MISSIONE 4
ISTRUZIONE
RICERCA

SPOKE 1

FUTURE HPC & BIG DATA

FLAGSHIP 2:
Survey of state-of-the-art approaches and gap analysis of Hardware platform for acceleration
Accel-HW-Spec Requirement for HPC

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
ICSC Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

EXECUTIVE SUMMARY

This document overviews the state-of-the-art approaches of hardware platforms for Deep Learning accelerators for HPC systems, according to the main objectives described in the milestone #4, Spoke 1 - Flagship 2 WP2: Flagship on heterogeneous acceleration, architecture, tools, and software (Leader: POLIMI).

Recent trends in deep learning (DL) imposed hardware accelerators as the most viable solution for several classes of high performance computing (HPC) applications such as image classification, computer vision and speech recognition. This survey summarizes and classifies the most recent advances in designing DL accelerators suitable to reach the performance requirements of HPC applications. In particular, we highlight the most advanced approaches to support deep learning accelerations including not only GPU and TPU-based accelerators, but also design-specific hardware accelerators such as FPGA-based and ASIC-based accelerators, Neural Processing Units, RISC-V based accelerators and co-processors. The survey also describes accelerators based on emerging memory technologies and computing paradigms, such as 3D-stacked Processor-In-Memory, non-volatile memories (mainly RRAM and PCM) to implement in-memory computing, Neuromorphic Processing Units and accelerators based on Multi-Chip Modules. The survey classifies the most influential architectures and technologies proposed in the last 20 years, with the purpose to offer to the reader a wide perspective in the rapid evolving field of deep learning. Finally, this survey provides some insights on the future challenging trends in DL accelerators.

The survey is structured in different categories and sub-categories belonging to the areas of computer architectures and hardware design. We start with Section 2 by providing an overview of deep learning concepts and terminology. In Section 3, we then review the most significant acceleration solutions based on GPUs and Tensor Processing Units. Section 4 introduces three types of hardware-based accelerators: FPGA-based, ASIC-based and accelerators based on the open-hardware RISC-V Instruction Set Architecture. Section 5 describes DL accelerators based on emerging computing paradigm and technologies. A final discussion on future trends on DL accelerators can be found in Section 6.

A Survey on Deep Learning Accelerators for Heterogeneous HPC Platforms

CRISTINA SILVANO, DANIELE IELMINI, FABRIZIO FERRANDI, LEANDRO FIORIN, and SERENA CURZEL, Politecnico di Milano, Italy
LUCA BENINI, FRANCESCO CONTI, and ANGELO GAROFALO, Università di Bologna, Italy
CRISTIAN ZAMBELLI, ENRICO CALORE, and SEBASTIANO FABIO SCHIFANO, Università degli Studi di Ferrara, Italy
MAURIZIO PALESI, GIUSEPPE ASCIA, and DAVIDE PATTI, Università degli Studi di Catania, Italy
STEFANIA PERRI, Università degli studi della Calabria, Italy
NICOLA PETRA and DAVIDE DE CARO, Università degli studi di Napoli Federico II, Italy
LUCIANO LAVAGNO and TEODORO URSO, Politecnico di Torino, Italy
VALERIA CARDELLINI and GIAN CARLO CARDARILLI, Università di Roma "Tor Vergata", Italy
ROBERT BIRKE, Università degli Studi di Torino, Italy

Recent trends in deep learning (DL) imposed hardware accelerators as the most viable solution for several classes of high performance computing (HPC) applications such as image classification, computer vision and speech recognition. This survey summarizes and classifies the most recent advances in designing DL accelerators suitable to reach the performance requirements of HPC applications. In particular, we highlight the most advanced approaches to support deep learning accelerations including not only GPU and TPU-based accelerators, but also design-specific hardware accelerators such as FPGA-based and ASIC-based accelerators, Neural Processing Units, RISC-V based accelerators and co-processors. The survey also describes accelerators based on emerging memory technologies and computing paradigms, such as 3D-stacked Processor-In-Memory, non-volatile memories (mainly Resistive RAM - RRAM and Phase Change Memories - PCM) to implement in-memory computing, Neuromorphic Processing Units and accelerators based on Multi-Chip Modules. The survey classifies the most influential architectures and technologies proposed in the last 20 years, with the purpose to offer to the reader a wide perspective in the rapid evolving field of deep learning. Finally, this survey provides some insights on the future challenging trends in DL accelerators.

1 INTRODUCTION

Since the advent of the Exascale era, we have witnessed the convergence between High Performance Computing (HPC) and Artificial Intelligence (AI). The ever increasing computing power of HPC systems and their ability to manage large amounts of data made the development of more and more sophisticated machine learning (ML) techniques possible. Deep Learning (DL) is a subset of machine learning and uses artificial Deep Neural Networks (DNNs) with multiple layers of artificial neurons to attempt to mimic the human brain behavior by learning from large amounts of data. Thanks to technological and architectural improvements, not only an increasing number of parallel high-end

Authors' addresses: Cristina Silvano, cristina.silvano@polimi.it; Daniele Ielmini, daniele.ielmini@polimi.it; Fabrizio Ferrandi, fabrizio.ferrandi@polimi.it; Leandro Fiorin, leandro.fiorin@polimi.it; Serena Curzel, serena.curzel@polimi.it, Politecnico di Milano, Italy; Luca Benini, luca.benini@unibo.it; Francesco Conti, f.conti@unibo.it; Angelo Garofalo, angelo.garofalo@unibo.it, Università di Bologna, Italy; Cristian Zambelli, cristian.zambelli@unife.it; Enrico Calore, enrico.calore@fe.infn.it; Sebastiano Fabio Schifano, sebastiano.fabio.schifano@unife.it, Università degli Studi di Ferrara, Italy; Maurizio Palesi, maurizio.palesi@unict.it; Giuseppe Ascia, giuseppe.ascia@unict.it; Davide Patti, davide.patti@unict.it, Università degli Studi di Catania, Italy; Stefania Perri, s.perri@unical.it, Università degli studi della Calabria, Italy; Nicola Petra, nicola.petra@unina.it; Davide De Caro, dadecaro@unina.it, Università degli studi di Napoli Federico II, Italy; Luciano Lavagno, luciano.lavagno@polito.it; Teodoro Urso, teodoro.urso@polito.it, Politecnico di Torino, Italy; Valeria Cardellini, cardellini@ing.uniroma2.it; Gian Carlo Cardarilli, g.cardarilli@uniroma2.it, Università di Roma "Tor Vergata", Italy; Robert Birke, robert.birke@unito.it, Università degli Studi di Torino, Italy.

processors, but also co-processors such as graphics processing units (GPUs) and vector/tensor computing units have been integrated into the nodes of HPC systems. This supercomputing power enabled to speed up the automatic training phase of DNN models and their subsequent inference phase in the target application scenarios.

The introduction of the pioneering AlexNet [150] at the ImageNet challenge in 2012, made clear the need of acceleration during the training phase. Since then, a multitude of DNN models have been developed for various tasks including image recognition and classification, Natural Language Processing, and Generative AI. These applications require specialized *hardware accelerators*, to efficiently handle the heavy computational demands of DNN algorithms. DL accelerators are currently in use in several types of computing systems spanning from ultra-low-power and resource-constraints devices on-the-edge up to servers, HPC infrastructures and data centers.

Scope of the survey. This survey is an attempt to provide an extensive overview of the most influential architectures to accelerate DL for high-performance applications. The survey highlights various approaches that support DL acceleration including GPU-based accelerators, Tensor Processor Units, FPGA-based accelerators and ASIC-based accelerators, such as Neural Processing Units and co-processors on the open-hardware RISC-V architecture. The survey also includes accelerators based on emerging technologies and computing paradigms, such as 3D-stacked PIM, emerging non-volatile memories such as the Resistive switching Random Access Memory (RRAM) and the Phase Change Memory (PCM), Neuromorphic Processing Units and Multi-Chip Modules.

Overall, we have reviewed the research on DL accelerators from the past two decades, covering a significant time span of literature in this field. We have described and referenced about 250 works proposed for DL acceleration. Being DL acceleration such a prolific and rapid evolving field, we do not claim to cover exhaustively all the research works appeared so far, but we focused on the most influential contributions. Moreover, this survey can be leveraged as a connecting point for some previous surveys on accelerators on the AI and DL field [44, 84, 109, 225] and other surveys focused on some more specific aspects of DL, such as the architecture-oriented optimization of sparse matrices [226] and the Neural Architecture Search [50].

Organization of the survey. The survey is structured in different categories and sub-categories belonging to the areas of computer architectures and hardware design. As shown in Figure 1, the proposed classification is based on several representative features of the accelerators, in order to highlight their similarities and differences. To this aim, we organized the material in a way that all research papers corresponding to multiple types of classifications are cited under each classification. For example, let us consider the work W, which primarily belongs to the sub-category X where it makes its primary contribution. According to our classification policy, this work could be cited again in another sub-category Y, where it makes its secondary contribution. Moreover, under each classification, we have selectively chosen the most notable and influential works and, for each work, we focused on its innovative contributions.

The survey is structured as follows: Section 2 introduces some background on DL concepts and terminology, while Section 3 reviews the most significant acceleration solutions based on GPUs and TPUs. Section 4 introduces three types of hardware-based accelerators: FPGA-based, ASIC-based and accelerators based on the open-hardware RISC-V Instruction Set Architecture. Section 5 describes DL accelerators based on emerging computing paradigm and technologies. A final discussion on future trends on DL accelerators can be found in Section 6.

To conclude, we hope this survey could be useful for a wide range of readers, including computer architects, hardware developers, HPC engineers, researchers, and technical professionals. A major effort was spent to use a clear and concise technical writing style: we hope this effort could be useful in particular to the young generations of master and PhD students. To facilitate the reading, a list of acronyms is reported in Table 1.

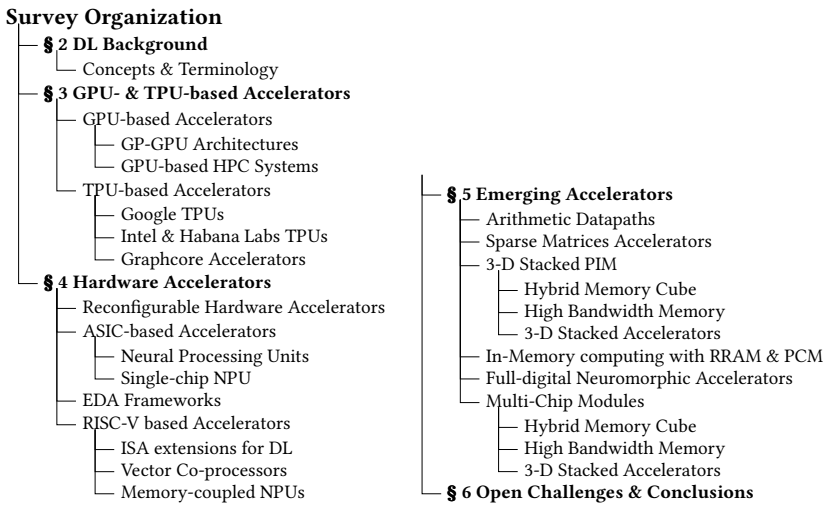

Fig. 1. Organization of the survey

2 DEEP LEARNING BACKGROUND: CONCEPTS AND TERMINOLOGY

DL [157, 231] is a subset of ML methods which uses artificial DNNs for automatically discovering the representations needed for feature detection or classification from large data sets, by employing multiple layers of processing to extract progressively higher level features. DNNs mimic the human brain functionalities, in which neurons are interconnected with each other to receive information, process it, and pass it to other neurons. As shown in Figure 2a, in a way similar to the brain's neuron, the simple model of a perceptron (artificial neuron) receives information from a set of inputs, and apply a nonlinear function F (activation function) on a weighted (W) sum of the inputs (X) [228]. DNNs are composed of a number of layers of artificial neurons (hidden layers), organized between the input layer, which brings the initial data into the system, and the output layer, in which the desired predictions are obtained (see Figure 2b). In *feed-forward networks*, the outputs of one layer become the inputs of the next layer in the model, while in *recurrent networks*, the output of a neuron can be the input of neurons in the same or previous layers. The term "deep" in DNNs

Table 1. List of acronyms

Acronym	Acronym	Acronym
AI: Artificial Intelligence	**ASIC:** Application Specific Integrated Circuit	**BRAM:** Block Random Access Memory
CMOS: Complementary Metal Oxide Semiconductor	**CNN:** Convolutional Neural Network	**CPU:** Central Processing Unit
DL: Deep Learning	**DP:** Double Precision	**DNN:** Deep Neural Network
DRAM: Dynamic Random Access Memory	**EDA:** Electronic Design Automation	**FLOPS:** Floating Point Operations per Second
FMA: Fused Multiply-Add	**FPGA:** Field-Programmable Gate Array	**GEMM:** General Matrix Multiply
GP-GPU: General-Purpose Graphics Processing Unit	**GPU:** Graphics Processing Unit	**HBM:** High Bandwidth Memory
HDL: Hardware Description Language	**HLS:** High Level Synthesis	**HMC:** Hybrid Memory Cube
HPC: High-Performance Computing	**MLP:** Multi-Layer Perceptron	**NPU:** Neural Processing Unit
IMC: In-Memory Computing	**IoT:** Internet of Things	**ISA:** Instruction Set Architecture
MCM: Multi-Chip Module	**ML:** Machine Learning	**NDP:** Near Data Processing
NN: Neural Network	**NoC:** Network on Chip	**PCM:** Phase Change Memory
PCU: Programmable Computing Unit	**PIM:** Processing In-Memory	**PULP:** Parallel Ultra Low Power
QC: Quantum Computing	**QML:** Quantum Machine Learning	**QNN:** Quantized Neural Network
QPU: Quantum Processing Unit	**RAM:** Random Access Memory	**RRAM:** Resistive RAM
RISC: Reduced Instruction Set Computer	**RNN:** Recurrent Neural Network	**SoC:** System on Chip
SP: Single Precision	**SIMD:** Single Instruction Multiple Data	**SIMT:** Single Instruction Multiple Thread
SNN: Spiking Neural Network	**SRAM:** Static Random Access Memory	**TPU:** Tensor Processing Unit
TNN: Ternary Neural Network	**VPU:** Vector Processing Unit	**VRAM:** Video Random Access Memory

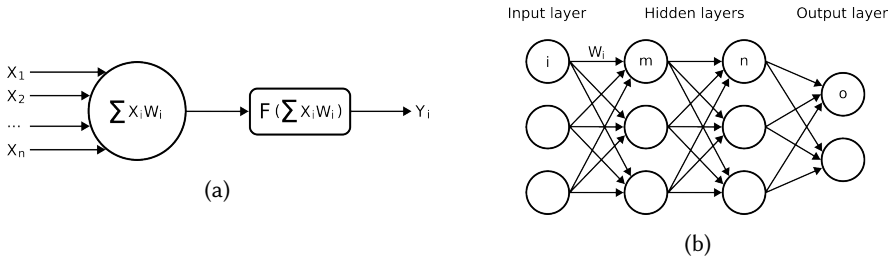

Fig. 2. Model of a perceptron (artificial neuron) (a) and of a multi-layer DNN (b).

refers to the use of a large number of layers, which results in more accurate models that capture complex patterns and concepts.

There are two phases in DNNs' operations: training, and inference. In the *training* phase, the neural network model is fed on a curated data set so that it can "learn" everything it needs to about the type of data it will analyze. In the case of *supervised* learning, a large set of examples and their corresponding labels indicating the correct classification are passed as input to the DNN. A forward pass is executed, and the error against the correct labels is measured. Then, the error is used in the DNN's backward pass to update the weights. This loop is performed repeatedly, until the DNN model achieves the desired accuracy. In *unsupervised* learning, the DNN uses unlabeled data to create an encoded self-organization of weights and activations that captures patterns as probability densities. With *semi-supervised* learning, during training a small amount of labeled data is combined with a large amount of unlabeled data. In the *inference* phase, the trained DNN model is used to make predictions on unseen data. When it comes to deployment, the trained model is often modified and simplified to meet real-world power and performance requirements. The two phases present different computational characteristics. On the one hand, the training phase of a model is computationally expensive, but usually performed only once. On the other hand, the trained model is used for predictions on multiple input data, often under strict latency and/or energy constraints.

Three general types of DNN are mostly used today: Multi-Layer Perceptrons (MLPs), Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs). MLPs [228] are feed-forward ANNs composed of a series of fully connected layers, where each layer is a set of nonlinear functions of a weighted sum of all outputs of the previous one. On the contrary, in a CNN [158], a convolutional layer extracts the simple features from the inputs by executing convolution operations. Each layer is a set of nonlinear functions of weighted sums of different subsets of outputs from the previous layer, with each subset sharing the same weights. Each convolutional layer in the model can capture a different high-level representation of input data, allowing the system to automatically extract the features of the inputs to complete a specific task, e.g., image classification, face authentication, and image semantic segmentation. Finally, RNNs [231] address the time-series problem of sequential input data. Each RNN layer is a collection of nonlinear functions of weighted sums of the outputs of the previous layer and the previous state, calculated when processing the previous samples, and stored in the RNN's internal memory. RNN models are widely used in Natural Language Processing (NLP) for natural language modeling, word embedding, and machine translation.

Each type of DNN is especially effective for a specific subset of cognitive applications, and, depending on the specific task, a model composed of a mix of the above mentioned three types of neural network can be deployed. Depending on the target application, and on the resource constraints of the computing system, different DNN models have been deployed.

On the one hand, DNNs such as AlexNet[150] and the most recent GoogLeNet[256] are composed of tens of layers, with millions of weights to be trained and used in every prediction, requiring tens to hundreds of megabytes (or even gigabytes) of memory for their storage. The calculation of the weighted sums requires a large number of data movements between the different levels of the memory hierarchy and the processing units, often posing a challenge on the available energy, memory bandwidth, and memory storage of the compute architecture.

On the other hand, Tiny machine learning (TinyML) DNN models [284] have been investigated to run on small, battery-operated devices like microcontrollers, trading off prediction accuracy with respect to low-latency, low-power and low-bandwidth model inference of sensor data on edge devices.

3 GPU- AND TPU-BASED ACCELERATORS

3.1 GPU-based accelerators

GPUs are specific-purpose processors introduced to compute efficiently graphics-related tasks, such as 3D rendering. They became widely used since the nineties as co-processors, working alongside CPUs, to offload graphics-related computations. The introduction of programmable shaders into GPU architectures, increased their flexibility paving the way for their adoption to perform general-purpose computations. Despite being specifically designed for computer graphics, their highly-parallel architecture is well suited to tackle a wide range of applications. Consequently, in the early 2000s, GPUs started to be used to accelerate data-parallel computations not necessarily related to graphics, which could benefit from their architecture as well. This practice is commonly referred as General-Purpose computing on GPUs (GP-GPU) and started to be increasingly popular since the early 2010s with the advent of the CUDA language.

The technological development of the last ten years significantly increased the compute power of GPU devices, which due to their highly parallel nature, are incidentally very well suited to accelerate neural networks training algorithms. The availability of such compute power allowed more complex neural network models to become practically usable, fostering the development of DNNs.

The impressive results obtainable with DNNs in the context of AI, followed by significant investments in this market sector, induced hardware manufacturers to modify GPU architectures in order to be even more optimized to compute such workloads, as an example implementing the support for lower-precision computations. This lead to a de-facto co-design of GPU architectures and neural network algorithms implementations, which is nowadays significantly boosting the performance, accuracy and energy efficiency of AI applications.

3.1.1 GP-GPU Architectures. In this sub-section, we review the basic features of NVIDIA GPU architectures to boost the performance of HPC and Deep Learning applications. The hardware architecture of a GPU is based on a multicore design of processing elements called *Streaming Multiprocessors* (SM). Each SM in turn includes a number of compute units, called CUDA-cores in NVIDIA jargon, to execute at each clock-cycle multiple warps, i.e. groups of 32 operations called CUDA-threads processed by the *Single Instruction Multiple Thread* (SIMT) fashion. SIMT execution enables different threads of a group to take different branches (with a performance penalty). By varying CPU threads, context switches among active CUDA-threads are very fast. Typically one CUDA-thread processes one element of the data-set of the application. This helps to exploit the available parallelism of the algorithm and to hide the latency by swapping among threads waiting for data coming from memory and threads ready to run. This structure remained stable across generations, with several enhancements implemented in the most recent architectures making available more registers addressable to each CUDA-thread. Considering each generation of NVIDIA

architecture, some minor differences occurred. The C2050 and C2070 boards based on the *Fermi* processor architecture differ in the amount of available global memory. Both cards have a peak performance of ≈ 1 Tflops in single-precision (SP), and ≈ 500 Gflops in double-precision (DP), and the peak memory bandwidth is 144 GB/s.

The K20, K40 and K80 are boards based on the *Kepler* architecture. The K40 processor has more global memory than the K20 and slightly improves memory bandwidth and floating-point throughput, while the K80 has two enhanced *Kepler* GPUs with more registers and shared memory than K20/K40 and extended GPUBoost features. On the *Kepler* K20 and K40, the peak SP (DP) performance is ≈ 5 Tflops (≈ 1.5 Tflops), while on the K80 the aggregate performance of the two GPUs delivers a peak SP (DP) of ≈ 5.6 Tflops (≈ 1.9 Tflops). The peak memory bandwidth is 250 and 288 GB/s respectively for the K20X and the K40 while on the K80 the aggregate peak is 480 GB/s.

The P100 board is based on the *Pascal* architecture, engineered to tackle memory challenges using stacked memory, a technology which enables multiple layers of DRAM components to be integrated vertically on the package along with the GPU. The P100 is the first GPU accelerator to use High Bandwidth Memory 2 (HBM2) to provide greater bandwidth, more than twice the capacity, and higher energy efficiency, compared to off-package GDDR5 used in previous generations. The SXM-2 version of P100 board also integrates the NVLinks, NVIDIA's new high-speed interconnect technology for GPU-accelerated computing significantly increasing performance for both GPU-to-GPU communications, and for GPU access to system memory. The P100 delivers a peak performance of ≈ 10.5 Tflops SP and ≈ 5.3 in DP, while the peak memory bandwidth has been increased to 732 GB/s.

The *Volta* architecture has been developed and engineered for the convergence of HPC and AI. Key compute features of Tesla V100 include new SM Architecture Optimized for Deep Learning, integrating Tensor Cores designed specifically for deep learning. Also the Tesla V100 board integrate second-generation of NVLink supporting up to 6 links at 25 GB/s for a total of 300 GB/s, and1 6GB of HBM2 memory subsystem delivering 900 GB/sec peak memory bandwidth provides 1.5x delivered memory bandwidth versus Pascal GP100. *Volta* increases the computing throughput to 7.5 Tflops DP, and the memory bandwidth to 900 GB/s, respectively a factor 1.4X and 1.2X w.r.t. the Pascal architecture.

The *Ampere* architecture adds powerful new generation of Tensor Core that boosts throughput over V100 for Deep Learning applications running 10x faster. The peak performance in DP has been increased to 9.7 TFlops, and to 19.5 TFlops using Tensor Core or single precision FP32 operations. The A100 40 GB of high-speed HBM2 memory with a peak bandwidth of 1555 GB/sec, corresponding to a 73% increase compared to Tesla V100. It also support a third-generation of NVIDIA NVLink with a data rate of 50 Gbit/sec per signal pair, nearly doubling the 25.78 Gbits/sec rate in V100.

The *Hopper* is the latest architecture developed by NVidia providing a new generation of streaming multiprocessors with several new features. Tensor Cores are up to 6x faster chip-to-chip compared to A100, the memory subsystem is based on HBM3 modules providing nearly a 2x bandwidth increase over the previous generation, and integrate a fourth-generation of NVlinks providing a 3x bandwidth increase. The peak performance is boosted up to 24 TFlops in DP, and 48 TFlops using FP64 tensor core and FP32 operations. The H100 SXM5 GPU raises the bar considerably by supporting 80 GB (five stacks) of fast HBM3 memory, delivering over 3 TB/sec of memory bandwidth, effectively a 2x increase over the memory bandwidth of A100 that was launched just two years ago. The PCIe H100 provides 80 GB of fast HBM2e with over 2 TB/sec of memory bandwidth. The H100 also introduces DPX instructions to accelerate the performance of Dynamic Programming algorithms. These new instructions provide support for advanced fused operands for the inner loop of many dynamic programming algorithms. This leads to dramatically faster

Architecture GPU	Fermi GF100	Kepler GK110	Kepler GK110B	Kepler GK210 × 2	Pascal P100	Volta V100	Ampere A100	Hopper H100
Year	2011	2012	2013	2014	2016	2017	2021	2022
#SMs	16	14	15	13 × 2	56	80	108	132
#CUDA-cores	448	2688	2880	2496 × 2	3584	5120	6912	16896
Base clock (MHz)	1.15	735	745	562	1328	1370	1700	1600
Base DP (Gflops)	515	1310	1430	935 × 2	4755	7000	9700	30000
Total available memory (GB)	3	6	12	12 × 2	16	16	40	80
Memory bus width (bit)	384	384	384	384 × 2	4096	4096	5120	5120
Peak mem. BW (GB/s)	144	250	288	240 × 2	732	900	1555	3072

Table 2. Summary of hardware features of NVIDIA GPU architectures.

times-to-solution in disease diagnosis, logistics routing optimizations, and even graph analytics. For a more complete description, we can refer to [1, 3, 18, 71, 107, 149], while the work in Table 2 summarizes just a few relevant parameters of NVIDIA GPU architectures.

GPUs can execute multiple, simultaneous computations. This enables the distribution of training processes and can significantly speed up the ML operations. With GPUs, you can accumulate many cores that use fewer resources without sacrificing neither efficiency nor power.

When designing a deep learning architecture, the decision to include GPUs relies on several factors as follows:

- Memory bandwidth: including GPUs can provide the bandwidth needed to accommodate large datasets. This is because GPUs include dedicated video RAM (VRAM), enabling to retain CPU memory for other tasks.
- Dataset size: GPUs in parallel can scale more easily than CPUs, enabling to process massive datasets faster. The larger your datasets are, the greater the benefit you can gain from GPUs.
- Optimization: a downside of GPUs is that optimization of long-running individual tasks is sometimes more difficult than with CPUs.

The performance of GPU accelerators could be compared in different ways. As first approximation, their theoretical peak performance and memory bandwidth could be used, as shown in Table 2. Anyhow several other architectural characteristics could affect the final performance of an actual algorithm implementation. In fact, to get a better overview of their expected performance, running a specific workload, it could be preferable to use reference benchmarks, possibly made of representative sets of commonly used algorithms implementations. For this reason, different benchmarks have been developed, each of them able to test the obtainable performance with respect to a given workload characteristic, or a given set of application kernels. In the context of machine learning, one of the most used benchmark is MLPerf [181], which have a specific set of training phase tasks [180]. Its results on two different systems, embedding the latest GPU architecture and its predecessor (i.e. Nvidia Hopper and Ampere) are shown in Table 4, highlighting on average an approximate 2× factor of performance improvement.

Different vendors, like AMD and Intel, have also developed GP-GPU architectures mostly orientend to HPC and more recently to AI computing. Yet the terminology used by different vendors is not the same, they share most of the hardware details. For example AMD names Compute Unit what NVIDIA calls Streaming Multiprocessor and Intel calls Compute Slice or Execution-Unite (EU). Further, NVIDIA names Warp the set of instructions scheduled and executed at each cycle, while AMD uses the term Wavefront, and Intel uses the term EU-Thread. Concerning the execution model, NVIDIA uses the Single Instruction Multiple Thread (SIMT), while AMD and Intel use the Single

Instruction Multiple Data (SIMD) [140]. In Table 3, we report the main hardware features of the three most recent GP-GPU architectures developed by NVIDIA H100 [18], AMD [15] and Intel [120]. We compare the peak performance related to the 32-bit single- and 64-bit double-precision, and the peak performance achieved using half-precision.

3.1.2 GPU-based Platforms for AI. Over the last years, Nvidia deployed the DGX [196] line of server and workstation platforms specialized in using GPUs to accelerate deep learning applications. The DGX systems are based on high-performance commodity CPUs, and a set of GPUs interconnected using a motherboard integrated network based on high speed NVLink [197] technology developed by NVidia. The number of GPU modules varies from 4 to 16 Tesla daughter cards integrated into the system using a version of the high-bandwidth SMX[219] socket solution. The DGX-1 server, the first of DGX line, was announced in 2016, and it was first based on 8 Pascal cards, after upgraded to Volta, interconneced by an NVLink mesh network. The Pascal based DGX-1 delivered 170 TFlops using FP16 half-precision processing, while the Volta based upgrade increased this to 960 TFlops unsing FP16 tensor computing. The DGX-2, the successor of DGX-1, was announced in 2018; it is based on 16 V100 32 GB GPU cards in a single unit interconnected by a NVSwitch [197] for high-bandwidth GPU-to-GPU communications, and delivers nearly 2 PFlops using FP16 tensor processing, ans assemble a total of 512 GB of HBM2 memory. The DGX Station is a workstation designed as a deskside AI system that can operate completely independent without the typical infrastructure of a datacenter. The DGX Station is a tower chassis, and the first available was including four Testa V100 accelerators each with 16 GB of HBM2 memory, delivering an aggregate computing performance of nearly 500 TFops using FP16 tensor computing. The Ampere version of the DGX Station include four A100 accelerators configured with either 40 or 80 GB of memory each, resulting either in 160 GB or 320 GB variants, and a peak FP16-tensor computing performance of approximately 1 PFlops. The DGX A100 server is the 3rd generation of DGX servers announced in 2002. It includes 8

Model Vendor	H100 NVIDIA	Instinct MI250X AMD	Arc 770 Intel
#physical-cores	132	220	32
#logical-cores	16896	14080	4096
Clock (GHz)	1.6	1.7	2.4
Peak perf. DP (TF)	30	47.9	4.9
Peak perf. SP (TF)	60	95.8	19.7
Peak perf. FP16 (TF)	120	383	39.3
Max Memory (GB)	80 HBM2e	128GB HBM2e	16GB GDDR6
Mem BW (TB/s)	2.0	3.2	0.56
TDP Power (Watt)	350	560	225

Table 3. Selected hardware features of most recent GP-GPU systems developed by NVidia, AMD and Intel

	ImageNet ResNet	KiTS19 3D U-Net	OpenImages RetinaNet	COCO Mask R-CNN	LibriSpeech RNN-T	Wikipedia BERT	Go Minigo
8 × A100	30.8	25.6	89.1	43.1	32.5	24.2	161.6
8 × H100	14.7	13.1	38.0	20.3	18.2	6.4	174.6

Table 4. MLPerf Training v2.1 Benchmark Results (minutes)

Platform	#GPUs	FP16 Tensor	F32	FP64
DGX1-P100	8x P100	–	85	42
DGX1-V100	8x V100	1000	124	62
DGX2	16x V100	2000	248	124
DGX-A100 Server	8x A100	2496	154	77
DGX-H100 Server	8x H100	16000	544	272
Supercomputer				
Selene	2240x A100	698880	43120	21560
Eos	4608x H100	9216000	313344	156672

Table 5. Performance in TFlops of DGX based platforms; for H100 platforms, sparsity features are used.

A100 accelerators, and it is the first DGX server replacing the Intel Xeon CPUs with the AMD EPYC CPUs, delivering a peak FP16-tensor computing performance of approximately 2.5 PFlops. The DGX H100 Server has been announced in 2022, and it is the 4th generation of DGX servers. It includes 8 Hopper H100 cards delivering a total of 16 PFlops of FP16-tensor AI computing, and assembling a total of 640 GB of HBM3 memory. The DGX SuperPod is a high performance turnkey supercomputer solution based on DGX hardware, combining high performance DGX compute nodes with fast storage and high bandwidth networking, that can be used as building-block to assemble large supercomputer systems. The Selene Supercomputer, installed at the Argonne National Laboratory, is one example of a DGX SuperPod based system, built from 280 DGX A100 nodes. The new version of SuperPod based on H100 DGX can scale up to 32 nodes, for a total of 256 H100 GPUs and 64 x86 CPUs. This gives the complete SuperPod a total 20TB of HBM3 memory, 70.4 TB/s of bisection bandwidth, and up to 1 EFlop of FP8 and 500 PFlops of FP16 tensor AI compute. The Eos[195] supercomputer announced in March 2022, designed, built, and operated by Nvidia, is based on 18 H100 SuperPods, for a total of 576 DGX H100 systems. This allows Eos to deliver approximately 18 EFlops of FP8 and 9 EFLOPs of FP16 compute, making Eos the fastest AI supercomputer in the world. Table Table 5 summarizes the computing performance of few DGX systems. We report the peak computing performance using tensor FP16 operations relevant for AI applications, and the standard FP32 and FP64 relevant for many scientific applications.

3.2 TPU-based accelerators

Tensor Processing Units (TPUs) dedicated to training and inference have been proposed very early after the emergence of the first large CNN-based applications. This is due to the observation that these workloads are dominated by linear algebra kernels that can be refactored as matrix multiplications (particularly if performed in batches) and that their acceleration is particularly desirable for high-margin applications in datacenters. More recently, the emergence of exponentially larger models with each passing year (e.g., the GPT-2, GPT-3, GPT-4 Transformer-based large language models) required a continuous investment in higher-performance training architectures in data centers.

Google showcased the first TPU [133, 134] at ISCA in 2017, but according to the original paper the first deployment occurred in 2015 – just three years after the "AlexNet revolution". The architecture of the TPU is centered on a large (256×256) systolic array operating on signed or unsigned 8-bit integers and targeting exclusively data center inference applications; this is coupled with a large amount of on-chip SRAM for activations (24 MiB) and a high-bandwidth (30 GiB/s) dedicated path to off-chip L3 DRAM for weights. The next design iterations (TPUv2, TPUv3) [135] forced to

move from an inference-oriented design to a more general engine tuned for both inference and training, employing the 16-bit BF16 floating-point format, more cores (2 per chip) using each one or two 4× smaller arrays than TPUv1 (128×128, to reduce under-usage inefficiencies). TPUv2/v3 also introduced high-bandwidth memory support, which results in more than 20× increase in the available off-chip memory bandwidth.

In 2019, Habana Labs and Intel proposed Goya and Gaudi as microarchitectures for the acceleration of inference [182]. Goya relies on PCIe 4.0 to interface to a host processor and exploits a design that uses a heterogenous approach comprising of a large General Matrix Multiply (GMM) engine, TPUs, and a large shared DDR4 memory pool. Each TPU also incorporates its own local memory that can be either hardware-managed or fully software-managed, allowing the compiler to optimize the residency of data and reducing movement. Each of the individual TPUs is a VLIW design that has been optimized for AI applications. The TPU supports mixed-precision operations including 8-bit, 16-bit, and 32-bit SIMD vector operations for both integer and floating-point. Gaudi has an enhanced version of the TPUs and uses HBM global memories rather than the DDR used in Goya, increasing the support towards bfloat16 data types and by including more operations and functionalities dedicated for training operations.

While Google and Intel rely on a mixture of in-house designs and GPUs, the other main data center providers typically relied on NVIDIA GPUs, as discussed above, to serve Deep Learning workloads. Starting from the Volta architecture [52] and continuing with Ampere [53] and Hopper [51, 72], NVIDIA has embedded inside the GPU Streaming Multiprocessors the counterpart of smaller TPUs, i.e., *TensorCores*. Following the GPU architectural template, NVIDIA TensorCores are small units, designed to perform a 4×4×4 FP16 GEMM operation per cycle in Volta (doubled in Ampere and quadrupled in Hopper, adding also support for other data types). Performance is then obtained by parallelisation: each Streaming Multiprocessor includes eight TensorCores controlled by 32 threads; and, depending on the specific chip, GPUs can contain tens of Streaming Multiprocessors.

GraphCore Colossus Mk1 and Mk2 IPUs [127, 147] target specifically the niche of Graph Neural Networks (as well as DNNs and Transformers) training employing a tiled many-core architecture of relatively simple processors. GraphCore focuses on a highly power- and cost-efficient memory hierarchy that does not rely on high-bandwidth off-chip HBM, but on cheaper DRAM chips combined with a large amount of on-chip SRAM (in the order of 1 GiB per chip). According to GraphCore, this design achieves ~2× the energy efficiency of an NVIDIA Ampere GPU and ~3× that of a Google TPUv3 on sustained workloads.

Concerning academic and research-proposed architectures, IBM Research focused on introducing techniques to reduce the precision of data formats used for training [8, 272], introducing Hybrid-FP8 formats in training ASICs and tensor processors. A similar effort is performed by the authors of Cambricon-Q [300], which also introduce further improvements to exploit the statistical properties of tensors to minimize bandwidth consumption and maximize efficiency. Finally, Gemmini [89, 94] and RedMulE [259, 260] are efforts to introduce tensor processor hardware IPs (respectively, generated from a template and hand-tuned) that can be integrated inside System-on-Chips, similarly to what NVIDIA does with TensorCores.

4 HARDWARE ACCELERATORS

Typical HPC workloads, like genomics, astrophysics, finance, and cyber security, require the elaboration of massive amount of data and they can take advantage of DL methods with results that can surpass human ability [21, 95, 232, 247]. However, an ever-increasing computing power, a rapid change of the data analysis approaches, and the introduction of novel computational paradigms are needed. DL models rely on remarkable computational complexities that can be efficiently supported, without renouncing to a good trade-off between speed, energy efficiency, design effort,

and cost, by optimized hardware platforms which are able to provide high levels of parallelism and a considerable amount of memory resources.

These platforms can be developed using CPUs, GPUs, FPGAs, CGRAs and ASICs [66, 69, 95, 170, 175, 266, 282]. CPUs may have higher cache size and higher on-chip bandwidth than GPUs and reconfigurable architectures, but they show a limited ability to process large amounts of data in parallel. On the other hand, with their high throughput and parallelism, GPUs are extremely efficient in terms of performance, but, as a drawback, they consume a lot of power and are much more expensive than their counterparts. Heterogeneous computing platforms based on modern FPGAs achieve moderate speed and consume less energy compared to GPUs, despite limited computing and memory resources [66] [276] [266]. Conversely, ASICs take longer design times, require higher design efforts and do not offer flexibility, but they provide optimum computational speed and power consumption. A good trade-off between speed, power consumption and design effort is offered by CGRAs that exhibit near-ASIC energy efficiency and performances with near-FPGA reconfigurability level.

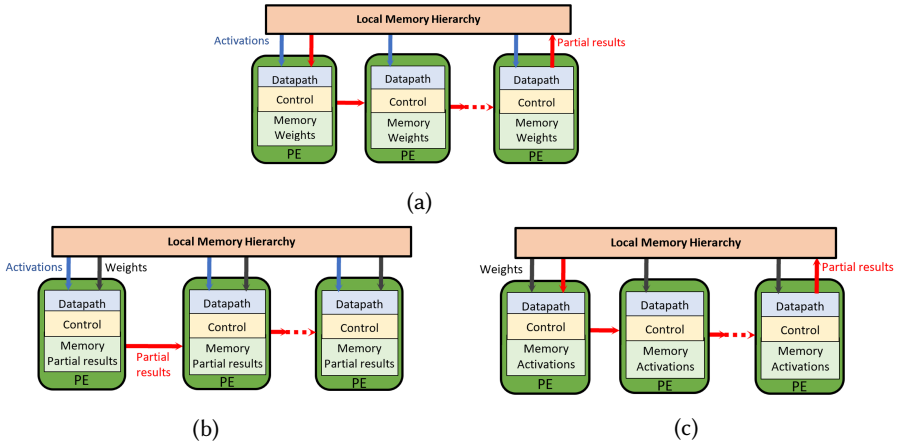

Fig. 3. Dataflows in Deep Learning accelerators: (a) Weights stationary; (b) Output stationary; (c) Input stationary.

Independently of the technology used, a common problem in the design of the accelerators is the high energy cost and delay of accessing off-chip DRAM memory, in particular considering the significant amount of data that the target applications need to process. As schematized in Figure 3, several data reuse and stationary strategies can be exploited to reduce the number of accesses, each strategy offering a certain benefit [38, 100, 208, 212, 230, 251]. For example, in the weight stationary dataflow, convolutional weights (i.e. the filter coefficients) are fixed and stored in the local memory of the Processing Elements (PEs) and reused on the input activations uploaded step-by-step from the external DRAM. Conversely, in the output stationary dataflow, partial outputs produced by the PEs are stored locally and reused step-by-step until the entire computation is completed. Then, just the final results are moved to the external DRAM. An efficient alternative is the input stationary dataflow: in this case, the input activations are stored in the local memory of the PEs while the weights are uploaded from the external DRAM and brought to the PEs.

Another approach common to many accelerator implementations is the use of quantization to reduce the width of data types. Quantization represents an open problem in the implementation of deep learning models on and many studies today address this topic [90] [172]. Integer or fixed-point

data formats are generally preferred over the more computationally intensive floating-point ones. This guarantees better memory occupation, lower computational cost and improves the robustness of the model [130]. Extreme quantization techniques that use only one bit for the data stored (Binary Neural Networks [221]) are widely used for the implementation of very large networks but with comparable accuracy they require 2-11× the number of parameters and operations [263], making them not suitable for complex problems.

4.1 Reconfigurable Hardware Accelerators

FPGAs and CGRAs are highly sought-after solutions to hardware accelerate a wide range of applications, including DL. The main feature of such reconfigurable platforms is the ability to support different computational requirements by repurposing the underlying hardware accelerators also at runtime.

FPGAs are semiconductor devices that provide a unique combination of flexibility and performances thanks to their fundamental building blocks, known as Configurable Logic Blocks (CLBs) or simply Logic Elements (LEs). They consist of look-up tables (LUTs) and flip-flops that can be used to implement arbitrary combinational and sequential bit-level operations, on the basis of user-defined tasks. Programmable interconnects provide the necessary routing resources to establish connections between different elements within the device and to facilitate the seamless flow of data and control signals. Appropriate storage capabilities are also available on-chip as BRAMs and distributed RAM, which serve to implement several storage elements, like data buffers and FIFOs. Moreover, FPGAs provide the designers with specialized macros, such as Digital Signal Processors (DSPs) and embedded multipliers, that can be exploited to enhance processing capabilities, improve power efficiency, and increase flexibility of hardware accelerators for DL. The latter exploit FPGAs mostly to accelerate inference, while training is delegated to GPUs: this reflects the differences between the two phases, as training is only executed once and requires high throughput, while for inference, especially on edge devices, latency and power consumption become critical [24, 99]. FPGAs are also often used as a prototyping platform to explore different architectures before committing to ASIC manufacturing [246].

Several FPGA-based hardware accelerators for DL are structured as heterogeneous embedded systems [173] [163] [10] [291] [214] that mainly consist of: a general-purpose processor, responsible for running software workloads; a computational module, purposely designed to speed up common DL operators, like convolutions [270][223], de-convolutions [39, 234], pooling, fully connected operations, activation and softmax functions [248, 249]; and a memory hierarchy needed to optimize data movement to/from an external DRAM that stores data to be processed and computational results. A typical approach to accelerate convolutions consists of a systolic array architecture (SA), a regular pattern which can be easily replicated [285]. Each PE in the array is a SIMD vector accumulation module to which inputs and weights are supplied in each cycle by shifting from the horizontally and vertically adjacent PE (Figure 4a). The use of pipelined groups of PE with short local communication and regular architecture enables a high clock frequency and limited global data transfer (Figure 4b).

Although FPGAs have traditionally been proposed as accelerators for edge applications, they are starting to be adopted also in datacenters. Microsoft's Project Brainwave [79] uses several FPGA boards to accelerate the execution of recurrent neural networks in the cloud, exploiting the reconfigurability to adapt the platform to different DL models. One way to face the limitations imposed by the capability of FPGAs to effectively map very large DL models is to use a deeply pipelined multi-FPGA design. Recent studies focus on optimizing this type of architecture and maximizing the overall throughput [296] [224][236].

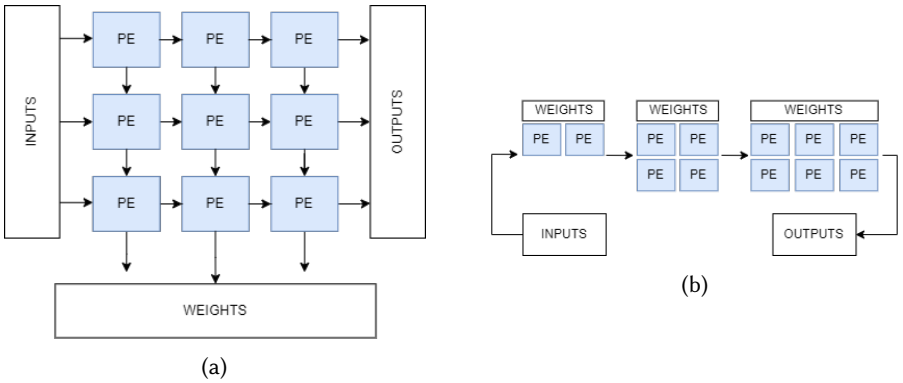

Fig. 4. FPGA accelerators: (a) Systolic array accelerator; (b) Pipelined dataflow accelerator.

In these applications contexts, CGRAs represent an alternative to FPGAs, providing reconfigurability with coarser-grained functional units. They are based on an array of processing elements (PEs), performing basic arithmetic, logic, and memory operations at word level and using a small register file as temporary data storage. Neighboring PEs are connected to each other through reconfigurable routing that allows to transfer intermediate results of the computations towards the proper neighbors for the next computational step. CGRAs can represent a powerful solutions to accelerate dense linear algebra applications, such as ML, image processing, and computer vision [34, 86]. In fact, thanks to parallel computing and time-multiplexing, CGRAs can efficiently support and combine spatial and temporal computational models. Furthermore, they are flexible enough for specific domains, and their interconnections, being not as complex as those present on FPGAs, provide remarkable advantages in terms of speed, energy-efficiency, and resources utilization.

4.2 ASIC-based Accelerators

4.2.1 Neural Processing Units (NPUs). An NPU is processing architecture that includes all the control and the arithmetic logic components necessary to accelerate the performance and improve the energy-efficiency [243] of common DL tasks such as image classification, object detection, and many more [116], which are of paramount importance from edge and mobile computing to high-performance computing. The purpose of an NPU is to accelerate a segment of a program (e.g., a fully-connected layer of a large neural network) offloading the CPU. In particular, the NPU is designed to accommodate a reasonable amount of multiply/accumulate (MAC) units, that are the fundamental blocks devised in the convolutional and fully-connected layers of deep neural networks [45, 65].

Table 6. Summary of NPU accelerators.

NPU	Process	Area [mm^2]	Supply voltage [V]	Max. Freq. [MHz]	PP [TOPS]	Max EE [TOPS/W]	Max AE [TOPS/mm^2]
Samsung [243]	8 nm	5.5	0.8	933	6.9	3.4	1.25
UM+NVIDIA [297]	16 nm	2.4	0.8	480	-	3.6	-
MediaTek [167]	7 nm	3.04	0.825	880	3.6	6.55	1.18
Alibaba [129]	12 nm	709	-	700	825	499	1.16
Samsung [206]	5 nm	5.46	0.9	1196	29.4	13.6	2.69
Samsung [207]	4 nm	4.74	1	1197	39.3	11.59	6.9

Each PE contains a synaptic weight buffer and the MAC units to perform the computation of a neuron, namely, multiplication, accumulation, and an activation function (e.g., sigmoid). A PE can

be realized entirely with a full-CMOS design or by using emerging non-volatile memories such as RRAM and PCM to perform *in situ* matrix-vector multiplication as in the RENO chip [171] or as in the MAC units proposed in [192, 289]. The advantage of these architectures is that only the input and final output are digital; the intermediate results are all analog and are coordinated by analog routers. Data converters (DACs and ADCs) are required only when transferring data between the NPU and the CPU with an advantage in terms of energy-efficiency ([289] reports an energy efficiency of 53.17 TOPS/W), although there are insufficient experimental data to support this evidence in comparison with full-digital NPUs. In Table 6, we reported the main features of several full-digital NPUs designs by also highlighting their Peak Performance (PP), Energy Efficiency (EE), and Area Efficiency (AE).

4.2.2 Single-chip NPUs. In the DNN landscape, single-chip domain-specific accelerators have achieved a great success in both cloud and edge scenarios. These custom architectures offer better performance and energy efficiency with respect to CPUs/GPUs thanks to an optimized dataflow (or data reuse pattern) that reduces off-chip memory accesses, while improving the system efficiency [44].

The DianNao series is an example of a full digital stand-alone DNN accelerator that introduces a customized design to minimize the memory transfer latency and to enhance the system efficiency. DaDianNao [43] targets the datacenter scenario and integrates a large on-chip embedded dynamic random access memory (eDRAM) to avoid the long main memory access time. The same principle applies to the embedded scenario. ShiDianNao [43] is a DNN accelerator dedicated to CNN applications. Using a weight sharing strategy, its footprint is much smaller than the previous design. It is possible to map all of the CNN parameters onto a small on-chip static random access memory (SRAM) when the CNN model is small. In this way, ShiDianNao avoids expensive off-chip DRAM access time and achieves a 60 times energy efficiency compared to DianNao.

Furthermore, domain-specific instruction set architectures (ISAs) have been proposed to support a wide range of NN applications. Cambricon [298] and EIE [104] are examples of architectures that integrate scalar, vector, matrix, logical, data transfer, and control instructions. Their ISA considers data parallelism and the use of customized vector/matrix instructions.

Eyeriss is another notable accelerator to discuss [45]. The architecture is that of a CNN accelerator that can support high throughput inference and optimize the system-level energy efficiency, also including off-chip DRAMs. The main features of Eyeriss are a spatial architecture based on an array of 168 processing elements (PEs) that creates a four-level memory hierarchy, a dataflow that reconfigures the spatial architecture to map the computation of a given CNN and optimize towards the best energy efficiency, a network-on-chip (NoC) architecture that uses both multi-cast and point-to-point single-cycle data delivery, and run-length compression (RLC) and PE data gating that exploit the statistics of zero data in CNNs to further improve energy efficiency.

In [65], STMicroelectronics presented Orlando system-on-chip, a 28nm FDSOI-based CNN accelerator integrating an SRAM-based architecture with low-power features and adaptive circuitry to support a wide voltage range. Such DNN processor provides an energy-efficient set of convolutional accelerators supporting kernel compression, an on-chip reconfigurable data-transfer fabric, a power-efficient array of DSPs to support complete real-world computer vision applications, an ARM-based host subsystem with peripherals, a range of high-speed I/O interfaces, and a chip-to-chip multilink to pair multiple accelerators together.

IBM presented a processor core for AI training and inference tasks applicable to a broad range of neural networks (i.e., CNN, LSTM and RNN) [198]. High compute efficiency is achieved for robust fp16 training via efficient heterogeneous 2-D systolic array-SIMD compute engines that leverages DLFloat16 FPUs. A modular dual-corelet architecture with a shared scratchpad memory

and a software-controlled network/memory interface enables scalability to many-core SoCs for scale-out paradigms. In 2022, IBM also presented a 7-nm four-core mixed-precision AI chip [159] that demonstrates leading-edge power efficiency for low-precision training and inference without model accuracy degradation. The chip is based on high-bandwidth ring interconnect to enable efficient data transfers, while workload-aware power management with clock frequency throttling maximizes the application performance within a given power envelope.

Qualcomm presented an AI core that is a scalar 4-way VLIW architecture that includes vector/tensor units and lower precision to enable high-performance inference [40]. The design uses a 7 nm technology and is sought to be integrated in the AI 100 SoC to reach up to 149 TOPS with a power efficiency of 12.37 TOPS/W.

Table 7. Summary of single-chip digital DNN accelerators

Accelerator	Technology	Application	Area [mm^2]	Power [mW]	Performance [GOPS]	EE [GOPS/W]
DaDianNao [43]	28 nm	DNN	67.7	15970	5585	-
ShiDianNao [43]	65 nm	CNN	4.86	320	194	-
Cambricon [298]	65 nm	CNN	6.38	954	1111.92	-
EIE [104]	28 nm	CNN+LSTM	63.8	2360	1.6	177.78
Eyeriss [45]	65 nm	CNN	16	450	33.6	74.7
STM [65]	28 nm	CNN	34.8	39	750	2900
IBM [198]	14 nm	CNN+LSTM+RNN	9.84	-	3000	1100
IBM [159]	7 nm	CNN+RNN	19.6	-	16300	3580

4.3 EDA Frameworks

Implementing hardware accelerators for ML algorithms, particularly DNNs, is a complex task that is rarely addressed through manual coding in low-level Hardware Description Languages (HDL). When Register Transfer Level (RTL) design is required to achieve high performance, templated components may be used [174]. Instead, there are several electronic design automation (EDA) tools that bridge the gap between ML models and FPGAs/ASICs, allowing researchers to focus on developing the algorithms at a high level of abstraction [271].

Vitis AI, Xilinx's development environment for AI inference [16], supports models developed in major frameworks such as PyTorch [209], TensorFlow [7] and Caffe [126], and maps them on deep learning processor unit (DPU) cores present on modern Xilinx boards alongside the standard FPGA logic. The work in [261] describes the implementation of DeepSense, a framework that includes CNN and RNN, with a focus on the choice of parameters to define DPUs used by Vitis AI; [268] performs a parametric study of the DPU architecture used by Vitis AI and examines the tradeoffs between the resources used and the clock frequency, as well as their impact on power consumption; [279] compares the FPGA implementation of YOLOv3 provided by Vitis AI with its GPU counterpart, showing higher throughput and lower power consumption; [265] evaluates the implementation of three different CNNs in terms of precision, power consumption, throughput, and design man-hours, and compares these figures with their GPU counterparts.

High-Level Synthesis (HLS) plays a crucial role to automate the design of ML accelerators. HLS tools such as Vitis HLS [287], Bambu [77], Intel HLS Compiler [121], Catapult [241], Stratus HLS [31], or LegUp [32] provide users with a high level of abstraction where they can describe the desired functionality with a software programming language (C/C++/SystemC) and automatically obtain a corresponding high-performance HDL implementation. HLS thus boosts the productivity of hardware designers, who can benefit from faster design changes and functional verification. In fact, HLS allows to create accelerators for different platforms (e.g., larger or smaller FPGAs) without altering the C/C++ source code apart from a few design directives; this makes it possible to explore

the design space and find the best implementation much faster than with HDL design. Note that code must be written with hardware knowledge in mind in order to meet given performance and resource usage results. Arbitrary software code, written for a CPU target, could achieve very low performance, since it typically does not expose enough parallelism to exploit the spatial concurrency available on FPGA or ASIC.

In order to explore the acceleration of DNN inference on FPGAs, several frameworks and packages have been developed based on HLS. They can be divided into two categories: tools based on libraries of HLS templates, such as FINN [25] and hls4ml [70], and tools that use a compiler-based approach, such as SODA [26] and ScaleHLS [292]. In [176], a comparison between a custom implementation of two DNNs written in SystemVerilog and an implementation using the Xilinx tools FINN and Vitis AI is presented; a comparison between FINN and Vitis AI is reported in [103], where a ResNet model is implemented using a widely used set of configurations of FINN and Vitis AI. Both FINN and hls4ml use Vitis HLS as a backend; they parse a model exported from high-level ML frameworks and replace operators with C/C++ functions taken from a library of templates that already contains Vitis optimization directives. The HLS tool processes the C/C++ code and produces a corresponding accelerator design. The library of templates is necessarily tied to a specific HLS tool, and it requires expert HLS developers to implement in advance the best version of all necessary ML operators for a pre-determined backend tool. On the other hand, SODA and ScaleHLS use a compiler infrastructure (MLIR, the Multi-Level Intermediate Representation from the LLVM project [155]) to progressively translate the input model through representations at different levels of abstraction, until they can be passed to the HLS tool as a C++ representation or an LLVM IR. This second approach exploits the existing MLIR infrastructure for machine learning, without requiring to create and maintain a library of operators. A hybrid RTL–HLS approach has been proposed in [97] to improve performance and development time for various DL algorithms.

4.4 Accelerators based on open-hardware RISC-V

RISC-V is an open-source, modular instruction set architecture (ISA) that is gaining popularity in computer architecture research due to its flexibility and suitability for integration with acceleration capabilities for deep learning. The RISC-V ISA is designed with a small, simple core that can be extended with optional instruction set extensions (ISEs) to support various application domains.

RISC-V offers several advantages for deep learning acceleration research. First, the modular nature of the ISA allows researchers to easily integrate acceleration capabilities as ISEs, which can be customized to suit the specific needs of different deep learning models. Second, RISC-V supports a range of standard interfaces, such as AXI4, that can be used to interface with external acceleration units integrated on the same System-on-Chip at various levels of coupling. This makes it easy to integrate specialized hardware accelerators into RISC-V-based systems for deep learning. Moreover, the defining feature of the RISC-V ISA is its openness, meaning that anybody can design a RISC-V implementation without paying royalties or needing a particular license. Thanks to this non-technical advantage with respect to other ISAs (ARM, x86), RISC-V has gained significant traction from academia and from emerging startups. Due to the concurrent rise of Deep Learning, many specialized architectures dedicated to Deep Learning have been based on RISC-V.

Fig. 5 reports a synthetic taxonomy of the RISC-V based architectures for Deep Learning acceleration discussed in the present section, organized by the way that they are coupled with the core: from the tightest coupling (directly extending the core's ISA) to the loosest (sharing memory at L3). We use the same taxonomy in the following to discuss these architectures.

4.4.1 RISC-V ISA extensions for (Deep) Learning. The RISC-V ISA standard uses a modular design composed of basic instruction sets addressing the general purpose computing, e.g., base 32-bit

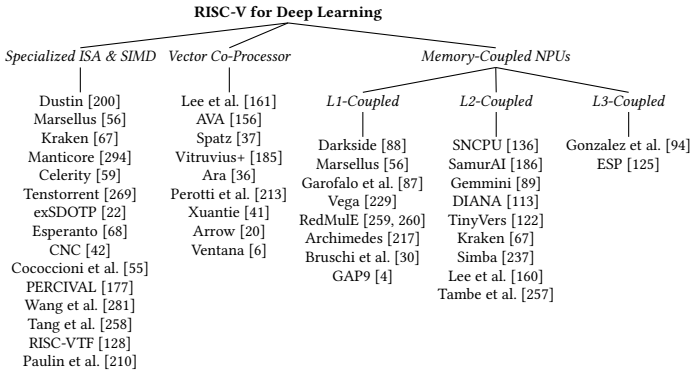

Fig. 5. Taxonomy of RISC-V based acceleration units discussed in Section 4.4

integer operations, plus a variety of extensions that can be seamlessly integrated to provide enhancements for specific computing needs, e.g., the H-extension for virtualization. This flexible design together with the openness of the standard brought forward a number of studies proposing custom extensions to accelerate specific applications. Due to the flexibility of this architectural template, the diversity of solutions proposed is significant. Here we review extensions aiming at (deep) learning.

(Deep) neural networks are often limited by the computing and memory resources used up by their large number of weights. Weight compression via alternative or quantized number representations is often adopted to speed up and optimize the performance of neural network models. [55, 177] propose ISA extensions for posit numbers which can be used to do weight compression. Posits are an alternative representation to the standard IEEE floating point formats for real numbers. Posits need fewer bits to obtain the same precision or dynamic range of IEEE floats allowing them to store more weights in a same-sized memory. For example, [55] provide an efficient conversion between 8- or 16-bit posits and 32-bit IEEE floats or fixed point formats with little loss in precision leading to a 10x speedup in inference time. Other works directly address the compute-intensive parts of different neural networks, in particular convolutional neural networks (CNNs), graph convolutional networks (GCN) and transformers. [281] proposes a new Winograd-based convolution instruction to speed up the time-consuming convolutional layers in CNNs. The matrix convolution between the CNN kernel and the input data can not be executed efficiently using the standard RISC-V instructions. The proposed extension allows to compute a convolution producing a 2×2 output using a 3×3 kernel on a 4×4 input in a single instruction using 19 clock cycles instead of multiple instructions totaling 140 cycles using the standard RISC-V ISA. [258] addresses the computational bottlenecks of GCNs. They design a set of general-purpose instructions for GCNs specifically addressing the compute inefficiencies in aggregating and combining feature vectors. As such the authors combine the software programmability given by the RISC-V ISA with the compute efficiency of GCN accelerators. Similarly, [128] focuses on transformer models. Notably, the extension comprises instructions to accelerate the popular ReLU activation and softmax functions. Paulin et al. [210] performs a similar task but focuses on Recurrent Neural Networks (RNNs).

Following the industry and academia trend to use aggressive quantization schemes to accelerate inference of Deep Neural Networks, many ISA extensions focus on low-bitwidth arithmetics to implement Quantized Neural Networks (QNNs), often combined with multi-core parallel execution to further boost performance and efficiency. Several developments in SoA augment the PULP RI5CY core used for example in Vega [229] to improve its energy efficiency on QNNs. Marsellus [56] (16

cores) and Kraken [67] (8 cores) use *Xpulpnn*, an ISA extension for low-bitwidth (2/4-bit) integer dot-products used to accelerate symmetric precision QNNs. Dustin [200] (16 cores) also exploits a similar concept, but it also introduces a lockstep mechanism to operate all the cores in a SIMD fashion, further increasing their efficiency.

Manticore [294], Celerity [59], Esperanto [68] and Tenstorrent [269] exploit ISA extensions for faster RISC-V based Deep Learning workload execution in the context of many-core architectures where a large number of very simple cores cooperate. As these architectures are targeted at server-based training as well as inference, they typically focus on floating point multiply-accumulate and dot-product operations, such as exSDOTP [22].

4.4.2 RISC-V Vector Co-processors. The main proponents of the RISC-V architecture have traditionally been strong proponents of Cray-style vector co-processors as an alternative to packed-SIMD extensions used in many competitor architectures (see e.g., Lee et al. [161]). Following this line of thought, vector co-processors have been from the start a natural architectural target for Deep Learning-oriented RISC-V acceleration. AVA [156], Vitruvius+ [185], Ara [36, 213] are research-based vector co-processors meant to accelerate the full RISC-V *V* extension for general-purpose vectorizable applications; while these include Deep Learning, such attempts (generally focusing on 64-bit computation) are not particularly tailored in terms of energy efficiency when compared to more compact solutions specifically tuned for DNNs. Commercial RISC-V vector processors, mainly targeted at High-Performance Computing markets, have recently started appearing, such as Xuantie [41] and Ventana [6]. Spatz [37] and Arrow [20], on the other hand, are examples of vector co-processors explicitly tailored for Deep Learning. The former in particular focuses only on a subset of the *V* extension and on 32-bit data, capturing better opportunities for energy efficiency.

4.4.3 RISC-V Memory-coupled Neural Processing Units (NPUs). Memory-coupled NPUs have been very often connected to RISC-V processors, exploiting the latter's availability to engineer innovative architectural templates and solutions. Concerning the tightest kind of memory coupling, at L1, most proposals in the state-of-the-art are based on the PULP template, and devote significant effort to enabling fast communication between RISC-V cores and accelerators. Vega [229] is a prototype system based on the Parallel Ultra-Low Power (PULP) multi-core template coupling 9 RI5CY cores with a quantized DNN convolutional NPU sharing directly L1 scratchpad memory with the cores. GreenWaves Technologies GAP9 [4] is a commercial product, targeted at the hearables market, that follows the same line with many architectural improvements and a redesigned AI NPU for QNNs – leading to the product achieving the best performance and energy efficiency in the public TinyMLPerf Tiny 1.0 challenge as of this writing [5]. Archimedes [217] proposes a large AI hardware accelerator for performance-hungry Extended Reality applications. RedMulE [259, 260], integrated in the Darkside prototype [88], follows the same principles but focusing on floating-point computation to support training as well as inference. Garofalo et al. [87] and Bruschi et al. [30] integrate in-memory-computing PCM-based NPUs, the latter simulating a system scaled up to match the size of server-class hardware.

Loosening the memory coupling, i.e., moving the memory shared between cores from L1 to L2/L3, there are many other proposed NPU solutions in the State-of-the-Art. In these instances, RISC-V cores serve mainly as they are well available (also as fully open-source verified cores) and at the same time flexible and easy to integrate into larger systems. Systems exploiting this template include, for example, SNCPU [136], which builds a hybrid system that can act as either a set of 10 RISC-V cores or be reconfigured in a systolic NPU. Gonzalez et al. [94] and Genc et al. [89] exploit a systolic array generation tool, Gemmini, to generate systolic arrays used to accelerate DNNs and coupled with a RISC-V core by exploiting a shared L3 or L2 memory, respectively. Simba [237] follows a similar template, and is also meant to be scaled towards server-grade performance by means of integration

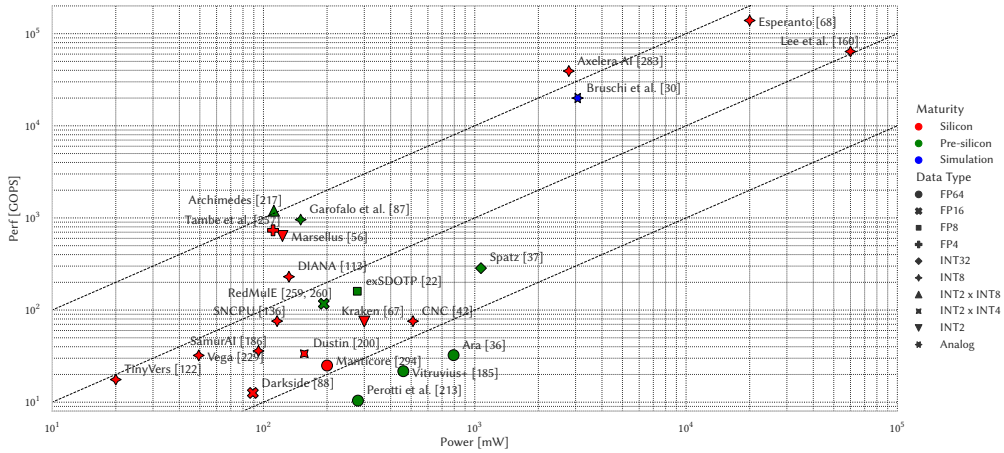

Fig. 6. Performance and power consumption of prototypes of several State-of-the-Art Deep Learning acceleration architectures discussed in Section 4.4. Dot color indicates maturity (silicon, pre-silicon, simulation); dot shape indicates data type employed.

Table 8. Summary of RISC-V Deep Learning acceleration architectures

Category	Accelerator	Tech [nm]	Area [mm2]	Freq [MHz]	Voltage [V]	Power [mW]	Perf [GOPS]	Eff [GOPS/W]	# MAC units	Data Type	Maturity
ISA	Dustin [200]	65nm	10	205	1.2	156	33.6	215	128	INT2 x INT4	Silicon
	Kraken (RISC-V cores) [67]	22nm	9	330	0.8	300	75	750	128	INT2	Silicon
	Manticore [294]	22nm	888	500	0.6	200	25	188	24	FP64	Silicon
	Celerity [59]	16nm	25	1050	-	1900	-	-	496	INT32	Silicon
	Tenstorrent [269]	12nm	477	-	-	-	92000	-	-	FP16	Silicon
	exSDOTP [22]	12nm	0.52	1260	0.8	*278*	160	575	16	FP8	Pre-silicon
	Esperanto [68]	7nm	570	1000	-	20000	139000	6.95	69632	INT8	Silicon
	CNC [42]	4nm	1.92	1150	0.85	510	75.8	149	512	INT8	Silicon
Vector	Lee et al. [160]	14nm	181	2000	0.8	60000	64000	1450	16384	INT8	Silicon
	AVA [156]	22nm	3.9	-	-	-	-	-	-	FP64	Pre-silicon
	Spatz [37]	22nm	20	594	0.8	1070	285	266	256	INT32	Pre-silicon
	Vitruvius+ [185]	22nm	1.3	1400	0.8	459	21.7	47.3	8	FP64	Pre-silicon
	Ara [36]	22nm	10735 kGE	1040	0.8	794	32.4	40.8	16	FP64	Pre-silicon
	Perotti et al. [213]	22nm	0.81	1340	0.8	280	10.4	37.1	4	FP64	Pre-silicon
L1 NPU	Darkside [88]	65nm	3.85	200	1.2	89.1	12.6	152	32	FP16	Silicon
	Marsellus (NPU) [56]	22nm	18.7	420	0.8	123	637	7600	10368 1–bit	INT2	Silicon
	Garofalo et al. [87]	22nm	30	500	0.8	150	958	6390	36 (DW)	INT8	Pre-silicon
	Vega [229]	22nm	12	450	0.8	49.4	32.2	651	27	INT8	Silicon
	RedMulE [259, 260]	22nm	0.73	613	0.8	193	117	608	96	FP16	Pre-silicon
	Archimedes [217]	22nm	3.38	270	0.65	112	1198	10.6	5184	INT2 x INT8	Pre-silicon
	Bruschi et al. [30]	5nm	480	-	-	3070	20000	6500	3.35×10^7	Analog	Simulation
L2 NPU	SNCPU [136]	65nm	4.47	400	1	116	75.9	655	100	INT8	Silicon
	SamurAI [186]	28nm	4.52	350	0.9	94.7	36	380	64	INT8	Silicon
	Gemmini [89]	22nm	1.03	1000	-	-	-	-	256	INT8	Pre-silicon
	DIANA (digital) [113]	22nm	10.24	280	0.8	132	230	1740	256	INT8	Silicon
	DIANA (analog) [113]	22nm	10.24	350	0.8	132	18100	176000	256	INT8	Silicon
	TinyVers [122]	22nm	6.25	150	0.8	20	17.6	863	64	INT8	Silicon
	Simba [237]	16nm	6	161	0.42	-	-	9100	1024	INT8	Silicon
	Axelera AI [283]	12nm	9	800	-	2787	39300	14100	-	INT8	Silicon
	Tambe et al. [257]	12nm	4.59	717	1	111	734	6612	-	FP4	Silicon
L3 NPU	Gonzalez et al. [94]	22nm	16	961	-	-	-	106.1	256	INT8	Silicon
	ESP [125]	12nm	21.6	1520	1	1830	-	-	3x NVDLA	INT8	Silicon

of chiplets on multi-chip modules. ESP [91, 92] and Tambe et al. [257] also focus on integration of hardware accelerators and NPUs in larger-scale Network-on-Chips using RISC-V cores as the primary computing engines. On the other end of the spectrum, SamurAI [186], TinyVers [122], and DIANA [113] build up AI-IoT systems composed of a microcontroller and L2-coupled NPUs (in the case of DIANA both an analog SRAM-IMC-based accelerator and a conventional digital one). Kraken [67] couples the aforementioned RISC-V ISA-extended cluster with specialized L2-coupled Spiking Neural Network (SNN) and Ternary Neural Network (TNN) accelerators.

Fig. 6 summarizes the performance, power, technological maturity and data types used in the architectures discussed in this Section. We can observe that RISC-V-based architectures for Deep Learning occupy essentially the full spectrum of Deep Learning architectures from 10 mW microcontrollers up to 100 W SoC's meant to be integrated as part of high-performance computing systems. Most of the research has, so far, focused on the lower end of this spectrum, striving for the best energy efficiency. We can observe efficiency is strongly correlated with architectural techniques yielding accuracy (e.g., data bitwidth reduction & quantization). Table 8 summarizes all quantitative information available on the discussed architectures; where multiple operating points were reported for a single architecture, the table reports always the highest-performance one (for consistency, this applies to both performance and energy efficiency numbers).

5 ACCELERATORS BASED ON EMERGING COMPUTING PARADIGMS

5.1 Arithmetic Data-paths

Performance achievable with ASIC accelerators for inference of deep learning circuits are mainly dependent on the structure of the arithmetic data-path. At its core deep learning systems perform several finite impulse response operations over a large set of data. Hence the accelerator can be optimized by exploiting the techniques used for the efficient implementation of the underlying arithmetic operations. As shown in Fig. 7, three main types of optimization can be performed on the arithmetic data-path. Convolution is one of the main operations performed in a deep learning system. Mono-dimensional convolution can be efficiently implemented using the approach proposed in [47, 262]. In these papers, the overall result of the convolution is obtained with a reduced number of multiplications. A reduced number of multiplications allows improving the trade-off between the throughput of the circuit and the amount of hardware resources needed for its implementation. The technique has been further developed in [48, 278, 280] where it is applied to multi-dimensional convolution used in neural networks.

The multiplication itself can be implemented with optimized circuits. In [132, 215] the area and power dissipation of the multiplier circuit is reduced by discarding part of the partial products used to compute the result. These circuits trade-off precision and circuit complexity in order to improve the design. As a general methodology this approach is often referred as Approximate Computing Paradigm, providing a way to approximate the design at the cost of a tolerable accuracy loss. The Approximate Computing techniques proposed in [151, 295] provide a reduced complexity multiplier by modifying the way the partial products are computed. In [293] a recursive approach is proposed, in which the multiplier is decomposed into small approximate units. In the approach proposed in [74] the approximation is implemented in the way the partial products are summed.

Finally, the Approximate Computing Paradigm can also be implemented in the 4-2 compressors [9, 102, 148, 203, 254, 290] that represent the atomic blocks used for the compression of the partial products of the multiplier.

Different from the previous works, the segmentation method is aimed at reducing the bit-width of the multiplicands. The papers [108, 267] describe a dynamic segmentation method in which the segment is selected starting from the leading one of the multiplicand binary representation. On the contrary, the paper [191] proposes a static segmentation method, which reduces the complexity of the selection mechanism by choosing between two segments with a fixed number of bits. The paper [255] improves the accuracy of the static segmentation method multipliers by reducing the maximum approximation error, whereas in [165] the authors propose a hybrid approach in which a static stage is cascaded to a dynamic stage.

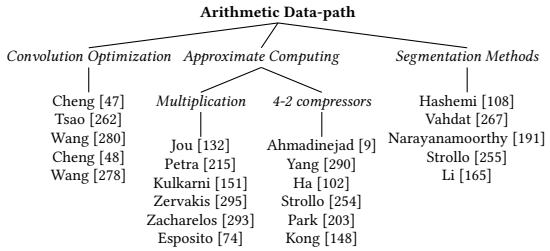

Fig. 7. Taxonomy of the data-path architectures described in Section 5.1

5.2 Accelerators for Sparse Matrices

Sparse matrices represent a main staple for scientific computations and are at the hearth of widely used computational kernels, which are also applied in modern DL workloads. The most famous definition of sparse matrix is attributed to James Wilkinson and dates back to more than 50 years ago [61]: any matrix with enough zeros that it pays to take advantage of them. A more recent and quantitative definition by Filippone et al. [78] states that a matrix is sparse if its number of non-zero coefficients is $O(n)$, where n is the number of rows (columns) of the matrix.

Sparse matrices are usually stored in compressed format in order to avoid redundant storage as well as a lot of useless calculations. That is, the *storage format* attempts to take advantage of the zeros by avoiding their explicit storage. The counterpart is that the traditional simple mapping between the index pair of each matrix coefficient and the position of the coefficient in memory is destroyed. Therefore, all sparse matrix storage formats are devised around means of rebuilding this mapping using some auxiliary index information. This rebuilding has a non-negligible cost and impacts on the matrix operations to be performed. Therefore, the performance of sparse matrix computations depends critically on the selected storage format.

Widely used sparse matrix storage formats include COOrdinate (COO), Compressed Sparse Rows (CSR), and Compressed Sparse Columns (CSC) [78]. For example, CSR, which is perhaps the most popular sparse matrix representation, compresses the sparse matrix into three different arrays. The first one represents the non-zero values, the second contains the column indexes, while the third marks the boundaries of each row in the matrix. The above formats can be considered as general-purpose, meaning that they can be used on most hardware with little or no changes. However, additional and hardware-oriented formats become attractive when moving onto special computing architectures such as accelerators. For example, many storage formats, such as ELLPACK, were specifically developed for vector machines, in order to introduce a certain degree of regularity in the data structure to allow the efficient exploitation of vector instructions.

A factor that can drive the choice of the storage format is the *sparsity pattern* of the matrix that is, the pattern of non-zero entries contained in the matrix. Common sparsity patterns include unstructured (where nonzeros are randomly and irregularly scattered), diagonal (where nonzeros are restricted to a small number of matrix diagonals), and block sparse (either coarse-grain or fine-grain). Each of these sparsity patterns is best addressed using different formats. For instances, the diagonal format (DIA) is an appropriate representation for diagonal matrices.

DNN models are composed of large, dense matrices which are typically used in matrix multiplication and convolutions. In the last years, state-of-the-art DL models have dramatically increased in size, with hundreds of billions of parameters (e.g., large language models as GPT-3 require 175B parameters [29]) and trillions of compute operations per input sample. In order to reduce DNN model sizes and computation requirements (including the energy footprint), *pruning* (i.e., setting to zero)

of DNN weights has emerged as a particularly effective and promising technique. Pruning entails to identify unnecessary redundancy in DNN trained model weights and zero out these nonessential weights [106, 250], thus allowing to discard zero values from storage and computations. Therefore, pruning induces sparsity in the DL model, in which a large proportion (typically between 50%[204] to 90% [105]) of the weights are zero. Pruning methods allow to keep model accuracy with little loss in model quality, thus achieving the same expressive power as dense model counterparts, while leading to models that are more efficient in terms of computing and storage resources demand.

The second factor that induces sparsity in DNN models is the ReLU (rectified linear unit) operator, which is frequently used as activation function. Indeed, ReLU resets all the negative values in the matrices of the activations[1] to zero.

Because of network pruning and zero-valued activations, sparsity has become an active area of research in DNN models. These two techniques allow to reduce both the memory size and the memory accesses, the latter thanks to the removal of useless operations (i.e., multiply by zero), which also save processing power and energy consumption. As regards the memory size, the number of non-zero entries in the resulting sparse matrices can be reduced to 20-80% and 50-70% for weights and activations, respectively [106, 202]. Sparse matrices can thus be stored using a compressed representation, thus leading to at least 2-3x memory size reduction. However, the main disadvantage of a sparse matrix is that the indexes become relative, which adds extra levels of indirection that add complexity and need to be carefully managed to avoid inefficiency.

As regards the sparsity pattern of DNN models, it can range from unstructured as a result of fine-grain pruning, which maintains model accuracy, to structured when coarse-grain pruning is applied to improve execution efficiency at the cost of downgrading the model accuracy [106, 286]. Randomly distributed nonzeros can lead to irregular memory accesses, that are unfriendly on commodity architectures, e.g., GPU, as well as to irregular computations that introduce conditional branches to utilize the sparsity. The latter are hardly applicable for accelerators, which are designed for fine-grained data or thread parallelism rather than flexible data path control. On the other hand, hardware-friendly structured sparsity can efficiently accelerate the DNN evaluation at the cost of model accuracy degradation.

Moreover, sparsity is becoming ubiquitous in modern deep learning workloads (i.e., not only because of the application of compression techniques such as network pruning and zero-valued activations) due to the application of deep learning to graphs for modeling relations (in social networks, proteins, etc.) using highly sparse matrices, such as in Graph Neural Networks (GNNs).

The key computational kernel within most DL workloads is general matrix-matrix multiplications (GEMM) [84]. It appears frequently during both the forward pass (inference and training) and backward pass (training); for instance, experiments reported in [220] show that GEMM comprises around 70% of the total compute cycles during training for Transformer and Google Neural Machine Translation workloads. Therefore, GEMM represents a primary target for hardware acceleration in order to speed up training and inference. However, GEMM in DL is characterized by sparsity of matrices, which arises from pruning as explained above, and non-square matrix dimensions, which arise from minibatches and weight factorization [205]. A popular computational kernel for convolutional neural networks (CNNs) is sparse vector-vector dot product. Sparse-dense matrix multiplication (SpMM) and sampled dense dense matrix multiplication (SDDMM) are two of the most generic kernels in GNNs.

Spatial-architecture-based hardware accelerators that exploit sparsity have different architectures that allow to adapt the computation to sparse matrices. In the following, we review their main features.

[1]The activations are the output values of an individual layer that are passed as inputs to the next layer.

Some accelerators (e.g., Cnvlutin, Cambricon-X, Eyeriss) handle only one-sided sparsity, which stems either from zero-valued activations or network pruning, thus achieving only a partial reduction in compute and data reduction. On the other hand, other accelerators (e.g., SCNN, SparTen, Eyeriss v2) target two-sided sparsity, which originates from network pruning and zero-valued activations. In addition to the different approaches of exploiting sparsity, these architectures also employ distinct dataflows to execute the DNN layers. Due to the complexity of the logic, existing hardware accelerators for sparse processing are typically limited to a specific layer type (e.g., fully-connected layers, convolutional layers).

Eyeriss [45] targets CNN acceleration by storing in DRAM only nonzero-valued activations in CSC format and by skipping zero-valued activations (by means of gating the datapath switching and memory accesses) in order to save energy. Eyeriss v2 [46], which targets DNNs on mobile devices, supports also sparse network models. It utilizes the CSC format to store weights and activations, which are kept compressed not only in memory but also during processing. To improve flexibility, it uses a hierarchical mesh for the PEs interconnections. By means of these optimizations, Eyeriss v2 is significantly faster and more energy-efficient than the original Eyeriss.

Cnvlutin [12], which also targets CNN acceleration, uses hierarchical data-parallel units, skips computation cycles for zero-valued activations and employs a co-designed data storage format based on CSR to compress the activations in DRAM. However, it does not consider the sparsity of the weights.

On the contrary, Cambricon-X architecture [298] exploits sparsity of CNNs by letting the PEs store the compressed weights in CSR format for asynchronous computation. However, it does not exploit activations sparsity.

EIE [104] targets energy-efficient acceleration of DNN inference. To this purpose, it employs a scalable array of PEs, where each PE stores a partition of the DNN network in SRAM, whose usage allows to obtain high energy savings with respect to DRAM. It compresses the weights using a variant of CSC sparse matrix representation and has skipping ability for zero-valued activations.

NullHop [10] is a CNN accelerator architecture that applies the Compressed Image Size (CIS) format to the weights and skips the null activations, similarly to EIE.

Sparse CNN (SCNN) [202] is an accelerator architecture for inference in CNNs. It employs a cluster of asynchronous PEs connected via simple interconnections and comprising several multipliers and accumulators. SCNN exploits sparsity in both weights and activations, which are stored in the classic CSR representation. It employs a Cartesian product-based computation architecture that maximizes the reuse of weights and activations within the cluster of PEs; the values are delivered to an array of multipliers, and the resulting scattered products are summed using a dedicated interconnection mesh. By exploiting two-sided sparsity, SCNN achieves to improve performance and energy over dense architectures.

SparTen [93] is based on SCNN [202]. It addresses some considerable overheads of SCNN in performing the sparse vector-vector dot product by improving the distribution of the operations to the multipliers and allows to use any convolutional stride (not being limited to unit-stride convolutions as SCNN). It also addresses unbalanced sparsity distribution across the PEs by means of an offline software scheme.

The PermDNN architecture [63] addresses the generation and execution of hardware-friendly structured sparse DNN models using permuted diagonal matrices. In this way, it does not incur into load imbalance which is caused by the irregularity of unstructured sparse DNN models.

SqueezeFlow [164] is an accelerator architecture that exploits sparsity of CNN models. Differently from the accelerators described above, it takes an alternative approach with the goal to reduce the hardware complexity. To this end, it exploits concise convolution rules to benefit from the reduction of computation and memory accesses as well as the acceleration of existing dense CNN

architectures without intrusive PE modifications. The Run Length Compression (RLC) format is used to compress activations and weights.

A different strategy is also pursued by the Unique Weight CNN (UCNN) accelerator [111], which proposes a generalization of the sparsity problem. Rather than considering only the repetition of zero-valued weights, it exploits repeated weights with any value by reusing CNN sub-computations and reducing the model size in memory.

SIGMA [220] is an accelerator for DNN training which is characterized by a flexible and scalable architecture that offers high utilization of its PEs regardless of kernel shape (i.e., matrices of arbitrary dimensions) and sparsity pattern. It targets acceleration of GEMMs with unstructured sparsity.

Bit-Tactical [62] is a DNN accelerator where the responsibility for exploiting weight sparsity is shared between a static scheduling middleware and a co-designed hardware front-end, with a lightweight sparse shuffling network which comprises two multiplexers per activation input. Unlike SIGMA and other accelerators, Bit-tactical leverages scheduling in software to align inputs and weights.

While the above accelerators are statically tailored to a particular dataflow, which make them better suited to different data sets, Flexagon [189] is a reconfigurable accelerator that is capable of performing sparse-sparse matrix multiplication computation by using the particular dataflow that best matches each case.

Besides the design of specialized hardware accelerators to exploit model sparsity, a parallel trend is to use GPU architectures. Pruned sparse models with unstructured sparse patterns introduce irregular memory accesses that are unfriendly on commodity GPUs architectures. A first direction to tackle this issue on commodity DNN accelerators is at the software layer, by means of pruning algorithms which enforce a particular sparsity pattern, such as tile sparsity [98], on the model that allows to leverage existing GEMM accelerators.

A second direction to leverage the sparsity in DNN models on GPUs, is to introduce new architectural support, such as Sparse Tensor Cores [187]. The NVIDIA Ampere architecture introduces this Sparse Tensor Core design with a fixed 50% weight pruning target and achieves a better accuracy and performance trade-off. However, sparsity from activations, which are dynamic and unpredictable, is challenging to leverage on GPUs. Indeed, the current sparse Tensor Core is only able to take advantage of weight sparsity but not activation sparsity.

Reconfigurability appears to be a keyword for the design of new sparse accelerators, because some network models exhibit *dynamic sparsity* [76], where the position of non-zero elements changes overtime.

5.3 Emerging 3D-stacked Processing-in-memory Technologies

3D integration technologies [137] allow to stack as many as 16 or more 2D integrated circuits and interconnect them vertically using, for instance, through-silicon vias (TSVs), microbumps, or Cu-Cu connections. In this way, a 3D circuit behaves as a single device achieving a smaller area footprint than conventional 2D circuits, while reducing power and latency in data transfer. In general, 3D integration is a term that includes such technologies as 3D wafer-level packaging (3DWLP) [245], 2.5D and 3D interposer-based integration [2], 3D stacked ICs (3D-SICs), 3D heterogeneous integration, and 3D systems integration, as well as true monolithic 3D ICs [145, 179].

These technologies have been employed in the development of new memory devices, which stack layers of conventional 2D DRAM or other memory types (for instance, Non-Volatile Memory (NVM) based on ReRAM [54]) together with one or more optional layers of logic circuits. These logic layers are often implemented with a different process technology and can include buffer circuitry, test logic, and processing elements. Compared to 2D memories, 3D stacking increases the memory capacity and bandwidth, reduces the access latency due to the shorter on-chip wiring interconnection and

the use of wider buses, and potentially improves performance and power-efficiency of system. In fact, 3D stacking of DRAM memory provides an order of magnitude higher bandwidth and up to 5× better energy efficiency then conventional 2D solutions, making the technology an excellent option for meeting the requirements in terms of high throughput and low energy of DNN accelerators [109].

Two main 3D stacked memory standards have been recently proposed: the Hybrid Memory Cube (HMC) and the High Bandwidth Memory (HBM). 3D stacked processing-in-memory accelerator proposals modify the architecture of the 3D memory block inserting processing logic near the memory elements. Two approaches can be commonly found. In the first approach, the computing logic is embedded into the logic die (logic die-level processing-in-memory); in the second approach, processing logic is integrated into each DRAM die at the level of memory banks, after the column decoder and selector blocks (bank-level processing-in-memory). We present in the following subsections the main characteristics of the two 3D stacked memory standards, and overview the existing accelerators adopting them.

5.3.1 Hybrid Memory Cube. The Hybrid Memory Cube is a single package containing four to eight DRAM die and one logic die, all stacked together using thousands of TSVs, achieving a much desired high memory bandwidth [118, 123]. As shown in Figure 8a, in a HMC, memory is organized vertically, and portions of each memory die are combined with the corresponding portions of the other memory dies and the logic die. This creates a 2D grid of vertical partitions, referred as vaults [211]. Each vault is functionally and operationally independent and includes in the logic layer a memory controller that manages all memory reference operations within that vault, as well as determining timing requirements and dealing with refresh operations, eliminating these functions from the host memory controller. The independence of each vault allows to exploit memory level parallelism, as multiple partitions in the DRAM dies can be accessed simultaneously.

Commands and data are transmitted from and to the host across external I/O links consisting of up to four serial links, each with a default of 16 input lanes and 16 output lanes for full duplex operation (HMC2 specifications [118]). All in-band communication across a link is packetized. According to specifications, up to 320 GB/s effective bandwidth can be achieved by considering 30 Gb/s SerDes I/O interfaces, with a storage capacity, depending on the number of stacked layers, of 4GB and 8GB [118, 183].

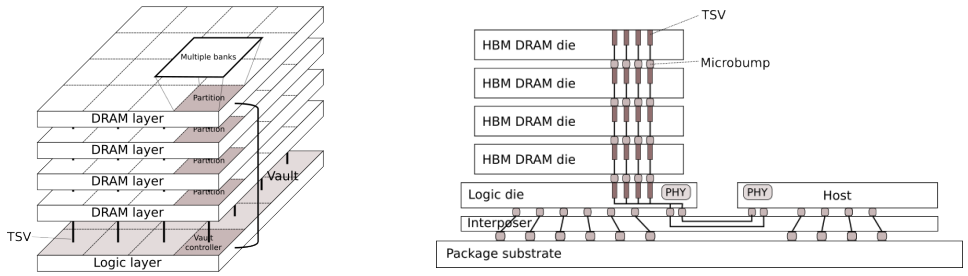

Fig. 8. (a) High-level architecture of the Hybrid Memory Cube. (b) Cut through image of a computing system with HBM.

5.3.2 High Bandwidth Memory. The High Bandwidth Memory is a high-speed computer memory interface for 3D-stacked synchronous dynamic random-access memory (SDRAM) [143, 242]. Each

Table 9. Summary of 3D-staked Processing-in-memory DNN accelerators.

PIM	Year	Integration Level	3D Mem. Tech.	Functions	Data Type	Tech. Node	Performance [GOPs/s]	Power [W]	Maturity
Neurocube[141]	2016	Logic die	HMC	MAC	16-bit fixed point	15nm	132	3.4 + HMC	Layout
Tetris[85]	2017	Logic die	HMC	ALU/MAC	16-bit fixed point	45nm	-	8.42	Simulation
NeuralHMC[184]	2019	Logic die	HMC	MAC	32-bit floating point	-	-	-	Simulation
VIMA[57]	2021	Logic die	HMC	ALU/MULT/DIV	32-bit integer/floating point	-	-	3.2 + HMC	Simulation
Newton[110]	2020	Bank	HBM	MAC	bfloat16	-	-	-	Simulation
HBM-PIM[153]	2020	Bank	HBM	ALU/MAC	16-bit floating point	20nm	1200	-	Silicon

memory module is composed by stacking up to eight DRAM dies and an optional base die including buffer circuitry and test logic. Dies are vertically interconnected by TSVs and microbumps, in a way similar to the HMC. As shown in Figure 8b, the memory stack is often connected to the memory controller on the host (e.g., GPU or CPU) through purpose-built silicon chip, called interposer [2], which is effectively a miniature PCB that goes inside the package and decreases the memory paths by allowing the host and the memory to be physically close. However, as semiconductor device fabrication is significantly more expensive than printed circuit board manufacture, this adds cost to the final product. Alternatively, the memory die can be stacked directly on the host processor chip.

The HBM DRAM is tightly coupled to the host computer through a distributed interface, which is divided into independent channels. The HBM DRAM uses a wide-interface architecture to achieve high-speed, low-power operation. Each channel interface maintains a 128-bit (HMB2) or 64-bit (HMB3) data bus operating at double data rate (DDR). The latest version (2022) of the HBM (HBM3) supports up to 16 channels of 64 bits, with a total number of data pins equal to 1024, and with an overall package bandwidth of 600 GB/s [124, 218]. Depending on the producer, the HBM stack consists of 8 or 12 16Gb DRAMs, with a total maximum memory capacity of 24 GB [227].

5.3.3 3D stacked Accelerators: some considerations. The HMC and HBM provide highly parallel access to the memory which is well suited to the highly parallel architecture of the DNN accelerators. The processing elements of 3D stacked DNN accelerators can be embedded in the logic die or in the memory dies, reducing significantly the latency of accessing data in main memory, and improving the energy efficiency of the system. However, there are some challenges and limitations to be taken into account when using this technology [142]. First, the amount of processing elements that can be integrated into 3D stacked memories is limited by the size of the package. Moreover, the overall power dissipation of these elements is limited by thermal issues of 3D stacking, as an increase in the operation temperature would result in performance degradation from overheating [112]. Second, the stacking of multiple IC layers has a high manufacturing complexity, which leads to lower yield and problematic testability. Therefore, in order to support the adoption of this technology, proper cooling methods and better manufacturing solutions are required.

Apart from the above mentioned technological challenges, embedding processing elements into the memory and moving the computation closer to it requires to rethink the optimization of the system design in order to take into account the proximity of the processing logic to the main memory. Depending on the use case, this might involve the redesign of the on-chip buffers in the logic die, to support the lower latency and energy cost of the accesses to main memory, as well as the use of new approaches for representing, partitioning, and mapping the dataflow of the application in order to exploit the highly parallel system supported by the availability of multiple channels [109].

5.3.4 State-of-the-art on 3D-stacked Processor-in-memory solutions. We can distinguish two different approaches when integrating digital processing elements in a 3D stacked memory architecture [142]. The first approach, most commonly found in the literature, embeds the computing into the logic die of the memory block (logic die-level processing-in-memory). In the second approach

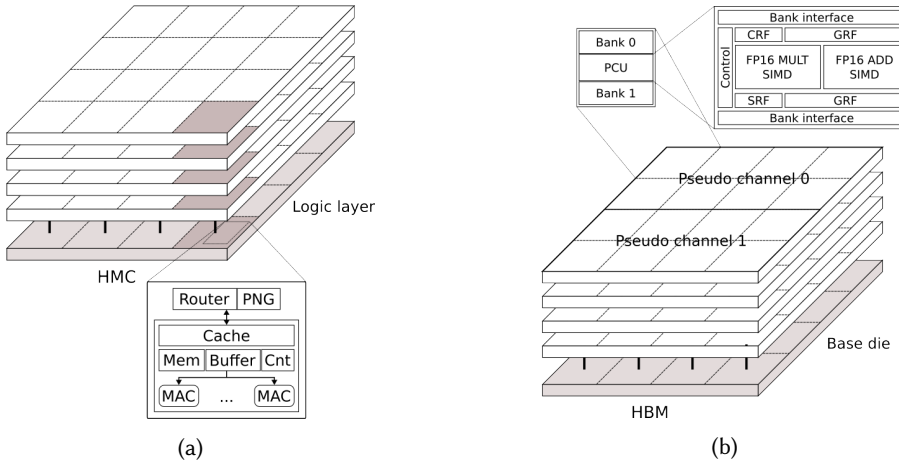

Fig. 9. (a) Neurocube architecture. (b) HBM-PIM architecture.

(bank-level processing-in-memory), processing logic is integrated into each DRAM die at the level of the memory banks, after the column decoder and selector blocks.

A first example of a 3D processing-in-memory implementation is Neurocube [141]. As shown in Figure 9a, the Neurocube architecture is embedded into the logic die of a HMC, and consists of a cluster of processing engines (PE) connected by a 2D mesh Network-on-Chip (NoC). The PE is composed of a row of multiply accumulator (MAC) units, a cache memory, a temporal buffer, and a memory module for storing shared synaptic weights. Each PE is associated to a single memory vault and can operate independently and communicate through the TSVs and the vault controller. A host communicates with the Neurocube through the external links of the HMC to configure the Neurocube for different neural network architectures. Each vault controller in the HMC has an associated programmable neurosequence generator (PNG), i.e., a programmable state-machine that controls the data movements required for neural computation. Neurocube implements an output stationary dataflow, meaning that each MAC from a PE is responsible for the computations of a different output neuron at a time.

Similarly to Neurocube, Tetris [85] uses a HMC memory stack organized into 16 vaults. Each vault is associated with a processing engine, connected to the vault controller, and composed of an systolic array of 14 ×14 processing elements and a small SRAM buffer, shared among the processing elements. A 2D mesh NoC connects all the processing engines. Differently from previous accelerator approaches, the dimension of the buffers in the logic layer is reduced and optimized to take into account the lower cost of accessing the DRAM layers, as well as the area constraints of the 3D package. Each PE has a register file and a MAC to locally store the inputs/weights and perform computations. Tetris implements a a row stationary dataflow which maps 1D convolutions onto a single PE and utilizes the PE register file for local data reuse. A 2D convolution is orchestrated on the 2D array interconnect so that the data propagation among PEs remains local. In [85], an optimal scheduling is discussed to maximize on-chip reuse of weights and/or activations, and resource utilization. However, a programming model is not presented.

NeuralHMC [184] adopts the same systolic architecture and row-stationary dataflow discussed in Tetris. However, the authors introduce the use of a weight sharing pipelined MAC design to lower the cost of accessing weight data, by reducing the original 32 bits floating points weights to a 5 or 8

bits cluster index, saving memory consumption. Moreover, they discuss a series of mechanisms to reduce and optimize packet scheduling and on-chip communication in multi-HMC architectures.

The authors in [57] study the benefits of migrating machine learning kernels on a near-data processing (NDP) architecture capable of large-vector operations. The work derives from previous work of the same authors [13], where they introduced the HIVE architecture, which extends the HMC ISA for performing common vector operations directly inside the HMC, avoiding contention on the interconnections as well as cache pollution. The newly introduced Vector-In-Memory Architecture (VIMA) supports all ARM NEON Integer and Floating-point instructions and operates over vectors of 8 KB of data by fetching data over the 32 channels (vaults) of the HMC in parallel. The authors extend and use an NDP intrinsics library that supports validation of NDP architectures based on large vectors, and provide insights and show benefits of migrating Machine Learning algorithms to vector-based NDP architectures. Their simulated results show a significant speed up and energy reduction with respect to an x86 baseline.

The second approach to PIM acceleration found in the literature integrates the processing logic near the memory banks, after column decoder and selector, allowing the logic to benefit of the entire width of the cell array.

Newton [110] proposes a fixed data flow accelerator that computes matrix-vector multiplication effectively. It employs a minimal compute of only MAC units and buffers and a DRAM-like command interface for the host CPU to issue commands to the PIM compute, avoiding the overhead and granularity issues of offloading-based accelerators, e.g., the delay in the launch of the kernel and the switching between the PIM/non-PIM operational modes. Newton is implemented, at the bank level, on HBM, and each of its bank includes 16 multipliers, 16 adders in a reduction tree, and a 16-bit accumulator register. Two input operands of the multipliers come from the memory cell array after the column selector and the global buffer, which broadcasts an input vector to all memory banks, implementing a multiplication between the vector and the matrix rows stored in the banks. The results of the vector-matrix multiplications are stored in an output vector. To reduce the output vector write traffic with minimal output buffering, Newton employs an unusually-wide interleaved layout (DRAM row-wide). In Newton, both the input and output vectors have high reuse while the matrix has no reuse.

HBM-PIM [153] implements a function-in-memory DRAM (FIMDRAM) that integrates a 16-wide single-instruction multiple-data engine within the memory banks and that exploits bank-level parallelism to provide 4 × higher processing bandwidth than an off-chip memory solution (Figure 9b). In their design, half of the cell array in each bank of the HBM was removed and replaced by a programmable computing unit (PCU), placed adjacent to the cell array to utilize bank-level parallelism. Each PCU is shared among two banks, and there are 8 PCUs per pseudo-channel. The PCU is divided into a register group, an execution unit, a decoding unit for parsing instructions needed to perform operations, and interface units to control data flow. The register group consists of a command-register file for instruction memory (CRF), a general-purpose register file for weight and accumulation (GRF), and a scalar register file to store constants for MAC operations (SRF). The PIM controller is integrated to support the programmability of the PCU and, similarly to Newton, the seamless integration with the host by determining the switching between the PIM/non-PIM operational modes. If the PIM mode is asserted, the PCUs execute the instructions pre-stored in the CRF, incrementing the program counter every time a DRAM's read command is issued.

3D-stacked processing-in-memory has been also proposed for accelerating application loosely related to DNNs. We present hereafter a brief overview of these accelerators.

The use of processing elements in the logic layer of an HMC is discussed in [199], to support the simulation of large networks on neurons. The proposed Neuron In-Memory (NIM) architecture is composed of 2,048 functional units, operating on integer and floating-point data, and a small

register file with 8 × 16 registers of 32 bits each per vault. Fast vector elements operation is also supported. When compared with traditional multi-core environments, NIM provides overall system acceleration and reduces overall energy consumption, taking advantages of the broad bandwidth available in 3D-stacked memory devices.

Millipede [193] is an NDP architecture for Big data Machine Learning Analytics (BMLA) that implements its processors in the logic layer of 3D-stacked memories. These processors have a local memory, register file, pipeline, cache, and prefetch buffers.

The authors in [179] explore the design trade-offs and thermal implications of 3D stacking in different configurations layers of SRAM buffers and systolic accelerators composed of MAC elements, while targeting Deep learning applications. The main memory (DRAM) is however not necessarily stacked with the rest of the system. Their simulations show that stacking PE array on top of the SRAM stack in a logic-over-memory fashion can not only achieve low energy but also mitigate the thermal impact of 3-D.

iPIM [96] uses a near-bank architecture for image processing. The control and the execution are decoupled to obtain a higher bank-level bandwidth and maximize the parallel execution of processing engines on the memory dies. Each vault contains a control core in the logic die, while the execution logic is placed in the memory die of each vault. Each control core manages intra/inter-vault data communication and executes instruction decoding with the support of the single-instruction-multiple-bank (SIMB) instruction set architecture (ISA).

Neurosensor [17] is a 3D CMOS image sensor system with an integrated convolutional neural network computation. The image sensor, read-out circuits, memory, and neural computation logic layers are integrated in a single stack. The DNN computation platform is an adaptation from Neurocube [141], and consists of a global controller, processing elements, a 2D mesh NoC connecting the PEs, and a programmable neurosequence generator for DRAM. The DNN computation is split between the sensor and the host, and the optimal task distribution depends on the processing capabilities of the sensor, the available amount of in-sensor memory for storing the synaptic weights, and the available bandwidth between the sensor and the host.

5.4 In-memory computing accelerators based on emerging memories

In-memory computing (IMC) has been proposed to break both the memory and the compute wall in data-driven AI workloads, using either SRAM or emerging memory technologies such as PCM and RRAM, offering different trade-offs when used as an integrated computing device at the system level. Full-digital IMC designs offer a fast path for the integration of next generation of neural processing systems like in NPUs. An example of IMC architecture targeting NPU design has recently been proposed by STMicroelectronics in [64], where a scalable and design time parametric NPU for edge AI relying on digital SRAM IMC has been manufactured in 18 nm FDSOI technology achieving an end-to-end system-level energy efficiency of 77 TOPS/W and an area efficiency of 13.6 TOPS/mm^2. This IMC-NPU is the evolution of the Orlando system-on-chip proposed in [65]. Another digital IMC design is NeuroCIM [144], an energy-efficient processor with four key features achieving 310.4 TOPS/W: Most significant bit (MSB) Word Skipping to reduce the BL activity; early stopping to enable lower bitline activity; mixed-mode firing for multi-macro aggregation; voltage folding to extend the dynamic range.

Besides conventional CMOS designs, emerging non-volatile memories such as the RRAM and the PCM have been recently explored for integration in stand-alone DNN accelerators. The RRAM device structure (see Fig.10a) is a metal-insulator-metal (MIM) structure that consists of a top electrode (TE), a bottom electrode (BE), and a metal-oxide layer sandwiched between them. By applying a proper voltage across the electrodes and setting the maximum current flowing in the MIM stack (through a series transistor), an RRAM cell can modulate the shape of a conductive

filament created in the metal-oxide layer. In PCM the active material is a chalcogenide phase change material, which can remain in either crystalline or amorphous states for long periods of time at moderately high temperature. Starting from the amorphous state, the application of voltage pulses with relatively low amplitude causes the crystallization induced by Joule heating, whereas the application of pulses at higher amplitudes can lead to local melting and consequent amorphization. A typical PCM cell has a mushroom shape shown in Fig. 10a, where the pillar-like bottom electrode confines heat and current, thus resulting in a hemispherical shape of the molten material. In both technologies, their resistance state can be tuned not only as a digital memory but also as a continuous analog memory with multiple states to perform in-memory computing [119]. This characteristic allows efficient matrix-vector multiplication when RRAM and PCM are arranged in crossbar structures (see Fig.10b).

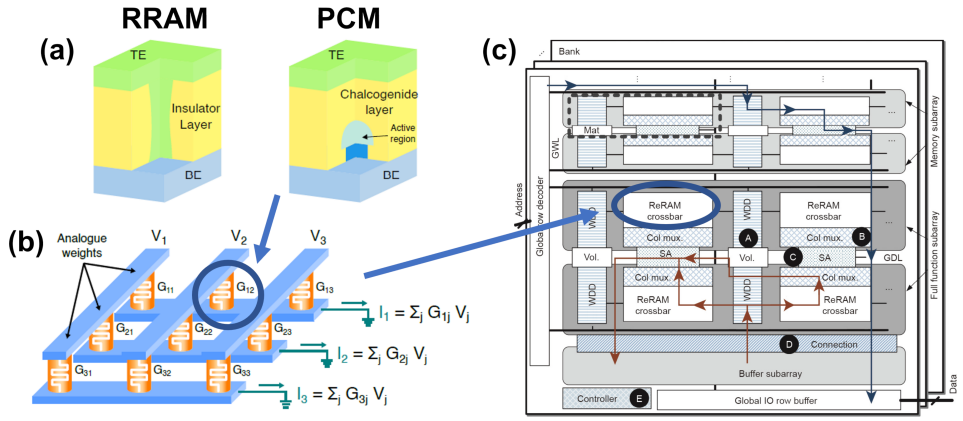

Fig. 10. (a) RRAM and PCM devices structure and (b) their arrangment in a crossbar structure for matrix-vector multiplication. (c) Example of a stand-alone DNN accelerator (i.e., PRIME [49]) using RRAM crossbars for *in situ* MAC operations. Reprinted from [162] and [44] under Creative Commons License.

A first example of an RRAM-based accelerator is the ISAAC [235] tile-based architecture that proposes a pipeline design for CNN processing, which combines the data encoding and the processing steps within *in situ* multiply and accumulate units (IMA). In the first pipeline step, data are fetched from on an chip eDRAM to the computation tile. The data format in ISAAC is fixed 16 bit. In computation, in each cycle, 1 bit is input to the IMA, and the computation result from the IMA is converted to digital format, thus requiring 16 clock cycles to process the input. The nonlinear activation is then applied, and the results are written back to eDRAM. Tiled computation is widely used from RRAM to improve the throughput. The PipeLayer [244] architecture introduces intra-layer parallelism and an inter-layer pipeline for tiled architecture, using duplicates of processing units featuring the same weights to process multiple data in parallel.

RRAM-based accelerators have been also designed for RNN applications such as the works in [277]. Here, all the decomposed operations were formulated into in-situ MAC operations to provide high throughput. Further, designs like PRIME [49] takes part of the RRAM memory arrays to serve as the accelerator instead of adding an extra processing unit for computation. This can be considered as an architecture which is borderline between NPUs and stand-alone.

However, [33] noted that existing PIM RRAM accelerators suffer from frequent and energy-intensive analog-to-digital (A/D) conversions, severely limiting their performance. To this extent, they presented a new architecture to efficiently accelerate deep learning tasks by minimizing the

required A/D conversions with analog accumulation and neural approximated peripheral circuits. Using a new dataflow they remarkably reduced the required A/D conversions for matrix-vector multiplications by extending shift and add (S+A) operations into the analog domain before the final quantizations.

The first PCM-based silicon demonstrator for DNN inference is Hermes, which appeared in 2021 [139]. The in-memory computing accelerator consists of a 256x256 PCM cross-bar and optimized ADC circuitry to reduce the read-out latency and energy penalty. The SoC is implemented in 14nm technology, showing energy efficiency of 10.5 TOPS/W and performance density of 1.59 TOPS/mm2 on inference tasks of multi-layer perceptrons and ResNet-9 models trained on MNIST and CIFAR-10 datasets, with comparable accuracies as software baseline. The same 256x256 PCM cross-bar has been integrated into a scaled-up mixed-signal architecture that targets inference of long short-term memory (LSTM) and ResNet-based neural networks [83]. The chip, implemented in the same 14nm technology, consists of 64 analog cores interconnected via an on-chip communication network and complemented with digital logic to execute activation functions, normalization and other kernels than Matrix-Vector Multiplications (MVMs). The accelerator achieves a peak throughput of 63.1 TOPS with a energy efficiency of 9.76 TOPS for 8-bit input/8-bit output MVM operations.

Besides silicon stand-alone demonstrators, the PCM technology is evaluated from a broader perspective in heterogeneous architectures that target different classes of devices, from IoT end-nodes to many-core HPC systems. Such studies aim to highlight and overcome the system-level challenges that arise when the PCM technology is integrated in more complex mixed-signal systems. For example, Garofalo et al. [87] analyze the limited flexibility of AIMC cores that can only sustain MVM-oriented workloads, but they are inefficient to execute low-reuse kernels, and other ancillary functions such as batch-normalization and activation functions. To better balance the Amdahl's effects that show up on the execution of end-to-end DNN inference workloads, they propose as a solution an analog-digital edge system that complements the computing capabilities of PCM-based accelerators with the flexibility of general-purpose cores. The architecture, benchmarked on a real-world MobileNetV2 model, demonstrates significant advantages over purely digital solutions.

Bruschi et al. [30] leave the edge domain to study the potentiality of PCM-based AIMC in much more powerful HPC many-core systems. The work presents a general-purpose chiplet-oriented architecture of 512 processing clusters, each composed of RISC-V cores for digital computations and nvAIMC cores for analog-amenable operations, such as 2D convolutions. This system is benchmarked on a ResNet18 DNN model, achieving 20.2 TOPS and 6.5 TOPS/W.

Table 10. Summary of IMC accelerators based on RRAM and PCM memories

Accelerator	Technology	Process	Application	Area [mm^2]	Power [mW]	Performance [GOPS]	EE [GOPS/W]	AE [GOPS/mm^2]
ISAAC [235]	RRAM+CMOS	32 nm	CNN	85.4	65800	-	380.7	466.8
PipeLayer [244]	RRAM+CMOS	-	CNN	82.63	-	-	140	1485
NeuralPIM [33]	RRAM+CMOS	32 nm	CNN+RNN	86.4	67700	-	2040.6	1904
PRIME [49]	RRAM+CMOS	65 nm	MLP+CNN	-	-	-	2100	1230
NeuRRAM [277]	RRAM+CMOS	130 nm	CNN+RNN+RBN	159	49.7	2135	43000	-
Hermes [139]	PCM+CMOS	14 nm	MLP+CNN+LSTM	-	-	-	10500	1590

5.5 Full-digital Neuromorphic Accelerators

Neuromorphic computing aims at a paradigm shift from Von Neumann-based architectures to distributed and co-integrated memory and processing elements, the granularity at which this paradigm shift is achieved in digital implementations strongly varies between a distributed Von Neumann or full-custom approach, from high to low processing and memory separation [80]. Neuromorphic chip architectures enable the hardware implementation of spiking neural networks

(SNNs) [225] and advanced bio-inspired computing systems that have the potential to achieve even higher energy efficiency with respect to DNN stand-alone accelerators described so far [11].

A first example of a digital architecture for SNN and neuroscience simulation acceleration is the SpiNNaker chip [201]. It follows a distributed von-Neumann approach using a globally asynchronous locally synchronous (GALS) design for efficient handling of asynchronous spike data and is based on a 130 nm technology. SpiNNaker has been optimized for large-scale SNN experiments while keeping a high degree of flexibility. The evolution of the architecture using 22 nm technology embedding 4 ARM Cortex M4F cores out of the 152 per chip is planned for the final SpiNNaker 2 system [169]. The objective is to simulate two orders of magnitude more neurons per chip compared to [201]. However, it has been demonstrated that GPU-based accelerators can compare favorably to a SpiNNaker-based system when it comes to large scale SNN and cortical-scale simulations [146].

Full-custom digital hardware allows for higher-density and more energy-efficient neuron and synapse integration for spiking neural networks (SNN) compared to the two formerly described accelerators [80]. All the accelerators to be reported in this document benefit from moving the memory (generally SRAM elements) closer to computation. The 45 nm design in [233] is a small-scale architecture for SNN acceleration embedding 256 Leaky-Integration-Fire (LIF) neurons and up to 64k synapses based on the Stochastic Synaptic Time Dependant Plasticity (S-STDP) concept. It achieves a reasonably high neuron and synapse densities, despite the use of a custom SRAM and given is energy-efficiency figures is an ideal choice especially for edge computing scenario. At the same integration scale, the ODIN chip embeds 256 neurons and 64k Spike Driven Synaptic Plasticity (SDSP)-based 4-bit synapses in a 28 nm CMOS process [81]. A first attempt to scale up the NPU for SNN applications is represented by the 65 nm MorphIC chip, that bases on the ODIN core integrated in a quadcore design [82].

Concerning large-scale neuromorphic platforms required for cognitive computing applications, there are currently two designs offered: the 28 nm IBM TrueNorth [11] and the 14 nm Intel Loihi [60] neuromorphic chips. TrueNorth is a GALS design embedding as high as 1M neurons and 256M binary non-plastic synapses per chip, where neurons rely on a custom model that allows modifying their behaviors by combining up to three neurons [35]. Loihi is a fully asynchronous design embedding up to 180k neurons and 114k (9- bit) to 1M (binary) synapses per chip. Neurons rely on a LIF model with a configurable number of compartments to which several functionalities such as axonal and refractory delays, spike latency and threshold adaptation have been added. The spike-based plasticity rule used for synapses is programmable.

In digital designs for neuromorphic chips, versatility can be obtained with a joint optimization comprising power and area efficiencies. This flexibility in optimizing between versatility and efficiency in digital designs is highlighted with platforms going from versatility-driven (e.g., SpiNNaker) to efficiency-driven (e.g., ODIN and MorphIC), through platforms aiming at a well-balanced trade-off on both sides (e.g., Loihi). Table 11 summarizes the main characteristics of the neuromorphic chips described so far with a particular insight on the Energy per spike operation (SOP) that is seen as a primary benchmarking factor for these architectures.

Table 11. Summary of Neuromorphic chip characteristics based on [80]

Chip name	Technology	Cores	Core Area [mm^2]	Neurons per core	Synapses per core	Weights storage	Supply Voltage [V]	Energy per SOP [J]
SpiNNaker [201]	0.13 μm	18	3.75	1000	-	Off-chip	1.2	>11.3n/26.6n
[233]	45 nm SOI	1	0.8	256	64k	1-bit SRAM	0.53 - 1.0	-
ODIN [81]	28 nm FDSOI	1	0.086	256	64k	(3+1)-bits (SRAM)	0.55 - 1.0	8.4p/12.7p
MorphIC [82]	65 nm LP	4	0.715	512	528k	1-bit (SRAM)	0.8 - 1.2	30p/51p
TrueNorth [11]	28 nm	4096	0.095	256	64k	1-bit (SRAM)	0.7 - 1.05	26p
Loihi [60]	14 nm FinFET	128	0.4	1024	1M	1- to 9 bits (SRAM)	0.5 - 1.25	>23.6p

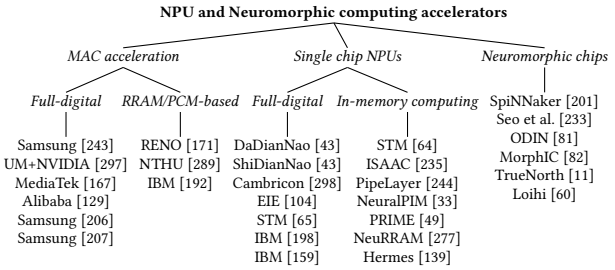

Fig. 11. Taxonomy of neural accelerators discussed in Sections 4.2.1, 4.2.2, 5.3, and 5.4.

5.6 Accelerators based on Multi-Chip Modules

The semiconductor industry has grown significantly as a result of increased integration complexity, resulting in improved performance and cost-effectiveness of transistors. Unfortunately, the trend of increasing the number of transistors per die is slowing down, leading to a power-efficiency driven design era known as "dark silicon" [73]. While the number of transistors per die continues to increase, many foundries are struggling to achieve the targeted area scaling per transistor, and new process technologies are expected to slow down. The cost per transistor may no longer hold, resulting in yield challenges and additional wafer costs. Circuit designers and computer architects can no longer rely on the free availability of additional transistors and integration opportunities with each new process node, and non-recurring engineering costs have also increased due to fabrication and system complexity challenges [131].

5.6.1 Alternate Integration Technologies. Alternate integration technologies can provide cost reductions and increase the number of transistors per circuit. These technologies include die-level integration such as 3D die stacking with connections through micro-bumps or Through-Silicon Vias (TSVs) [115], or through interposer-based 2.5D integration [299]. By partitioning a monolithic SoC across multiple small dies, namely *chiplets*, (see Fig. 12a), yield per die can be improved and metal layer count can be reduced, which can lead to a lower total IC cost [252]. In fact, larger chips cost more due to two main factors: geometry and manufacturing defects. Fewer larger chips can fit in a wafer, while defects in larger chips waste more silicon than defects in smaller chips [138]. Smaller chips can be packed more tightly, resulting in more chips that work. In general, making smaller chips results in a higher yield of functioning chips (see Fig. 12b).

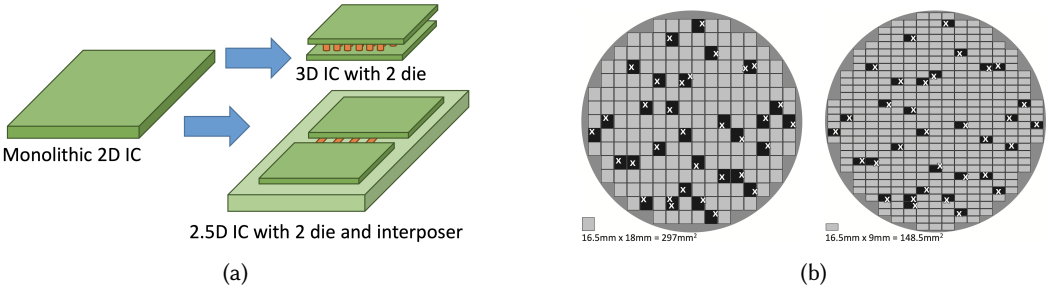

Fig. 12. Die-level integration through TSV-based 3D and interposer-based 2.5D technologies 12a [252] and overall number of chips and impact on yield of an example defect distribution for two different chip sizes [138]

Die-level integration provides new integration strategies like heterogeneous process integration between dies that can improve performance and lower costs [299]. Additionally, this technology can be used for the reuse of intellectual property to configure SoCs with different die combinations and reduce non-recurring overheads.

In multichip-module (MCM) silicon interposer-based integration, the interposer uses microbumps to connect the smaller chips, which have a higher density than traditional C4 bumps. The impedance across the interposer is the same as conventional on-chip interconnects. The only downside is the additional cost of the interposer. Vertical 3D chip stacking involves combining multiple chips with through-silicon vias (TSVs) for vertical interconnects. This technique has the potential to offer the highest bandwidth but it requires significant cost and overall process complexity as each die must be thinned and processed for TSVs. Overall, as 3D stacking is more expensive and complex, while also potentially causing thermal issues, we focus on MCM silicon interposer-based design in the following.

5.6.2 MCM Silicon Interposer-based Design. In 2.5D integration technology, an interposer is a substrate that connects multiple dies (chiplets) together. There are two types of interposers: passive interposers and active interposers [253]. Passive interposers are simple substrates that connect multiple dies together without adding any active components. They mainly provide electrical connections, signal routing, and thermal management between the dies. On the other hand, active interposers contain active components such as transistors, capacitors, and inductors, in addition to the electrical connections and signal routing provided by passive interposers. Active interposers can perform some processing and signal conditioning functions between the dies. Regardless of the interposer type used, a design based on a MCM silicon interposer offers several advantages over single-chip designs, which can be summarized as follows.

- Increased Functionality: By using MCM, designers can combine multiple chips and functionalities into a single package, thereby reducing the overall footprint and increasing functionality. This can help reduce the overall cost of the product by reducing the number of components required.
- Reduced Power Consumption: MCM-based designs can offer better power efficiency compared to single-chip designs. This is because multiple chips can be optimized for different power requirements, which helps reduce the overall power consumption of the system.
- Higher Performance: MCM-based designs can offer higher performance compared to single-chip designs. This is because multiple chips can be optimized for different tasks, which helps increase the overall performance of the system.
- Improved Reliability: MCM-based designs can offer improved reliability compared to single-chip designs. This is because multiple chips can be used redundantly to improve fault tolerance and increase system reliability.
- Cost Savings: MCM-based designs can often be less expensive than single-chip designs. This is because MCMs can be manufactured using existing processes and technology, which helps reduce the overall manufacturing cost. Additionally, MCMs can be designed to use off-the-shelf components, which helps reduce the cost of custom components.

Figure 13 presents a synthesized taxonomy of the MCM-based accelerators that will be introduced in the following subsections.

5.6.3 General Purpose Chiplet-based Architectures. Chiplet-based designs are being utilized across a wide range of platforms, tailored to support various application contexts. The challenges of creating integrated SoCs for aerospace platforms in advanced semiconductor nodes are reported in [188] in which authors highlight the possibility of creating heterogeneous mixtures of chiplets, including

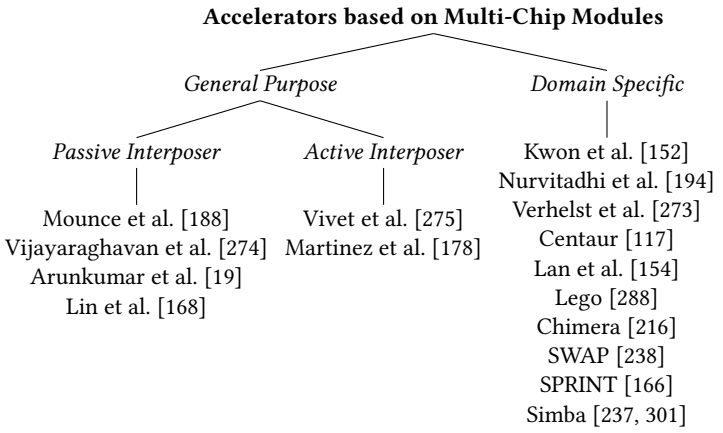

Fig. 13. Taxonomy of MCM based accelerators discussed in Section 5.6

different embodiments of processors, ultradense memory servers, field-programmable gate array clusters, and configurable analog and radiofrequency functional blocks. Further, some of the features necessary to support scalability and heterogeneity with multi-domain, hybrid architectures involving a mixture of semiconductor technologies and advanced packaging approaches are also outlined. In [274] a chiplet-based computing system for climate prediction is presented. It integrates high-throughput and energy-efficient GPU chiplet, high-performance multi-core CPU chiplet, and large capacity 3D memory. The system can achieve a bandwidth of 3 TB/s and power consumption of 160 W at 1 GHz.

GPU platforms are also benefiting from chiplet-based integration. In [19] a single-chip multi-core GPU is broken down into multiple GPU chiplets to improve both performance and energy efficiency by increasing hardware resource utilization for both the GPU and DRAM chiplets, while also mitigating the dark silicon effect. Additionally, breaking the larger GPU into multiple smaller chiplets has resulted in improved wafer yield.

The design and implementation of a dual-chiplet Chip-on-Wafer-on-Substrate is presented in [168] where each chiplet has four Arm Cortex-A72 processors operating at 4 GHz. The on-die interconnect mesh bus operates above 4 GHz at a 2 mm distance and the inter-chiplet connection features a scalable, power-efficient, high-bandwidth interface achieving 8 Gb/s/pin and 320 GB/s bandwidth.

The above work use use 2.5D integration technology based on passive interposer. In [275] the authors observe that current passive interposer solutions still lack flexibility and efficiency when it comes to long-distance communication, smooth integration of chiplets with incompatible interfaces, and easy integration of less-scalable analog functions, such as power management and system input/output signals (IOs). Thus, they present a CMOS Active Interposer that integrates power management and distributed interconnects, enabling a scalable cache-coherent memory hierarchy. The proposed platform integrates six chiplets onto the active interposer, offering a total of 96 cores.

The exploitation of active interposer as a way to address energy efficiency and computing density issues in high performance computing (HPC) for exascale architectures is discussed in [178]. The authors suggest that the integration of chiplets, active interposer, and FPGA can lead to dense, efficient, and modular compute nodes. They detail the ExaNoDe multi-chip-module which combines various components and demonstrate that multi-level integration allows for tight integration of hardware accelerators in a heterogeneous HPC compute node.

5.6.4 Domain Specific Chiplet-based Architectures. In the realm of deep learning, chiplet-based design is utilized to create hardware accelerator platforms that are both efficient and scalable.

Designing AI processors for data explosion computing due to the physical limitations of semiconductors and high costs is challenging. In [152] authors propose chiplet-based design as a viable solution to this problem. They outline various aspects of designing a chiplet AI processor, including incorporating NPU chiplets, HBM chiplets, and 2.5D interposers, ensuring signal integrity for high-speed interconnections, power delivery network for chiplets, bonding reliability, thermal stability, and interchiplet data transfer on heterogeneous integration architecture. They conclude that chiplet-based design provides higher performance at a lower cost compared to IP-based design.

Data intensive deep learning (DL) algorithms with strict latency constraints require balancing both data movement and compute capabilities. Thus, persistent approaches that keep the entire DL model on-chip are becoming the new norm for real time services to avoid the expensive off-chip memory accesses. In [194] it is shown how the integration of FPGA with ASIC chiplets outperform GPU based platforms (NVIDIA Volta) in terms of latency by enhancing on-chip memory capacity and bandwidth, and provide compute throughput matching the required bandwidth. Specifically, it is reported that the GPU and chiplet-based FPGA computing capabilities are 6% and 57% of their peak, respectively. In terms of delay and energy efficiency, the FPGA outperforms the GPU with a delay that is 1/16 and energy efficiency that is 34x better than the GPU's peak performance.

In [273] the authors investigate the recent multi-core trend in deep-learning accelerators evolution as a solution to further increase throughput and match the ever-growing computational demands. Chiplet integration is considered a promising implementation strategy for both homogeneous and heterogeneous multi-core accelerators.

Personalized recommendations, a crucial component of various application domains, are powered by machine learning algorithms. In [117] authors present a new chiplet-based hybrid sparse-dense accelerator called Centaur, which addresses memory-intensive embedding layers and compute-intensive multi-layer perceptron layers. The proposed accelerator demonstrates significant performance speedup and energy efficiency improvement compared to conventional approaches monolithic approaches.

The trend towards developing high throughput and energy-efficient neural network hardware accelerators due to the growing complexity and dimension of neural network algorithms is analyzed in [154]. The authors propose a chiplet-based architecture for a multi-core neuromorphic processor with a chip-package co-design flow. It is shown how the proposed design is reusable for different neuromorphic computing applications by scaling the number of chips in a package and by reusing existing IPs from different technology nodes with 2.5D integration technology.

The challenges of using modern deep neural network (DNN) accelerators in multi-tenant DNN data centers are investigated in [288]. The MCM architecture is proposed as a promising approach to address this issue, but highlights the challenge of distributing DNN model layers with different parameters across chiplets. Thus the authors present Lego MCM architecture with a dynamic scheduler that adapts to the size of DNN model layers and increases chiplet utilization. The results show that Lego MCM achieves a 1.51x speedup over a monolithic DNN accelerator.

Chimera [216] is a non-volatile chip for DNN training and inference that does not require off-chip memory. Multiple Chimera accelerator chiplets can be combined in a multi-chip system to enable inference on models larger than the single-chip memory with only 5% energy overhead.

Chiplet-based processing-in-memory DNN hardware accelerator have also been proposed. SWAP [238] is a DNN inference accelerator based on the 2.5D integration of multiple resistive RAM chiplets that allows fabrication cost reductions. The authors also propose a design space exploration flow to optimize the interconnection Network-on-Package, minimizing inter-chiplet communications and enabling link pruning.

Inter-chiplet communication remains one of the main challenges in multi-chiplet architectures. Authors in [166] investigate photonic-based interconnects as an alternative to metallic-based inter-chiplet networks and propose a DNN inference accelerator namely SPRINT.

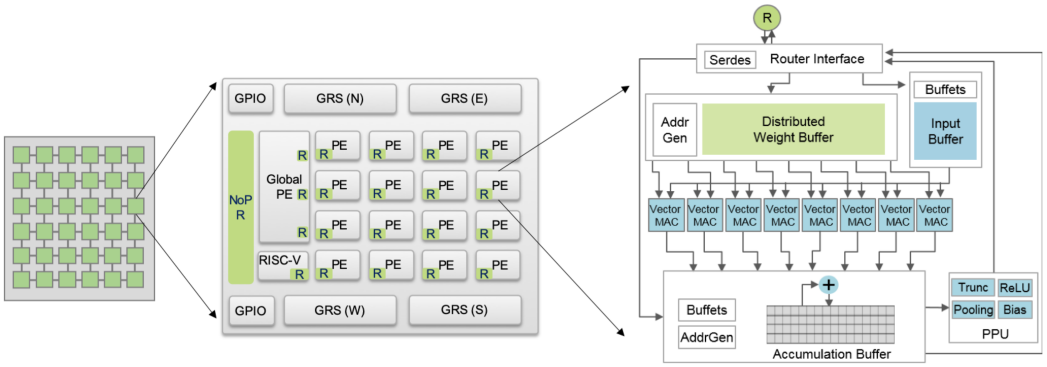

Fig. 14. Simba architecture [237] from left to right: package with 36 chiplets, chiplet, and processing element.

Finally, as a representative chiplet-based DNN hardware accelerator, we report Simba [237, 301]. Simba is a scalable deep neural network (DNN) accelerator consisting of 36 chiplets connected in a mesh network on a multi-chip-module using ground-referenced signaling. Simba enables flexible scaling for efficient inference on a wide range of DNNs, from mobile to data center domains. The prototype achieves high area efficiency, energy efficiency, and peak performance for both one-chiplet and 36-chiplet systems. Simba architecture is shown in Fig. 14. It implements a tile-based architecture and adopts a hierarchical interconnect to efficiently connect different processing elements (PEs). This hierarchical interconnect consists of a network-on-chip (NoC) that connects PEs on the same chiplet and a network-on-package (NoP) that connects chiplets together on the same package. Each Simba chiplet contains an array of PEs, a global PE, a NoP router, and a controller, all connected by a chiplet-level interconnect.

Table 12 presents a summary of the key characteristics of a representative subset of chiplet-based DNN accelerators that were reviewed earlier.

6 OPEN CHALLENGES AND CONCLUSIONS

The Deep Learning ecosystem based on advanced computer architectures and memory technologies spans from edge computing solutions to high-performance servers, supercomputers, up to large data centers for data analytics. In this context, the main objective of this survey is to provide an overview of the leading computing platforms utilized for accelerating the execution and enhancing the efficiency of high-performance Deep Learning applications. More in detail, this survey includes GPU-based accelerators, Tensor Processor Units, FPGA-based accelerators, Neural Processing Units and co-processors based on the open-hardware RISC-V architecture. The survey also describes accelerators based on emerging technologies and computing paradigms, such as 3D-stacked Processor-In-Memory, emerging non-volatile memories such as the Resistive switching Random Access Memory (RRAM) and the Phase Change Memory (PCM), Neuromorphic Processing Units and Multi-Chip Modules.

In the following part of the section, we briefly discuss open challenges of promising technologies that can be used for the acceleration of Deep Learning workloads: quantum computing and photonic computing.

Table 12. Summary of Chiplet-based DNN Accelerators.

	Simba [237]	Centaur [117]	Lego [288]	Chimera [216]	SWAP [238]	SPRINT [166]
Technology	16nm	FPGA	FPGA	40nm	-	28nm
Area	6 mm^{2*}	-	19016 LUT, 16916 FF 0.5 BRAM, 28 DSP†	29.2 mm^2	-	-
Power Efficiency	9.1 TOPS/W**	-	-	2.2 TOPS/W	-	-
Throughput	4–128 TOPS	0.313 TOPS	-	0.92 TOPS	-	-
Frequency	161 MHz–1.8 GHz	200 MHz	-	200 MHz	-	-
Precisions	int8	-	-	int8, fp16	-	-
On-chip Memory	752 KiB*	-	-	2.5 MB‡	-	128 KiB*
Chiplet Bandwidth	100 GB/s	-	-	1.9 Gbps	-	180 Gbps
Processing Type	Near-Memory	Near-Memory	Near-Memory	Near-Memory	In-Memory	Near-Memory
Sparsity	✗	✓	✗	✗	✗	✗
Interconnect	Wired Mesh (GRS)	-	-	Wired App. specific‡‡	Wired Pruned	Optical§
Applications	CNN Inference	Recommendation Inference	Multi-Tenant Inference	Inference Training	Multiple Applications	CNN Inference

*One chiplet, **When operating at a minimum voltage of 0.42 V with a 161 MHz PE frequency
†16×16 chiplet
‡2 MB RRAM, 0.5 MB SRAM, ‡‡C2C links (77 pJ/bit, 1.9 Gbits/s)
§0.77 pJ/bit

There is a general agreement that quantum computers will not replace classical computing systems, but they will be used in combination with supercomputers to accelerate some hard-to-compute problems. Quantum computers will play the role of unconventional accelerators with the goal to outperform conventional supercomputers, thanks to the improved parallelism which enables the so-called *quantum speedup*.

An example of quantum speedup is provided by the Shor's algorithm [240], which allows to find the prime factors of a large number in polynomial-time, therefore faster than any classical algorithm. Being the prime factorization at the hearth of breaking the RSA-based cryptography, the Shor's factorization algorithm immediately attracted the attention of national governments and opened up the era of post-quantum cryptography. Other complex problems, where quantum computing can help, include some fundamental numerical problems used in chemistry and physics for the development of new drugs and materials. So far, research efforts have been spent not only in developing new algorithms, but also robust, reliable and scalable quantum bits, as well as the quantum software stack (i.e. programming languages, compilers and runtime systems) needed to take full advantage of the quantum speedup to solve complex real-world problems.

Recently, a survey on Quantum Computing technology appeared in [101], while another survey on QC frameworks appeared in [264]. More specifically, there is a research trend on the so-called *Quantum Machine Learning* [23] which aims at developing quantum algorithms that outperform classical computing algorithms on machine learning tasks such as recommendation systems. More in detail, classical deep neural networks inspired the development of *Deep Quantum Learning* methods. The main advantage of these methods is that they do not require a large, general-purpose quantum computer. Quantum annealers, such as the D-Wave commercial solutions [58], are well-suited for implementing deep quantum learners. Quantum annealers are special-purpose quantum processors that are significantly easier to construct and to scale up than general-purpose quantum computers. To answer to this research trend, Google proposed TensorFlow Quantum (TFQ) [28], an open-source quantum machine learning library that could be used for prototyping hybrid quantum-classical ML models for classical or quantum data

Governments, supercomputing centers and companies around the world have also started to investigate *How*/*When*/*Where* quantum processing units (QPUs) could fit into HPC infrastructures

to speed up some heavy tasks, such as Deep Learning workloads. Emerging trends and commercial solutions related to *hybrid* quantum-classical supercomputers are described in [114].

To address this challenging trend, in October 2022, the EuroHPC Joint Undertaking initiative has selected six supercomputing centers across the European Union to host the first European quantum computers by the end of 2023. The selected supercomputing centers will integrate quantum computers and simulators (QCS) to build the so-called European Quantum Computing & Simulation infrastructure (EuroQCS), offering researchers the cloud access to these resources. The view of the European members of the EuroQCS community has recently been expressed in [75].

IBM Research was the first provider to offer a cloud-based QC service: Qiskit [222], an open-source SDK for working with quantum computers based on a library of quantum gates/circuits and pre-built solutions for research and application developers. The remote users can develop quantum programs and execute them on quantum simulators and cloud-based quantum processors.

Cloud providers have also jumped in the quantum race. In this way, Quantum Computing is becoming accessible by cloud providers offering the software stack for customers to explore the feasibility for their applications. As an example, Amazon Braket [27] is a quantum computing service based on different types of quantum systems and simulators (including the quantum annealer from D-Wave) to speed up development of scientific research and software exploiting quantum computing solutions.

On this trend, there is a general agreement that GPUs will play a key role in hybrid quantum-classical computing systems. GPU company NVIDIA is not developing a quantum processor, but is offering the CuQuantum software kit for quantum simulation. The NVIDIA's CuQuantum DGX hardware appliance integrates a software container on a full-stack quantum circuit simulator. The system uses NIVIDIA's A100 GPUs to accelerate quantum simulation workloads.

The second challenging and promising research direction is represented by the use of photonic computing to further accelerate DL tasks on HPC infrastructures. Photonic computing relies on the computation of electromagnetic waves typically via non-linear modulation and interference effects. Photonic computing was originally introduced in the 1980s to address optical pattern recognition and optical Fourier transform processing [14]. Despite the potential advantages of processing parallelism and speed, optical computing has never translated in a commercial technology. Only recently, due to the emergence of data intensive computing tasks such as AI, DL and ML, optical computing has seen a renewed interest by the research community of DL.

There are two main advantages of optical computing, namely (i) the inherent speed of signal transmission, where light pulses can be transferred without the typical RC delays and IR drop of electrical interconnects, and (ii) the inherent parallelism, where multiple wavelengths, polarizations and modes can be processed by the same hardware (e.g., waveguides, interferometers, etc.), without interfering with each other. These properties can provide strong benefits to data-intensive computing tasks such as DL. For instance, multiple activations can be modulated in different wavelengths and be processed in parallel within the same optical network of synaptic weights, thus allowing a high processing density. Finally, optical computing can overcome the memory wall of the von Neumann architecture, since computation is done physically within the data, thus largely reducing the amount of data that need to be transferred from the memory to the processing unit.

Photonic computing techniques have been reported by using different approaches for the main computational bottlenecks, i.e., matrix-vector multiplication (MVM), in conventional AI solutions. For instance, MVM can be operated by microring resonators (MRRs) used as tunable filters, where the weight is encoded in a change of refractive index via thermo-optic, electro-optic, or phase-change effect [239]. The nonlinear activation, where an artificial photonic neuron stimulated by an optical input MVM signal nonlinearly generates an optical output, is significantly more challenging. As a result, non-linear activation is generally operated in the electrical domain by converting the

optical MVM signals into electrical signals. Similarly, the electrical activation needs to be converted to the optical domain by suitable coherent or incoherent light sources.

Photonic computing represents a promising platform for accelerating AI. For instance, it has been estimated that photonic multiply-accumulate operations can show significant improvements over digital electronics in terms of energy efficiency ($> 10^2$), speed ($> 10^3$), and compute density ($> 10^2$) [190]. However there are still many challenges toward developing an industrially feasible photonic system. The main challenge is the area/energy inefficiency of processing across the mixed optical/electronic domain. Optical-electrical conversion and vice versa results in a considerable overhead in terms of area and power consumption. To bridge this gap, the research is developing silicon photonic integrated circuits (PICs) with increasing robustness, manufacturability and scalability. From the accuracy viewpoint, it should be noted that photonic computing essentially operates in the analog domain thus accuracy is deeply affected by accumulated noise and imprecision of the various optical devices, such as electro-optic and phase change modulators. These challenges, which are similar to those arising in analog IMC, might be addressed and mitigated by suitable hardware-software co-design, *e.g.*, hardware-aware training or other system-level calibration techniques.

ACKNOWLEDGMENTS

This work has been supported by the Spoke 1 *FutureHPC & BigData* of the Italian Research Center on High-Performance Computing, Big Data and Quantum Computing (ICSC) funded by MUR Mission 4 - Next Generation EU.

REFERENCES

[1] 2009. *NVIDIA, Fermi.* Retrieved Apr 16, 2023 from http://www.nvidia.com/content/PDF/fermi_white_papers/NVIDIA_Fermi_Compute_Architecture_Whitepaper.pdf

[2] 2011. *The Most Cost-Effective Integrator (TSV Interposer) for 3D IC Integration System-in-Package (SiP).* International Electronic Packaging Technical Conference and Exhibition, Vol. ASME 2011 Pacific Rim Technical Conference and Exhibition on Packaging and Integration of Electronic and Photonic Systems, MEMS and NEMS: Volume 1. https://doi.org/10.1115/IPACK2011-52189 arXiv:https://asmedigitalcollection.asme.org/InterPACK/proceedings-pdf/InterPACK2011/44618/53/4597366/53_1.pdf

[3] 2012. *NVIDIA, Kepler GK110.* Retrieved Apr 16, 2023 from http://www.nvidia.com/content/PDF/kepler/NVIDIA-Kepler-GK110-Architecture-Whitepaper.pdf

[4] 2023. GreenWaves Technologies GAP9 Processor. https://greenwaves-technologies.com/gap9_processor/. Accessed: 2023-04-18.

[5] 2023. MLPerf Inference Tiny v1.0 Results. https://mlcommons.org/en/inference-tiny-10/. Accessed: 2023-04-18.

[6] 2023. Ventana Micro. https://www.ventanamicro.com/. Accessed: 2023-04-18.

[7] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.

[8] Ankur Agrawal, Sae Kyu Lee, Joel Silberman, Matthew Ziegler, Mingu Kang, Swagath Venkataramani, Nianzheng Cao, Bruce Fleischer, Michael Guillorn, Matthew Cohen, et al. 2021. 9.1 A 7nm 4-core AI chip with 25.6 TFLOPS hybrid FP8 training, 102.4 TOPS INT4 inference and workload-aware throttling. In *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, Vol. 64. IEEE, 144–146. https://doi.org/10.1109/ISSCC42613.2021.9365791

[9] Mohammad Ahmadinejad, Mohammad Hossein Moaiyeri, and Farnaz Sabetzadeh. 2019. Energy and area efficient imprecise compressors for approximate multiplication at nanoscale. *AEU - International Journal of Electronics and Communications* 110 (2019), 152859. https://doi.org/10.1016/j.aeue.2019.152859

[10] Alessandro Aimar, Hesham Mostafa, Enrico Calabrese, Antonio Rios-Navarro, Ricardo Tapiador-Morales, Iulia-Alexandra Lungu, Moritz B. Milde, Federico Corradi, Alejandro Linares-Barranco, Shih-Chii Liu, and Tobi Delbruck. 2019. NullHop: A Flexible Convolutional Neural Network Accelerator Based on Sparse Representations of Feature

Maps. *IEEE Transactions on Neural Networks and Learning Systems* 30, 3 (2019), 644–656. https://doi.org/10.1109/TNNLS.2018.2852335

[11] Filipp Akopyan, Jun Sawada, Andrew Cassidy, Rodrigo Alvarez-Icaza, John Arthur, Paul Merolla, Nabil Imam, Yutaka Nakamura, Pallab Datta, Gi-Joon Nam, Brian Taba, Michael Beakes, Bernard Brezzo, Jente B. Kuang, Rajit Manohar, William P. Risk, Bryan Jackson, and Dharmendra S. Modha. 2015. TrueNorth: Design and Tool Flow of a 65 mW 1 Million Neuron Programmable Neurosynaptic Chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34, 10 (2015), 1537–1557. https://doi.org/10.1109/TCAD.2015.2474396

[12] Jorge Albericio, Patrick Judd, Tayler Hetherington, Tor Aamodt, Natalie Enright Jerger, and Andreas Moshovos. 2016. Cnvlutin: Ineffectual-Neuron-Free Deep Neural Network Computing. In *Proceedings of the 43rd International Symposium on Computer Architecture* (Seoul, Republic of Korea) *(ISCA'16)*. 13 pages. https://doi.org/10.1109/ISCA.2016.11

[13] Marco A. Z. Alves, Matthias Diener, Paulo C. Santos, and Luigi Carro. 2016. Large vector extensions inside the HMC. In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 1249–1254.

[14] Pierre Ambs. 2010. Optical Computing: A 60-Year Adventure. *Advances in Optical Technologies* 2010, 372652 (May 2010). https://doi.org/10.1155/2010/372652

[15] AMD. 2021. *AMD INSTINCT MI200 SERIES ACCELERATOR*. Retrieved May 25, 2023 from https://www.amd.com/system/files/documents/amd-instinct-mi200-datasheet.pdf

[16] AMD-Xilinx. 2023. VitisAI develop environment. https://www.xilinx.com/products/design-tools/vitis/vitis-ai.html

[17] Mohammad Faisal Amir, Jong Hwan Ko, Taesik Na, Duckhwan Kim, and Saibal Mukhopadhyay. 2018. 3-D Stacked Image Sensor With Deep Neural Network Computation. *IEEE Sensors Journal* 18, 10 (2018), 4187–4199. https://doi.org/10.1109/JSEN.2018.2817632

[18] Michael Andersch, Greg Palmer, Ronny Krashinsky, Nick Stam, Vishal Mehta, Gonzalo Brito, and Sridhar Ramaswamy. 2022. *NVIDIA Hopper Architecture In-Depth*. Retrieved Apr 16, 2023 from https://developer.nvidia.com/blog/nvidia-hopper-architecture-in-depth/

[19] Akhil Arunkumar, Evgeny Bolotin, Benjamin Cho, Ugljesa Milic, Eiman Ebrahimi, Oreste Villa, Aamer Jaleel, Carole-Jean Wu, and David Nellans. 2017. MCM-GPU: Multi-chip-module GPUs for continued performance scalability. In *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*. 320–332.

[20] Imad Al Assir, Mohamad El Iskandarani, Hadi Rayan Al Sandid, and Mazen A. R. Saghir. 2021. Arrow: A RISC-V Vector Accelerator for Machine Learning Inference. https://doi.org/10.48550/arXiv.2107.07169 arXiv:arXiv:2107.07169

[21] Yoshua Bengio. 2009. Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning* 2, 1 (2009), 1–127. https://doi.org/10.1561/2200000006

[22] Luca Bertaccini, Gianna Paulin, Tim Fischer, Stefan Mach, and Luca Benini. 2022. MiniFloat-NN and ExSdotp: An ISA Extension and a Modular Open Hardware Unit for Low-Precision Training on RISC-V Cores. In *2022 IEEE 29th Symposium on Computer Arithmetic (ARITH)*. 1–8. https://doi.org/10.1109/ARITH54963.2022.00010

[23] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. 2017. Quantum machine learning. *Nature* 549, 7671 (sep 2017), 195–202. https://doi.org/10.1038/nature23474

[24] Ahmed Ghazi Blaiech, Khaled Ben Khalifa, Carlos Valderrama, Marcelo AC Fernandes, and Mohamed Hedi Bedoui. 2019. A survey and taxonomy of FPGA-based deep learning accelerators. *Journal of Systems Architecture* 98 (2019), 331–345.

[25] Michaela Blott, Thomas B Preußer, Nicholas J Fraser, Giulio Gambardella, Kenneth O'brien, Yaman Umuroglu, et al. 2018. FINN-R: An end-to-end deep-learning framework for fast exploration of quantized neural networks. *ACM Transactions on Reconfigurable Technology and Systems* 11, 3 (2018), 1–23.

[26] Nicolas Bohm Agostini, Serena Curzel, Jeff Jun Zhang, Ankur Limaye, Cheng Tan, Vinay Amatya, Marco Minutoli, Vito Giovanni Castellana, Joseph Manzano, David Brooks, Gu-Yeon Wei, and Antonino Tumeo. 2022. Bridging Python to Silicon: The SODA Toolchain. *IEEE Micro* 42, 5 (2022), 78–88.

[27] Braket 2023. Quantum Computing Service - Amazon Braket - AWS. https://aws.amazon.com/braket/

[28] Michael Broughton, Guillaume Verdon, Trevor McCourt, Antonio J. Martinez, Jae Hyeon Yoo, Sergei V. Isakov, Philip Massey, Ramin Halavati, Murphy Yuezhen Niu, Alexander Zlokapa, Evan Peters, Owen Lockwood, Andrea Skolik, Sofiene Jerbi, Vedran Dunjko, Martin Leib, Michael Streif, David Von Dollen, Hongxiang Chen, Shuxiang Cao, Roeland Wiersema, Hsin-Yuan Huang, Jarrod R. McClean, Ryan Babbush, Sergio Boixo, Dave Bacon, Alan K. Ho, Hartmut Neven, and Masoud Mohseni. 2021. TensorFlow Quantum: A Software Framework for Quantum Machine Learning. https://doi.org/10.48550/arXiv.2003.02989 arXiv:2003.02989 [quant-ph]

[29] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. *CoRR* abs/2005.14165

(2020). https://arxiv.org/abs/2005.14165

[30] Nazareno Bruschi, Giuseppe Tagliavini, Angelo Garofalo, Francesco Conti, Irem Boybat, Luca Benini, and Davide Rossi. 2022. End-to-End DNN Inference on a Massively Parallel Analog In Memory Computing Architecture. https://doi.org/10.48550/arXiv.2211.12877 arXiv:arXiv:2211.12877

[31] Cadence. 2022. *Stratus High-Level Synthesis.* https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/synthesis/stratus-high-level-synthesis.html

[32] Andrew Canis, Jongsok Choi, Mark Aldham, Victor Zhang, Ahmed Kammoona, Tomasz S. Czajkowski, Stephen Dean Brown, and Jason Helge Anderson. 2013. LegUp: An open-source high-level synthesis tool for FPGA-based processor/accelerator systems. *ACM Trans. Embed. Comput. Syst.* 13, 2 (2013), 24:1–24:27. https://doi.org/10.1145/2514740

[33] Weidong Cao, Yilong Zhao, Adith Boloor, Yinhe Han, Xuan Zhang, and Li Jiang. 2022. Neural-PIM: Efficient Processing-In-Memory With Neural Approximation of Peripherals. *IEEE Trans. Comput.* 71, 9 (2022), 2142–2155. https://doi.org/10.1109/TC.2021.3122905

[34] Alex Carsello, Kathleen Feng, Taeyoung Kong, Kalhan Koul, Qiaoyi Liu, Jackson Melchert, Gedeon Nyengele, Maxwell Strange, Keyi Zhang, Ankita Nayak, Jeff Setter, James Thomas, Kavya Sreedhar, Po-Han Chen, Nikhil Bhagdikar, Zachary Myers, Brandon D'Agostino, Pranil Joshi, Stephen Richardson, Rick Bahr, Christopher Torng, Mark Horowitz, and Priyanka Raina. 2022. Amber: A 367 GOPS, 538 GOPS/W 16nm SoC with a Coarse-Grained Reconfigurable Array for Flexible Acceleration of Dense Linear Algebra. In *2022 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*. 70–71. https://doi.org/10.1109/VLSITechnologyandCir46769.2022.9830509

[35] Andrew S. Cassidy, Paul Merolla, John V. Arthur, Steve K. Esser, Bryan Jackson, Rodrigo Alvarez-Icaza, Pallab Datta, Jun Sawada, Theodore M. Wong, Vitaly Feldman, Arnon Amir, Daniel Ben-Dayan Rubin, Filipp Akopyan, Emmett McQuinn, William P. Risk, and Dharmendra S. Modha. 2013. Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*. 1–10. https://doi.org/10.1109/IJCNN.2013.6707077

[36] Matheus Cavalcante, Fabian Schuiki, Florian Zaruba, Michael Schaffner, and Luca Benini. 2020. Ara: A 1-GHz+ Scalable and Energy-Efficient RISC-V Vector Processor With Multiprecision Floating-Point Support in 22-Nm FD-SOI. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 28, 2 (Feb. 2020), 530–543. https://doi.org/10.1109/TVLSI.2019.2950087

[37] Matheus Cavalcante, Domenic Wüthrich, Matteo Perotti, Samuel Riedel, and Luca Benini. 2022. Spatz: A Compact Vector Processing Unit for High-Performance and Energy-Efficient Shared-L1 Clusters. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*. ACM, San Diego California, 1–9. https://doi.org/10.1145/3508352.3549367

[38] Lukas Cavigelli and Luca Benini. 2017. Origami: A 803-GOp/s/W Convolutional Network Accelerator. *IEEE Transactions on Circuits and Systems for Video Technology* 27, 11 (2017), 2461–2475. https://doi.org/10.1109/TCSVT.2016.2592330

[39] Jung-Woo Chang, Keon-Woo Kang, and Suk-Ju Kang. 2020. An Energy-Efficient FPGA-Based Deconvolutional Neural Networks Accelerator for Single Image Super-Resolution. *IEEE Transactions on Circuits and Systems for Video Technology* 30, 1 (2020), 281–295. https://doi.org/10.1109/TCSVT.2018.2888898

[40] Karam Chatha. 2021. Qualcomm® Cloud Al 100 : 12TOPS/W Scalable, High Performance and Low Latency Deep Learning Inference Accelerator. In *2021 IEEE Hot Chips 33 Symposium (HCS)*. 1–19. https://doi.org/10.1109/HCS52781.2021.9567417

[41] Chen Chen, Xiaoyan Xiang, Chang Liu, Yunhai Shang, Ren Guo, Dongqi Liu, Yimin Lu, Ziyi Hao, Jiahui Luo, Zhijian Chen, Chunqiang Li, Yu Pu, Jianyi Meng, Xiaolang Yan, Yuan Xie, and Xiaoning Qi. 2020. Xuantie-910: A Commercial Multi-Core 12-Stage Pipeline Out-of-Order 64-Bit High Performance RISC-V Processor with Vector Extension : Industrial Product. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. 52–64. https://doi.org/10.1109/ISCA45697.2020.00016

[42] Gregory K. Chen, Phil C. Knag, Carlos Tokunaga, and Ram K. Krishnamurthy. 2022. An Eight-Core RISC-V Processor With Compute Near Last Level Cache in Intel 4 CMOS. *IEEE Journal of Solid-State Circuits* (2022), 1–12. https://doi.org/10.1109/JSSC.2022.3228765

[43] Yunji Chen, Tianshi Chen, Zhiwei Xu, Ninghui Sun, and Olivier Temam. 2016. DianNao Family: Energy-Efficient Hardware Accelerators for Machine Learning. *Commun. ACM* 59, 11 (Oct 2016), 105–112. https://doi.org/10.1145/2996864

[44] Yiran Chen, Yuan Xie, Linghao Song, Fan Chen, and Tianqi Tang. 2020. A Survey of Accelerator Architectures for Deep Neural Networks. *Engineering* 6, 3 (2020), 264–274. https://doi.org/10.1016/j.eng.2020.01.007

[45] Yu-Hsin Chen, Tushar Krishna, Joel S. Emer, and Vivienne Sze. 2017. Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks. *IEEE Journal of Solid-State Circuits* 52, 1 (Jan. 2017), 127–138. https://doi.org/10.1109/JSSC.2016.2616357

[46] Yu-Hsin Chen, Tien-Ju Yang, Joel Emer, and Vivienne Sze. 2019. Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9, 2 (2019), 292–308. https://doi.org/10.1109/JETCAS.2019.2910232

[47] Chao Cheng and K.K. Parhi. 2004. Hardware efficient fast parallel FIR filter structures based on iterated short convolution. *IEEE Transactions on Circuits and Systems I: Regular Papers* 51, 8 (2004), 1492–1500. https://doi.org/10.1109/TCSI.2004.832784

[48] Chao Cheng and Keshab K. Parhi. 2020. Fast 2D Convolution Algorithms for Convolutional Neural Networks. *IEEE Transactions on Circuits and Systems I: Regular Papers* 67, 5 (2020), 1678–1691. https://doi.org/10.1109/TCSI.2020.2964748

[49] Ping Chi, Shuangchen Li, Cong Xu, Tao Zhang, Jishen Zhao, Yongpan Liu, Yu Wang, and Yuan Xie. 2016. PRIME: A Novel Processing-in-Memory Architecture for Neural Network Computation in ReRAM-Based Main Memory. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. 27–39. https://doi.org/10.1109/ISCA.2016.13

[50] Krishna Teja Chitty-Venkata and Arun K. Somani. 2022. Neural Architecture Search Survey: A Hardware Perspective. *ACM Comput. Surv.* 55, 4, Article 78 (nov 2022), 36 pages. https://doi.org/10.1145/3524500

[51] Jack Choquette. 2023. NVIDIA Hopper H100 GPU: Scaling Performance. *IEEE Micro* (2023), 1–13. https://doi.org/10.1109/MM.2023.3256796

[52] Jack Choquette, Olivier Giroux, and Denis Foley. 2018. Volta: Performance and Programmability. *IEEE Micro* 38, 2 (March 2018), 42–52. https://doi.org/10.1109/MM.2018.022071134

[53] Jack Choquette, Edward Lee, Ronny Krashinsky, Vishnu Balan, and Brucek Khailany. 2021. 3.2 The A100 Datacenter GPU and Ampere Architecture. In *2021 IEEE International Solid- State Circuits Conference (ISSCC)*, Vol. 64. 48–50. https://doi.org/10.1109/ISSCC42613.2021.9365803

[54] Peter Clarke. 2015. Intel, Micron Launch "Bulk-Switching" ReRAM. https://www.eetimes.com/intel-micron-launch-bulk-switching-reram/. Accessed: 18/04/2023.

[55] Marco Cococcioni, Federico Rossi, Emanuele Ruffaldi, and Sergio Saponara. 2022. A Lightweight Posit Processing Unit for RISC-V Processors in Deep Neural Network Applications. *IEEE Transactions on Emerging Topics in Computing* 10, 4 (2022), 1898–1908. https://doi.org/10.1109/TETC.2021.3120538

[56] Francesco Conti, Davide Rossi, Gianna Paulin, Anaelo Garofalo, Alfio Di Mauro, Georg Rutishauer, Gian marco Ottavi, Manuel Eggimann, Hayate Okuhara, Vincent Huard, Olivier Montfort, Lionel Jure, Nils Exibard, Pascal Gouedo, Mathieu Louvat, Emmanuel Botte, and Luca Benini. 2023. 22.1 A 12.4TOPS/W @ 136GOPS AI-IoT System-on-Chip with 16 RISC-V, 2-to-8b Precision-Scalable DNN Acceleration and 30%-Boost Adaptive Body Biasing. In *2023 IEEE International Solid- State Circuits Conference (ISSCC)*. 21–23. https://doi.org/10.1109/ISSCC42615.2023.10067643

[57] Aline S. Cordeiro, Sairo R. dos Santos, Francis B. Moreira, Paulo C. Santos, Luigi Carro, and Marco A. Z. Alves. 2021. Machine Learning Migration for Efficient Near-Data Processing. In *2021 29th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*. 212–219. https://doi.org/10.1109/PDP52278.2021.00041

[58] D-WAVE 2023. D-Wave Systems - The Practical Quantum Computing Company. https://www.dwavesys.com/

[59] Scott Davidson, Shaolin Xie, Christopher Torng, Khalid Al-Hawai, Austin Rovinski, Tutu Ajayi, Luis Vega, Chun Zhao, Ritchie Zhao, Steve Dai, Aporva Amarnath, Bandhav Veluri, Paul Gao, Anuj Rao, Gai Liu, Rajesh K. Gupta, Zhiru Zhang, Ronald Dreslinski, Christopher Batten, and Michael Bedford Taylor. 2018. The Celerity Open-Source 511-Core RISC-V Tiered Accelerator Fabric: Fast Architectures and Design Methodologies for Fast Chips. *IEEE Micro* 38, 2 (March 2018), 30–41. https://doi.org/10.1109/MM.2018.022071133

[60] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, Yuyun Liao, Chit-Kwan Lin, Andrew Lines, Ruokun Liu, Deepak Mathaikutty, Steven McCoy, Arnab Paul, Jonathan Tse, Guruguhanathan Venkataramanan, Yi-Hsin Weng, Andreas Wild, Yoonseok Yang, and Hong Wang. 2018. Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. *IEEE Micro* 38, 1 (2018), 82–99. https://doi.org/10.1109/MM.2018.112130359

[61] Tim Davis. 2007. Wilkinson's Sparse Matrix Definition. *NA Digest* 7, 12 (2007), 379–401.

[62] Alberto Delmas Lascorz, Patrick Judd, Dylan Malone Stuart, Zissis Poulos, Mostafa Mahmoud, Sayeh Sharify, Milos Nikolic, Kevin Siu, and Andreas Moshovos. 2019. Bit-Tactical: A Software/Hardware Approach to Exploiting Value and Bit Sparsity in Neural Networks. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems* (Providence, RI, USA) *(ASPLOS '19)*. 749–763. https://doi.org/10.1145/3297858.3304041

[63] Chunhua Deng, Siyu Liao, Yi Xie, Keshab K. Parhi, Xuehai Qian, and Bo Yuan. 2018. PermDNN: Efficient Compressed DNN Architecture with Permuted Diagonal Matrices. In *Proceedings of the 51st Annual IEEE/ACM International Symposium on Microarchitecture* (Fukuoka, Japan) *(MICRO-51)*. 189–202. https://doi.org/10.1109/MICRO.2018.00024

[64] Giuseppe Desoli, Nitin Chawla, Thomas Boesch, Manui Avodhyawasi, Harsh Rawat, Hitesh Chawla, VS Abhijith, Paolo Zambotti, Akhilesh Sharma, Carmine Cappetta, Michele Rossi, Antonio De Vita, and Francesca Girardi. 2023. A

40-310TOPS/W SRAM-Based All-Digital Up to 4b In-Memory Computing Multi-Tiled NN Accelerator in FD-SOI 18nm for Deep-Learning Edge Applications. In *2023 IEEE International Solid- State Circuits Conference (ISSCC)*. 260–262. https://doi.org/10.1109/ISSCC42615.2023.10067422

[65] Giuseppe Desoli, Nitin Chawla, Thomas Boesch, Surinder-pal Singh, Elio Guidetti, Fabio De Ambroggi, Tommaso Majo, Paolo Zambotti, Manuj Ayodhyawasi, Harvinder Singh, and Nalin Aggarwal. 2017. A 2.9TOPS/W deep convolutional neural network SoC in FD-SOI 28nm for intelligent embedded systems. In *2017 IEEE International Solid-State Circuits Conference (ISSCC)*. 238–239. https://doi.org/10.1109/ISSCC.2017.7870349

[66] Pudi Dhilleswararao, Srinivas Boppu, M. Sabarimalai Manikandan, and Linga Reddy Cenkeramaddi. 2022. Efficient Hardware Architectures for Accelerating Deep Neural Networks: Survey. *IEEE Access* 10 (2022), 131788–131828. https://doi.org/10.1109/ACCESS.2022.3229767

[67] Alfio Di Mauro, Moritz Scherer, Davide Rossi, and Luca Benini. 2022. Kraken: A Direct Event/Frame-Based Multi-sensor Fusion SoC for Ultra-Efficient Visual Processing in Nano-UAVs. https://doi.org/10.48550/arXiv.2209.01065 arXiv:arXiv:2209.01065

[68] David R. Ditzel and the Esperanto team. 2022. Accelerating ML Recommendation With Over 1,000 RISC-V/Tensor Processors on Esperanto's ET-SoC-1 Chip. *IEEE Micro* 42, 3 (May 2022), 31–38. https://doi.org/10.1109/MM.2022.3140674

[69] Li Du, Yuan Du, Yilei Li, and Mau-Chung Frank Chang. 2017. A Reconfigurable Streaming Deep Convolutional Neural Network Accelerator for Internet of Things. *IEEE Transactions on Circuits and Systems I: Regular Papers* PP (07 2017). https://doi.org/10.1109/TCSI.2017.2735490

[70] Javier Duarte et al. 2018. Fast inference of deep neural networks in FPGAs for particle physics. *JINST* 13, 07 (2018), P07027. https://doi.org/10.1088/1748-0221/13/07/P07027 arXiv:1804.06913 [physics.ins-det]

[71] Luke Durant, Olivier Giroux, Mark Harris, and Nick Stam. 2017. *Inside Volta: The World's Most Advanced Data Center GPU*. Retrieved Apr 16, 2023 from https://developer.nvidia.com/blog/inside-volta/

[72] Anne C. Elster and Tor A. Haugdahl. 2022. Nvidia Hopper GPU and Grace CPU Highlights. *Computing in Science & Engineering* 24, 2 (March 2022), 95–100. https://doi.org/10.1109/MCSE.2022.3163817

[73] Hadi Esmaeilzadeh, Emily Blem, Renée St. Amant, Karthikeyan Sankaralingam, and Doug Burger. 2011. Dark silicon and the end of multicore scaling. In *2011 38th Annual International Symposium on Computer Architecture (ISCA)*. 365–376.

[74] Darjn Esposito, Antonio G. M. Strollo, and Massimo Alioto. 2017. Low-power approximate MAC unit. In *2017 13th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*. 81–84. https://doi.org/10.1109/PRIME.2017.7974112

[75] EuroQCS 2022. European Quantum Computing and Simulation Infrastructure. https://qt.eu/media/pdf/20220202_HPC-QCS-JWP-final.pdf?m=1674828481&

[76] William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *Journal of Machine Learning Research* 23, 120 (2022), 1–39. http://jmlr.org/papers/v23/21-0998.html

[77] Fabrizio Ferrandi, Vito Giovanni Castellana, Serena Curzel, Pietro Fezzardi, Michele Fiorito, Marco Lattuada, Marco Minutoli, Christian Pilato, and Antonino Tumeo. 2021. Invited: Bambu: an Open-Source Research Framework for the High-Level Synthesis of Complex Applications. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. 1327–1330. https://doi.org/10.1109/DAC18074.2021.9586110

[78] Salvatore Filippone, Valeria Cardellini, Davide Barbieri, and Alessandro Fanfarillo. 2017. Sparse Matrix-Vector Multiplication on GPGPUs. *ACM Trans. Math. Softw.* 43, 4, Article 30 (2017), 49 pages. https://doi.org/10.1145/3017994

[79] Jeremy Fowers, Kalin Ovtcharov, Michael Papamichael, Todd Massengill, Ming Liu, Daniel Lo, Shlomi Alkalay, Michael Haselman, Logan Adams, Mahdi Ghandi, Stephen Heil, Prerak Patel, Adam Sapek, Gabriel Weisz, Lisa Woods, Sitaram Lanka, Steven K. Reinhardt, Adrian M. Caulfield, Eric S. Chung, and Doug Burger. 2018. A Configurable Cloud-Scale DNN Processor for Real-Time AI. In *ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. 1–14.

[80] Charlotte Frenkel, David Bol, and Giacomo Indiveri. 2021. Bottom-Up and Top-Down Neural Processing Systems Design: Neuromorphic Intelligence as the Convergence of Natural and Artificial Intelligence. *CoRR* abs/2106.01288 (2021).

[81] Charlotte Frenkel, Martin Lefebvre, Jean-Didier Legat, and David Bol. 2019. A 0.086-mm^2 12.7-pJ/SOP 64k-Synapse 256-Neuron Online-Learning Digital Spiking Neuromorphic Processor in 28-nm CMOS. *IEEE Transactions on Biomedical Circuits and Systems* 13, 1 (2019), 145–158. https://doi.org/10.1109/TBCAS.2018.2880425

[82] Charlotte Frenkel, Jean-Didier Legat, and David Bol. 2019. MorphIC: A 65-nm 738k-Synapse/mm^2 Quad-Core Binary-Weight Digital Neuromorphic Processor With Stochastic Spike-Driven Online Learning. *IEEE Transactions on Biomedical Circuits and Systems* 13, 5 (2019), 999–1010. https://doi.org/10.1109/TBCAS.2019.2928793

[83] Manuel Le Gallo, Riduan Khaddam-Aljameh, Milos Stanisavljevic, Athanasios Vasilopoulos, Benedikt Kersting, Martino Dazzi, Geethan Karunaratne, Matthias Braendli, Abhairaj Singh, Silvia M Mueller, et al. 2022. A 64-core mixed-signal in-memory compute chip based on phase-change memory for deep neural network inference. *arXiv preprint arXiv:2212.02872* (2022).

[84] Jianhua Gao, Weixing Ji, Fangli Chang, Shiyu Han, Bingxin Wei, Zeming Liu, and Yizhuo Wang. 2023. A Systematic Survey of General Sparse Matrix-Matrix Multiplication. *ACM Comput. Surv.* 55, 12, Article 244 (2023), 36 pages. https://doi.org/10.1145/3571157

[85] Mingyu Gao, Jing Pu, Xuan Yang, Mark Horowitz, and Christos Kozyrakis. 2017. TETRIS: Scalable and Efficient Neural Network Acceleration with 3D Memory *(ASPLOS '17)*. Association for Computing Machinery, New York, NY, USA, 751–764. https://doi.org/10.1145/3037697.3037702

[86] Mingyu Gao, Xuan Yang, Jing Pu, Mark Horowitz, and Christos Kozyrakis. 2019. TANGRAM: Optimized Coarse-Grained Dataflow for Scalable NN Accelerators. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems* (Providence, RI, USA) *(ASPLOS '19)*. Association for Computing Machinery, New York, NY, USA, 807–820. https://doi.org/10.1145/3297858.3304014

[87] Angelo Garofalo, Gianmarco Ottavi, Francesco Conti, Geethan Karunaratne, Irem Boybat, Luca Benini, and Davide Rossi. 2022. A Heterogeneous In-Memory Computing Cluster for Flexible End-to-End Inference of Real-World Deep Neural Networks. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 12, 2 (June 2022), 422–435. https://doi.org/10.1109/JETCAS.2022.3170152

[88] Angelo Garofalo, Yvan Tortorella, Matteo Perotti, Luca Valente, Alessandro Nadalini, Luca Benini, Davide Rossi, and Francesco Conti. 2022. DARKSIDE: A Heterogeneous RISC-V Compute Cluster for Extreme-Edge On-Chip DNN Inference and Training. *IEEE Open Journal of the Solid-State Circuits Society* 2 (2022), 231–243. https://doi.org/10.1109/OJSSCS.2022.3210082

[89] Hasan Genc, Seah Kim, Alon Amid, Ameer Haj-Ali, Vighnesh Iyer, Pranav Prakash, Jerry Zhao, Daniel Grubb, Harrison Liew, Howard Mao, Albert Ou, Colin Schmidt, Samuel Steffl, John Wright, Ion Stoica, Jonathan Ragan-Kelley, Krste Asanovic, Borivoje Nikolic, and Yakun Sophia Shao. 2021. Gemmini: Enabling Systematic Deep-Learning Architecture Evaluation via Full-Stack Integration. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. 769–774. https://doi.org/10.1109/DAC18074.2021.9586216

[90] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. 2021. A Survey of Quantization Methods for Efficient Neural Network Inference. arXiv:2103.13630 [cs.CV]

[91] Davide Giri, Kuan-Lin Chiu, Giuseppe Di Guglielmo, Paolo Mantovani, and Luca P. Carloni. 2020. ESP4ML: Platform-Based Design of Systems-on-Chip for Embedded Machine Learning. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 1049–1054. https://doi.org/10.23919/DATE48585.2020.9116317

[92] Davide Giri, Kuan-Lin Chiu, Guy Eichler, Paolo Mantovani, and Luca P. Carloni. 2021. Accelerator Integration for Open-Source SoC Design. *IEEE Micro* 41, 4 (July 2021), 8–14. https://doi.org/10.1109/MM.2021.3073893

[93] Ashish Gondimalla, Noah Chesnut, Mithuna Thottethodi, and T. N. Vijaykumar. 2019. SparTen: A Sparse Tensor Accelerator for Convolutional Neural Networks. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture* (Columbus, OH, USA) *(MICRO'52)*. 151–165. https://doi.org/10.1145/3352460.3358291

[94] Abraham Gonzalez, Jerry Zhao, Ben Korpan, Hasan Genc, Colin Schmidt, John Wright, Ayan Biswas, Alon Amid, Farhana Sheikh, Anton Sorokin, Sirisha Kale, Mani Yalamanchi, Ramya Yarlagadda, Mark Flannigan, Larry Abramowitz, Elad Alon, Yakun Sophia Shao, Krste Asanovic, and Borivoje Nikolic. 2021. A 16mm ^{2} 106.1 GOPS/W Heterogeneous RISC-V Multi-Core Multi-Accelerator SoC in Low-Power 22nm FinFET. In *ESSCIRC 2021 - IEEE 47th European Solid State Circuits Conference (ESSCIRC)*. IEEE, Grenoble, France, 259–262. https://doi.org/10.1109/ESSCIRC53450.2021.9567768

[95] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[96] Peng Gu, Xinfeng Xie, Yufei Ding, Guoyang Chen, Weifeng Zhang, Dimin Niu, and Yuan Xie. 2020. iPIM: Programmable In-Memory Image Processing Accelerator Using Near-Bank Architecture. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. 804–817. https://doi.org/10.1109/ISCA45697.2020.00071

[97] Yijin Guan, Hao Liang, Ningyi Xu, Wenqiang Wang, Shaoshuai Shi, Xi Chen, Guangyu Sun, Wei Zhang, and Jason Cong. 2017. FP-DNN: An Automated Framework for Mapping Deep Neural Networks onto FPGAs with RTL-HLS Hybrid Templates. In *2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. 152–159. https://doi.org/10.1109/FCCM.2017.25

[98] Cong Guo, Bo Yang Hsueh, Jingwen Leng, Yuxian Qiu, Yue Guan, Zehuan Wang, Xiaoying Jia, Xipeng Li, Minyi Guo, and Yuhao Zhu. 2020. Accelerating Sparse DNN Models without Hardware-Support via Tile-Wise Sparsity. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (Atlanta, Georgia) *(SC'20)*. Article 16, 15 pages.

[99] Kaiyuan Guo, Shulin Zeng, Jincheng Yu, Yu Wang, and Huazhong Yang. 2019. [DL] A survey of FPGA-based neural network inference accelerators. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)* 12, 1 (2019), 1–26.

[100] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. 2015. Deep Learning with Limited Numerical Precision. In *Proceedings of the 32nd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 37)*, Francis Bach and David Blei (Eds.). 1737–1746.

[101] Laszlo Gyongyosi and Sandor Imre. 2019. A Survey on quantum computing technology. *Computer Science Review* 31 (2019), 51–71. https://doi.org/10.1016/j.cosrev.2018.11.002

[102] Minho Ha and Sunggu Lee. 2018. Multipliers With Approximate 4–2 Compressors and Error Recovery Modules. *IEEE Embedded Systems Letters* 10, 1 (2018), 6–9. https://doi.org/10.1109/LES.2017.2746084

[103] Fumio Hamanaka, Takashi Odan, Kenji Kise, and Thiem Van Chu. 2023. An Exploration of State-of-the-Art Automation Frameworks for FPGA-Based DNN Acceleration. *IEEE Access* 11 (2023), 5701–5713. https://doi.org/10.1109/ACCESS.2023.3236974

[104] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A. Horowitz, and William J. Dally. 2016. EIE: Efficient Inference Engine on Compressed Deep Neural Network. In *Proceedings of the 43rd International Symposium on Computer Architecture (ISCA'16)*. 243–254. https://doi.org/10.1109/ISCA.2016.30

[105] Song Han, Huizi Mao, and William J. Dally. 2015. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *CoRR* abs/1510.00149 (2015). http://arxiv.org/abs/1510.00149

[106] Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both Weights and Connections for Efficient Neural Network. In *Advances in Neural Information Processing Systems*, Vol. 28. Curran Associates, Inc.

[107] Mark Harris. 2016. *Inside Pascal: NVIDIA's Newest Computing Platform.* Retrieved Apr 16, 2023 from https://developer.nvidia.com/blog/inside-pascal/

[108] Soheil Hashemi, R. Iris Bahar, and Sherief Reda. 2015. DRUM: A Dynamic Range Unbiased Multiplier for approximate applications. In *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 418–425. https://doi.org/10.1109/ICCAD.2015.7372600

[109] Mehdi Hassanpour, Marc Riera, and Antonio González. 2022. A Survey of Near-Data Processing Architectures for Neural Networks. *Machine Learning and Knowledge Extraction* 4, 1 (2022), 66–102. https://doi.org/10.3390/make4010004

[110] Mingxuan He, Choungki Song, Ilkon Kim, Chunseok Jeong, Seho Kim, Il Park, Mithuna Thottethodi, and T. N. Vijaykumar. 2020. Newton: A DRAM-maker's Accelerator-in-Memory (AiM) Architecture for Machine Learning. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 372–385. https://doi.org/10.1109/MICRO50266.2020.00040

[111] Kartik Hegde, Jiyong Yu, Rohit Agrawal, Mengjia Yan, Michael Pellauer, and Christopher W. Fletcher. 2018. UCNN: Exploiting Computational Reuse in Deep Neural Networks via Weight Repetition. In *Proceedings of the 45th Annual International Symposium on Computer Architecture* (Los Angeles, California) *(ISCA'18)*. 674–687. https://doi.org/10.1109/ISCA.2018.00062

[112] Karen Heyman. 2022. DRAM Thermal Issues Reach Crisis Point. https://semiengineering.com/dram-thermal-issues-reach-crisis-point/. Accessed: 18/04/2023.

[113] Pouya Houshmand, Giuseppe M. Sarda, Vikram Jain, Kodai Ueyoshi, Ioannis A. Papistas, Man Shi, Qilin Zheng, Debjyoti Bhattacharjee, Arindam Mallik, Peter Debacker, Diederik Verkest, and Marian Verhelst. 2023. DIANA: An End-to-End Hybrid DIgital and ANAlog Neural Network SoC for the Edge. *IEEE Journal of Solid-State Circuits* 58, 1 (Jan. 2023), 203–215. https://doi.org/10.1109/JSSC.2022.3214064

[114] HPCWIRE 2022. Quantum computers emerging as accelerators in HPC. https://www.hpcwire.com/2022/06/07/quantum-computers-emerging-as-accelerators-in-hpc/

[115] Xing Hu, Dylan Stow, and Yuan Xie. 2018. Die Stacking Is Happening. *IEEE Micro* 38, 1 (2018), 22–28. https://doi.org/10.1109/MM.2018.011441561

[116] Je-Min Hung, Chuan-Jia Jhang, Ping-Chun Wu, Yen-Cheng Chiu, and Meng-Fan Chang. 2021. Challenges and Trends of Nonvolatile In-Memory-Computation Circuits for AI Edge Devices. *IEEE Open Journal of the Solid-State Circuits Society* 1 (2021), 171–183. https://doi.org/10.1109/OJSSCS.2021.3123287

[117] R. Hwang, T. Kim, Y. Kwon, and M. Rhu. 2020. Centaur: A Chiplet-based, Hybrid Sparse-Dense Accelerator for Personalized Recommendations. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE Computer Society, Los Alamitos, CA, USA, 968–981. https://doi.org/10.1109/ISCA45697.2020.00083

[118] Hybrid Memory Cube Consortium. 2014. Hybrid Memory Cube specification 2.1.

[119] Daniele Ielmini and H.-S. Philip Wong. 2018. In-memory computing with resistive switching devices. *Nature Electronics* 1, 6 (Jun 2018), 333–343. https://doi.org/10.1038/s41928-018-0092-2

[120] Intel. 2022. *Intel Arc A770 Graphics 16GB.* Retrieved May 25, 2023 from https://ark.intel.com/content/www/us/en/ark/products/229151/intel-arc-a770-graphics-16gb.html

[121] Intel. 2022. Intel High Level Synthesis Compiler Reference Manual. https://www.intel.com/content/www/us/en/docs/programmable/683349/21-4/pro-edition-reference-manual.html

[122] Vikram Jain, Sebastian Giraldo, Jaro De Roose, Linyan Mei, Bert Boons, and Marian Verhelst. 2023. TinyVers: A Tiny Versatile System-on-Chip With State-Retentive eMRAM for ML Inference at the Extreme Edge. *IEEE Journal of Solid-State Circuits* (2023), 1–12. https://doi.org/10.1109/JSSC.2023.3236566

[123] Joe Jeddeloh and Brent Keeth. 2012. Hybrid memory cube new DRAM architecture increases density and performance. In *2012 Symposium on VLSI Technology (VLSIT)*. 87–88. https://doi.org/10.1109/VLSIT.2012.6242474

[124] JEDEC Solid State Technology Association. 2023. High Bandwidth Memory DRAM (HBM3), JESD238A.

[125] Tianyu Jia, Paolo Mantovani, Maico Cassel Dos Santos, Davide Giri, Joseph Zuckerman, Erik Jens Loscalzo, Martin Cochet, Karthik Swaminathan, Gabriele Tombesi, Jeff Jun Zhang, Nandhini Chandramoorthy, John-David Wellman, Kevin Tien, Luca Carloni, Kenneth Shepard, David Brooks, Gu-Yeon Wei, and Pradip Bose. 2022. A 12nm Agile-Designed SoC for Swarm-Based Perception with Heterogeneous IP Blocks, a Reconfigurable Memory Hierarchy, and an 800MHz Multi-Plane NoC. In *ESSCIRC 2022- IEEE 48th European Solid State Circuits Conference (ESSCIRC)*. 269–272. https://doi.org/10.1109/ESSCIRC55480.2022.9911456

[126] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093* (2014).

[127] Zhe Jia, Blake Tillman, Marco Maggioni, and Daniele Paolo Scarpazza. 2019. Dissecting the Graphcore IPU Architecture via Microbenchmarking. https://doi.org/10.48550/arXiv.1912.03413 arXiv:1912.03413 [cs]

[128] Qiang Jiao, Wei Hu, Fang Liu, and Yong Dong. 2021. RISC-VTF: RISC-V Based Extended Instruction Set for Transformer. In *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 1565–1570. https://doi.org/10.1109/SMC52423.2021.9658643

[129] Yang Jiao, Liang Han, Rong Jin, Yi-Jung Su, Chiente Ho, Li Yin, Yun Li, Long Chen, Zhen Chen, Lu Liu, Zhuyu He, Yu Yan, Jun He, Jun Mao, Xiaotao Zai, Xuejun Wu, Yongquan Zhou, Mingqiu Gu, Guocai Zhu, Rong Zhong, Wenyuan Lee, Ping Chen, Yiping Chen, Weiliang Li, Deyu Xiao, Qing Yan, Mingyuan Zhuang, Jiejun Chen, Yun Tian, Yingzi Lin, Wei Wu, Hao Li, and Zesheng Dou. 2020. A 12nm Programmable Convolution-Efficient Neural-Processing-Unit Chip Achieving 825TOPS. In *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*. 136–140. https://doi.org/10.1109/ISSCC19947.2020.9062984

[130] Qing Jin, Jian Ren, Richard Zhuang, Sumant Hanumante, Zhengang Li, Zhiyu Chen, Yanzhi Wang, Kaiyuan Yang, and Sergey Tulyakov. 2022. F8Net: Fixed-Point 8-bit Only Multiplication for Network Quantization. In *International Conference on Learning Representations*. https://openreview.net/forum?id=_CfpJazzXT2

[131] Shalf John. 2020. The future of computing beyond Moore's Law. *Phil. Trans. R. Soc.* (2020). https://doi.org/10.1098/rsta.2019.0061

[132] Jer Min Jou, Shiann Rong Kuang, and Ren Der Chen. 1999. Design of low-error fixed-width multipliers for DSP applications. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* 46, 6 (1999), 836–842. https://doi.org/10.1109/82.769795

[133] Norman Jouppi, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Cliff Young, Tara Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Ho, Doug Hogberg, John Hu, and Nan Boden. 2017. In-Datacenter Performance Analysis of a Tensor Processing Unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA '17)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3079856.3080246

[134] Norman Jouppi, Cliff Young, Nishant Patil, and David Patterson. 2018. Motivation for and Evaluation of the First Tensor Processing Unit. *IEEE Micro* 38, 3 (May 2018), 10–19. https://doi.org/10.1109/MM.2018.032271057

[135] Norman P. Jouppi, Doe Hyun Yoon, George Kurian, Sheng Li, Nishant Patil, James Laudon, Cliff Young, and David Patterson. 2020. A Domain-Specific Supercomputer for Training Deep Neural Networks. *Commun. ACM* 63, 7 (June 2020), 67–78. https://doi.org/10.1145/3360307

[136] Yuhao Ju and Jie Gu. 2023. A Systolic Neural CPU Processor Combining Deep Learning and General-Purpose Computing With Enhanced Data Locality and End-to-End Performance. *IEEE Journal of Solid-State Circuits* 58, 1 (Jan. 2023), 216–226. https://doi.org/10.1109/JSSC.2022.3214170

[137] Morihiro Kada. 2015. *Research and Development History of Three-Dimensional Integration Technology*. Springer International Publishing, Cham, 1–23. https://doi.org/10.1007/978-3-319-18675-7_1

[138] Ajaykumar Kannan, Natalie Enright Jerger, and Gabriel H. Loh. 2015. Enabling interposer-based disintegration of multi-core processors. In *2015 48th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 546–558.

[139] Riduan Khaddam-Aljameh, Milos Stanisavljevic, Jordi Fornt Mas, Geethan Karunaratne, Matthias Brändli, Feng Liu, Abhairaj Singh, Silvia M Müller, Urs Egger, Anastasios Petropoulos, et al. 2022. HERMES-core—A 1.59-TOPS/mm 2 PCM on 14-nm CMOS in-memory compute core using 300-ps/LSB linearized CCO-based ADCs. *IEEE Journal of*

Solid-State Circuits 57, 4 (2022), 1027–1038.

[140] Mahmoud Khairy, Amr G. Wassal, and Mohamed Zahran. 2019. A survey of architectural approaches for improving GPGPU performance, programmability and heterogeneity. *J. Parallel and Distrib. Comput.* 127 (2019), 65–88. https://doi.org/10.1016/j.jpdc.2018.11.012

[141] Duckhwan Kim, Jaeha Kung, Sek Chai, Sudhakar Yalamanchili, and Saibal Mukhopadhyay. 2016. Neurocube: A Programmable Digital Neuromorphic Architecture with High-Density 3D Memory. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. 380–392. https://doi.org/10.1109/ISCA.2016.41

[142] Donghyuk Kim, Chengshuo Yu, Shanshan Xie, Yuzong Chen, Joo-Young Kim, Bongjin Kim, Jaydeep P. Kulkarni, and Tony Tae-Hyoung Kim. 2022. An Overview of Processing-in-Memory Circuits for Artificial Intelligence and Machine Learning. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 12, 2 (2022), 338–353. https://doi.org/10.1109/JETCAS.2022.3160455

[143] Joonyoung Kim and Younsu Kim. 2014. HBM: Memory solution for bandwidth-hungry processors. In *2014 IEEE Hot Chips 26 Symposium (HCS)*. 1–24. https://doi.org/10.1109/HOTCHIPS.2014.7478812

[144] Sangyeob Kim, Sangjin Kim, Soyeon Um, Soyeon Kim, Kwantae Kim, and Hoi-Jun Yoo. 2022. Neuro-CIM: A 310.4 TOPS/W Neuromorphic Computing-in-Memory Processor with Low WL/BL activity and Digital-Analog Mixed-mode Neuron Firing. In *2022 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*. 38–39. https://doi.org/10.1109/VLSITechnologyandCir46769.2022.9830276

[145] Johann Knechtel, Ozgur Sinanoglu, Ibrahim (Abe) M. Elfadel, Jens Lienig, and Cliff C. N. Sze. 2017. Large-Scale 3D Chips: Challenges and Solutions for Design Automation, Testing, and Trustworthy Integration. *IPSJ Transactions on System and LSI Design Methodology* 10 (2017), 45–62. https://doi.org/10.2197/ipsjtsldm.10.45

[146] James C. Knight and Thomas Nowotny. 2018. GPUs Outperform Current HPC and Neuromorphic Solutions in Terms of Speed and Energy When Simulating a Highly-Connected Cortical Model. *Frontiers in Neuroscience* 12 (2018). https://doi.org/10.3389/fnins.2018.00941

[147] Simon Knowles. 2021. Graphcore. In *2021 IEEE Hot Chips 33 Symposium (HCS)*. 1–25. https://doi.org/10.1109/HCS52781.2021.9567075

[148] Tianqi Kong and Shuguo Li. 2021. Design and Analysis of Approximate 4–2 Compressors for High-Accuracy Multipliers. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 29, 10 (2021), 1771–1781. https://doi.org/10.1109/TVLSI.2021.3104145

[149] Ronny Krashinsky, Olivier Giroux, Stephen Jones, Nick Stam, and Sridhar Ramaswamy. 2020. *NVIDIA Ampere Architecture In-Depth*. Retrieved Apr 16, 2023 from https://developer.nvidia.com/blog/nvidia-ampere-architecture-in-depth/

[150] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 1097–1105. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[151] Parag Kulkarni, Puneet Gupta, and Milos Ercegovac. 2011. Trading Accuracy for Power with an Underdesigned Multiplier Architecture. In *2011 24th Internatioal Conference on VLSI Design*. 346–351. https://doi.org/10.1109/VLSID.2011.51

[152] Youngsu Kwon, Jinho Han, Yongcheol Peter Cho, Juyeob Kim, Jaehoon Chung, Jaewoong Choi, Sujin Park, Igyeong Kim, Hyunjeong Kwon, Jinkyu Kim, Hyunmi Kim, Won Jeon, Youngdeuk Jeon, Minhyung Cho, and Minseok Choi. 2023. Chiplet Heterogeneous-Integration AI Processor. In *2023 International Conference on Electronics, Information, and Communication (ICEIC)*.

[153] Young-Cheon Kwon, Suk Han Lee, Jaehoon Lee, Sang-Hyuk Kwon, Je Min Ryu, Jong-Pil Son, O Seongil, Hak-Soo Yu, Haesuk Lee, Soo Young Kim, Youngmin Cho, Jin Guk Kim, Jongyoon Choi, Hyun-Sung Shin, Jin Kim, BengSeng Phuah, HyoungMin Kim, Myeong Jun Song, Ahn Choi, Daeho Kim, SooYoung Kim, Eun-Bong Kim, David Wang, Shinhaeng Kang, Yuhwan Ro, Seungwoo Seo, JoonHo Song, Jaeyoun Youn, Kyomin Sohn, and Nam Sung Kim. 2021. 25.4 A 20nm 6GB Function-In-Memory DRAM, Based on HBM2 with a 1.2TFLOPS Programmable Computing Unit Using Bank-Level Parallelism, for Machine Learning Applications. In *2021 IEEE International Solid- State Circuits Conference (ISSCC)*, Vol. 64. 350–352. https://doi.org/10.1109/ISSCC42613.2021.9365862

[154] Jingjing Lan, Vishnu P. Nambiar, Rheeshaalaen Sabapathy, Mihai Dragos Rotaru, and Anh Tuan Do. 2021. Chiplet-based Architecture Design for Multi-Core Neuromorphic Processor. In *2021 IEEE 23rd Electronics Packaging Technology Conference (EPTC)*. 410–412. https://doi.org/10.1109/EPTC53413.2021.9663898

[155] Chris Lattner, Mehdi Amini, Uday Bondhugula, Albert Cohen, Andy Davis, Jacques Pienaar, River Riddle, Tatiana Shpeisman, Nicolas Vasilache, and Oleksandr Zinenko. 2021. MLIR: Scaling Compiler Infrastructure for Domain Specific Computation. In *IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*. 2–14.

[156] Cristóbal Ramírez Lazo, Enrico Reggiani, Carlos Rojas Morales, Roger Figueras Bagué, Luis A. Villa Vargas, Marco A. Ramírez Salinas, Mateo Valero Cortés, Osman Sabri Ünsal, and Adrián Cristal. 2022. Adaptable Register File

Organization for Vector Processors. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. 786–799. https://doi.org/10.1109/HPCA53966.2022.00063

[157] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436.

[158] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324. https://doi.org/10.1109/5.726791

[159] Sae Kyu Lee, Ankur Agrawal, Joel Silberman, Matthew Ziegler, Mingu Kang, Swagath Venkataramani, Nianzheng Cao, Bruce Fleischer, Michael Guillorn, Matthew Cohen, Silvia M. Mueller, Jinwook Oh, Martin Lutz, Jinwook Jung, Siyu Koswatta, Ching Zhou, Vidhi Zalani, Monodeep Kar, James Bonanno, Robert Casatuta, Chia-Yu Chen, Jungwook Choi, Howard Haynie, Alyssa Herbert, Radhika Jain, Kyu-Hyoun Kim, Yulong Li, Zhibin Ren, Scot Rider, Marcel Schaal, Kerstin Schelm, Michael R. Scheuermann, Xiao Sun, Hung Tran, Naigang Wang, Wei Wang, Xin Zhang, Vinay Shah, Brian Curran, Vijayalakshmi Srinivasan, Pong-Fei Lu, Sunil Shukla, Kailash Gopalakrishnan, and Leland Chang. 2022. A 7-nm Four-Core Mixed-Precision AI Chip With 26.2-TFLOPS Hybrid-FP8 Training, 104.9-TOPS INT4 Inference, and Workload-Aware Throttling. *IEEE Journal of Solid-State Circuits* 57, 1 (2022), 182–197. https://doi.org/10.1109/JSSC.2021.3120113

[160] Sang Min Lee, Hanjoon Kim, Jeseung Yeon, Juyun Lee, Younggeun Choi, Minho Kim, Changjae Park, Kiseok Jang, Youngsik Kim, Yongseung Kim, Changman Lee, Hyuck Han, Won Eung Kim, Rui Tang, and Joon Ho Baek. 2022. A 64-TOPS Energy-Efficient Tensor Accelerator in 14nm With Reconfigurable Fetch Network and Processing Fusion for Maximal Data Reuse. *IEEE Open Journal of the Solid-State Circuits Society* 2 (2022), 219–230. https://doi.org/10.1109/OJSSCS.2022.3216798

[161] Yunsup Lee, Andrew Waterman, Rimas Avizienis, Henry Cook, Chen Sun, Vladimir Stojanović, and Krste Asanović. 2014. A 45nm 1.3 GHz 16.7 double-precision GFLOPS/W RISC-V processor with vector accelerators. In *ESSCIRC 2014-40th European Solid State Circuits Conference (ESSCIRC)*. IEEE, 199–202.

[162] N. Lepri, A. Glukhov, L. Cattaneo, M. Farronato, P. Mannocci, and D. Ielmini. 2023. In-memory computing for machine learning and deep learning. *IEEE Journal of the Electron Devices Society* (2023), 1–1. https://doi.org/10.1109/JEDS.2023.3265875

[163] Gang Li, Zejian Liu, Fanrong Li, and Jian Cheng. 2021. Block Convolution: Towards Memory-Efficient Inference of Large-Scale CNNs on FPGA. *CoRR* abs/2105.08937 (2021). arXiv:2105.08937 https://arxiv.org/abs/2105.08937

[164] Jiajun Li, Shuhao Jiang, Shijun Gong, Jingya Wu, Junchao Yan, Guihai Yan, and Xiaowei Li. 2019. SqueezeFlow: A Sparse CNN Accelerator Exploiting Concise Convolution Rules. *IEEE Trans. Comput.* 68, 11 (2019), 1663–1677. https://doi.org/10.1109/TC.2019.2924215

[165] Ling Li, Issam Hammad, and Kamal El-Sankary. 2021. Dual segmentation approximate multiplier. *Electronics Letters* 57, 19 (2021), 718–720. https://doi.org/10.1049/ell2.12243 arXiv:https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/ell2.12243

[166] Yuan Li, Ahmed Louri, and Avinash Karanth. 2021. SPRINT: A high-performance, energy-efficient, and scalable chiplet-based accelerator with photonic interconnects for CNN inference. *IEEE Transactions on Parallel and Distributed Systems* 33, 10 (2021), 2332–2345.

[167] Chien-Hung Lin, Chih-Chung Cheng, Yi-Min Tsai, Sheng-Je Hung, Yu-Ting Kuo, Perry H Wang, Pei-Kuei Tsung, Jeng-Yun Hsu, Wei-Chih Lai, Chia-Hung Liu, Shao-Yu Wang, Chin-Hua Kuo, Chih-Yu Chang, Ming-Hsien Lee, Tsung-Yao Lin, and Chih-Cheng Chen. 2020. A 3.4-to-13.3TOPS/W 3.6TOPS Dual-Core Deep-Learning Accelerator for Versatile AI Applications in 7nm 5G Smartphone SoC. In *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*. 134–136. https://doi.org/10.1109/ISSCC19947.2020.9063111

[168] Mu-Shan Lin, Tze-Chiang Huang, Chien-Chun Tsai, King-Ho Tam, Cheng-Hsiang Hsieh, Tom Chen, Wen-Hung Huang, Jack Hu, Yu-Chi Chen, Sandeep Kumar Goel, Chin-Ming Fu, Stefan Rusu, Chao-Chieh Li, Sheng-Yao Yang, Mei Wong, Shu-Chun Yang, and Frank Lee. 2019. A 7nm 4GHz Arm-core-based CoWoS Chiplet Design for High Performance Computing. In *2019 Symposium on VLSI Circuits*.

[169] Chen Liu, Guillaume Bellec, Bernhard Vogginger, David Kappel, Johannes Partzsch, Felix Neumärker, Sebastian Höppner, Wolfgang Maass, Steve B. Furber, Robert Legenstein, and Christian G. Mayr. 2018. Memory-Efficient Deep Learning on a SpiNNaker 2 Prototype. *Frontiers in Neuroscience* 12 (2018). https://doi.org/10.3389/fnins.2018.00840

[170] L. Liu, J. Zhu, Z. Li, Y. Lu, Y. Deng, J. Han, S. Yin, and S. Wei. 2019. A Survey of Coarse-Grained Reconfigurable Architecture and Design: Taxonomy, Challenges, and Applications. *ACM Comput. Surv.* 52, 6 (2019). https://doi.org/10.1145/3357375

[171] Xiaoxiao Liu, Mengjie Mao, Beiye Liu, Hai Li, Yiran Chen, Boxun Li, Yu Wang, Hao Jiang, Mark Barnell, Qing Wu, and Jianhua Yang. 2015. RENO: A high-efficient reconfigurable neuromorphic computing accelerator design. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. 1–6. https://doi.org/10.1145/2744769.2744900

[172] Zechun Liu, Kwang-Ting Cheng, Dong Huang, Eric Xing, and Zhiqiang Shen. 2022. Nonuniform-to-Uniform Quantization: Towards Accurate Quantization via Generalized Straight-Through Estimation. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 4932–4942. https://doi.org/10.1109/CVPR52688.2022.

00489

[173] Yufei Ma, Yu Cao, Sarma Vrudhula, and Jae-sun Seo. 2018. Optimizing the Convolution Operation to Accelerate Deep Neural Networks on FPGA. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 26, 7 (2018), 1354–1367. https://doi.org/10.1109/TVLSI.2018.2815603

[174] Yufei Ma, Naveen Suda, Yu Cao, Jae-sun Seo, and Sarma Vrudhula. 2016. Scalable and modularized RTL compilation of Convolutional Neural Networks onto FPGA. In *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*. 1–8. https://doi.org/10.1109/FPL.2016.7577356

[175] Raju Machupalli, Masum Hossain, and Mrinal Mandal. 2022. Review of ASIC accelerators for deep neural network. *Microprocessors and Microsystems* 89 (2022), 104441. https://doi.org/10.1016/j.micpro.2022.104441

[176] Michal Machura, Michal Danilowicz, and Tomasz Kryjak. 2022. Embedded Object Detection with Custom LittleNet, FINN and Vitis AI DCNN Accelerators. *Journal of Low Power Electronics and Applications* 12, 2 (2022). https://doi.org/10.3390/jlpea12020030

[177] David Mallasén, Raul Murillo, Alberto A. Del Barrio, Guillermo Botella, Luis Piñuel, and Manuel Prieto-Matias. 2022. PERCIVAL: Open-Source Posit RISC-V Core With Quire Capability. *IEEE Transactions on Emerging Topics in Computing* 10, 3 (2022), 1241–1252. https://doi.org/10.1109/TETC.2022.3187199

[178] P. Y. Martinez, Y. Beilliard, M. Godard, D. Danovitch, D. Drouin, J. Charbonnier, P. Coudrain, A. Garnier, D. Lattard, P. Vivet, S. Cheramy, E. Guthmuller, C. Fuguet Tortolero, V. Mengue, J. Durupt, A. Philippe, and D. Dutoit. 2020. ExaNoDe: Combined Integration of Chiplets on Active Interposer with Bare Dice in a Multi-Chip-Module for Heterogeneous and Scalable High Performance Compute Nodes. In *2020 IEEE Symposium on VLSI Technology*.

[179] Rahul Mathur, Ajay Krishna Ananda Kumar, Lizy John, and Jaydeep P. Kulkarni. 2021. Thermal-Aware Design Space Exploration of 3-D Systolic ML Accelerators. *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits* 7, 1 (2021), 70–78. https://doi.org/10.1109/JXCDC.2021.3092436

[180] Peter Mattson, Christine Cheng, Gregory Diamos, Cody Coleman, Paulius Micikevicius, David Patterson, Hanlin Tang, Gu-Yeon Wei, Peter Bailis, Victor Bittorf, et al. 2020. Mlperf training benchmark. *Proceedings of Machine Learning and Systems* 2 (2020), 336–349.

[181] Peter Mattson, Vijay Janapa Reddi, Christine Cheng, Cody Coleman, Greg Diamos, David Kanter, Paulius Micikevicius, David Patterson, Guenther Schmuelling, Hanlin Tang, Gu-Yeon Wei, and Carole-Jean Wu. 2020. MLPerf: An Industry Standard Benchmark Suite for Machine Learning Performance. *IEEE Micro* 40, 2 (2020), 8–16. https://doi.org/10.1109/MM.2020.2974843

[182] Eitan Medina. 2019. [Habana Labs presentation]. In *2019 IEEE Hot Chips 31 Symposium (HCS)*. 1–29. https://doi.org/10.1109/HOTCHIPS.2019.8875670

[183] Micron. 2018. Hybrid Memory Cube – HMC Gen2 HMC Memory Features.

[184] Chuhan Min, Jiachen Mao, Hai Li, and Yiran Chen. 2019. NeuralHMC: An Efficient HMC-Based Accelerator for Deep Neural Networks. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference* (Tokyo, Japan) *(ASPDAC '19)*. Association for Computing Machinery, New York, NY, USA, 394–399. https://doi.org/10.1145/3287624.3287642

[185] Francesco Minervini, Oscar Palomar, Osman Unsal, Enrico Reggiani, Josue Quiroga, Joan Marimon, Carlos Rojas, Roger Figueras, Abraham Ruiz, Alberto Gonzalez, Jonnatan Mendoza, Ivan Vargas, César Hernandez, Joan Cabre, Lina Khoirunisya, Mustapha Bouhali, Julian Pavon, Francesc Moll, Mauro Olivieri, Mario Kovac, Mate Kovac, Leon Dragic, Mateo Valero, and Adrian Cristal. 2023. Vitruvius+: An Area-Efficient RISC-V Decoupled Vector Coprocessor for High Performance Computing Applications. *ACM Transactions on Architecture and Code Optimization* 20, 2 (March 2023), 28:1–28:25. https://doi.org/10.1145/3575861

[186] Ivan Miro-Panades, Benoit Tain, Jean-Frédéric Christmann, David Coriat, Romain Lemaire, Clément Jany, Baudouin Martineau, Fabrice Chaix, Guillaume Waltener, Emmanuel Pluchart, Jean-Philippe Noel, Adam Makosiej, Maxime Montoya, Simone Bacles-Min, David Briand, Jean-Marc Philippe, Yvain Thonnart, Alexandre Valentian, Frédéric Heitzmann, and Fabien Clermidy. 2022. SamurAI: A Versatile IoT Node With Event-Driven Wake-Up and Embedded ML Acceleration. *IEEE Journal of Solid-State Circuits* (2022), 1–0. https://doi.org/10.1109/JSSC.2022.3198505

[187] Asit K. Mishra, Jorge Albericio Latorre, Jeff Pool, Darko Stosic, Dusan Stosic, Ganesh Venkatesh, Chong Yu, and Paulius Micikevicius. 2021. Accelerating Sparse Deep Neural Networks. *CoRR* abs/2104.08378 (2021). https://arxiv.org/abs/2104.08378

[188] Gabriel Mounce, Jim Lyke, Stephen Horan, Wes Powell, Rich Doyle, and Rafi Some. 2016. Chiplet based approach for heterogeneous processing and packaging architectures. In *2016 IEEE Aerospace Conference*. 1–12.

[189] Francisco Muñoz Martínez, Raveesh Garg, Michael Pellauer, José L. Abellán, Manuel E. Acacio, and Tushar Krishna. 2023. Flexagon: A Multi-Dataflow Sparse-Sparse Matrix Multiplication Accelerator for Efficient DNN Processing. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3* (Vancouver, BC, Canada) *(ASPLOS 2023)*. 252–265. https://doi.org/10.1145/3582016.3582069

[190] Mitchell A. Nahmias, Thomas Ferreira de Lima, Alexander N. Tait, Hsuan-Tung Peng, Bhavin J. Shastri, and Paul R. Prucnal. 2020. Photonic Multiply-Accumulate Operations for Neural Networks. *IEEE Journal of Selected Topics in Quantum Electronics* 26, 1 (2020), 1–18. https://doi.org/10.1109/JSTQE.2019.2941485

[191] Srinivasan Narayanamoorthy, Hadi Asghari Moghaddam, Zhenhong Liu, Taejoon Park, and Nam Sung Kim. 2015. Energy-Efficient Approximate Multiplication for Digital Signal Processing and Classification Applications. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 23, 6 (2015), 1180–1184. https://doi.org/10.1109/TVLSI.2014.2333366

[192] P. Narayanan, S. Ambrogio, A. Okazaki, K. Hosokawa, H. Tsai, A. Nomura, T. Yasuda, C. Mackin, S. C. Lewis, A. Friz, M. Ishii, Y. Kohda, H. Mori, K. Spoon, R. Khaddam-Aljameh, N. Saulnier, M. Bergendahl, J. Demarest, K. W. Brew, V. Chan, S. Choi, I. Ok, I. Ahsan, F. L. Lie, W. Haensch, V. Narayanan, and G. W. Burr. 2021. Fully On-Chip MAC at 14 nm Enabled by Accurate Row-Wise Programming of PCM-Based Weights and Parallel Vector-Transport in Duration-Format. *IEEE Transactions on Electron Devices* 68, 12 (2021), 6629–6636. https://doi.org/10.1109/TED.2021.3115993

[193] . Nitin, Mithuna Thottethodi, and T. N. Vijaykumar. 2018. Millipede: Die-Stacked Memory Optimizations for Big Data Machine Learning Analytics. In *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 160–171. https://doi.org/10.1109/IPDPS.2018.00026

[194] Eriko Nurvitadhi, Dongup Kwon, Ali Jafari, Andrew Boutros, Jaewoong Sim, Phillip Tomson, Huseyin Sumbul, Gregory Chen, Phil Knag, Raghavan Kumar, Ram Krishnamurthy, Sergey Gribok, Bogdan Pasca, Martin Langhammer, Debbie Marr, and Aravind Dasu. 2019. Why Compete When You Can Work Together: FPGA-ASIC Integration for Persistent RNNs. In *2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. 199–207.

[195] NVidia. 2022. *NVIDIA Announces DGX H100 Systems – World's Most Advanced Enterprise AI Infrastructure.* Retrieved May 11, 2023 from https://nvidianews.nvidia.com/news/nvidia-announces-dgx-h100-systems-worlds-most-advanced-enterprise-ai-infrastructure

[196] NVidia. 2023. *NVIDIA DGX Platform The best of NVIDIA AI—all in one place.* Retrieved May 11, 2023 from https://www.nvidia.com/en-us/data-center/dgx-platform/

[197] NVidia. 2023. *NVLink and NVSwitch.* Retrieved May 11, 2023 from https://www.nvidia.com/en-us/data-center/nvlink/

[198] Jinwook Oh, Sae Kyu Lee, Mingu Kang, Matthew Ziegler, Joel Silberman, Ankur Agrawal, Swagath Venkataramani, Bruce Fleischer, Michael Guillorn, Jungwook Choi, Wei Wang, Silvia Mueller, Shimon Ben-Yehuda, James Bonanno, Nianzheng Cao, Robert Casatuta, Chia-Yu Chen, Matt Cohen, Ophir Erez, Thomas Fox, George Gristede, Howard Haynie, Vicktoria Ivanov, Siyu Koswatta, Shih-Hsien Lo, Martin Lutz, Gary Maier, Alex Mesh, Yevgeny Nustov, Scot Rider, Marcel Schaal, Michael Scheuermann, Xiao Sun, Naigang Wang, Fanchieh Yee, Ching Zhou, Vinay Shah, Brian Curran, Vijayalakshmi Srinivasan, Pong-Fei Lu, Sunil Shukla, Kailash Gopalakrishnan, and Leland Chang. 2020. A 3.0 TFLOPS 0.62V Scalable Processor Core for High Compute Utilization AI Training and Inference. In *2020 IEEE Symposium on VLSI Circuits*. 1–2. https://doi.org/10.1109/VLSICircuits18222.2020.9162917

[199] Geraldo F. Oliveira, Paulo C. Santos, Marco A. Z. Alves, and Luigi Carro. 2017. NIM: An HMC-Based Machine for Neuron Computation. In *Applied Reconfigurable Computing*, Stephan Wong, Antonio Carlos Beck, Koen Bertels, and Luigi Carro (Eds.). Springer International Publishing, Cham, 28–35.

[200] Gianmarco Ottavi, Angelo Garofalo, Giuseppe Tagliavini, Francesco Conti, Alfio Di Mauro, Luca Benini, and Davide Rossi. 2023. Dustin: A 16-Cores Parallel Ultra-Low-Power Cluster With 2b-to-32b Fully Flexible Bit-Precision and Vector Lockstep Execution Mode. *IEEE Transactions on Circuits and Systems I: Regular Papers* (2023), 1–14. https://doi.org/10.1109/TCSI.2023.3254810

[201] Eustace Painkras, Luis A. Plana, Jim Garside, Steve Temple, Francesco Galluppi, Cameron Patterson, David R. Lester, Andrew D. Brown, and Steve B. Furber. 2013. SpiNNaker: A 1-W 18-Core System-on-Chip for Massively-Parallel Neural Network Simulation. *IEEE Journal of Solid-State Circuits* 48, 8 (2013), 1943–1953. https://doi.org/10.1109/JSSC.2013.2259038

[202] Angshuman Parashar, Minsoo Rhu, Anurag Mukkara, Antonio Puglielli, Rangharajan Venkatesan, Brucek Khailany, Joel Emer, Stephen W. Keckler, and William J. Dally. 2017. SCNN: An Accelerator for Compressed-Sparse Convolutional Neural Networks. In *Proceedings of the 2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA'17)*. 27–40. https://doi.org/10.1145/3079856.3080254

[203] Gunho Park, Jaeha Kung, and Youngjoo Lee. 2021. Design and Analysis of Approximate Compressors for Balanced Error Accumulation in MAC Operator. *IEEE Transactions on Circuits and Systems I: Regular Papers* 68, 7 (2021), 2950–2961. https://doi.org/10.1109/TCSI.2021.3073177

[204] Jongsoo Park, Sheng R. Li, Wei Wen, Hai Li, Yiran Chen, and Pradeep Dubey. 2016. Holistic SparseCNN: Forging the Trident of Accuracy, Speed, and Size. *CoRR* abs/1608.01409 (2016). http://arxiv.org/abs/1608.01409

[205] Jongsoo Park, Maxim Naumov, Protonu Basu, Summer Deng, Aravind Kalaiah, Daya Shanker Khudia, James Law, Parth Malani, Andrey Malevich, Nadathur Satish, Juan Miguel Pino, Martin Schatz, Alexander Sidorov, Viswanath Sivakumar, Andrew Tulloch, Xiaodong Wang, Yiming Wu, Hector Yuen, Utku Diril, Dmytro Dzhulgakov, Kim M.

Hazelwood, Bill Jia, Yangqing Jia, Lin Qiao, Vijay Rao, Nadav Rotem, Sungjoo Yoo, and Mikhail Smelyanskiy. 2018. Deep Learning Inference in Facebook Data Centers: Characterization, Performance Optimizations and Hardware Implications. *CoRR* abs/1811.09886 (2018). http://arxiv.org/abs/1811.09886

[206] Jun-Seok Park, Jun-Woo Jang, Heonsoo Lee, Dongwoo Lee, Sehwan Lee, Hanwoong Jung, Seungwon Lee, Suknam Kwon, Kyungah Jeong, Joon-Ho Song, SukHwan Lim, and Inyup Kang. 2021. A 6K-MAC Feature-Map-Sparsity-Aware Neural Processing Unit in 5nm Flagship Mobile SoC. In *2021 IEEE International Solid- State Circuits Conference (ISSCC)*, Vol. 64. 152–154. https://doi.org/10.1109/ISSCC42613.2021.9365928

[207] Jun-Seok Park, Changsoo Park, Suknam Kwon, Hyeong-Seok Kim, Taeho Jeon, Yesung Kang, Heonsoo Lee, Dongwoo Lee, James Kim, YoungJong Lee, Sangkyu Park, Jun-Woo Jang, SangHyuck Ha, MinSeong Kim, Jihoon Bang, Suk Hwan Lim, and Inyup Kang. 2022. A Multi-Mode 8K-MAC HW-Utilization-Aware Neural Processing Unit with a Unified Multi-Precision Datapath in 4nm Flagship Mobile SoC. In *2022 IEEE International Solid- State Circuits Conference (ISSCC)*, Vol. 65. 246–248. https://doi.org/10.1109/ISSCC42614.2022.9731639

[208] Seong-Wook Park, Junyoung Park, Kyeongryeol Bong, Dongjoo Shin, Jinmook Lee, Sungpill Choi, and Hoi-Jun Yoo. 2015. An Energy-Efficient and Scalable Deep Learning/Inference Processor With Tetra-Parallel MIMD Architecture for Big Data Applications. *IEEE Transactions on Biomedical Circuits and Systems* 9, 6 (2015), 838–848. https://doi.org/10.1109/TBCAS.2015.2504563

[209] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[210] Gianna Paulin, Renzo Andri, Francesco Conti, and Luca Benini. 2021. RNN-Based Radio Resource Management on Multicore RISC-V Accelerator Architectures. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 29, 9 (Sept. 2021), 1624–1637. https://doi.org/10.1109/TVLSI.2021.3093242

[211] J. Thomas Pawlowski. 2011. Hybrid memory cube (HMC). In *2011 IEEE Hot Chips 23 Symposium (HCS)*. 1–24. https://doi.org/10.1109/HOTCHIPS.2011.7477494

[212] Maurice Peemen, Arnaud A. A. Setio, Bart Mesman, and Henk Corporaal. 2013. Memory-centric accelerator design for Convolutional Neural Networks. In *2013 IEEE 31st International Conference on Computer Design (ICCD)*. 13–19. https://doi.org/10.1109/ICCD.2013.6657019

[213] Matteo Perotti, Matheus Cavalcante, Nils Wistoff, Renzo Andri, Lukas Cavigelli, and Luca Benini. 2022. A "New Ara" for Vector Computing: An Open Source Highly Efficient RISC-V V 1.0 Vector Processor Design. In *2022 IEEE 33rd International Conference on Application-specific Systems, Architectures and Processors (ASAP)*. 43–51. https://doi.org/10.1109/ASAP54787.2022.00017

[214] Stefania Perri, Cristian Sestito, Fanny Spagnolo, and Pasquale Corsonello. 2020. Efficient Deconvolution Architecture for Heterogeneous Systems-on-Chip. *Journal of Imaging* 6, 9 (2020). https://doi.org/10.3390/jimaging6090085

[215] Nicola Petra, Davide De Caro, Valeria Garofalo, Ettore Napoli, and Antonio Giuseppe Maria Strollo. 2011. Design of Fixed-Width Multipliers With Linear Compensation Function. *IEEE Transactions on Circuits and Systems I: Regular Papers* 58, 5 (2011), 947–960. https://doi.org/10.1109/TCSI.2010.2090572

[216] Kartik Prabhu, Albert Gural, Zainab F Khan, Robert M Radway, Massimo Giordano, Kalhan Koul, Rohan Doshi, John W Kustin, Timothy Liu, Gregorio B Lopes, et al. 2022. CHIMERA: A 0.92-TOPS, 2.2-TOPS/W edge AI accelerator with 2-MByte on-chip foundry resistive RAM for efficient training and inference. *IEEE Journal of Solid-State Circuits* 57, 4 (2022), 1013–1026.

[217] Arpan Prasad, Luca Benini, and Francesco Conti. 2023. Specialization Meets Flexibility: A Heterogeneous Architecture for High-Efficiency, High-flexibility AR/VR Processing. In *Proceedings of the 2023 Design Automation Conference (DAC 2023), to Appear*.

[218] Morgan Prickett. 2022. The HBM3 roadmap is just getting started. https://www.nextplatform.com/2022/04/06/the-hbm3-roadmap-is-just-getting-started/. Accessed: 18/04/2023.

[219] Matt Proud. 2018. *ACHIEVING MAXIMUM COMPUTE THROUGHPUT: PCIE VS. SXM2.* Retrieved May 11, 2023 from https://www.nextplatform.com/micro-site-content/achieving-maximum-compute-throughput-pcie-vs-sxm2/

[220] Eric Qin, Ananda Samajdar, Hyoukjun Kwon, Vineet Nadella, Sudarshan Srinivasan, Dipankar Das, Bharat Kaul, and Tushar Krishna. 2020. SIGMA: A Sparse and Irregular GEMM Accelerator with Flexible Interconnects for DNN Training. In *Proceedings of the 2020 IEEE International Symposium on High Performance Computer Architecture (HPCA'20)*. 58–70. https://doi.org/10.1109/HPCA47549.2020.00015

[221] Haotong Qin, Ruihao Gong, Xianglong Liu, Xiao Bai, Jingkuan Song, and Nicu Sebe. 2020. Binary neural networks: A survey. *Pattern Recognition* 105 (2020), 107281. https://doi.org/10.1016/j.patcog.2020.107281

[222] QISKIT 2023. IBM Qiskit Simulator. https://qiskit.org/

[223] Jiantao Qiu, Jie Wang, Song Yao, Kaiyuan Guo, Boxun Li, Erjin Zhou, Jincheng Yu, Tianqi Tang, Ningyi Xu, Sen Song, Yu Wang, and Huazhong Yang. 2016. Going Deeper with Embedded FPGA Platform for Convolutional Neural Network. *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays* (2016).

[224] Atul Rahman, Sangyun Oh, Jongeun Lee, and Kiyoung Choi. 2017. Design space exploration of FPGA accelerators for convolutional neural networks. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. 1147–1152. https://doi.org/10.23919/DATE.2017.7927162

[225] Nitin Rathi, Indranil Chakraborty, Adarsh Kosta, Abhronil Sengupta, Aayush Ankit, Priyadarshini Panda, and Kaushik Roy. 2023. Exploring Neuromorphic Computing Based on Spiking Neural Networks: Algorithms to Hardware. *ACM Comput. Surv.* 55, 12 (2023). https://doi.org/10.1145/3571155

[226] Albert Reuther, Peter Michaleas, Michael Jones, Vijay Gadepally, Siddharth Samsi, and Jeremy Kepner. 2022. AI and ML Accelerator Survey and Trends. In *2022 IEEE High Performance Extreme Computing Conference (HPEC)*. 1–10. https://doi.org/10.1109/HPEC55821.2022.9926331

[227] Cliff Robinson. 2022. NVIDIA H100 Hopper Details at HC34 as it Waits for Next-Gen CPUs. https://www.servethehome.com/nvidia-h100-hopper-details-at-hc34-as-it-waits-for-next-gen-cpus/. Accessed: 18/04/2023.

[228] F. Rosenblatt. 1957. *The perceptron - A perceiving and recognizing automaton.* Technical Report 85-460-1. Cornell Aeronautical Laboratory, Ithaca, New York.

[229] Davide Rossi, Francesco Conti, Manuel Eggiman, Alfio Di Mauro, Giuseppe Tagliavini, Stefan Mach, Marco Guermandi, Antonio Pullini, Igor Loi, Jie Chen, Eric Flamand, and Luca Benini. 2022. Vega: A Ten-Core SoC for IoT Endnodes With DNN Acceleration and Cognitive Wake-Up From MRAM-Based State-Retentive Sleep Mode. *IEEE Journal of Solid-State Circuits* 57, 1 (Jan. 2022), 127–139. https://doi.org/10.1109/JSSC.2021.3114881

[230] Murugan Sankaradas, Venkata Jakkula, Srihari Cadambi, Srimat Chakradhar, Igor Durdanovic, Eric Cosatto, and Hans Peter Graf. 2009. A Massively Parallel Coprocessor for Convolutional Neural Networks. In *Proceedings of the 2009 20th IEEE International Conference on Application-Specific Systems, Architectures and Processors*. IEEE Computer Society, USA, 53–60. https://doi.org/10.1109/ASAP.2009.25

[231] Jürgen Schmidhuber. 2014. Deep Learning in Neural Networks: An Overview. *CoRR* abs/1404.7828 (2014). arXiv:1404.7828 http://arxiv.org/abs/1404.7828

[232] Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural Networks* 61 (2015), 85–117. https://doi.org/10.1016/j.neunet.2014.09.003

[233] Jae-sun Seo, Bernard Brezzo, Yong Liu, Benjamin D. Parker, Steven K. Esser, Robert K. Montoye, Bipin Rajendran, José A. Tierno, Leland Chang, Dharmendra S. Modha, and Daniel J. Friedman. 2011. A 45nm CMOS neuromorphic chip with a scalable architecture for learning in networks of spiking neurons. In *2011 IEEE Custom Integrated Circuits Conference (CICC)*. 1–4. https://doi.org/10.1109/CICC.2011.6055293

[234] Cristian Sestito, Fanny Spagnolo, and Stefania Perri. 2021. Design of Flexible Hardware Accelerators for Image Convolutions and Transposed Convolutions. *Journal of Imaging* 7, 10 (2021). https://www.mdpi.com/2313-433X/7/10/210

[235] Ali Shafiee, Anirban Nag, Naveen Muralimanohar, Rajeev Balasubramonian, John Paul Strachan, Miao Hu, R. Stanley Williams, and Vivek Srikumar. 2016. ISAAC: A Convolutional Neural Network Accelerator with in-Situ Analog Arithmetic in Crossbars. In *Proceedings of the 43rd International Symposium on Computer Architecture*. 14–26. https://doi.org/10.1109/ISCA.2016.12

[236] Junnan Shan, Mario R. Casu, Jordi Cortadella, Luciano Lavagno, and Mihai T. Lazarescu. 2019. Exact and Heuristic Allocation of MuIti-kernel Applications to Multi-FPGA Platforms. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*. 1–6.

[237] Yakun Sophia Shao, Jason Clemons, Rangharajan Venkatesan, Brian Zimmer, Matthew Fojtik, Nan Jiang, Ben Keller, Alicia Klinefelter, Nathaniel Pinckney, Priyanka Raina, Stephen G. Tell, Yanqing Zhang, William J. Dally, Joel Emer, C. Thomas Gray, Brucek Khailany, and Stephen W. Keckler. 2019. Simba: Scaling Deep-Learning Inference with Multi-Chip-Module-Based Architecture. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture* (Columbus, OH, USA) *(MICRO '52)*. Association for Computing Machinery, New York, NY, USA, 14–27. https://doi.org/10.1145/3352460.3358302

[238] Harsh Sharma, Sumit K Mandal, Janardhan Rao Doppa, Umit Y Ogras, and Partha Pratim Pande. 2022. SWAP: A Server-Scale Communication-Aware Chiplet-Based Manycore PIM Accelerator. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 41, 11 (2022), 4145–4156.

[239] Bhavin J. Shastri, Alexander N. Tait, T. Ferreira de Lima, Wolfram H. P. Pernice, Harish Bhaskaran, C. D. Wright, and Paul R. Prucnal. 2021. Photonics for artificial intelligence and neuromorphic computing. *Nature Photonics* 15 (2021), 102–114. https://doi.org/10.1038/s41566-020-00754-y

[240] Peter W. Shor. 1997. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.* 26, 5 (oct 1997), 1484–1509. https://doi.org/10.1137/s0097539795293172

[241] Siemens. 2022. *Catapult C++/Systemc Synthesis.* https://eda.sw.siemens.com/en-US/ic/catapult-high-level-synthesis/hls/c-cplus/

[242] Kyomin Sohn, Won-Joo Yun, Reum Oh, Chi-Sung Oh, Seong-Young Seo, Min-Sang Park, Dong-Hak Shin, Won-Chang Jung, Sang-Hoon Shin, Je-Min Ryu, Hye-Seung Yu, Jae-Hun Jung, Hyunui Lee, Seok-Yong Kang, Young-Soo Sohn, Jung-Hwan Choi, Yong-Cheol Bae, Seong-Jin Jang, and Gyoyoung Jin. 2017. A 1.2 V 20 nm 307 GB/s HBM DRAM With At-Speed Wafer-Level IO Test Scheme and Adaptive Refresh Considering Temperature Distribution. *IEEE Journal of Solid-State Circuits* 52, 1 (2017), 250–260. https://doi.org/10.1109/JSSC.2016.2602221

[243] Jinook Song, Yunkyo Cho, Jun-Seok Park, Jun-Woo Jang, Sehwan Lee, Joon-Ho Song, Jae-Gon Lee, and Inyup Kang. 2019. An 11.5TOPS/W 1024-MAC Butterfly Structure Dual-Core Sparsity-Aware Neural Processing Unit in 8nm Flagship Mobile SoC. In *2019 IEEE International Solid- State Circuits Conference - (ISSCC)*. 130–132. https://doi.org/10.1109/ISSCC.2019.8662476

[244] Linghao Song, Xuehai Qian, Hai Li, and Yiran Chen. 2017. PipeLayer: A Pipelined ReRAM-Based Accelerator for Deep Learning. In *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 541–552. https://doi.org/10.1109/HPCA.2017.55

[245] P. Soussan, Deniz Sabuncuoglu Tezcan, Francois Iker, Wouter Ruythooren, Bart Swinnen, Bivragh Majeed, and Eric Beyne. 2008. 3D Wafer Level Packaging: Processes and Materials for Trough Silicon Vias & Thin Die Embedding. In *MRS Online Proceedings Library*, Vol. 1112. https://doi.org/10.1557/PROC-1112-E01-05

[246] F. Spagnolo, P. Corsonello, F. Frustaci, and S. Perri. 2023. Design of a Low-Power Super-Resolution Architecture for Virtual Reality Wearable Devices. *IEEE Sensors Journal* 23, 8 (2023), 9009–9016. https://doi.org/10.1109/JSEN.2023.3256524

[247] Fanny Spagnolo, Stefania Perri, and Pasquale Corsonello. 2020. Design of a real-time face detection architecture for heterogeneous systems-on-chips. *Integration* 74 (2020), 1–10. https://doi.org/10.1016/j.vlsi.2020.04.008

[248] F. Spagnolo, S. Perri, and P. Corsonello. 2022. Aggressive Approximation of the SoftMax Function for Power-Efficient Hardware Implementations. *IEEE Transactions on Circuits and Systems II: Express Briefs* 69, 3 (2022), 1652–1656. https://doi.org/10.1109/TCSII.2021.3120495

[249] F. Spagnolo, S. Perri, and P. Corsonello. 2022. Approximate Down-Sampling Strategy for Power-Constrained Intelligent Systems. *IEEE Access* 10 (2022), 7073–7081. https://doi.org/10.1109/ACCESS.2022.3142292

[250] Suraj Srinivas and R. Venkatesh Babu. 2015. Data-free Parameter Pruning for Deep Neural Networks. *CoRR* abs/1507.06149 (2015). http://arxiv.org/abs/1507.06149

[251] Vinay Sriram, David Cox, Kuen Tsoi, and Wayne Luk. 2011. Towards an embedded biologically-inspired machine vision processor. 273–278. https://doi.org/10.1109/FPT.2010.5681487

[252] Dylan Stow, Itir Akgun, Russell Barnes, Peng Gu, and Yuan Xie. 2016. Cost analysis and cost-driven IP reuse methodology for SoC design based on 2.5D/3D integration. In *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*.

[253] Dylan Stow, Yuan Xie, Taniya Siddiqua, and Gabriel H. Loh. 2017. Cost-effective design of scalable high-performance systems using active and passive interposers. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 728–735. https://doi.org/10.1109/ICCAD.2017.8203849

[254] Antonio Giuseppe Maria Strollo, Ettore Napoli, Davide De Caro, Nicola Petra, and Gennaro Di Meo. 2020. Comparison and Extension of Approximate 4-2 Compressors for Low-Power Approximate Multipliers. *IEEE Transactions on Circuits and Systems I: Regular Papers* 67, 9 (2020), 3021–3034. https://doi.org/10.1109/TCSI.2020.2988353

[255] Antonio Giuseppe Maria Strollo, Ettore Napoli, Davide De Caro, Nicola Petra, Gerardo Saggese, and Gennaro Di Meo. 2022. Approximate Multipliers Using Static Segmentation: Error Analysis and Improvements. *IEEE Transactions on Circuits and Systems I: Regular Papers* 69, 6 (2022), 2449–2462. https://doi.org/10.1109/TCSI.2022.3152921

[256] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1–9. https://doi.org/10.1109/CVPR.2015.7298594

[257] Thierry Tambe, Jeff Zhang, Coleman Hooper, Tianyu Jia, Paul N. Whatmough, Joseph Zuckerman, Maico Cassel Dos Santos, Erik Jens Loscalzo, Davide Giri, Kenneth Shepard, Luca Carloni, Alexander Rush, David Brooks, and Gu-Yeon Wei. 2023. 22.9 A 12nm 18.1TFLOPs/W Sparse Transformer Processor with Entropy-Based Early Exit, Mixed-Precision Predication and Fine-Grained Power Management. In *2023 IEEE International Solid- State Circuits Conference (ISSCC)*. IEEE, San Francisco, CA, USA, 342–344. https://doi.org/10.1109/ISSCC42615.2023.10067817

[258] Wenkai Tang and Peiyong Zhang. 2022. GPGCN: A General-Purpose Graph Convolution Neural Network Accelerator Based on RISC-V ISA Extension. *Electronics* 11, 22 (2022). https://doi.org/10.3390/electronics11223833

[259] Yvan Tortorella, Luca Bertaccini, Luca Benini, Davide Rossi, and Francesco Conti. 2023. RedMule: A Mixed-Precision Matrix-Matrix Operation Engine for Flexible and Energy-Efficient On-Chip Linear Algebra and TinyML Training Acceleration. https://doi.org/10.48550/arXiv.2301.03904 arXiv:arXiv:2301.03904

[260] Yvan Tortorella, Luca Bertaccini, Davide Rossi, Luca Benini, and Francesco Conti. 2022. RedMulE: A Compact FP16 Matrix-Multiplication Accelerator for Adaptive Deep Learning on RISC-V-based Ultra-Low-Power SoCs. In *Proceedings of the 2022 Conference & Exhibition on Design, Automation & Test in Europe (DATE '22)*. European Design and Automation Association, Leuven, BEL, 1099–1102.

[261] Mehdi Trabelsi Ajili and Yuko Hara-Azumi. 2022. Multimodal Neural Network Acceleration on a Hybrid CPU-FPGA Architecture: A Case Study. *IEEE Access* 10 (2022), 9603–9617. https://doi.org/10.1109/ACCESS.2022.3144977

[262] Yu-Chi Tsao and Ken Choi. 2012. Area-Efficient VLSI Implementation for Parallel Linear-Phase FIR Digital Filters of Odd Length Based on Fast FIR Algorithm. *IEEE Transactions on Circuits and Systems II: Express Briefs* 59, 6 (2012), 371–375. https://doi.org/10.1109/TCSII.2012.2195062

[263] Yaman Umuroglu, Nicholas J. Fraser, Giulio Gambardella, Michaela Blott, Philip Leong, Magnus Jahre, and Kees Vissers. 2017. FINN: A Framework for Fast, Scalable Binarized Neural Network Inference. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA '17)*. Association for Computing Machinery, New York, NY, USA, 65–74. https://doi.org/10.1145/3020078.3021744

[264] Paramita Basak Upama, Md Jobair Hossain Faruk, Mohammad Nazim, Mohammad Masum, Hossain Shahriar, Gias Uddin, Shabir Barzanjeh, Sheikh Iqbal Ahamed, and Akond Rahman. 2022. Evolution of Quantum Computing: A Systematic Survey on the Use of Quantum Computing Tools. arXiv:2204.01856 [cs.SE]

[265] Akihiko Ushiroyama, Minoru Watanabe, Nobuya Watanabe, and Akira Nagoya. 2022. Convolutional neural network implementations using Vitis AI. In *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*. 0365–0371. https://doi.org/10.1109/CCWC54503.2022.9720794

[266] T. J. Yang J. S. Emer V. Sze, Y. H. Chen. 2017. Efficient Processing of Deep Neural Networks: A Tutorial and Survey. *Proc. IEEE* 105, 12 (2017), 2295–2329. https://doi.org/10.1109/JPROC.2017.2761740

[267] Shaghayegh Vahdat, Mehdi Kamal, Ali Afzali-Kusha, and Massoud Pedram. 2019. TOSAM: An Energy-Efficient Truncation- and Rounding-Based Scalable Approximate Multiplier. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 27, 5 (2019), 1161–1173. https://doi.org/10.1109/TVLSI.2018.2890712

[268] Jurgen Vandendriessche, Bruno Da Silva, and Abdellah Touhafi. 2022. Frequency Evaluation of the Xilinx DPU Towards Energy Efficiency. In *IECON 2022 – 48th Annual Conference of the IEEE Industrial Electronics Society*. 1–6. https://doi.org/10.1109/IECON49645.2022.9968811

[269] Jasmina Vasiljevic, Ljubisa Bajic, Davor Capalija, Stanislav Sokorac, Dragoljub Ignjatovic, Lejla Bajic, Milos Trajkovic, Ivan Hamer, Ivan Matosevic, Aleksandar Cejkov, Utku Aydonat, Tony Zhou, Syed Zohaib Gilani, Armond Paiva, Joseph Chu, Djordje Maksimovic, Stephen Alexander Chin, Zahi Moudallal, Akhmed Rakhmati, Sean Nijjar, Almeet Bhullar, Boris Drazic, Charles Lee, James Sun, Kei-Ming Kwong, James Connolly, Miles Dooley, Hassan Farooq, Joy Yu Ting Chen, Matthew Walker, Keivan Dabiri, Kyle Mabee, Rakesh Shaji Lal, Namal Rajatheva, Renjith Retnamma, Shripad Karodi, Daniel Rosen, Emilio Munoz, Andrew Lewycky, Aleksandar Knezevic, Raymond Kim, Allan Rui, Alexander Drouillard, and David Thompson. 2021. Compute Substrate for Software 2.0. *IEEE Micro* 41, 2 (March 2021), 50–55. https://doi.org/10.1109/MM.2021.3061912

[270] Stylianos I. Venieris and Christos-Savvas Bouganis. 2019. fpgaConvNet: Mapping Regular and Irregular Convolutional Neural Networks on FPGAs. *IEEE Transactions on Neural Networks and Learning Systems* 30, 2 (2019), 326–342. https://doi.org/10.1109/TNNLS.2018.2844093

[271] Stylianos I Venieris, Alexandros Kouris, and Christos-Savvas Bouganis. 2018. Toolflows for mapping convolutional neural networks on FPGAs: A survey and future directions. *ACM Computing Surveys (CSUR)* 51, 3 (2018), 1–39.

[272] Swagath Venkataramani, Vijayalakshmi Srinivasan, Wei Wang, Sanchari Sen, Jintao Zhang, Ankur Agrawal, Monodeep Kar, Shubham Jain, Alberto Mannari, Hoang Tran, Yulong Li, Eri Ogawa, Kazuaki Ishizaki, Hiroshi Inoue, Marcel Schaal, Mauricio Serrano, Jungwook Choi, Xiao Sun, Naigang Wang, Chia-Yu Chen, Allison Allain, James Bonano, Nianzheng Cao, Robert Casatuta, Matthew Cohen, Bruce Fleischer, Michael Guillorn, Howard Haynie, Jinwook Jung, Mingu Kang, Kyu-hyoun Kim, Siyu Koswatta, Saekyu Lee, Martin Lutz, Silvia Mueller, Jinwook Oh, Ashish Ranjan, Zhibin Ren, Scot Rider, Kerstin Schelm, Michael Scheuermann, Joel Silberman, Jie Yang, Vidhi Zalani, Xin Zhang, Ching Zhou, Matt Ziegler, Vinay Shah, Moriyoshi Ohara, Pong-Fei Lu, Brian Curran, Sunil Shukla, Leland Chang, and Kailash Gopalakrishnan. 2021. RaPiD: AI Accelerator for Ultra-low Precision Training and Inference. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. 153–166. https://doi.org/10.1109/ISCA52012.2021.00021

[273] Marian Verhelst, Man Shi, and Linyan Mei. 2022. ML Processors Are Going Multi-Core: A performance dream or a scheduling nightmare? *IEEE Solid-State Circuits Magazine* 14, 4 (2022), 18–27.

[274] Thiruvengadam Vijayaraghavan, Yasuko Eckert, Gabriel H. Loh, Michael J. Schulte, Mike Ignatowski, Bradford M. Beckmann, William C. Brantley, Joseph L. Greathouse, Wei Huang, Arun Karunanithi, Onur Kayiran, Mitesh Meswani, Indrani Paul, Matthew Poremba, Steven Raasch, Steven K. Reinhardt, Greg Sadowski, and Vilas Sridharan. 2017. Design and Analysis of an APU for Exascale Computing. In *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 85–96.

[275] Pascal Vivet, Eric Guthmuller, Yvain Thonnart, Gael Pillonnet, César Fuguet, Ivan Miro-Panades, Guillaume Moritz, Jean Durupt, Christian Bernard, Didier Varreau, Julian Pontes, Sébastien Thuries, David Coriat, Michel Harrand, Denis Dutoit, Didier Lattard, Lucile Arnaud, Jean Charbonnier, Perceval Coudrain, Arnaud Garnier, Frédéric Berger, Alain Gueugnot, Alain Greiner, Quentin L. Meunier, Alexis Farcy, Alexandre Arriordaz, Séverine Chéramy, and Fabien Clermidy. 2021. IntAct: A 96-Core Processor With Six Chiplets 3D-Stacked on an Active Interposer With Distributed Interconnects and Integrated Power Management. *IEEE Journal of Solid-State Circuits* 56, 1 (2021), 79–97.

[276] Ofer Shacham Preeth Venkatesan Christos Kozyrakis Mark Horowitz Wajahat Qadeer, Rehan Hameed. 2015. Convolution engine: balancing efficiency and flexibility in specialized computing. *Commun. ACM* 58, 4 (2015), 85–93. https://doi.org/10.1145/2735841

[277] Weier Wan, Rajkumar Kubendran, Clemens Schaefer, Sukru Burc Eryilmaz, Wenqiang Zhang, Dabin Wu, Stephen Deiss, Priyanka Raina, He Qian, Bin Gao, Siddharth Joshi, Huaqiang Wu, H.-S. Philip Wong, and Gert Cauwenberghs. 2022. A compute-in-memory chip based on resistive random-access memory. *Nature* 608, 7923 (01 Aug 2022), 504–512. https://doi.org/10.1038/s41586-022-04992-8

[278] Huizheng Wang, Weihong Xu, Zaichen Zhang, Xiaohu You, and Chuan Zhang. 2022. An Efficient Stochastic Convolution Architecture Based on Fast FIR Algorithm. *IEEE Transactions on Circuits and Systems II: Express Briefs* 69, 3 (2022), 984–988. https://doi.org/10.1109/TCSII.2021.3121081

[279] Jin Wang and Shenshen Gu. 2021. FPGA Implementation of Object Detection Accelerator Based on Vitis-AI. In *2021 11th International Conference on Information Science and Technology (ICIST)*. 571–577. https://doi.org/10.1109/ICIST52614.2021.9440554

[280] Jichen Wang, Jun Lin, and Zhongfeng Wang. 2018. Efficient Hardware Architectures for Deep Convolutional Neural Network. *IEEE Transactions on Circuits and Systems I: Regular Papers* 65, 6 (2018), 1941–1953. https://doi.org/10.1109/TCSI.2017.2767204

[281] Shihang Wang, Jianghan Zhu, Qi Wang, Can He, and Terry Tao Ye. 2021. Customized Instruction on RISC-V for Winograd-Based Convolution Acceleration. In *2021 IEEE 32nd International Conference on Application-specific Systems, Architectures and Processors (ASAP)*. 65–68. https://doi.org/10.1109/ASAP52443.2021.00018

[282] Yizhi Wang, Jun Lin, and Zhongfeng Wang. 2019. FPAP: A Folded Architecture for Energy-Quality Scalable Convolutional Neural Networks. *IEEE Transactions on Circuits and Systems I: Regular Papers* 66 (2019), 288–301.

[283] Sally Ward-Foxton. 2022. Axelera Demos AI Test Chip After Taping Out in Four Months. (2022).

[284] Pete Warden and Daniel Situnayake. 2019. *TinyML*. O'Reilly Media, Inc.

[285] Xuechao Wei, Cody Hao Yu, Peng Zhang, Youxiang Chen, Yuxin Wang, Han Hu, Yun Liang, and Jason Cong. 2017. Automated systolic array architecture synthesis for high throughput CNN inference on FPGAs. In *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*. 1–6. https://doi.org/10.1145/3061639.3062207

[286] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2016. Learning Structured Sparsity in Deep Neural Networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems* (Barcelona, Spain) *(NIPS'16)*. Curran Associates Inc., Red Hook, NY, USA, 2082–2090.

[287] Xilinx Inc. 2022. *Vitis High-Level Synthesis User Guide*. https://www.xilinx.com/support/documentation/sw_manuals/xilinx2022_2/ug1399-vitis-hls.pdf

[288] Zhou Yu Xuan, Ching-Jui Lee, and Tsung Tai Yeh. 2022. Lego: Dynamic Tensor-Splitting Multi-Tenant DNN Models on Multi-Chip-Module Architecture. In *2022 19th International SoC Design Conference (ISOCC)*. 173–174. https://doi.org/10.1109/ISOCC56007.2022.10031596

[289] Cheng-Xin Xue, Wei-Hao Chen, Je-Syu Liu, Jia-Fang Li, Wei-Yu Lin, Wei-En Lin, Jing-Hong Wang, Wei-Chen Wei, Ting-Wei Chang, Tung-Cheng Chang, Tsung-Yuan Huang, Hui-Yao Kao, Shih-Ying Wei, Yen-Cheng Chiu, Chun-Ying Lee, Chung-Chuan Lo, Ya-Chin King, Chorng-Jung Lin, Ren-Shuo Liu, Chih-Cheng Hsieh, Kea-Tiong Tang, and Meng-Fan Chang. 2019. A 1Mb Multibit ReRAM Computing-In-Memory Macro with 14.6ns Parallel MAC Computing Time for CNN Based AI Edge Processors. In *2019 IEEE International Solid- State Circuits Conference - (ISSCC)*. 388–390. https://doi.org/10.1109/ISSCC.2019.8662395

[290] Zhixi Yang, Jie Han, and Fabrizio Lombardi. 2015. Approximate compressors for error-resilient multiplier design. In *2015 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*. 183–186. https://doi.org/10.1109/DFT.2015.7315159

[291] Amir Yazdanbakhsh, Kambiz Samadi, Nam Sung Kim, Hadi Esmaeilzadeh, Hajar Falahati, and Philip J. Wolfe. 2018. GANAX: A Unified MIMD-SIMD Acceleration for Generative Adversarial Networks. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. 650–661. https://doi.org/10.1109/ISCA.2018.00060

[292] Hanchen Ye, HyeGang Jun, Hyunmin Jeong, Stephen Neuendorffer, and Deming Chen. 2022. ScaleHLS: A Scalable High-Level Synthesis Framework with Multi-Level Transformations and Optimizations. In *Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC)*. 1355–1358.

[293] Efstratios Zacharelos, Italo Nunziata, Gerardo Saggese, Antonio G.M. Strollo, and Ettore Napoli. 2022. Approximate Recursive Multipliers Using Low Power Building Blocks. *IEEE Transactions on Emerging Topics in Computing* 10, 3

(2022), 1315–1330. https://doi.org/10.1109/TETC.2022.3186240

[294] Florian Zaruba, Fabian Schuiki, and Luca Benini. 2021. Manticore: A 4096-Core RISC-V Chiplet Architecture for Ultraefficient Floating-Point Computing. *IEEE Micro* 41, 2 (March 2021), 36–42. https://doi.org/10.1109/MM.2020.3045564

[295] Georgios Zervakis, Kostas Tsoumanis, Sotirios Xydis, Dimitrios Soudris, and Kiamal Pekmestzi. 2016. Design-Efficient Approximate Multiplication Circuits Through Partial Product Perforation. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 24, 10 (2016), 3105–3117. https://doi.org/10.1109/TVLSI.2016.2535398

[296] Chen Zhang, Di Wu, Jiayu Sun, Guangyu Sun, Guojie Luo, and Jason Cong. 2016. Energy-Efficient CNN Implementation on a Deeply Pipelined FPGA Cluster. In *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*. Association for Computing Machinery, New York, NY, USA, 326–331. https://doi.org/10.1145/2934583.2934644

[297] Jie-Fang Zhang, Ching-En Lee, Chester Liu, Yakun Sophia Shao, Stephen W. Keckler, and Zhengya Zhang. 2019. SNAP: A 1.67 — 21.55TOPS/W Sparse Neural Acceleration Processor for Unstructured Sparse Deep Neural Network Inference in 16nm CMOS. In *2019 Symposium on VLSI Circuits*. C306–C307. https://doi.org/10.23919/VLSIC.2019.8778193

[298] Shijin Zhang, Zidong Du, Lei Zhang, Huiying Lan, Shaoli Liu, Ling Li, Qi Guo, Tianshi Chen, and Yunji Chen. 2016. Cambricon-X: An accelerator for sparse neural networks. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)* (Taipei, Taiwan) *(MICRO-49)*. Article 20, 12 pages. https://doi.org/10.1109/MICRO.2016.7783723

[299] Xiaowu Zhang, Jong Kai Lin, Sunil Wickramanayaka, Songbai Zhang, Roshan Weerasekera, Rahul Dutta, Ka Fai Chang, King-Jien Chui, Hong Yu Li, David Soon Wee Ho, Liang Ding, Guruprasad Katti, Suryanarayana Bhattacharya, and Dim-Lee Kwong. 2015. Heterogeneous 2.5D integration on through silicon interposer. *Applied Physics Reviews* 2, 2 (2015). https://doi.org/10.1063/1.4921463

[300] Yongwei Zhao, Chang Liu, Zidong Du, Qi Guo, Xing Hu, Yimin Zhuang, Zhenxing Zhang, Xinkai Song, Wei Li, Xishan Zhang, Ling Li, Zhiwei Xu, and Tianshi Chen. 2021. Cambricon-Q: A Hybrid Architecture for Efficient Training. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. 706–719. https://doi.org/10.1109/ISCA52012.2021.00061

[301] Brian Zimmer, Rangharajan Venkatesan, Yakun Sophia Shao, Jason Clemons, Matthew Fojtik, Nan Jiang, Ben Keller, Alicia Klinefelter, Nathaniel Pinckney, Priyanka Raina, Stephen G. Tell, Yanqing Zhang, William J. Dally, Joel S. Emer, C. Thomas Gray, Stephen W. Keckler, and Brucek Khailany. 2020. A 0.32–128 TOPS, Scalable Multi-Chip-Module-Based Deep Neural Network Inference Accelerator With Ground-Referenced Signaling in 16 nm. *IEEE Journal of Solid-State Circuits* 55, 4 (2020), 920–932. https://doi.org/10.1109/JSSC.2019.2960488