IEEE Open Journal of the
Solid-State Circuits Society

Received 30 June 2022; revised 2 September 2022; accepted 21 September 2022. Date of publication 27 September 2022;
date of current version 13 December 2022.

Digital Object Identifier 10.1109/OJSSCS.2022.3210082

DARKSIDE: A Heterogeneous RISC-V Compute Cluster for Extreme-Edge On-Chip DNN Inference and Training

ANGELO GAROFALO[1], YVAN TORTORELLA[1], MATTEO PEROTTI[2], LUCA VALENTE[1], ALESSANDRO NADALINI[1], LUCA BENINI[1,2], DAVIDE ROSSI[1] (Member, IEEE), AND FRANCESCO CONTI[1] (Member, IEEE)

[1]Department of Electrical, Electronic and Information Engineering, University of Bologna, 40126 Bologna, Italy

[2]IIS Integrated Systems Laboratory, ETH Zürich, 8092 Zürich, Switzerland

CORRESPONDING AUTHOR: A. GAROFALO (e-mail: angelo.garofalo@unibo.it)

This work was supported in part by EU Horizon 2020 Research and Innovation through the European Pilot Project under Grant 101034126; in part by WiPLASH under Grant 863337; in part by ECSEL Horizon 2020 Project AI4DI under Grant 826060; and in part by GreenWaves Technologies.

ABSTRACT On-chip deep neural network (DNN) inference and training at the Extreme-Edge (*TinyML*) impose strict latency, throughput, accuracy, and flexibility requirements. Heterogeneous clusters are promising solutions to meet the challenge, combining the flexibility of DSP-enhanced cores with the performance and energy boost of dedicated accelerators. We present DARKSIDE, a System-on-Chip with a heterogeneous cluster of eight RISC-V cores enhanced with 2-b to 32-b mixed-precision integer arithmetic. To boost the performance and efficiency on key compute-intensive DNN kernels, the cluster is enriched with three digital accelerators: 1) a specialized engine for low-data-reuse depthwise convolution kernels (up to 30 MAC/cycle); 2) a minimal overhead datamover to marshal 1–32-b data on-the-fly; and 3) a 16-b floating-point tensor product engine (TPE) for tiled matrix-multiplication acceleration. DARKSIDE is implemented in 65-nm CMOS technology. The cluster achieves a peak integer performance of 65 GOPS and a peak efficiency of 835 GOPS/W when working on 2-b integer DNN kernels. When targeting floating-point tensor operations, the TPE provides up to 18.2 GFLOPS of performance or 300 GFLOPS/W of efficiency—enough to enable on-chip floating-point training at competitive speed coupled with ultralow power quantized inference.

INDEX TERMS Heterogeneous cluster, tensor product engine (TPE), ultralow-power AI.

I. INTRODUCTION

THE RECENT mega-trend aiming at deploying machine learning (ML) and deep learning (DL) at the extreme edge of the Internet of Things (IoT), usually referred to as tiny machine learning (*TinyML*), has reached outstanding results. For example, MobileNets [1] have rapidly become state-of-the-art (SoA) compute workloads used for classification and object detection inference tasks but also as a flexible template for tasks not related to vision [2], [3], [4].

Next-generation TinyML IoT devices, however, will likely require also the capability to adapt the deployed DL model to new data directly in the field. Retraining the model on data centers with data collected on-field from the distributed IoT end nodes might be expensive in terms of latency and power and inconvenient from the privacy and security viewpoints. Therefore, a common direction of TinyML is to rethink the deployed deep neural network (DNN) as a dynamic model that can adapt by learning from newly sensed data directly on the device. Recent progress in this research area concerns DNN model tuning, partial on-chip training [5] or unsupervised continual learning [6], which have been applied successfully to many IoT applications, such as anomaly detection tasks [7].

Satisfying both the needs of TinyML inference and on-device adaptation requires devices that are highly flexible and efficient simultaneously on these two very different

tasks. Inference in TinyML devices typically adopts low-bitwidth integer arithmetic, relying on well-established quantization-aware training [8] and post-training quantization techniques [9]. Mixed-precision approaches [10], [11], where the activations and weights of all DNN layers can be quantized with different precisions, are State-of-the-Art (SoA) solutions to reduce the accuracy drop compared to full-precision models (e.g., within a 3%–6% range in ImageNet Top-1), while cutting the model footprint by a significant factor (∼7× on MobileNets [10]).

Specialized digital accelerators like [12], [13], [14], [15] achieve outstanding performance (1–50 TOPS/W) and energy efficiency (10–100 TOPS/W) on DNN kernels by exploiting low-bitwidth integer arithmetic. Recently, this approach has also been adopted in analog-digital mixed-signal solutions [16], [17], boosting energy efficiency up to hundreds of TOPS/W. However, these hardware units are highly specialized in terms of supported functionality and numerical precision and leak the flexibility needed to adapt to rapidly evolving TinyML models.

A different solution, exploiting clusters of parallel fully programmable architectures, would ensure the highest flexibility while still achieving competitive efficiency by leveraging instruction extensions supporting multiple formats to cover multiple data precision combinations in arithmetic instructions. Garofalo *et al.* [18] proposed parallel RISC-V cores with SIMD sum-of-dot-product instructions and custom mac-load operations to achieve ASIC-like efficiency on symmetric DNN convolutions. To reduce the overhead of instruction decoding for multiple-precision combinations, Ottavi *et al.* [19] proposed lightweight status-based mixed-precision computing support to a RISC-V processor, showing two orders of magnitude better efficiency than existing commercial microcontroller solutions.

Supporting multiple, mixed-precision computation is not the only flexibility challenge. Unlike the previous generation of TinyML DNN models, SoA MobileNets and derived networks feature more heterogeneous workloads, with standard convolutions combined with pointwise and depthwise kernels. Although they have less computation complexity and smaller memory footprint, depthwise layers are characterized by low intrinsic data reuse [20]. For this reason, they are harder to accelerate with massive arrays of processing elements. As a result, in the DNN processing pipeline, Amdahl's effect moves the acceleration bottleneck toward depthwise kernels. Likewise, data marshalling operations (e.g., low-bitwidth transpose) commonly used in DNNs heavily rely on sub-byte swap operations, which also contribute to reducing the utilization of the arithmetic units.

Introducing on-device training to the picture imposes yet different constraints in terms of performance and footprint, as training has stricter requirements on the data representation: integer arithmetic can not be used due to its limited dynamic range. To develop extreme-edge novel learning algorithms, a decisive effort is underway to adapt learning algorithms to lower precision like floating point (FP)16 and FP8 [21], [22].

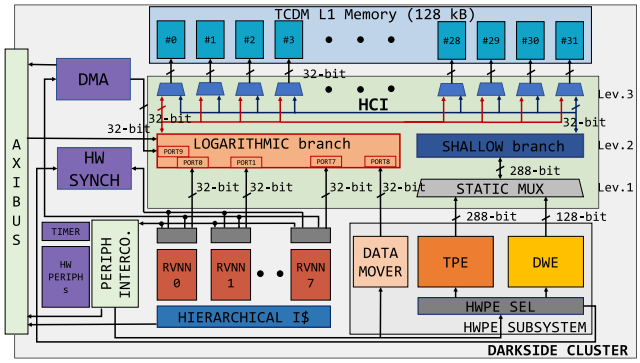


FIGURE 1. Overview of the DARKSIDE cluster architecture, featuring eight RVNN cores, TPE, DWE, and DataMover accelerators.

Despite this, *TinyML* on-chip training workload is still 10–100× larger than inference [5], and the performance requirements remain very high. Accelerating these workloads with general-purpose processors would require massive cores, blowing up the SoC's area and power consumption unacceptably. Hence, fixed-function custom designs still are the most suitable solutions to deliver significantly high performance within a TinyML compatible area and power budget.

We argue that a single *catch-it-all* solution is infeasible with all these competing requirements. Instead, boosting end-to-end AI-enhanced applications will require *heterogeneous systems* combining different acceleration engines for different kernels, coping with strict power and cost constraints [23]: multiple programmable cores provide flexible and efficient execution for generic parallel kernels, while specialized hardware accelerators provide extra performance and efficiency boost on essential kernels that dominate the computational workload.

In this work, we present DARKSIDE, a parallel ultra-low-power (PULP)-based [23] heterogeneous computing system on chip (SoC) that targets emerging TinyML inference and on-chip training applications. We introduce four main innovations in DARKSIDE: 1) RISC-V cores with advanced low-bitwidth mixed-precision integer computing capabilities; 2) a depthwise convolution engine (DWE); 3) a low-overhead DataMover for marshalling operations; and 4) a low-power tensor product engine (TPE) for efficient FP16 matrix multiplications (MatMuls). The cores and accelerators are tightly integrated into a shared-L1 cluster to enable advanced hardware/software cooperation. The chip has been fabricated in TSMC 65-nm technology and achieves peak integer performance (2 bit) of 65 GOPS at 1.2 V with an efficiency of 835 GOPS/W at 0.75 V. On TPE-accelerated FP16 workloads, it achieves up to 18.2 GFLOPS at 1.2 V and a peak efficiency of 300 GFLOPS/W and 2.6 GFLOPS at 0.75 V, achieving peak performance and efficiency similar to 8-bit integer operations.

II. DARKSIDE SOC ARCHITECTURE

Fig. 1 shows the architecture of the DARKSIDE cluster. It is built around eight 81-kGE RISC-V-based 32-bit processors

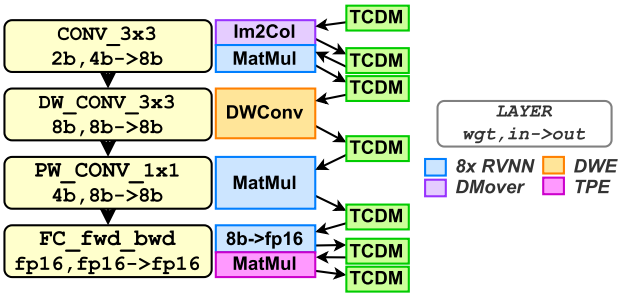


FIGURE 2. DARKSIDE heterogeneous operation. Figure shows a DNN with three mixed-precision quantized layers and a final fully connected layer using all the cluster blocks and communicating through software-managed buffers allocated on the shared L1 memory.

(*RVNN* cores), described in detail in Section II-B, and three specialized digital accelerators: 1) the *TPE*; 2) the *DWE*; and 3) the *DataMover*. The heterogeneous cluster can be used to support complex ML models, such as those depicted in Fig. 2, through cooperation among its hardware compute units.

To achieve high computing efficiency on a wide range of workloads, the key goal is to minimize the area and power impact of the specialized accelerators integrated into DARKSIDE's cluster and improve their efficiency on data movements.

To save area, we design the three accelerators to have small internal buffers, the minimal necessary to guarantee datapath utilization rate close to 100%, while they use the 128-kB scratch-pad multibanked L1 tightly coupled data memory (TCDM) of the cluster as the primary data buffer. Moreover, to minimize their power consumption, especially when the accelerators are not used, we added clock gating cells and operand isolation gates. This strategy cut the dynamic power consumption of the *idle* accelerators with minimal additional logic. To improve the performance and energy efficiency of the accelerators in data movement operations, and to east their integration into the cluster, each of the three accelerators is incorporated as a hardware processing engine (HWPE), using a standardized interface.[1] Such an interface exposes to the rest of the cluster a wide data transfer port (typically much wider than 32-bit) to optimize the access to the primary data buffer and a control port that allows the RISC-V cores to program the accelerator through memory-mapped control registers, as visible in Fig. 1. In each HWPE, specialized internal *Streamers* move data between the accelerators and L1 TCDM memory through the data port, converting the memory accesses into data streams to feed the accelerator's datapath.

The TCDM is divided into 32 4-kB SRAM banks, capable of serving 32 requests in parallel and it is shared among the three specialized accelerators and the eight RVNN general-purpose cores. The memory requests of all the cluster's compute units are routed through a one-cycle latency hierarchical heterogeneous cluster interconnect (HCI) (see Section II-A), which leverages a request/grant protocol and a world-level interleaving scheme to evenly distribute the requests, minimizing the access contentions toward the SRAM banks.

The cluster also features a two-level hierarchical instruction cache (I$), implemented with latch-based SCM to improve the energy efficiency over energy-expensive SRAM cuts. It includes eight 512-B private per-core plus 4 kB of the two-cycle shared cache to maximize the efficiency with the data-parallel code. A dedicated DMA controller, featuring a similar size as the cores (∼84 kGE), efficiently manages the data transfer between the L2 (off-the-cluster) and L1 memory. The DMA supports 2-D data transfers and up to 16 outstanding transactions, hiding the latency of L2–L1 data transfers on data-intensive kernels [24], while saving energy compared to cached-based systems. The cluster integrates also a small Hardware Synchronization Unit (∼30 kGE) which manages fine-grained parallel thread dispatching and clock-gating of idle cores waiting for synchronization, enabling low-overhead and fine-grained parallelism, thus high energy efficiency. The cluster resides in a dedicated power and clock domain. It is surrounded by other IPs integrated into a different power and clock domain, namely, the *Fabric* domain. The latter includes a controlling RISC-V processor, FLLs for clock generation, a standard set of I/O peripherals, and 256 kB of L2 memory containing the code executed by both the compute cluster and the controlling RISC-V core. In the context of this work, the *Fabric* domain serves as a programmable testbench for the cluster, which is the main architectural contribution of this work. The communication bus between the *Fabric* and the cluster domain is AXI4-based, and dual clock first-in–first-out (FIFO) buffers are used for clock domain crossing.

A. HETEROGENEOUS CLUSTER INTERCONNECT

To reduce the area and simplify the arbitration scheme, the HCI is organized hierarchically in three different levels. At the first level, the TPE and DWE are *statically multiplexed* to share the same physical HWPE 288-bit data port, which is sized to meet the bandwidth requirements of the two accelerators. Since in our computing model, reported in Fig. 2, the DWE and TPE are never used concurrently, the static multiplexing strategy is not a concern from a performance perspective. On the contrary, it allows exposing the accelerators' data interface toward the higher levels of the HCI with limited area and power costs.

The second level of the HCI is organized in two branches: 1) *logarithmic* and 2) *shallow*, as shown in Fig. 1: the cores, the cluster DMA, and the DataMover access the L1 banks from the *logarithmic* branch through 9 32-bit initiator ports. This branch allows all-to-all single-cycle access from the initiator ports to each word interleaved memory bank. Conflicts are handled by granting one initiator per bank at a time through a round-robin scheme. Instead, the

1. https://hwpe-doc.readthedocs.io

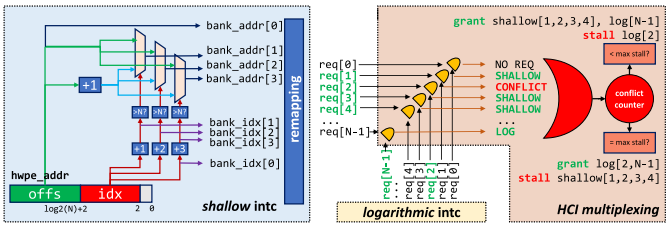


FIGURE 3. Simplified example of HCI *shallow* routing and arbitration between *shallow* and *logarithmic* branches, considering *N* TCDM banks. The example shows a *shallow* 128-bit (four ports) wide access starting on bank 1 and two 32-bit accesses from the *logarithmic* side.

288-bit muxed data port is connected to the dedicated *shallow* branch, routed to nine adjacent memory banks without arbitration. Considering a total of N TCDM banks, routing works by splitting the address of the 288-bit wide word in an *index* [bits 2 through $\log_2(N)+2$] and an *offset* part (upper bits). The index is used to select which TCDM banks are targeted, while the offset is used to compute the bank level address, considering the possibility that the wide word "rolls over" the set of banks (if the index corresponds to one of the last banks).

The third level of the HCI is at the memory side, where the TCDM banks are connected to the two HCI branches via multiplexers, granting access to one branch or the other according to a configurable-latency starvation-free rotation scheme. Ports from the logarithmic branch are stalled individually, whereas those from the shallow branch are stalled collectively (a single collision will result in no grant for the whole branch) to reflect the fact that they are actually a single access. Priority is given to a branch configurable via a memory-mapped register, and switched for one access after a configurable number of cycles. Fig. 3 showcases this mechanism in an example.

The heterogeneous organization of the interconnect serves two purposes. On the one hand, the HCI can be configured in software (by writing a memory-mapped register) to prioritize either the *shallow* or the *logarithmic* branch and guarantee a minimum quality of service (in terms of consecutive stall cycles) to the nonpriority branch (by setting a register with the maximum number of stalls that the less priority branch can tolerate). This enables to control and tune the interconnect's performance at a fine granularity. For example, setting priority to the *shallow* branch and maximum stall of ten cycles in the *logarithmic* branch means that after ten collisions the priority will be switched to *logaritmic* side for one cycle, hence guaranteeing a 9.1% collision rate, delivering up to 20.9 GB/s at 290 MHz in the configuration used in DARKSIDE (i.e., 288-bit wide shallow branch and nine 32-bit initiator ports in the logarithmic branch), even on data-intensive kernels.

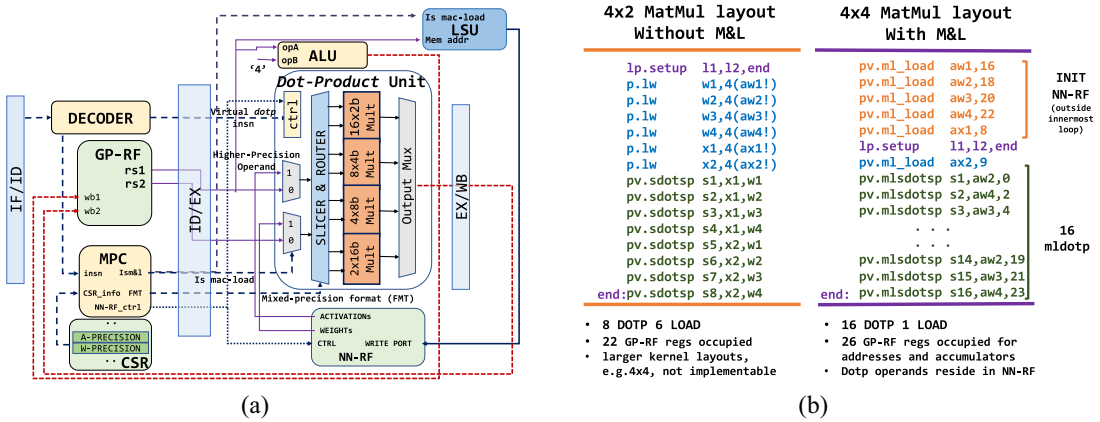
On the other hand, the scalability of the *logarithmic* interconnect is limited: attaching the accelerators to a non-hierarchical interconnect would result in a much more complex, larger, and power-hungry interconnect circuit, leading to poor cluster-level performance-per-area and per power. The HCI occupies 7.3% (∼220 kGE) of the total cluster area; our synthesis trials have shown that the overall complexity of the interconnect is reduced by 15% with respect to a purely logarithmic interconnect, which combined with easier timing closure and extended functionality led to the choice of this design.

B. DYNAMIC BIT-SCALABLE FUSED MAC-LOAD SIMD OPERATIONS

The DARKSIDE cluster's core, namely, RVNN, is a 4-stage in-order single-issue pipeline, depicted in Fig. 4(a), that implements the RV32IMCF RISC-V instruction set architecture (ISA), plus custom mixed-precision SIMD instructions operating on vector elements with power-of-two precision formats from 2 to 32 bit and all their possible permutations, supported through a dynamic bit-scalable execution [19]: the instruction encoded into the ISA identifies only the type of SIMD operations to be performed (denoted as *Virtual Instruction*), while its format (i.e., the precision of the operands) is specified at runtime by reading the content of a specific control&status register (CSR) of the core, which is writable by the programmer to set the desired precision, including mixed-formats. The SIMD instructions include *dot-product (dotp)*-based operations relevant to speed-up low-bitwidth compute-intensive kernels like matrix–matrix and matrix–vector multiplications.

The microarchitecture of RVNN is built on the baseline of the RI5CY [25] core and is reported in Fig. 4(a) we extend the ALU and the *Dot-product* Unit to process 2 and 4-bit SIMD operations which are not supported by RI5CY, we add extra CSR registers to store the instruction formats' information and we integrate the mixed-precision controller (MPC) into the ID-STAGE of the pipeline. When a mixed-precision *dotp* SIMD operation is performed, the decoder issues the *Virtual Instruction* to select the specific compute unit to be used in the EX-STAGE of the pipeline, and the format of the operands is specified by the CSR, while other control signals required for the execution are provided by the MPC. The *Dot-Product* Unit, as shown in Fig. 4(a), is preceeded by a *Slicer and Router* network, controlled by the MPC, which slices the registers according to the format (FMT) specified by the MPC; it selects the subportion of the vector RS2 to be used in the current operation and sign-extends (or zero-extends) it to match the size of the vector in RS1; afterward, the network routes the operands to the appropriate set of multipliers. To minimize the logic necessary to implement the new extensions, the first operand of the mixed-precision operations (RS1) is designated to be always the highest precision operand, without loss of generality given the commutative property of add and multiply operators. The extended pipeline entails 17% area overhead and 3% power overhead compared to RI5CY, but it improves the performance on sub-byte and mixed-precision kernels by a significant factor (up to 7.7×).

The key enhancement of RVNN is a fused MAC-load (M&L) operation that applies to any mixed-precision SIMD



(a) (b)

FIGURE 4. **(a) Pipeline extension to the RI5CY core to support mixed-precision and M&L instructions. (b) Example of a *MatMul* kernel using M&L instruction, compared to the same kernel implemented without M&L instruction. Thanks to the M&L operating on the dedicated NN-RF, we can implement larger layouts of *MatMul* kernels (right-sided) with a significant gain in terms of throughput.**

dotp instruction supported. The design of the M&L collapses the SIMD MAC and the load operations into a single one-cycle latency instruction since the datapath activated by the MAC operation would not interfere with the Load-Store Unit of the processor, and the two units can run in parallel. Fig. 4(a) shows the microarchitectural modifications to the cores' datapath to enable the M&L. When the M&L is executed, the two operands for the *dotp* operation are fetched from a dedicated register file, namely, the neural network register file (NN-RF), and routed to the *Dotp*-Unit through a multiplexer controlled by the MPC. At the same time, the accumulators reside in the GP-RF. The NN-RF consists of six 32-bit registers and is sized to maximize the innermost loop's performance of the PULP-NN [26] convolution routines, dedicating four out of six registers to layer's weights and two out of six registers to input activations. As visible in Fig. 4(a), this choice constraints the activations of the convolution layers always to feature higher precision than the weights in mixed-precision operations, which however is the common case in current SoA software solutions to deploy DNN models at the extreme-edge of the IoT [27].

Since the M&L operates on the NN-RF, the occupancy of the 32 32-bit registers of the general-purpose register file (GP-RF) is reduced by a significant factor, since it would only host the accumulators of the *dotp* operations and the addresses for the memory accesses. As a consequence, we can implement compute kernels with a higher amount of data reuse without incurring overheads to move data back and forth from the stack in the innermost hot loop (Fig. 4, right-sided kernel). This solution guarantees up to 1.7× performance improvements over the execution of the same kernel without M&L, with an extra area overhead of only 5%, necessary to integrate the NN-RF in the EX-STAGE of the core pipeline. When an M&L instruction is executed, one of the two source operands from the neural network register file (NN-RF) can eventually be updated with new data fetched from memory by the LSU, extended to operate on the NN-RF with negligible area overhead. However, the data stored in the NN-RF registers can be kept until necessary to allow a higher degree of flexibility for data reuse strategies: a second M&L instruction encoded into the ISA performs only the *dotp* branch, with no register update.

From an instruction count perspective, the M&L brings significant advantages, as shown in Fig. 4(b). After out-of-the-loop initialization of the dedicated Neural NN-RF, we perform 16 SIMD *dotp*-like operations and only a single explicit load (with no concurrent MAC) instruction. Therefore, we reduce the number of pure load instructions in the innermost loop of the kernel by a factor of 6, at the same time doubling the throughput, with an overall dot-product/cycle improvement of 57% compared to the same core not featuring the M&L. On the contrary, the impact of the M&L on the performance, power, and area (PPA) metrics of the RVNN core is minimal. Overall, the M&L implies a gate count increase of just 8.3%, without deteriorating the critical path of the core and with negligible power overhead. On the other hand, the core enhanced with the M&L achieves up to 94% dot-product unit utilization, compared with 58% of the RI5CY baseline.

C. TENSOR PRODUCT ENGINE

The TPE [28] accelerates MatMuls of the kind $Z = X \cdot W$. It is designed to use the IEEE 754 binary-16 representation (FP16 in the following) since it is understood that FP16 can be used to train Neural Networks without significant accuracy loss, but reducing the power consumption and time to computation [29]. Fig. 5 shows the implementation of the TPE. The datapath consists of 32 FP16 fused multiply-add (FMA) units [30] organized in eight rows, each of four columns. This configuration guarantees at the same time a good speed-up (from 9× up to 22×) with respect to the software parallel execution, and an area overhead bounded to 44.8% of the area occupied by all the RV-NN cores. The FMAs along each row are cascaded; each FMA passes the intermediate result as input to the unit to its right.

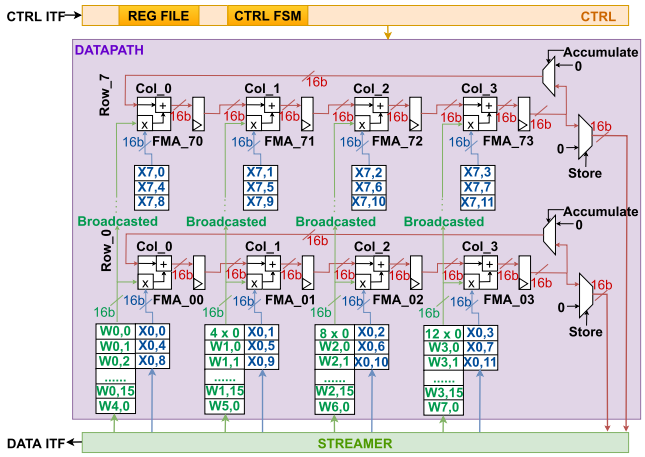


FIGURE 5. Architecture of the TPE, with a focus on the interconnection of the FMAs within the datapath.

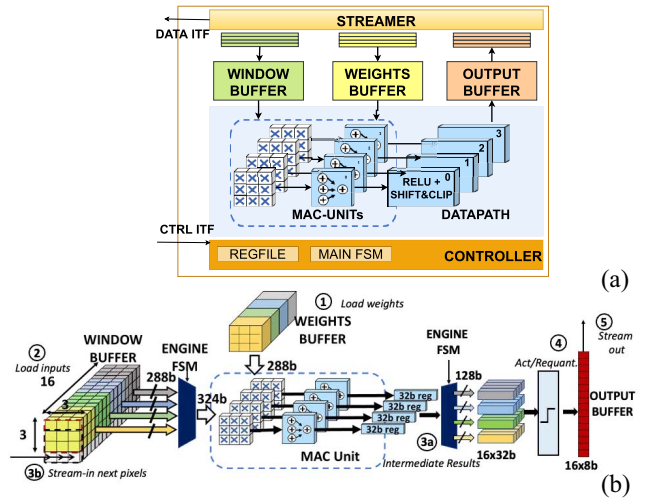


FIGURE 6. (a) Architecture overview of the depthwise digital accelerator, enclosed in the HWPE. (b) Execution flow of the depthwise operation.

To internally handle the computation of matrices larger than the array size, and to avoid intermediate store operations, the FMAs in each row are closed in a feedback loop so that the right-most FMA feeds back the computed partial product as accumulation input to the left-most FMA of the same row. Using this approach, the TPE can exploit the maximum reuse of both the **X**-matrix elements and the intermediate product, so that it stores the computed sub-blocks of the **Z**-matrix to the memory only at the very end of their computation. To match the critical path of the cores, each FMA features three internal pipeline registers. To maximize throughput, the **X**-matrix elements of each FMA are held steady for the number of cycles necessary to the FMAs of each row to compute the partial results. On the other hand, **W**-matrix operands are streamed-in at each cycle and broadcasted to all the FMAs of the same column. The memory accesses are scheduled so the load and store phases do not introduce overhead during the computation. This way, the TPE can reach an overall 98.8% utilization of the internal FMAs with a near-to-ideal performance (31.6 out of 32 MAC/cycle). The computed sub-blocks of the **Z**-matrix are stored in the memory only at the end of their computation, maximizing internal data reuse.

The TPE, as well as the other accelerators integrated in DARKSIDE, features a nonblocking event-based execution mode: the cores of the cluster, after programming the accelerator and starting its execution, can either go in sleep mode or resume software code execution. This mechanism enables complex execution models where the accelerator can be used in parallel with the general-purpose cores to boost the performance of the target kernel. This scenario is made possible also thanks to the dynamic arbitration mechanism provided by the HCI, which allows the requests to the memory by the TPE and the cores to be served simultaneously if there are no bank conflicts.

D. DEPTHWISE CONVOLUTION ENGINE

The DWE can process the low-reuse depthwise component of the depthwise + pointwise kernels often used in recent neural network models for mobile applications, leaving the much better-parallelizable pointwise kernels to the M&L-accelerated software. The DWE processes 8-bit signed input and weight tensors stored in L1 memory using the height–width-channel (HWC) layout, the same used by the cores to execute the pointwise kernels, therefore requiring no time-consuming intermediate on-the-fly marshalling operations. The 8-bit output tensors are generated after applying requantization steps.

The architecture is shown in Fig. 6(a). It employs a weight-stationary data flow to maximize the data reuse and targets 3×3 depthwise layers, the one most commonly encountered in DNNs. Although its datapath is optimized for 3×3 depthwise convolutions, the DWE can be used to run kernels with different sizes using the same approach presented in [31], at the cost of additional data manipulation on the intermediate results, hence less computation efficiency.

The execution flow is depicted in Fig. 6(b). The weights from 16 3×3 filters are loaded into the *Weights Buffer* (1), before the execution starts. The weights are kept in the buffer until they have been used to scan the whole input tensor. The input image is filtered through a vertical sliding window on the spatial dimensions, using a *window buffer* of $4 \times 3 \times 16$ registers. The first three rows are loaded at the beginning of the iteration (2) and consumed in four cycles by the datapath of the DWE consisting of 36 MAC units. The intermediate results are accumulated over 16 32-bit buffers, accessed 4 at a time in the 4-cycle operation loop (3a). Afterward, nonlinear activation functions and ancillary operations, such as shifting and clipping are applied to requantize the results to 8-bit precision (4). After four cycles of operation, the 16-channel 8-bit pixels stored in the *output buffer* are streamed out of the accelerator (5). Meanwhile, the streamer uses three cycles (overlapped with the computation) to fill the fourth row of the *window buffer* (3b), needed to implement the sliding window mechanism.

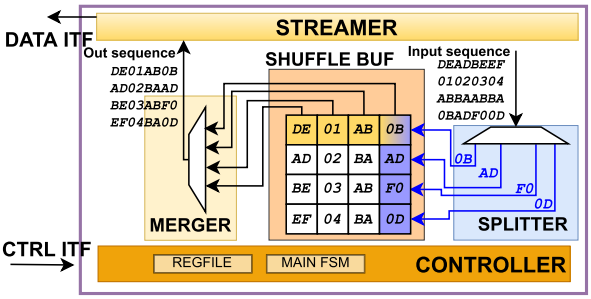
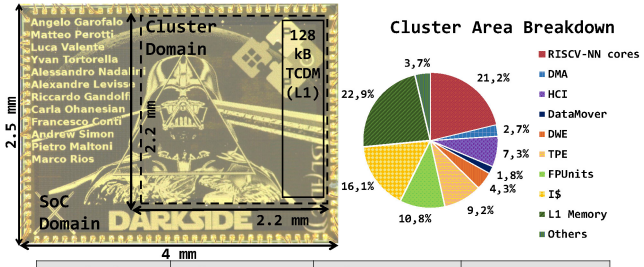


FIGURE 7. Overview of the DataMover architecture and execution flow.



Technology	Cluster Area	Cluster SRAM	Frequency Range
CMOS 65nm	4.84 mm²	128 kB	40 – 290 MHz
Chip Area	**Total SRAM**	**Vdd Range**	**Power Envelope**
12 mm²	384 kB	0.75 – 1.2 V	213 mW

FIGURE 8. Chip micrograph and specifications. Area breakdown of the cluster.

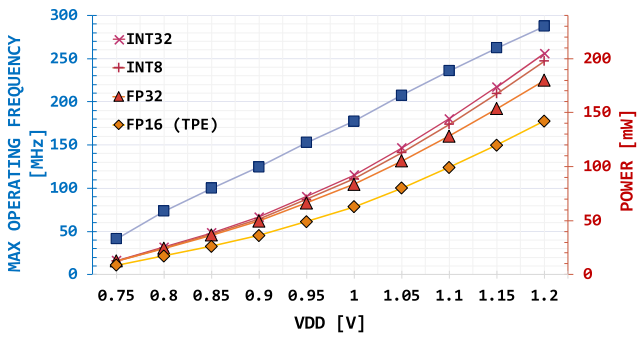


FIGURE 9. Voltage sweep versus max frequency versus power consumption.

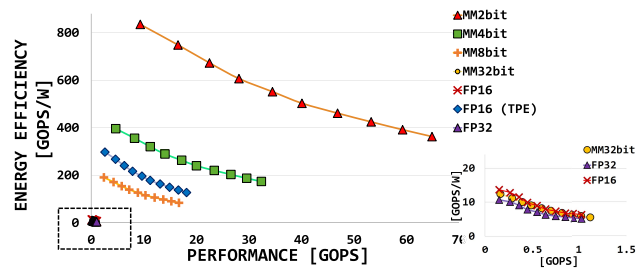


FIGURE 10. Performance and energy efficiency of the DARKSIDE cluster. The measurements are performed at the max frequency, sweeping the supply voltage between 0.75 and 1.2 V.

The DWE is designed to keep the datapath always active, in all stages and to fully exploit the memory bandwidth of 36 B per cycle available on the cluster, achieving the overall performance of 30 MAC/cycle, more than 10× better than a software execution of the depthwise kernels, on the eight RVNN cores.

E. DATAMOVER

On-the-fly data marshalling operations can dramatically reduce the performance of DNN workloads, during both inference and training tasks. To perform efficiently on-the-fly data transposition, the DARKSIDE's cluster is enhanced with a DataMover unit, exposed to the HCI with an additional master port on the *logarithmic* branch. The architecture is depicted in Fig. 7. It consists of a tiny accelerator of only 54 kGE, capable of transposing DNN 3-D tensors stored in the L1 memory, with 1.5–100× less time than eight RVNN cores and increased energy efficiency up to 50× (the lower the precision of chunks to transpose the more significant the advantages).

The accelerator works on data with configurable precision, d, in the range from 32 bit down to 1 bit. It splits incoming data streams from memory into chunks of size d, internally buffered into the *Shuffle Buffer* which features 32× 32-bit register with the transposed output format; only $32/d$ of them are used, depending on the data size configuration d. After $32/d$ input stream transactions, the transposed words are streamed out to the L1 memory and the accelerator continues the operations with new input streams.

III. CHIP IMPLEMENTATION AND MEASUREMENTS

Fig. 8 shows the chip micrograph of DARKSIDE. The SoC is implemented with TSMC 65-nm technology, targeting a clock frequency of 250 MHz in worst case operating conditions. The die area, including the *Fabric* domain, is 12 mm^2, while the cluster area is 4.84 mm^2, partitioned as shown in Fig. 8. The majority of the cluster's area is occupied by cores, hierarchical I$ and 128 kB of L1 memory, while the accelerators account only for 15.3% of the total area.

The measurements of the DARKSIDE's cluster are performed using an Advantest SoC hp9300 integrated circuit testing device, which precisely regulates the supply voltages delivered to the SoC and allows accurate current measurements of the SoC's cluster power domain. Fig. 9 reports the maximum operating frequency and the power consumption of the cluster over the 0.75–1.2-V voltage range. The operating frequency increases linearly with the supply voltage up to 290 MHz at 1.2 V. The power is measured on the silicon prototype, running integer and FP compute-intensive kernels (MatMuls). Offloading FP16 MatMuls on the TPE saves 30% of the power compared to the execution on the eight cores.

Fig. 10 shows the cluster's performance and efficiency on integer and FP MatMuls, sweeping all the main supported data formats. The measurements are taken at the maximum operating frequency, sweeping the supply voltage from 0.75 to 1.2 V. On ML workloads (i.e., integer 8/4/2 bits) deployed on RV-NN cores, DARKSIDE delivers up to 65 GOPS with a peak efficiency of 835 GOPS/W. The TPE boosts the performance and the efficiency of FP16 MatMuls

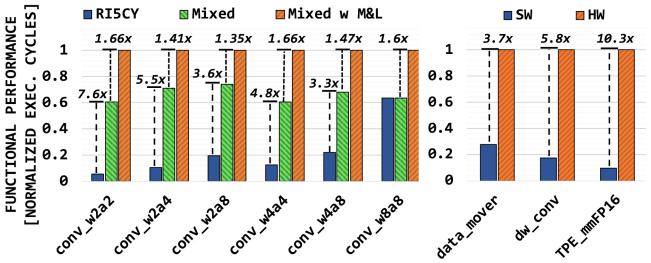


FIGURE 11. Left: Normalized single-core performance of mixed-precision convolutions. RVNN core (*Mixed w M&L*) is compared against a core featuring only mixed-precision *dotp* operations but no M&L (*Mixed*) and against *RI5CY*, which features no extensions for quantized neural networks. Right: performance improvement of execution on cluster-coupled accelerators over software (eight cores).

to 18.2 GFLOPS and 300 GFLOPS/W, respectively, 17.7× and 21.8× higher compared to a software scalar execution on the four FPUs of the DARKSIDE cluster.

IV. BENCHMARKING

To demonstrate the capabilities of DARKSIDE on SoA DNN workloads, we benchmark mixed-precision convolution kernels, end-to-end inference of the MobileNetV2, and one TinyML training use-case.

A. SINGLE DNN KERNELS

To highlight the features of the RVNN cores, we show improvements over the baseline, benchmarking several convolution kernels. To measure the computing performance, the layers operate on data stored in the L1 memory, with 64 $3 \times 3 \times 32$ filters applied on a $32 \times 16 \times 16$ input feature map, spanning different integer data formats (from 8 down to 2 bit) for the inputs and weights, including mixed-precision cases. Results are reported in Fig. 11 in terms of normalized execution cycles (with respect to the RVNN core with M&L). As shown, the M&L instruction improves the performance by up to 1.7× compared the execution with baseline mixed-precision SIMD *dotp* and load instructions (*Mixed* in the figure). Overall, ISA and microarchitecture design of RVNN leads to a cumulative performance improvement of up to 13× with respect to RI5CY, which supports only 8-bit SIMD operations and no M&L mechanisms.

Analyzing the depthwise kernels, in Fig. 11, we show that this workload achieves at least 5.8× better performance by offloading it to the dedicated digital accelerator presented in this work, the depthwise engine (DWE), instead of running it on eight RVNN cores. This conclusion is also strengthened in Section IV-B on a real-life *Bottleneck* layer use-case. Furthermore, we show that the DataMover can reduce by more than 3.7× execution cycles on 8-bit data marshalling operations, compared to the eight cores.

On 16-bit floating-point (FP16) MatMul workloads, the TPE boosts the performance by up to 10.3× with respect to the software execution of the same kernels exploiting the FP16 SIMD instructions available on the 4 FP units (FPUs) present in the cluster [30].

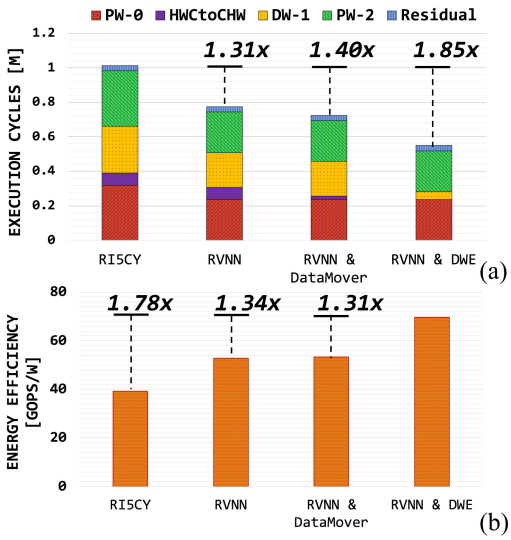


FIGURE 12. Comparison of the *Bottleneck* layer execution, offloaded to different hardware compute units. (a) reports the execution latency (in terms of compute cycles) and (b) reports the energy efficiency (in GOPS/W). The cluster is running at the best performance operating point: f_{CLU} = 290 MHz and V_{DD} = 1.2 V.

B. END-TO-END MOBILENETV2

First, we present the results of benchmarking the *Bottleneck* layer, the core building block of the MobileNetV2. We demonstrate our improvements incrementally by comparing our architectural solutions over a reference cluster that features eight RI5CY cores (without the mixed-precision SIMD operations and the M&L custom ISA extensions proposed in this work) and no dedicated accelerator. To implement the software to execute the *Bottleneck*, we use the PULP-NN library (which we use as-is to benchmark the reference cluster), extended to include additional kernels to exploit the new ISA instructions implemented in the RV-NN cores and a set of hardware-abstraction-layer (HAL) functions to program and start the accelerators that the programmer can easily insert into the C code. We adopt the 8-bit signed integer representation for all the tensors of the *Bottleneck*.

The results, in terms of execution cycles and energy efficiency, are reported in Fig. 12. The M&L improves the execution of pointwise and depthwise layers on eight RVNN cores by 1.31× compared to the execution on eight RI5CY cores. Additional 1.13× improvements are given by the data transposition (i.e., HWC to CHW data marshalling) performed by the DataMover, instead of transposing data via software. Finally, the DWE allows to speed-up the execution of depthwise convolution by 4.4× compared to the execution on eight RV-NN cores, with a final performance improvement of 1.85× on the whole *Bottleneck* layer compared to the RI5CY baseline.

To put the previous results in perspective, we benchmark DARKSIDE on the end-to-end inference task of the MobileNetV2 model. We employed the standard MobileNetV2 with depth multiplier 1.0 and input size 224×224, composed of 16 stacked *Bottleneck* layers. The

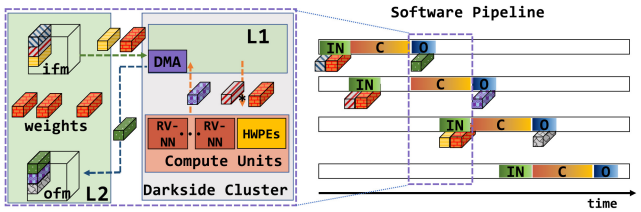


FIGURE 13. DNN tiling software pipeline. The figure shows the concurrent execution of weights and activations transfer (from L2 to L1 memory, indicated with *IN*), the computation of the kernel (indicated with *C*) and the copy of produced output tiles from L1 to L2 memory (indicated with *O*).

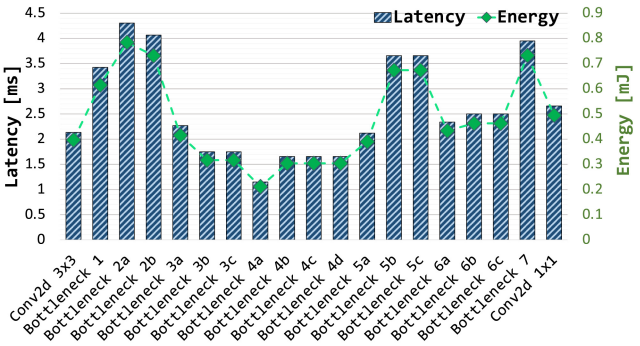


FIGURE 14. Layerwise execution latency and energy of the MobileNetV2 on DARKSIDE's cluster running at $f_{clk} = 290$ MHz and $V_{DD} = 1.2$ V.

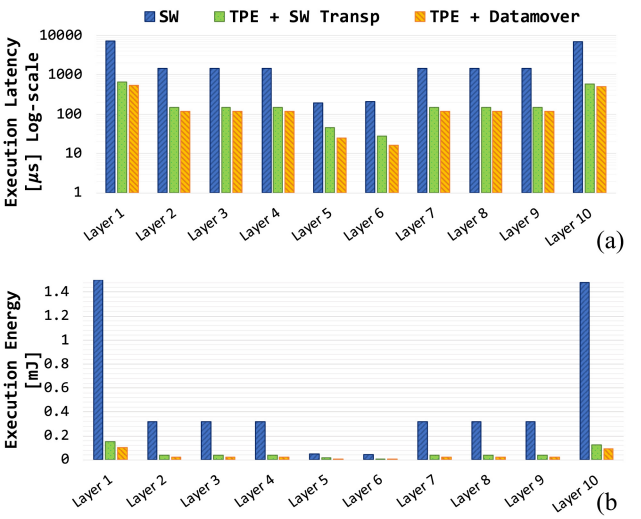


FIGURE 15. (a) Layerwise execution time of the AE TinyML model. (b) Execution Energy of the AE, including the energy for L2 to L1 (and vice-versa) data movements. Cluster running at 290 MHz and 1.2.

input and weight tensors feature 8-bit precision for all the depthwise and Conv2d layers. The pointwise layers feature 8-bit input tensors, while the weights are represented with reduced precision, i.e., 4 bits. This mixed-precision configuration of the model achieves similar Top-1 accuracy (only a 2.5% drop compared to the 8-bit version) while ensuring almost 2× less memory footprint (1.07 MB) for the network weights [11].

To enable the computation on the cluster, both weights and activations of the model must be divided into tiles that fit 128 kB of the L1 SRAM. Therefore, we assume the weights and feature maps for all the network layers to be stored in the off-the-cluster L2 memory, and we adopt the data and execution flow presented in Dory [24]. Dory is used to calculating the data tiling solutions fitting the L1 memory constraints and to schedule the data transfers from L2 to L1 and vice-versa, performed through the cluster DMA in double-buffering. The described software pipeline is represented in Fig. 13. For cases where the execution is not memory bound, data movements overlap with the computation, with negligible overhead (≤5%) to the execution latency.

However, since DARKSIDE's *Fabric* domain has the only purpose of acting as a programmable testbench for the cluster, it features a small L2 memory which is insufficient to host the entire MobileNetV2. Therefore, to benchmark the computing capabilities of the cluster on real-life end-to-end DNN models, we exploit our previous experience on explicit memory management, data tiling techniques [24], and on the deployment of real-sized DNN models on application chips such as Vega [23] to build a model of the system, with larger L2 memory, on which we run the experiments. The hardware-oriented description of the SoC is integrated into our open-source[2] event-based emulator, called GVSOC [32]; to run the experiments, the following measurements and considerations are taken.

1) We assume to have an L2 memory of 2 MB, necessary to host the entire MobileNetV2 model and to store the program code.
2) We analyze the traffic between L2 and L1 memories by running end-to-end simulations of the MobileNetV2 on the GVSOC; as expected, during the execution of the inference task, we are never memory bound; therefore, the contribution of the L2 to L1 (and vice-versa) data movements is relevant only for the total energy consumption.
3) We conduct silicon measurements, in terms of latency and energy, on all the L2 to L1 data transfers (and vice-versa) necessary to compute each tile and determined by the GVSOC simulations; we then include the measurements in the model.
4) We conduct silicon measurements, in terms of latency and the energy, on all the kernels necessary to compute each tile generated by the Dory framework; we then include the measurements in the model.

The layerwise compute time and energy of the inference task are shown in Fig. 14. DARKSIDE can perform the entire end-to-end task with a performance of more than 20 frame/s, with an energy budget of 11 mJ. The performance is 2× better than the one achieved on the Vega's cluster running at 250 MHz [23], thanks to our architectural contributions, the M&L extensions that accelerate the pointwise kernels

2. https://github.com/pulp-platform/gvsoc

TABLE 1. Comparison with SoA.

	SleepRunner [35]	*SamurAI [36]*	*VEGA [23]*	*Dustin [37]*	*This work*
Technology	CMOS 28nm FD-SOI	CMOS 28nm FD-SOI	CMOS 22nm FD-SOI	CMOS 65nm	CMOS 65nm
Die Area	0.68 mm2	4.5 mm2	12 mm2	10 mm2	12 mm2
Application	IoT GP	IoT GP + DNN	IoT GP + NSAA+DNN	IoT GP + DNN + QNNs	IoT GP + NSAA + DNN + QNNs
CPU	CM0DS	1x RI5CY	10 x RI5CY	16 x RI5CY MPIC CORES	8 x RI5CY-NN
ISA	Thumb-2 subset	RVC32IMFXpulp	RVC32IMFXpulp+S	RVC32IMCXmpic	RVC32IMFXpulpNN2
Int Precision (bits)	32	8, 16, 32	8, 16, 32	2, 4, 8, 16, 32 (plus Mixed-Precision)	2, 4, 8, 16, 32 (plus mixed-precision)
FP Precision	–	–	FP32, FP16, bfloat	–	FP32, FP16
Supply Voltage	0.4 - 0.8 V	0.45 - 0.9 V	0.5 – 0.8 V	0.8 - 1.2 V	0.75 - 1.2 V
Max Frequency	80 MHz	350 MHz	450 MHz	205 MHz	290 MHz
Power Envelope	320 µW	96 mW	49.4 mW	156 mW	213 mW
Best Integer Performance	31 MOPS (32b)	1.5 GOPS (8b)	15.6 GOPS (8b)	15 GOPS (8b) 30 GOPS (4b) 58 GOPS (2b)	17 GOPS (8b) 32 GOPS (4b) 65 GOPS (2b)
Best Integer Efficiency	97 MOPS/mW (32b)	230 GOPS/W (8b)	614 GOPS/W (8b)	303 GOPS/W (8b) 570 GOPS/W (4b) 1152 GOPS/W (2b)	191 GOPS/W (8b) 396 GOPS/W (4b) 835 GOPS/W (2b)
Best FP32 Performance	–	–	2 GFLOPS	–	1.03 GFLOPS
Best FP32 Efficiency	–	–	79 GFLOPS/W	–	12 GFLOPS/W
Best FP16 Performance	–	–	3.3 GFLOPS	–	18.2 GFLOPS
Best FP16 Efficiency	–	–	129 GFLOPS/W @ 1.27 GFLOPS	–	300 GFLOPS/W @ 2.6 GFLOPS

and the dedicated DWE that boosts the execution of depthwise convolutions. Despite Vega being implemented in the 22-nm technology node, our end-to-end energy consumption of 11 mJ remains still comparable, in the same order of magnitude.

C. TINYML ON-CHIP TRAINING

The TPE enhances the DARKSIDE cluster to support efficient FP matrix–matrix multiplications, enabling *de-facto* on-chip TinyML training workloads. To benchmark the SoC in terms of execution latency and energy on real-sized problems, we execute the autoencoder (AE) DNN model [7], commonly used within the TinyML scenario for unsupervised anomaly detection tasks. The TinyML AE consists of Encoder and Decoder layers (made by 128 unit Fully Connected layers with BatchNorm and ReLu activation functions) and a latent space layer of size 8. The input and output size is 640. We benchmark the whole training stage (forward and backward steps within one training *epoch*), adopting a batch size of 16, which is a reasonable tradeoff between performance and memory occupation for IoT multicore microcontroller-class devices. The tensors are represented with the FP16 format, and we adopt the same data flow explained above, which uses tilings and double-buffering.

To highlight the boost given by the TPE and DataMover, we first implement the AE on the eight general-purpose RVNN cores (we call this configuration *SW*), which share 4 FPUs supporting FP16 formats, using a software library optimized for on-chip training [33]. Then, we implement the AE offloading the MatMul workload to the TPE (*TPE* configuration), still performing on the cores control tasks (e.g., programming the DMA for double buffering, programming the TPE control units) and matrix transpositions. As a third execution mapping, we use the DataMover to speed-up also the matrix transpositions (*TPE + DataMover*).

The results are reported in Fig. 15 in terms of execution latency and energy consumption. As expected, the TPE delivers at least 10× speed-up with respect to a pure SW execution on all the layers of the AE except for the latent space layers, where the performance improvement is reduced to 4×–7× due to lower arithmetic intensity of those layers. The matrix transposition performed with the tiny DataMover accelerator contributes to an additional $1\times - 2\times$ of speed-up. Overall, combining the TPE and DataMover, the entire training epoch runs in 1.8 ms with an energy consumption of 345 μJ, 13 × faster than the *SW* execution of the AE on the eight RV-NN cores, with 14× lower energy consumption.

V. COMPARISON WITH THE STATE-OF-THE-ART

Table 1 compares DARKSIDE with a wide range of programmable embedded computing platforms that exploit either parallelism or heterogeneity to address the computing requirements of emerging TinyML applications.

Compared to a traditional low-power programmable IoT system such as [34], representative of a wide range of low-cost microcontrollers embedding CortexM0, DARKSIDE delivers several orders of magnitude better integer (8 bit) peak performance and also 1.9× better energy efficiency, despite *SleepRunner* [34] is implemented in a more scaled technology node (28-nm FD-SOI). Contrarily to DARKSIDE's cluster, the implementation strategy of SleepRunner is highly optimized to operate at very low voltage (i.e., down to 0.4 V). Its architecture features a simple memory hierarchy and interconnects scheme, which consumes very low power but poses severe limitations during the

execution of complex near-sensor data analytic applications, which are efficiently sustained on DARKSIDE.

With respect to hardware-accelerated IoT end nodes such as *SamurAI* [35], implemented in 28-nm FD-SOI technology, our SoC achieves similar energy efficiency on DNN workloads (only 1.2× less efficient despite the less scaled technology node used to implement Darkside, 65 nm) but with a significant gain of 10× in terms of peak performance. This gain is primarily due to the custom extensions of RV-NN cores and the parallel computing cluster over the sequential solution presented in [35].

Finally, we compare DARKSIDE with two SoCs that exploit a similar architectural template: Dustin [36] and Vega [23] implement a multicore RISC-V compute cluster in 65 and 22 nm, respectively. Compared to Vega [23], Darkside delivers a better performance on 8-bit integer workloads thanks to the M&L instruction. Contrarily to Vega, DARKSIDE can support also mixed and lower-precision (than 8 bit) integer workloads thanks to the enhanced mixed-precision ISA, enabling the computation of emerging DNN models that employ asymmetric quantization schemes [10]. On 32-bit FP workloads, Vega surpasses our solution in performance and energy efficiency due to the higher frequency operating mode and the much more scaled technology node. However, despite the previously mentioned advantages of Vega, the TPE of DARKSIDE ensures 2.32× better energy efficiency on FP16 workloads, with a considerable performance gain of up to 5.6×.

Compared to Dustin, featuring a cluster with 16 processors with mixed-precision extensions implemented in the same technology node, the proposed cluster shows slightly less energy efficiency due to the power reduction achieved by Dustin, thanks to the vector lockstep execution mode (VLEM).[3] However, DARKSIDE still achieves 1.13× better performance with half of the cores, thanks to the M&L extension.

VI. CONCLUSION

We presented DARKSIDE, a low-power heterogeneous compute cluster for *TinyML* DNN inference and on-chip training. The cluster features eight RISC-V cores, enhanced with 2–32-bit mixed-precision integer SIMD instructions and fused MAC-load operations. It also features specialized accelerators to boost the performance of integer depthwise convolutions, reduce the latency of data marshalling operations, and enhance the performance and efficiency of FP16 kernels. The proposed SoC, implemented in TSMC 65-nm technology, can achieve up to 65 GOPS peak performance on ML workloads, with 835 GOPS/W of energy efficiency. On FP16 kernels offloaded to the TPU, the SoC achieves 18.2 GFLOPS with 300 GFLOPS/W, surpassing the efficiency and performance of SoA SoCs implemented in much more scaled and expensive technology nodes.

3. The VLEM is not compatible with the cores unsed in Darkside (RVNN) and the two optimizations could be eventually combined to improve computing energy efficiency.

REFERENCES

[1] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetv2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4510–4520.

[2] H. Cai, L. Zhu, and S. Han, "Proxylessnas: Direct neural architecture search on target task and hardware," 2018, *arXiv:1812.00332*.

[3] A. Incze, H.-B. Jancsó, Z. Szilágyi, A. Farkas, and C. Sulyok, "Bird sound recognition using a convolutional neural network," in *Proc. IEEE 16th Int. Symp. Intell. Syst. Inf. (SISY)*, 2018, pp. 295–300.

[4] B. Zhang, Y. Zhang, and S. Wang, "A lightweight and discriminative model for remote sensing scene classification with multidilation pooling module," *IEEE J. Sel. Topics Appl. Earth Observ. in Remote Sens.*, vol. 12, no. 8, pp. 2636–2653, Aug. 2019.

[5] L. Ravaglia, M. Rusci, D. Nadalini, A. Capotondi, F. Conti, and L. Benini, "A TinyML platform for on-device continual learning with Quantized latent replays," *IEEE Trans. Emerg. Sel. Topics Circuits Syst.*, vol. 11, no. 4, pp. 789–802, Dec. 2021.

[6] H. Ren, D. Anicic, and T. A. Runkler, "Tinyol: Tinyml with online-learning on microcontrollers," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2021, pp. 1–8.

[7] C. Banbury *et al.*, "Mlperf tiny benchmark," 2021, *arXiv:2106.07597*.

[8] B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2704–2713.

[9] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 6869–6898, 2017.

[10] M. Rusci, M. Fariselli, A. Capotondi, and L. Benini, "Leveraging automated mixed-low-precision quantization for tiny edge microcontrollers," in *Proc. Int. Workshop IoT Edge Mobile Embedded Mach. Learn.*, Ghent, Belgium, 2020, pp. 296–308. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-66770-2_22

[11] C. J. Schaefer, S. Joshi, S. Li, and R. Blazquez, "Edge inference with fully differentiable quantized mixed precision neural networks," 2022, *arXiv:2206.07741*.

[12] Y.-H. Chen, T.-J. Yang, J. Emer, and V. Sze, "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 2, pp. 292–308, Jun. 2019.

[13] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "14.5 envision: A 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28nm FDSOI," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, 2017, pp. 246–247.

[14] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H.-J. Yoo, "UNPU: A 50.6 TOPS/W unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, 2018, pp. 218–220.

[15] G. Desoli *et al.*, "14.1 a 2.9 TOPS/W deep convolutional neural network SoC in FD-SOI 28nm for intelligent embedded systems," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, 2017, pp. 238–239.

[16] R. Khaddam-Aljameh *et al.*, "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 J. Solid-State Circuits*, vol. 57, no. 4, pp. 1027–1038, Apr. 2022.

[17] K. Ueyoshi *et al.*, "DIANA: An end-to-end energy-efficient digital and ANAlog hybrid neural network SoC," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, vol. 65, 2022, pp. 1–3.

[18] A. Garofalo, G. Tagliavini, F. Conti, L. Benini, and D. Rossi, "XpulpNN: Enabling energy efficient and flexible inference of Quantized neural networks on RISC-V based IoT end nodes," *IEEE Trans. Emerg. Topics Comput.*, vol. 9, no. 3, pp. 1489–1505, Jul.-Sep. 2021.

[19] G. Ottavi, A. Garofalo, G. Tagliavini, F. Conti, L. Benini, and D. Rossi, "A mixed-precision RISC-V processor for extreme-edge DNN inference," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, 2020, pp. 512–517.

[20] A. Garofalo *et al.*, "A heterogeneous in-memory computing cluster for flexible end-to-end inference of real-world deep neural networks," 2022, *arXiv:2201.01089*.

[21] A. Rodriguez *et al.*, "Lower numerical precision deep learning inference and training," *Intel White Paper*, vol. 3, pp. 1–19, Jan. 2018.

[22] X. Sun *et al.*, "Hybrid 8-bit floating point (HFP8) training and inference for deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 4901–4910.
[23] D. Rossi *et al.*, "Vega: A ten-core SoC for IoT Endnodes with DNN acceleration and cognitive wake-up from MRAM-based state-retentive sleep mode," *IEEE J. Solid-State Circuits*, vol. 57, no. 1, pp. 127–139, Jan. 2022.
[24] A. Burrello, A. Garofalo, N. Bruschi, G. Tagliavini, D. Rossi, and F. Conti, "Dory: Automatic end-to-end deployment of real-world dnns on low-cost IoT MCUs," *IEEE Trans. Comput.*, vol. 70, no. 8, pp. 1253–1268, Aug. 2021.
[25] M. Gautschi *et al.*, "Near-threshold RISC-V core with DSP extensions for scalable IoT endpoint devices," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 10, pp. 2700–2713, Oct. 2017.
[26] A. Garofalo, M. Rusci, F. Conti, D. Rossi, and L. Benini, "PULP-NN: Accelerating quantized neural networks on parallel ultra-low-power RISC-V processors," *Philosoph. Trans. Roy. Soc. A*, vol. 378, no. 2164, 2020, Art. no. 20190155.
[27] M. Rakka, M. E. Fouda, P. Khargonekar, and F. Kurdahi, "Mixed-precision neural networks: A survey," 2022, *arXiv:2208.06064*.
[28] Y. Tortorella, L. Bertaccini, D. Rossi, L. Benini, and F. Conti, "RedMulE: A compact FP16 matrix-multiplication accelerator for adaptive deep learning on RISC-V-based ultra-low-power SoCs," in *Proc. DATE*, 2022, pp. 1099–1102.
[29] "Training with mixed precision—NVIDA deep learning performance documentation." 2021. [Online]. Available: https://docs.nvidia.com/deeplearning/performance/mixed-precision-training/index.html#tensorop
[30] G. Tagliavini, S. Mach, D. Rossi, A. Marongiu, and L. Benini, "A transprecision floating-point platform for ultra-low power computing," in *Proc. Design Autom. Test Europe Conf. Exhibit. (DATE)*, 2018, pp. 1051–1056.
[31] P. Meloni *et al.*, "NEURAghe: Exploiting CPU-FPGA synergies for efficient and flexible CNN inference acceleration on Zynq SoCs," *ACM Trans. Reconfigurable Technol. Syst. (TRETS)*, vol. 11, no. 3, pp. 1–24, 2018.
[32] N. Bruschi, G. Haugou, G. Tagliavini, F. Conti, L. Benini, and D. Rossi, "GVSoC: A highly configurable, fast and accurate full-platform simulator for RISC-V based IoT processors," in *Proc. IEEE 39th Int. Conf. Comput. Design (ICCD)*, 2021, pp. 409–416.
[33] D. Nadalini, M. Rusci, G. Tagliavini, L. Ravaglia, L. Benini, and F. Conti, "PULP-TrainLib: Enabling on-device training for RISC-V multi-core MCUs through performance-driven Autotuning," in *Proc. 22nd Int. Conf. Embedded Comput. Syst. Archit. Model. Simulat. SAMOS XXII*, 2022, pp. 200–216.
[34] D. Bol *et al.*, "SleepRunner: A 28-nm FDSOI ULP cortex-M0 MCU with ULL SRAM and UFBR PVT compensation for 2.6–3.6-μW/DMIPS 40–80-MHz active mode and 131-nW/kB fully retentive deep-sleep mode," *IEEE J. Solid-State Circuits*, vol. 56, no. 7, pp. 2256–2269, Jul. 2021.
[35] I. Miro-Panades *et al.*, "SamurAI: A 1.7MOPS-36GOPS adaptive versatile IoT node with 15,000x peak-to-idle power reduction, 207ns wake-up time and 1.3TOPS/W ML efficiency," in *Proc. IEEE Symp. VLSI Circuits*, 2020, pp. 1–2.
[36] G. Ottavi *et al.*, "Dustin: A 16-cores parallel ultra-low-power cluster with 2b-to-32b fully flexible bit-precision and vector Lockstep execution mode," 2022, *arXiv:2201.08656*.

ANGELO GAROFALO received the B.Sc. and M.Sc. degrees in electronic engineering from the University of Bologna, Bologna, Italy, in 2016 and 2018, respectively, where he is currently pursuing the Ph.D. degree with the Department of Electrical, Electronic and Information Engineering.

His main research topic is on flexible computing systems for AI acceleration at the extreme edge of the IoT. His research interests include quantized neural networks, hardware-efficient machine learning, in-memory computing heterogeneous architectures, and fully programmable embedded architectures.

YVAN TORTORELLA received the master's degree in electronic engineering from the University of Bologna, Bologna, Italy, in October 2021, where he is currently pursuing the Ph.D. degree in digital systems design, in the group of Prof. L. Benini, with the Department of Electrical and Information Engineering.

His research interests include the design of parallel ultralow-power-based hardware accelerators for ultralow-power machine learning and the design of RISC-V-based computer architectures for satellite applications.

MATTEO PEROTTI received the M.Sc. degree in electronic engineering from the Polytechnic University of Turin, Turin, Italy, in 2019. He is currently pursuing the Ph.D. degree with the Integrated Systems Laboratory, ETH Zürich, Zürich, Switzerland.

His research interests include highly efficient compute architectures and computation with high-dynamic-range data types.

LUCA VALENTE received the M.Sc. degree in electronic engineering from the Politecnico di Turin, Turin, Italy, in 2020. He is currently pursuing the Ph.D. degree with the Department of Electrical, Electronic and Information Technologies Engineering, University of Bologna, Bologna, Italy.

His main research interests are hardware–software co-design of multiprocessors heterogenous systems on chip, parallel programming, and FPGA prototyping.

ALESSANDRO NADALINI received the B.Sc. and M.Sc. degrees in electronic engineering from the University of Bologna, Bologna, Italy, in 2018 and 2021, respectively.

He currently holds a research grant from the University of Bologna. His research regards lightweight extensions to the RISC-V ISA to boost the efficiency of heavily quantized neural networks inference on microcontroller-class cores.

LUCA BENINI received the Ph.D. degree from Stanford University, Stanford, CA, USA, in 1997.

He holds the Chair of Digital Circuits and Systems, ETH Zürich, Zürich, Switzerland, and is a Full Professor with the University of Bologna, Bologna, Italy. He has published more than 1000 peer-reviewed papers and five books. His research interests are in energy-efficient parallel computing systems, smart sensing microsystems, and machine learning hardware.

Dr. Benini is a Fellow of ACM and a member of the Academia Europaea.

DAVIDE ROSSI (Member, IEEE) received the Ph.D. degree from the University of Bologna, Bologna, Italy, in 2012.

He has been a Postdoctoral Researcher with the Department of Electrical, Electronic and Information Engineering "Guglielmo Marconi," University of Bologna since 2015, where he is currently an Assistant Professor. His research interests focus on energy-efficient digital architectures. In this field, he has published more than 100 papers in international peer-reviewed conferences and journals.

Dr. Rossi is a recipient of the Donald O. Pederson Best Paper Award 2018, the 2020 IEEE TCAS Darlington Best Paper Award, and the 2020 IEEE TVLSI Prize Paper Award.

FRANCESCO CONTI (Member, IEEE) received the Ph.D. degree in electronic engineering from the University of Bologna, Bologna, Italy, in 2016.

He is currently an Assistant Professor with the Department of Electrical and Information Engineering, University of Bologna, where he held a research grant from 2016 to 2020, and a position as a Postdoctoral Researcher with the Integrated Systems Laboratory, Digital Systems Group, ETH Zürich, Zürich, Switzerland. His research work has resulted in more than 40 publications in international conferences and journals. His research focuses on the development of deep-learning-based intelligence on top of ultralow power, and ultraenergy-efficient programmable systems-on-chip.

Dr. Conti has been awarded several times, including the 2020 IEEE TCAS-I Darlington Best Paper Award.

Open Access funding provided by 'Alma Mater Studiorum - Università di Bologna' within the CRUI CARE Agreement