple kernels in parallel while reconfiguring the fabric on a kernel-level granularity and power-gating unused units.

A Coarse-Grained Reconfigurable Architecture (CGRA) could be a good compromise between the flexibility of a general-purpose processor and the energy-efficiency of a fixed-function accelerator. A CGRA consists of a grid (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 the reconfiguration overhead in CGRAs is much lower over FPGAs. The class of CGRAs covers a wide range of reconfigurable architectures, each with a different degree of programmability. A recent overview is provided in [41].

The CONVOLVE CGRA will be based on the Blocks CGRA template [42] which will be extended for Dynamic Neural Network acceleration. An overview is depicted in Fig. 10. The Blocks CGRA template consists of a reconfigurable instruction and data network with programmable FUs. The physical fabric (physArch) is able to execute multiple application-specific processors or virtual cores (virtArchs) in parallel. These virtual cores can be VLIW cores with optional SIMD extensions sized towards the kernel or application at hand. The fabric supports flexible SIMD execution by broadcasting the same instructions to multiple FUs over the reconfigurable instruction network. Complex instructions are supported via specialized FUs (SFUs). To accommodate efficient acceleration for varying workloads, the fabric can be (partially) reconfigured on a kernel-level granularity with a tolerable reconfiguration penalty [43]. The fabric contains support for zero-overhead loops to enable spatial computation where the instruction stream remains static. Leakage power of unused FUs can be reduced using power gating.

To summarize, the use of a CGRA in CONVOLVE has

several advantages: (1) it supports highly parallel calculations, (2) it has good area-efficiency compared to an FPGA, (3) it has high energy-efficiency due to the static interconnect and spatial mapping of computation where possible, (4) and it is flexible, supporting all kinds of applications as long as the FUs are not too specialized. Future research is needed within the CONVOLVE project to optimize the CGRA fabric for Dynamic Neural Network acceleration and to improve the existing retargetable C-compiler for efficient code generation with support for run-time (partial) reconfiguration.

Generic vs Application specific hardware - Design hierarchy level: Architecture Most commercially available computing platforms are based on traditional general-purpose CPUs, which offer full flexibility and easy programmability. Such architectures are suitable for a variety of applications and hence allow mass-deployment and increased production volume which in the end helps to reduce the fabrication cost per unit. However, being general-purpose often results in mediocre performance and sub-optimal energy efficiency. In contrast, application-specific accelerators can achieve optimal throughput, area and energy efficiency for a specific hardwired algorithm. Such dedicated accelerators have very limited capabilities to adapt to next-generation algorithms, which — in the worst case — could lead to the expensive dedicated silicon area becoming completely unusable and requiring an expensive re-design of the accelerator. A balance between these two extremes is offered by FPGA solutions, where the datapath can be rewired to adapt to novel algorithms. However, FPGA platforms are still rather expensive when targeting a massive and dense deployment and the SoC integration is still rather difficult: Embedded FPGA solutions (e.g., offered by Menta, QuickLogic) which are offered as soft IPs are slower and more area-hungry than dedicated FPGAs chips. Another option is offered by general-purpose accelerators such as GPUs. Their flexibility allows their use for various applications in a variety of scenarios and hence offers a more cost-efficient alternative than FPGA or application-specific accelerators. Nevertheless, their somewhat hardwired datapath can limit accelerator utilization and hence deliver sub-optimal performance for certain kernels. The in CONVOLVE developed ULP acceleration blocks target a trade-off between application-specific and general-purpose computing by combining specialization with modularity, dynamic reconfigurability, and self-healing capabilities to fully harness the potential of machine learning. Furthermore, the selection of reconfigurable ULP blocks plan to support dynamic neural networks, improve reliability against process variations and provide real-time guarantees for safety-critical applications, and have provision for hardware security against e.g., side-channel attacks and security futureproofing. To keep the energy-efficiency high despite these reconfiguration overheads, CONVOLVE explores novel circuit and architecture-level techniques for efficient SRAM/ReRAM based CIM and CGRA.

C. Pillar: ULP blocks

In-memory vs Classical computing architecture- Design hierarchy level - circuit/architecture/device Moore's law has enabled the traditional von Neumann-based CPUs to deliver better performance for successive generations [44]. However, traditional CPU architectures are facing three major walls: (1) the memory wall due to the growing gap between processor and memory speed, and the limited memory bandwidth; (2) the Instruction-Level parallelism (ILP) wall due to the difficulty of extracting sufficient parallelism to fully exploit all the cores; (3) the power wall as the CPU clock frequency has reached the practical maximum value that is limited by cooling. In order for computing systems to continue delivering the required performance given the economical power constraints, novel computer architectures in the light of emerging non-volatile (practically no leakage) device technologies have to be explored. CIM has the potential to overcome those challenges by integrating computation and storage of data within the same physical location [7], [45]. 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 [3], [7]. 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 qualitative comparison of traditional von-Neumann based CPU and CIM architectures using different memory technologies [46], [47]. 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 [46], [47])

Comparison metric	Conventional CPU	CIM digital SRAM	CIM analog SRAM	CIM analog memristive
Device technology	CMOS	CMOS	CMOS	RRAM
Architecture	von-Neumann	von-Neumann	non von-Neumann	non von-Neumann
Mode of operation	Digital	Digital	Analog	Analog
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
Read latency	∼1ns	∼1ns	∼1ns	∼10ns

Exact vs Approximate computation - Design hierarchy level: Circuit Diminishing energy-efficiency gains from semiconductor scaling as per Moore's law and continued increase in compute-requirements, as evident from latest machine learning (ML) models like GPT3, Transformers etc., has forced 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.

While AxC techniques have shown benefits at all levels of the computing stack, zooming in on the circuit-level, most AxC techniques can be classified into three broad categories: (a) timing approximation, wherein the circuit is operated

at a lower supply voltage without reducing corresponding operational frequency, resulting in efficiency improvements for added timing errors [48], (b) functional approximation, wherein the logic functionality of the circuit is modified to trade off quality for added efficiency e.g. netlist modification [49], boolean rewriting [50], [51], precision-scaling [52]–[55] etc. and c) cross-level approximation where approximation knobs at both logic-, structural- and physical level are leveraged in a disciplined manner to boost energy efficiency [56] have been implemented for functional approximation.

AxC has been widely adopted for ML. Among all different approximations being investigated for ML, *precision-scaling* has emerged as a success story. ML models have been shown to 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 [52], DLFloat [53] etc. have been adopted for activation and weights. For inference, varying fixed-point formats are adopted scaling all the way down to ternary or binary bitwidths [54]. Accuracy lost due to approximations are regained using quantization-aware training (QAT) [55]. Execution engines with varying precision support have been proposed in both academia and industry [57]. Another widely adopted approximation technique is *pruning* which forces weight values in a neural network to zero, thereby introducing sparsity. Several studies have proposed the combination of weight pruning with precision scaling to achieve higher energy efficiency for ML inference [58], [59]. Pruning introduces irregularities in compute and memory access pattern. To tackle such challenges, specialized architectures with sparsity support have been proposed in literature [60]. Motivated by the promising returns proposed by the close synergy of AxC and ML, CONVOLVE aims at exploring novel AxC techniques at the circuit level to obtain extremely energy-efficient edge AI microprocessors.

Analog vs Digital computing - Design hierarchy level: Circuit/device: 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. Although state-of-the-art analog and mixed-signal architectures offer the lowest energy consumption, they have severe limitations that limited their applications. Firstly, they offer poor reliability as their performance is easily affected by noise induced by process, temperature, and voltage variations and hence require a complex calibration process. Secondly, they offer poor technology portability as they need to be re-designed when porting the design to a different technology node. Thirdly, they offer poor scalability as larger designs cannot be easily built using powerful design automation tools available for digital designs [61], [62]. However, analog implementations are proven to be more energy-efficient compared to digital. More recently, Wan et al. [13] demonstrate the exploitation of highly-integrated resistive random-access memory (ReRAM) devices to avoid power-hungry data movement between separate compute and memory, achieving inference accuracy comparable to software models trained with 4-bit weights across several AI benchmark tasks. These achievements demonstrate a simultaneous improvement in efficiency, flexibility, and accuracy over existing RRAM-CIM hardware by innovating across the entire design hierarchy, from a reconfigurable dataflow architecture to an energy- and area-efficient voltage-mode neuron circuit and to a series of algorithm-hardware co-optimization techniques. This demonstrates that the problem of efficient computing must be tackled simultaneously and at multiple levels of the stack.

V. Summary

On the short term ANNs are to be favored above SNNs. Online learning requires rethinking BPTT/TSpace. Dynamic NN will become dominant in low power edge computing. Quick and easy adaptation requires a Grey-box compiler which can efficiently deal with new accelerators in a comprehensive way. Architectures should and will become dynamic : 2-step code generation (i.e. Code $\rightarrow$ VA $\rightarrow$ PArch). The DL field is moving quickly and therefor requires flexible architectures. In-memory computing is promising and required, it means 2.5 D computing on a very short term, and Adapted memory periphery in the longer term. Analog computing has high potential for energy savings but is too inflexible on a short term, where DNNs do not fit spatially. Therefore time-sharing and reconfig flexibility is key.

References

[1] 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 *2022 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*, 2022, pp. 20–21.

[2] K. Ueyoshi *et al.*, "Diana: An end-to-end energy-efficient digital and analog hybrid neural network soc," in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 65, 2022, pp. 1–3.

[3] X. Si *et al.*, "A twin-8t sram computation-in-memory unit-macro for multibit cnn-based ai edge processors," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 1, pp. 189–202, 2019.

[4] 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 *2022 IEEE International Solid- State Circuits Conference (ISSCC)*, vol. 65, 2022, pp. 1–3.

[5] K. Roy *et al.*, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, no. 7784, pp. 607–617, 2019.

[6] A. Singh *et al.*, "Srif: Scalable and reliable integrate and fire circuit adc for memristor-based cim architectures," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 5, pp. 1917–1930, 2021.

[7] ——, "Low-power memristor-based computing for edge-ai applications," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2021, pp. 1–5.

[8] S. Diware *et al.*, "Unbalanced bit-slicing scheme for accurate memristor-based neural network architecture," in *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, 2021, pp. 1–4.

[9] C. Bengel *et al.*, "Reliability aspects of binary vector-matrix-multiplications using reram devices," *Neuromorphic Computing and Engineering*, 2022.

[10] M. Mayahinia *et al.*, "A voltage-controlled, oscillation-based adc design for computation-in-memory architectures using emerging rerams," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 18, no. 2, pp. 1–25, 2022.

[11] S. Diware *et al.*, "Accurate and energy-efficient bit-slicing for rram-based neural networks," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2022.

[12] S. D. Spetalnick *et al.*, "A 40nm 64kb 26.56 tops/w 2.37 mb/mm 2 rram binary/compute-in-memory macro with 4.23 x improvement in density and> 75% use of sensing dynamic range," in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 65. IEEE, 2022, pp. 1–3.

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

[14] 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 *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64. IEEE, 2021, pp. 245–247.

[15] 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.

[16] 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 *2022 IEEE Symposium in Low-Power and High-Speed Chips (COOL CHIPS)*. IEEE, 2022, pp. 1–3.

[17] J. Stuijt *et al.*, "μbrain: An event-driven and fully synthesizable architecture for spiking neural networks," *Frontiers in neuroscience*, vol. 15, p. 538, 2021.

[18] K. Guo *et al.*, "A survey of fpga-based neural network accelerator," *arXiv preprint arXiv:1712.08934*, 2017.

[19] A. Gebregiorgis *et al.*, "Dealing with non-idealities in memristor based computation-in-memory designs," in *2022 IFIP/IEEE 30th International Conference on Very Large Scale Integration (VLSI-SoC)*. IEEE, 2022, pp. 1–6.

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

[21] L. Mei *et al.*, "ZigZag: Enlarging joint architecture-mapping design space exploration for dnn accelerators," *IEEE Transactions on Computers*, vol. 70, no. 8, pp. 1160–1174, 2021.

[22] 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.

[23] E. O. Neftci *et al.*, "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 51–63, 2019.

[24] J. Kaiser *et al.*, "Synaptic Plasticity Dynamics for Deep Continuous Local Learning (DECOLLE)," *Frontiers in Neuroscience*, vol. 14, 2020.

[25] B. Yin *et al.*, "Accurate online training of dynamical spiking neural networks through forward propagation through time," *arXiv preprint arXiv:2112.11231*, 2021.

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

[27] B. Yin *et al.*, "Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks," *Nature Machine Intelligence*, vol. 3, no. 10, pp. 905–913, 2021.

[28] N. Perez-Nieves *et al.*, "Neural heterogeneity promotes robust learning," *bioRxiv*, p. 2020.12.18.423468, Jan. 2021.

[29] J. Lee *et al.*, "Resource-efficient deep learning: A survey on model-, arithmetic-, and implementation-level techniques," *arXiv.2112.15131*, 2021.

[30] E. Liberis *et al.*, "μnas: Constrained neural architecture search for microcontrollers," in *1st Workshop on Machine Learning and Systems*, 2021, pp. 70–79.

[31] D. Ariely *et al.*, "From thinking too little to thinking too much: a continuum of decision making," *Wiley Interdisciplinary Reviews: Cognitive Science*, vol. 2, no. 1, pp. 39–46, 2011.

[32] S. Scardapane *et al.*, "Why should we add early exits to neural networks?" *Cognitive Computing*, vol. 12, no. 5, pp. 954–966, 2020.

[33] J. Hu *et al.*, "queeze-and-excitation networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7132–7141.

[34] X. Gao *et al.*, "Dynamic channel pruning: Feature boosting and suppression," *arXiv:1810.05331v2*, 2018.

[35] W. Xia *et al.*, "Fully dynamic inference with deep neural networks," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 2, pp. 962–972, 2022.

[36] Y. Chen *et al.*, "Dynamic relu," in *16th European Conference on Computer Vision (ECCV)*, 2020, p. 351–367.

[37] Y. Han *et al.*, "Dynamic neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 11, p. 7436–7456, 2022.

[38] 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.

[39] C. Lattner *et al.*, "Llvm: A compilation framework for lifelong program analysis & transformation," in *International Symposium on Code Generation and Optimization, 2004. CGO 2004.* IEEE, 2004, pp. 75–86.

[40] T. Chen *et al.*, "TVM: An automated End-to-End optimizing compiler for deep learning," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 2018, pp. 578–594.

[41] M. Wijtvliet *et al.*, "Coarse grained reconfigurable architectures in the past 25 years: Overview and classification," in *2016 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS)*, 2016, pp. 235–244.

[42] ——, "Blocks: Challenging simds and vliws with a reconfigurable architecture," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 9, pp. 2915–2928, 2021.

[43] B. de Bruin *et al.*, "Multi-level optimization of an ultra-low power brainwave system for non-convulsive seizure detection," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 15, no. 5, pp. 1107–1121, 2021.

[44] S. Rai *et al.*, "Perspectives on emerging computation-in-memory paradigms," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2021, pp. 1925–1934.

[45] A. Singh *et al.*, "Cim-based robust logic accelerator using 28 nm stt-mram characterization chip tape-out," in *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, 2022, pp. 451–454.

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

[47] F. Oboril *et al.*, "Evaluation of hybrid memory technologies using sot-mram for on-chip cache hierarchy," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 3, pp. 367–380, 2015.

[48] K. Shi *et al.*, "Datapath synthesis for overclocking: Online arithmetic for latency-accuracy trade-offs," in *Proceedings of the 51st Annual Design Automation Conference*, ser. DAC '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 1–6.

[49] S. De *et al.*, "An automated approximation methodology for arithmetic circuits," in *2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, 2019, pp. 1–6.

[50] S. Venkataramani *et al.*, "Salsa: Systematic logic synthesis of approximate circuits," in *DAC Design Automation Conference 2012*, 2012, pp. 796–801.

[51] V. Leon *et al.*, "Cooperative arithmetic-aware approximation techniques for energy-efficient multipliers," in *Proceedings of the 56th Annual Design Automation Conference 2019, DAC 2019, Las Vegas, NV, USA, June 02-06, 2019*. ACM, 2019, p. 160.

[52] D. D. Kalamkar *et al.*, "A study of BFLOAT16 for deep learning training," *CoRR*, vol. abs/1905.12322, 2019.

[53] A. Agrawal *et al.*, "Dlfloat: A 16-b floating point format designed for deep learning training and inference," in *2019 IEEE 26th Symposium on Computer Arithmetic (ARITH)*, 2019, pp. 92–95.

[54] S. Zhu *et al.*, "Tab: Unified and optimized ternary, binary, and mixed-precision neural network inference on the edge," *ACM Trans. Embed. Comput. Syst.*, vol. 21, no. 5, oct 2022.

[55] J. Choi *et al.*, "PACT: Parameterized clipping activation for quantized neural networks," 2018.

[56] G. Zervakis *et al.*, "Multi-level approximate accelerator synthesis under voltage island constraints," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 66-II, no. 4, pp. 607–611, 2019.

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

[58] V. Leon *et al.*, "Max-dnn: Multi-level arithmetic approximation for energy-efficient DNN hardware accelerators," in *13th IEEE Latin America Symposium on Circuits and System, LASCAS 2022, Puerto Varas, Chile, March 1-4, 2022*. IEEE, 2022, pp. 1–4.

[59] V. Mrazek *et al.*, "Alwann: Automatic layer-wise approximation of deep neural network accelerators without retraining," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2019, pp. 1–8.

[60] I. Hubara *et al.*, "Accelerated sparse neural training: A provable and efficient method to find n:m transposable masks," in *Advances in Neural Information Processing Systems*, M. Ranzato *et al.*, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 21 099–21 111.

[61] C. D. Schuman *et al.*, "A survey of neuromorphic computing and neural networks in hardware," *CoRR*, vol. abs/1705.06963, 2017.

[62] M. Bouvier *et al.*, "Spiking neural networks hardware implementations and challenges: a survey," *CoRR*, vol. abs/2005.01467, 2020.