RedMulE: A Compact FP16 Matrix-Multiplication Accelerator for Adaptive Deep Learning on RISC-V-Based Ultra-Low-Power SoCs

Yvan Tortorella*, Luca Bertaccini†, Davide Rossi*, Luca Benini*†, Francesco Conti*
*University of Bologna, Bologna, Italy. *Email:* {*yvan.tortorella, davide.rossi, luca.benini, f.conti*}@*unibo.it*
†ETH Zurich, Zurich, Switzerland. *Email:* {*lbertaccini, lbenini*}@*iis.ethz.ch*

***Abstract*—The fast proliferation of extreme-edge applications using Deep Learning (DL) based algorithms required dedicated hardware to satisfy extreme-edge applications' latency, throughput, and precision requirements. While inference is achievable in practical cases, online finetuning and adaptation of general DL models are still highly challenging. One of the key stumbling stones is the need for parallel floating-point operations, which are considered unaffordable on sub-100 mW extreme-edge SoCs. We tackle this problem with RedMulE (Reduced-precision matrix Multiplication Engine), a parametric low-power hardware accelerator for FP16 matrix multiplications - the main kernel of DL training and inference - conceived for tight integration within a cluster of tiny RISC-V cores based on the PULP (Parallel Ultra-Low-Power) architecture. In 22 nm technology, a 32-FMA RedMulE instance occupies just 0.07 mm^2 (14% of an 8-core RISC-V cluster) and achieves up to 666 MHz maximum operating frequency, for a throughput of 31.6 MAC/cycle (98.8% utilization). We reach a cluster-level power consumption of 43.5 mW and a full-cluster energy efficiency of 688 16-bit GFLOPS/W. Overall, RedMulE features up to 4.65× higher energy efficiency and 22× speedup over SW execution on 8 RISC-V cores.**

I. Introduction & Related Works

In the last few years, the amount of Internet of Things (IoT) devices connected and executing Machine Learning (ML) and, in particular, Deep Learning (DL) based algorithms like Deep Neural Networks (DNNs) has considerably increased. Moving the computation from data centers to energy-efficient IoT end-nodes helps lower the amount of data sent over the network, improve energy efficiency, and prevent network congestion. Extreme-edge ML and Tiny-ML use efficient IoT endpoints to move data processing from the cloud to the edge.

Extreme-edge inference is achievable in practical cases since it can be performed with low precision integer operations that help increase the energy efficiency, reduce the memory footprint and the area overhead, with reduced accuracy loss [1], [2]. Eyeriss [3] is a hardware accelerator designed for Convolutional Neural Networks (CNNs) inference with INT16 arithmetic and implemented in 65 nm technology. Zeng *et al.* [4] also designed an accelerator for CNNs inference in the same technology using INT8, and the accelerator features 1.8× higher power consumption, 14.5× higher energy efficiency, and 25× higher throughput than Eyeriss. Also, EIE [5] is an inference chip based on INT8 arithmetic designed for compressed DNNs and implemented in 45 nm technology, but characterized by 590 mW of average power consumption. Simba [6] is a DNNs inference design implemented in 16 nm technology and featuring 9.1 TOPS/W energy efficiency with INT8 arithmetic.

All the chips mentioned so far are examples of accelerators designed only for extreme low-power inference, while on-chip training is still challenging because it imposes restrictive data accuracy and precision requirements. Single-precision and double-precision floating-point (FP) operations provide sufficient range and dynamic, but are highly time and energy-consuming. Graphic Processing Units (GPUs) like the NVIDIA

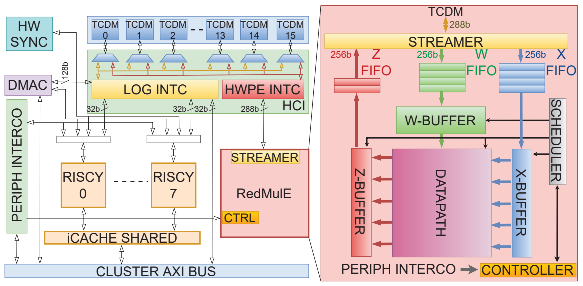

Fig. 1. PULP cluster overview with a focus on RedMulE architecture.

Ampere A100 [7] and HPC-oriented designs like Manticore [8] are examples of high-performance platforms for DL training providing industry-leading computing performances at the cost of high power consumption and area occupation. These facts limit their application on edge, imposing severe limitations on implementing novel learning algorithms [9] on extreme-edge end nodes. For this reason, a strong effort was recently made to adapt learning algorithms to low-precision FP16 [10], [11], and FP8 [12] with no accuracy loss, showing that massive gains can be achieved by lowering the precision to just the right amount needed [13].

Cambricon-Q [14] provides an example of a training-oriented chip for high accuracy and energy efficiency, but their training is based on 8-bit fixed-point arithmetic, while it is well-known that most common training algorithms require floating-point operations. IBM [15], [16] proposes an AI computing platform for training with FP16 and hybrid FP8 operands at the cost of overall power consumption that is not affordable on sub-100 mW extreme-edge devices. Anders *et al.* [17] propose a reconfigurable accelerator for matrix-multiplications supporting mixed-precision computations, targeting training-oriented applications thanks to low power consumption, low area occupation, and FP16 arithmetic.

We present *RedMulE* (Reduced-precision matrix Multiplication Engine), the first tightly-coupled and parametric hardware accelerator for FP16 (IEEE binary16) matrix multiplication (the main kernel in online learning), designed to be integrated into a PULP [18] cluster. It is optimized for better data reuse targeting ultra-low-power unified FP16 training and inference on edge. We prototyped our design within an 8-core PULP cluster in 22 nm CMOS technology, targeting a RedMulE instance with 32 FMAs [19]. We show that RedMulE occupies only 0.07 mm2 (14% of the entire cluster), achieves up to 22× speedup, and 4.65× higher energy efficiency than a software counterpart running on the 8 RISC-V cluster cores.

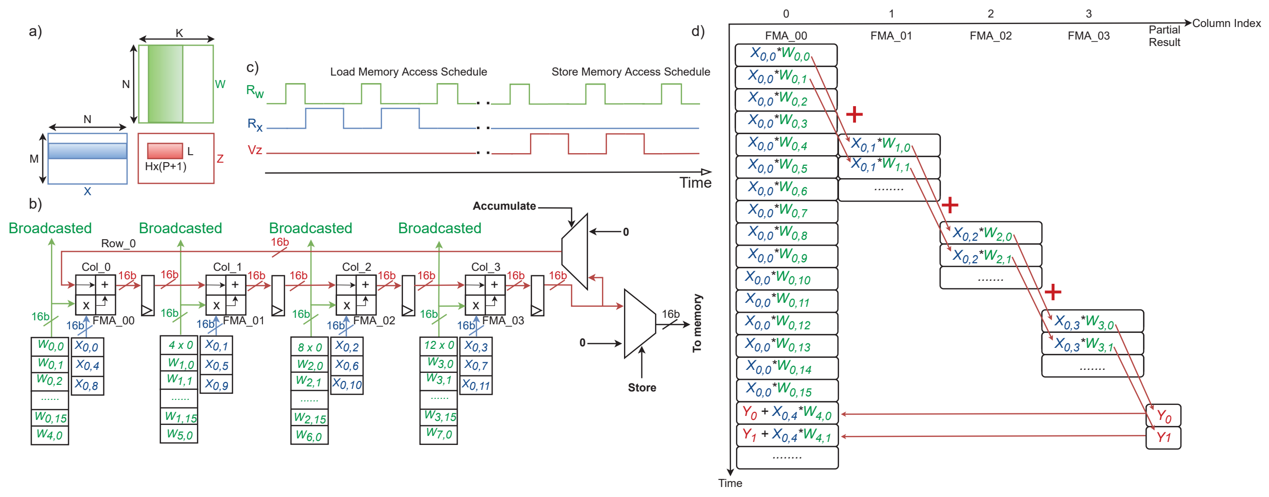

Fig. 2. a) Matrix-multiplication execution; b) Row of FMAs within RedMulE Datapath; c) Memory access schedule in load/store mode described in terms of R (ready) and V (valid) handshake signals; d) Pipeline evolution within a row of FMAs.

II. Architecture

A. PULP Cluster

In Fig. 1, we show the architecture of a PULP cluster featuring 8 RISC-V cores and enhanced with a tightly-coupled accelerator called Hardware Processing Engine (HWPE) [20], that shares the system memory with the cluster cores and is software-programmed by the cores. The Tightly-Coupled Data Memory (TCDM) is accessible via a single-cycle latency Heterogeneous Cluster Interconnect (HCI) designed for synergistic operation of the RISC-V cores and HWPEs. The HCI features two separated branches, *logarithmic* and *shallow*: the logarithmic branch allows all-to-all single-cycle access from 32-bit initiator ports (cores, DMA) to each of the word-interleaved memory banks; conflicts are managed by granting only one initiator per bank, with a round-robin scheme. The shallow branch features a single 288-bit port, routed to 9 adjacent memory banks treated like a single 288-bit bank without arbitration. The TCDM banks are connected to the two branches via a set of multiplexers, which grant access to one branch or the other according to a configurable-latency starvation-free rotation scheme.

B. RedMulE architecture

RedMulE targets matrix multiplications of the kind:

$$\mathbf{Z} = \mathbf{X} \cdot \mathbf{W}$$

where $\mathbf{X}$ is a matrix of size $M \times N$, $\mathbf{W}$ is a matrix of size $N \times K$, and $\mathbf{Z}$ has size $M \times K$, as shown in Fig 2a. This operation is performed using a semi-systolic array connected as an HWPE to a PULP cluster (Fig. 1). Notice that, in DNN training, $\mathbf{X}$ and $\mathbf{W}$ can assume either input and weight matrices indifferently: the accelerator is designed symmetrically and can be indifferently used as weight- or input-stationary.

The accelerator is divided in several modules. First, the *Datapath* is the core of RedMulE: it is an array of FP16 FMA units [19] interconnected in a semi-systolic fashion, as shown in Fig 2b. FMA units are organized in L rows, each made of H columns. Within each row, a number of H FMAs are wired together so that each FMA computing an intermediate product will pass its result to the next one. The partial product computed by the last FMA of a row is fed back as accumulation input of the first FMA of the same row. Each FMA features a design-time configurable number of internal pipeline registers (P), and to saturate the array, the $\mathbf{X}$-matrix elements of each FMA are held steady for $H \times (P+1)$ cycles, while new W-matrix operands are streamed-in cycle-by-cycle and broadcasted to all the FMAs of a column. Each row of FMAs computes $H \times (P+1)$ elements of a row of the $\mathbf{Z}$-matrix, that are stored in the memory only at the end of the computation, optimizing internal data reuse.

The *Streamer* is a specialized memory access unit that connects RedMulE to the HCI shallow branch through 9 32-bit memory ports used alternatively for load and store operations. The Streamer is connected to three internal buffers: a X-Buffer that changes all the L inputs of a column once every $H \times (P+1)$ cycles; a W-Buffer made of H shift registers, each broadcasting a new $\mathbf{W}$-element to all the L FMAs of a column every cycle; a Z-Buffer that buffers the computed $\mathbf{Z}$-elements.

The *Scheduler* and the *Controller* regulate the accelerator execution and contain the register file, accessed by the cores to program the accelerator.

We focus on a design with $H = 4$, $L = 8$, $P = 3$ parameters, resulting in 32 FMA units and 9 32-bit TCDM memory ports, for a 256-bit + 32-bit for non-word-aligned accesses.

C. Working Principle

RedMulE's operation starts by pre-loading the X-Buffer with L rows from $\mathbf{X}$-matrix, each row made of 16 FP16 elements (256-bit memory width/16-bit precision), namely $\mathbf{x_{0,0}}$ - $\mathbf{x_{0,15}}$ for Row_0, $\mathbf{x_{1,0}}$ - $\mathbf{x_{1,15}}$ for Row_1, and so on. Then, RedMulE loads a set of 16 $\mathbf{W}$-elements ($\mathbf{w_{0,0}}$ - $\mathbf{w_{0,15}}$) inside the first shift register of the W-buffer, broadcasting each of them to all the L FMAs of the first column.

After 4 (=$P+1$) cycles, all the L FMAs in the first column pass their computed partial products to all the L FMAs of the second column. The accelerator loads another set of 16 $\mathbf{W}$-elements ($\mathbf{w_{1,0}}$ - $\mathbf{w_{1,15}}$) to broadcast them to all the FMAs in the second column. Once all the H FMAs of a row have completed their computations, calculating a subset of 16 row-column intermediate products, RedMulE activates its feedback to provide the intermediate results to the accumulation input of the first FMA of the given row and reiterates the computation. Then, the X-Buffer provides a new $\mathbf{X}$-operand to the first

TABLE I
STATE OF THE ART COMPARISON. FIRST LINE = BEST EFFICIENCY; SECOND LINE = PEAK PERFORMANCE. 1 MAC = 2 OPS.

Category	**Design**	**Tech** *nm*	**Area** *mm²*	**Freq** *MHz*	**Volt** *V*	**Power** *mW*	**Perf** *GOPS*	**Energy Eff** *GOPS/W*	**MAC Units**	**Precision**
GPU	NVIDIA A100 [21]	7	-	1410	-	300000	-	-	256	FP16
Inference Chips	Eyeriss [3]	65	12.25	250	1.0	278	46	166	168	INT16
	EIE [5]	45	40.8	800	-	590	102	173	64	INT8
	Zeng etal. [4]	65	2.14	250	-	478	1152	2410	256	INT8
	Simba [6]	16	6	161	0.42	-	-	9100	1024	INT8
				2000	1.2	-	4000	-		
Training Chips	IBM [15]	7	19.6	1000	0.55	4400	8000	1800	4096	FP16
				1600	0.75	13000	12800	980		
	Cambricon-Q [14]	45	888	1000	0.6	1030	2000	2240	1024	INT8
HPC	Manticore [8]	22	888	500	0.6	200	25	188	24	FP64
				1000	0.9	900	54	50		
Mat-Mul Acc.	Anders et al. [17]	14	0.024	2.1	0.26	0.023	0.068	2970	16	FP16
				1090	0.9	82.7	34	420		
Our work	**PULP + RedMulE**	22	0.5	476	0.65	43.5	30	688	32	FP16
				666	0.8	90.7	42	462		
	PULP + RedMulE	65	3.85	200	1.2	89.1	12.6	152	32	FP16

column of FMAs, and a new set of 16 $\mathbf{W}$-elements is re-loaded in the first W shift register. After four cycles, all the L FMAs of the first column produce a new partial product and provide it to the FMAs in the second column. X-Buffer provides a new $\mathbf{X}$-operand at the input of the second column of FMAs, and the W-Buffer loads a new set of 16 $\mathbf{W}$-elements in the second W shift register for broadcasting, and the computation continues. Fig 2d shows the pipeline evolution inside a row of FMAs.

To guarantee a continuous data flow in the accelerator, the W-buffer accesses the memory once every 4-cycles to load a new set of 16 $\mathbf{W}$-elements. Once the X-Buffer is empty, ReMulE reuses the *Streamer* port to load the $\mathbf{X}$-operands. Such operation is made by interleaving the memory accesses to $\mathbf{X}$-matrix between two adjacent $\mathbf{W}$-matrix accesses (Fig 2c) until the complete fulfillment of the X-buffer, maximizing the memory port utilization. After an entire row-column multiplication concludes, the Z-Buffer buffers the output products, and then also store operations are interleaved between two adjacent $\mathbf{W}$ load accesses until the Z-Buffer is empty.

III. EXPERIMENTAL RESULTS

Our experiments target a $22\,\mathrm{nm}$ technology using Synopsys Design Compiler for synthesis (slow corner at $f_{\mathrm{targ}} = 208\,\mathrm{MHz}$, $V_{DD} = 0.59\,\mathrm{V}$, $T = 125\,°\mathrm{C}$) and Cadence Innovus for full-cluster Place&Route in the same operating point. We performed timing and power analysis with back annotated switching activity from post-layout simulation in typical corner at $25\,°\mathrm{C}$, and $0.65\,\mathrm{V}$ for peak energy efficiency or $0.8\,\mathrm{V}$ for peak throughput and frequency. In Table I, we resume the State-of-the-Art comparison, including a PULP cluster with RedMulE prototyped in $65\,\mathrm{nm}$.

A. Area, efficiency and performance

RedMulE occupies $0.07\,\mathrm{mm}^2$, corresponding to 14% of the entire PULP cluster. As can be noticed from Table I, at a system level, our design is the only one that occupies less than $1\,\mathrm{mm}^2$. The only exception is for Anders *et al.* [17], who show the results only for the systolic array alone. Fig. 3a and Fig. 3b show the area and power breakdown of the standalone RedMulE, while Fig. 3c shows the energy consumed by the cluster per FMA operation on RedMulE. At a cluster level, the average power consumption is $43.5\,\mathrm{mW}$, and the RedMulE contribution dominates it for 69%, while TCDM banks and the HCI interconnect contribution is 17.1%. Also in this case, our design outperforms the other accelerators. Only Anders *et al.* reach lower power consumption, but with a more scaled technology and in near-threshold operating conditions.

We reach a cluster peak energy efficiency of $688\,\mathrm{GFLOPS/W}$, which is $4.3\times$ lower than Anders *et al.*, whose results are obtained in near-threshold operating conditions and extremely reduced frequency ($2.1\,\mathrm{MHz}$). IBM [15] reaches $2.6\times$ better energy efficiency, at the cost of $440\,\mathrm{mW}$ of power consumption, $10\times$ higher than our design.

Fig. 3c and Fig. 3d highlight how the RedMulE energy efficiency decreases when lowering the computational burden. For small matrix sizes, the control overhead proportionally increases, reducing the performance and thus the energy efficiency. It is evident that the cluster energy per FMA operation considerably decreases when augmenting the amount of FMA computation since the utilization increases.

We evaluated RedMulE's throughput and computation cycles against the SW execution on 8 parallel RISC-V cores. RedMulE reaches a peak throughput of $31.6\,\mathrm{MACs/cycle}$ (98% utilization), meaning $21.1\,\mathrm{GMACS}$, or $42\,\mathrm{GFLOPS}$ at $666\,\mathrm{MHz}$ with $0.80\,\mathrm{V}$ supply. Even though targeting different precision, our system performance is comparable to HPC designs. The Manticore prototype [8] features just $1.3\times$ higher performance despite its higher frequency, but with $10\times$ higher power consumption. On the other hand, we reach $1.2\times$ higher throughput than Anders *et al.* accelerator that works at $1.6\times$ higher frequency in the same precision. To conclude, RedMulE introduces up to $22\times$ speedup over the software baseline and reaches 98.8% of the ideal case for a higher amount of computations (Fig. 4a).

Parametric area swipe: We studied the area overhead introduced when changing the number of FMAs within RedMulE, fixing the FMA's internal pipeline registers to $P = 3$ (Fig 4b). RedMulE's area occupation becomes comparable to the area of the entire PULP cluster with 256 FMAs ($H = 8$, $L = 32$), and doubles it with 512 ($H = 16, L = 32$). Changing the shape of the internal array also affects the number of memory ports. In particular, changing the H parameter from 4 to 5 results in including 4 ($= P + 1$) additional pipeline registers within each row. To keep a high FMAs utilization, the bandwidth towards the memory increases by $4 \times 16\text{-bit}$ (two additional memory ports), limiting the integration in the cluster.

B. Use Case: TinyMLPerf AutoEncoder

We evaluated RedMulE's performance on the TinyMLPerf [22] AutoEncoder benchmark as a possible use-case, compared with a software baseline executed on 8 RISC-V cores. First, we conducted the test with a batch size (B) of 1, i.e., a single input propagated forward and backward through the autoencoder

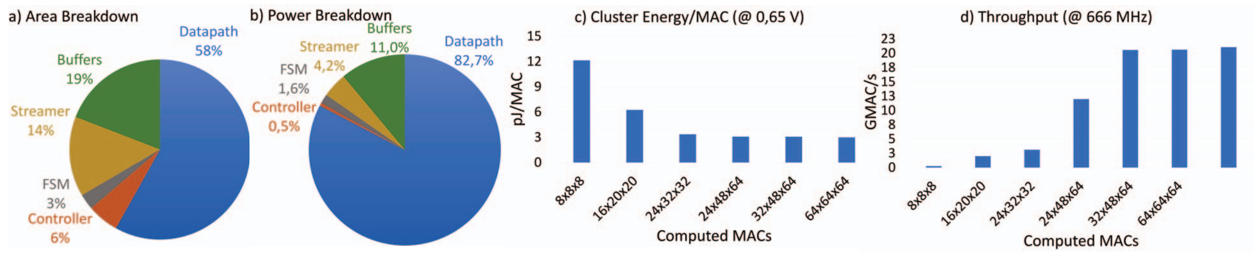

Fig. 3. a) RedMulE area breakdown; b) RedMulE power breakdown; c) Cluster energy per MAC operation; d) Throughput at maximum cluster frequency;

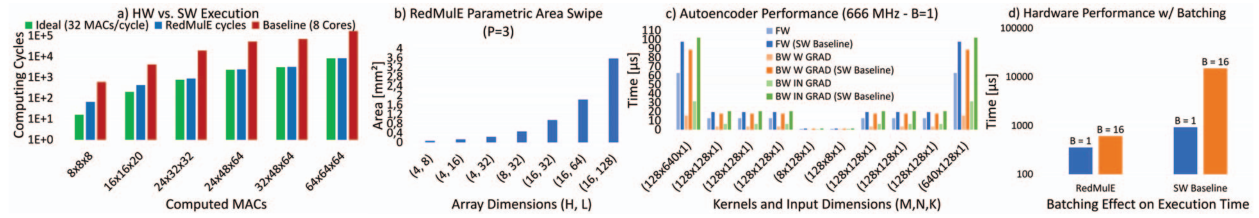

Fig. 4. a) HW vs. SW computational performancewith respect to ideal case (32 MACs/cycle); b) RedMulE area swipe as a function of H and L ($P = 3$); c) RedMulE performance tested on TinyML Perf Autoencoder benchmark; d) Effect of batching on HW/SW benchmark execution.

(Fig 4c). RedMulE speedup is $2.6\times$ with significant advantages in particular in backward operations. The accelerator has a smaller speedup during forward operations due to the K dimension, which is constant and equal to B. Consequently, RedMulE suffers from the effect of the latency introduced by the pipeline stages but does not benefit from the effect of the throughput because there are not sufficient elements in the activation matrix to fulfill the pipelines. We can improve the utilization of the accelerator by increasing B, at the cost of more memory required for activations. We compare $B = 1$ and $B = 16$ in Fig 4d; both configurations are well-fitting the L2 memory of a typical PULP-based system, with the $B = 16$ one having an overall footprint of $184\,\mathrm{kB}$. While the performance of the software baseline does not scale with a larger B, RedMulE's throughput is improved by almost $16\times$, achieving $24.4\times$ of speedup over the software counterpart.

IV. Conclusions

We presented RedMulE, a cluster-coupled accelerator for FP16 matrix multiplications that occupies $0.07\,\mathrm{mm}^2$ (14% of a PULP cluster with 8 RISC-V cores) and introduces up to $22\times$ speedup and $4.65\times$ higher energy efficiency than a software counterpart running on 8 RISC-V cores.

Acknowledgement

This work was supported in part by the EU H2020 "WiPLASH" (g.a. 863337), by the ECSEL H2020 "AI4DI" (g.a. 826060), and by Thales Alenia Space.

References

[1] Z. Zhou *et al.*, "Edge Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing," *Proceedings of the IEEE (Volume: 107, Issue: 8)*, 2019.
[2] M. G. S. Murshed *et al.*, "Machine Learning at the Network Edge: A Survey," *https://arxiv.org/abs/1908.00080*, 2019.
[3] Y. Chen *et al.*, "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," *IEEE Journal of Solid-State Circuits (Volume: 52, Issue: 1, Jan. 2017)*, 2017.
[4] Y. Zeng *et al.*, "Accelerating Convolutional Neural Network Inference Based on a Reconfigurable Sliced Systolic Array," *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2021.
[5] S. Han *et al.*, "EIE: Efficient Inference Engine on Compressed Deep Neural Network," *ISCA*, 2016.
[6] Y. S. Shao *et al.*, "Simba: Scaling Deep-Learning Inference with Multi-Chip-Module-Based Architecture," *MICRO '52: Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2019.
[7] "NVIDIA A100 Tensor Core GPU Architecture."
[8] F. Zaruba *et al.*, "Manticore: A 4096-Core RISC-V Chiplet Architecture for Ultraefficient Floating-Point Computing," *IEEE Micro (Volume: 41, Issue: 2, March-April 1)*, 2021.
[9] L. Ravaglia *et al.*, "Memory-Latency-Accuracy Trade-Offs for Continual Learning on a RISC-V Extreme-Edge Node," *IEEE Workshop on Signal Processing Systems (SiPS)*, 2020.
[10] "Training With Mixed Precision - NVIDIA Deep Learning Performance Documentation," 2021.
[11] A. F. Rodriguez Perez *et al.*, "Lower Numerical Precision Deep Learning Inference and Training ," 2018.
[12] X. Sun *et al.*, "Hybrid 8-bit Floating Point (HFP8) Training and Inference for Deep Neural Networks," *IBM T. J. Watson Research CenterYorktown Heights, NY 10598, USA*, 2019.
[13] G. Tagliavini *et al.*, "A Trans-precision Floating-Point Platform for Ultra-Low Power Computing," *2018 Design, Automation & Test in Europe Conference & Exhibition.*, 2018.
[14] Y. Zhao *et al.*, "Cambricon-Q: A Hybrid Architecture for Efficient Training," *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, 2021.
[15] A. Agrawal *et al.*, "9.1 A 7nm 4-Core AI Chip with 25.6TFLOPS Hybrid FP8 Training, 102.4TOPS INT4 Inference and Workload-Aware Throttling," *2021 IEEE International Solid- State Circuits Conference (ISSCC)*, 2021.
[16] S. Venkataramani *et al.*, "RaPiD: AI Accelerator for Ultra-low Precision Training and Inference," *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, 2021.
[17] M. Anders *et al.*, "2.9TOPS/W Reconfigurable Dense/Sparse Matrix-Multiply Accelerator with Unified INT8/INTI6/FP16 Datapath in 14NM Tri-Gate CMOS," *IEEE Symposium on VLSI Circuits*, 2018.
[18] F. Conti *et al.*, "PULP: A ultralow power parallel accelerator for energy-efficient and flexible embedded vision," *Journal of Signal Processing Systems, vol. 84, no. 3, pp. 339–354*, 2016.
[19] S. Mach *et al.*, "FPnew: An Open-Source Multi-Format Floating-Point Unit Architecture for Energy-Proportional Transprecision Computin," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 29, no. 4, pp. 774-78*, 2020.
[20] F. Conti *et al.*, "XNOR Neural Engine: A Hardware Accelerator IP for 21.6 fJ-per-operation Binary Neural Network Inference," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (Volume: 37, Issue: 11)*, 2018.
[21] J. Choquette *et al.*, " The A100 Datacenter GPU and Ampere Architecture," *2021 IEEE International Solid- State Circuits Conference (ISSCC)*, 2021.
[22] C. Banbury *et al.*, "MLPerf Tiny Benchmark," *NeurIPS 2021 Datasets and Benchmarks Track (Round 1)*, 2021.