Marsellus: A Heterogeneous RISC-V AI-IoT End-Node SoC With 2???8 b DNN Acceleration and 30%-Boost Adaptive Body Biasing

Published Version:
Marsellus: A Heterogeneous RISC-V AI-IoT End-Node SoC With 2???8 b DNN Acceleration and 30%-Boost Adaptive Body Biasing / Francesco Conti; Gianna Paulin; Angelo Garofalo; Davide Rossi; Alfio Di Mauro; Georg Rutishauser; Gianmarco Ottavi; Manuel Eggimann; Hayate Okuhara; Luca Benini. - In: IEEE JOURNAL OF SOLID-STATE CIRCUITS. - ISSN 0018-9200. - ELETTRONICO. - 59:1(2023), pp. 10269153.128-10269153.142. [10.1109/jssc.2023.3318301]

Availability:
This version is available at: https://hdl.handle.net/11585/953209 since: 2024-01-15

Published:
DOI: http://doi.org/10.1109/jssc.2023.3318301

MARSELLUS: A Heterogeneous RISC-V AI-IoT End-Node SoC with 2-to-8b DNN Acceleration and 30%-Boost Adaptive Body Biasing

Francesco Conti, *Member, IEEE,* Gianna Paulin, Angelo Garofalo, *Member, IEEE,* Davide Rossi, *Member, IEEE,* Alfio Di Mauro, *Member, IEEE,* Georg Rutishauser, *Graduate Student Member, IEEE,* Gianmarco Ottavi, Manuel Eggimann, *Member, IEEE,* Hayate Okuhara, *Member, IEEE,* and Luca Benini, *Fellow, IEEE*

***Abstract*—Emerging Artificial Intelligence-enabled Internet-of-Things (AI-IoT) System-on-a-Chips (SoCs) for augmented reality, personalized healthcare, and nano-robotics need to run many diverse tasks within a power envelope of a few tens of mW over a wide range of operating conditions: compute-intensive but strongly quantized Deep Neural Network (DNN) inference, as well as signal processing and control requiring high-precision floating-point. We present MARSELLUS, an all-digital heterogeneous SoC for AI-IoT end-nodes fabricated in GlobalFoundries 22nm FDX that combines 1) a general-purpose cluster of 16 RISC-V digital signal processing (DSP) cores attuned for the execution of a diverse range of workloads exploiting 4-bit and 2-bit arithmetic extensions (`XpulpNN`), combined with fused MAC&LOAD operations and floating-point support; 2) a 2-8bit Reconfigurable Binary Engine (RBE) to accelerate 3×3 and 1×1 (pointwise) convolutions in DNNs; 3) a set of On-Chip Monitoring (OCM) blocks connected to an Adaptive Body Biasing (ABB) generator and a hardware control loop, enabling on-the-fly adaptation of transistor threshold voltages. MARSELLUS achieves up to 180 Gop/s or 3.32 Top/s/W on 2-bit precision arithmetic in software, and up to 637 Gop/s or 12.4 Top/s/W on hardware-accelerated DNN layers.**

***Index Terms*—Deep Neural Networks (DNNs), Digital Dignal Processor (DSP), Internet of Things (IoT), Artificial Intelligence (AI), RISC-V, Heterogeneous Architecture, System-on-Chip (SoC)**

I. INTRODUCTION

The last few years have witnessed the emergence of a plethora of applications [1], such as augmented reality [2], [3], [4], personalized healthcare [5], [6], and nano-robotics [7], [8], [9], that require to combine two very distinct sets of characteristics: on the one hand, the low power profile, versatility and flexibility of Internet of Things (IoT) endpoint sensor nodes; and on the other hand, the computational power and energy efficiency of hardware accelerators, combined with advanced memory hierarchies, required to enable at-edge inference of Artificial Intelligence (AI) algorithms such as Deep Neural Networks (DNNs). AI-IoT System-on-a-Chips (SoCs) designed to meet this challenge need to be able to run real-world neural workloads in the range of hundreds of millions of multiply-accumulate (MAC) operations while respecting real-time constraints in the order of milliseconds and staying within an ultra-tight peak power envelope of a few tens of mW to enable long-term operation of battery operated nodes.

To add up to this challenge, AI-IoT applications often combine a primary task employing DNNs with other ancillary ones; for instance, on autonomous robots, control-oriented tasks are mixed with DNNs [9], while many audio processing applications combined DNNs with digital signal processing (DSP) tasks such as mel-frequency cepstrum coefficients (MFCC) [10] computation. The variety of tasks adds another dimension to the requirements of AI-IoT nodes, which must be capable of quickly ramp-up their performance in a few key computationally intensive kernels, selected at design time; deliver a generally good throughput on other compute-bound tasks; and minimize power consumption in all other states. Not only the intrinsic workload of such diverse kernels is highly variable, but they also show extremely different characteristics in terms of precision requirements. In particular, while many signal processing algorithms require high-precision floating-point computations, DNNs are generally tolerant to aggressive bit-precision reduction; several techniques for Post-Training Quantization [11] and Quantization-Aware Training [12] targeting Quantized Neural Networks (QNNs) have been recently proposed.

State-of-the-Art AI-IoT SoC's [13], [14], [15], [16], [17] typically combine a microcontroller, used to marshal data from multiple sensors and connect with other devices, with a hardware acceleration engine to exploit the intrinsic data-level parallelism of compute-intensive DSP and DNN kernels and the latter's tolerance to bit-precision reduction. Both fixed-function [16] and programmable accelerators [13] based on

Pre-print manuscript submitted for review to the IEEE Journal of Solid-State Circuits. This work was supported in part by the EU Horizon 2020 project WiPLASH under Grant 863337, in part by the EU Horizon Europe project NeuroSoC under Grant 101070634, in part by the KDT Joint Undertaking project TRISTAN under Grant 101095947, and in part by the Convolve project evaluated by the EU Horizon Europe under Grant 101070374 and supported by the Swiss State Secretariat for Education Research and Innovation under contract number 22.00150. *(Corresponding author: Francesco Conti.)*

F. Conti, D. Rossi, A. Garofalo, and G. Ottavi are with the Department of Electrical, Electronic, and Information Engineering (DEI), University of Bologna, 40126 Bologna, Italy; e-mail: f.conti@unibo.it.

G. Paulin, A. Di Mauro, G. Rutishauer, M. Eggimann are with the Integrated Systems Laboratory, ETH Zürich, 8092 Zürich, Switzerland; e-mail lbenini@iis.ee.ethz.ch.

H. Okuhara is currently with Department of Electrical and Computer Engineering, National University of Singapore, Singapore, and performed this work while at the University of Bologna.

L. Benini is with the University of Bologna, 40126 Bologna, Italy, and also with ETH Zürich, 8092 Zürich, Switzerland.

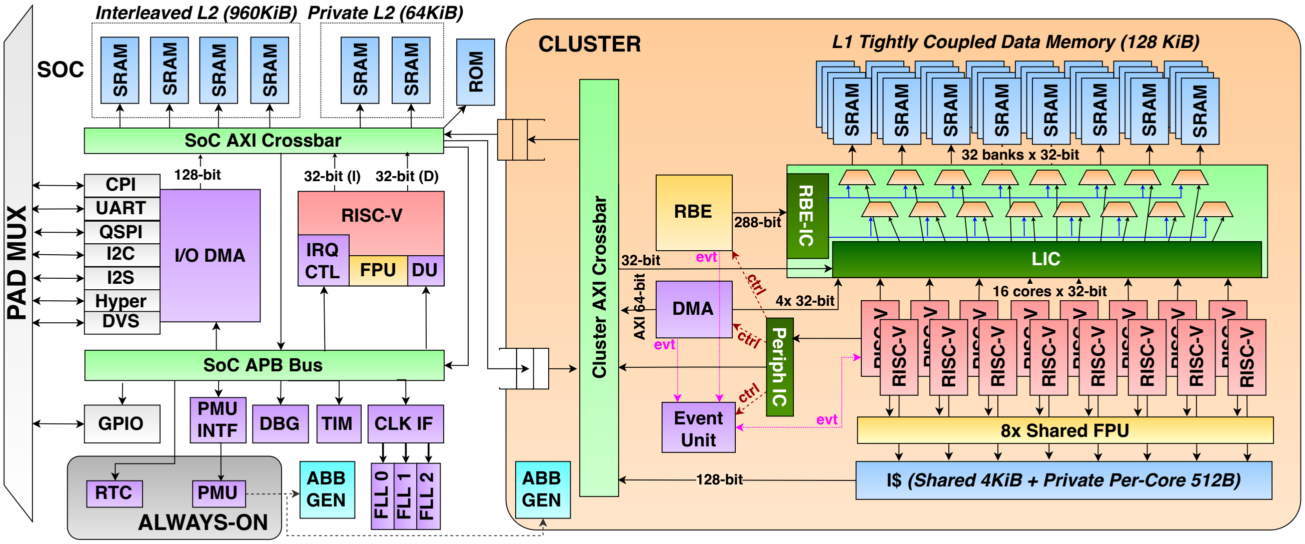


Fig. 1. Marsellus SoC architecture detailing the three power domains (ALWAYS-ON in light grey, SOC in white, CLUSTER in light orange) and their respective main component blocks.

multi-core clusters have been proposed; however, neither solution is perfect: the former are not flexible to post-fabrication algorithmic changes, e.g., due to the fast pace of development of new AI algorithms; the latter cannot deliver enough performance or performance-per-watt to enable breakthrough AI-based applications at the edge. A strategy to sidestep these limitations is to employ aggressive architectural heterogeneity, with heterogeneous acceleration introduced into software-flexible engines at the level of Instruction Set Architecture (ISA) extensions and coarse-grain hardware accelerators.

Fully Depleted Silicon-on-Insulator (FD-SOI) technology [18], [19], [20] presents a complementary, orthogonal angle of attack to maximize energy efficiency on AI-IoT SoC's executing complex applications composed of multiple tasks: the availability of Forward or Reverse Body Biasing techniques, using which it is possible to tune the effective transistor threshold voltage after fabrication to boost performance or improve energy efficiency without scaling the operating frequency, and therefore without performance loss.

In this work, we extend our ISSCC'23 paper [21] presenting Marsellus, an all-digital AI-IoT end-node heterogeneous SoC fabricated in GlobalFoundries 22nm FDX technology. Marsellus combines parallel and heterogeneous acceleration with aggressive body-biasing-enabled performance and voltage scalability, leading to state-of-the-art overall performance and efficiency in all application scenarios. In detail, we introduce the following contributions:

1) A general-purpose high-performance cluster of 16× software-programmable RISC-V cores, attuned for execution of a diverse range of DSP workloads exploiting integer (8-bit, 16-bit, 32-bit) and floating-point (16-bit, 32-bit);
2) `XpulpNN`, a set of extensions to the RISC-V Instruction Set Architecture (ISA) introducing single-instruction multiple-data execution of low-bitwidth (4-bit, 2-bit) dot-product operations in DNNs, with overlapped MAC&LOAD operations;
3) the Reconfigurable Binary Engine (RBE), a dedicated accelerator for 3×3 and 1×1 (pointwise) convolution layers with a re-configurable datapath to support 2-8bit activation and 2-8bit weight precisions enabling full exploitation of the bitwidth compressibility of QNNs.
4) an Adaptive Body Biasing (ABB) mechanism based on a set of On-Chip Monitoring (OCM) blocks and a hardware control loop, enabling automatic on-the-fly adaptation of transistor threshold voltages depending on application requirements.

Our experimental results, measured on a fabricated prototype, show that Marsellus delivers leading performance and efficiency on parallelizable software (up to 180 GOPS or up to 3.32TOPS/W with 2-bit precision exploiting the MAC&LOAD), combined with up to 637 GOPS of performance or up to 12.4 TOPS/W of energy efficiency on key DNN kernels supported by RBE: a result comparable to state-of-the-art digital accelerators, with no sacrifice in terms of flexibility and programmability. Combined with this, the ABB mechanism provides up to 30% performance boost in terms of operating frequency even when dropping the operating voltage to save energy.

The rest of this paper is organized as follows. Section II discusses the architecture of the Marsellus SoC. Section III details the experimental setup and results obtained on the SoC. Section IV discusses mapping of DNNs on Marsellus. Section V performs a detailed comparison of our work with some of the main related works in the State-of-the-Art. Section VI draws conclusions.

II. Marsellus SoC Architecture

Fig. 1 details the architecture of the Marsellus SoC. Marsellus is designed for the diverse needs of IoT devices, typically featuring a microcontroller unit and AI accelerators, focusing on parallel and heterogeneous computing. Accordingly, MARSELLUS is organized in three power domains (ALWAYS-ON in light grey, SOC in white, CLUSTER in light orange), the latter two of which correspond to an advanced microcontroller and to a heterogeneous accelerator, respectively. SOC and CLUSTER constitute also distinct clock domains, communicating through a set of dual-clock AXI FIFOs.

The SOC implements an advanced microcontroller based on a RISC-V `RV32IMCFXpulp` core [22], based on an in-order 4-stage pipeline designed to achieve high Instructions Per Cycle (IPC) in general-purpose and arithmetics-oriented applications. The SOC core features a Floating Point Unit (FPU) supporting the full RISC-V floating-point ISA extension and is augmented with the DSP-oriented `Xpulp` extension, which implements two nested hardware loops, post-increment load/store instructions, fused integer MAC instructions, and dot-product instructions for 16-bit and 8-bit data. The SOC includes also a large L2 Static Random Access Memory (SRAM)-based scratchpad memory divided in a 4-bank word-interleaved section (960 KiB) and a private bank-interleaved section (64 KiB). Both instructions and data (stacks, heap) are managed by the core and can be allocated to either section; the boot code is allocated on a small (8 KiB) boot ROM. All memories are accessed from the SOC core by means of a 64-bit AXI4 crossbar.

The SOC includes an I/O Direct Memory Access (DMA) controller [23] capable of marshaling data to/from the L2 memory with up to 128 bit/cycle bandwidth, from/to several I/O interfaces (including QSPI, I2C, I2S, Cypress' HyperRAM protocol for external memory, and an interface for DVS cameras [24]). The SOC is completed by a set of peripherals accessed via an APB peripheral bus: GPIOs, debug unit, and timers. Finally, the aforementioned ALWAYS-ON island contains a Real-Time Clock (RTC) and a power management unit (PMU), which controls the ABB generator and enables coarse-grain power gating of the SOC and CLUSTER domains. It also contains three frequency-locked loops (FLLs) to generate separate clocks for the SOC core & memories, for SOC peripherals and for the CLUSTER domain.

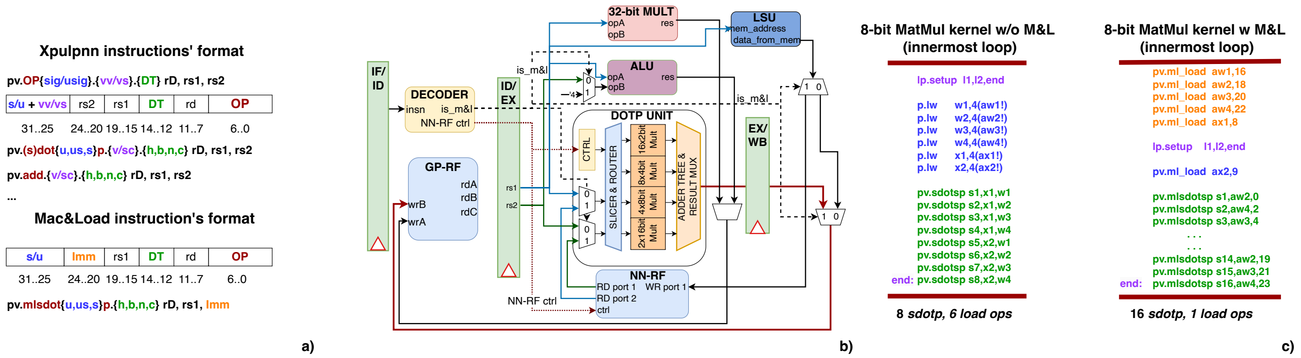
The CLUSTER is a separate power and frequency island hosting 16 identical RISC-V DSP cores implementing the `RV32IMFCXpulpnn` ISA. Similarly to the SOC core, the CLUSTER ones are based on the RI5CY baseline architecture; however, they are further augmented with the `Xpulpnn` ISA extension. `Xpulpnn` is a superset of `Xpulp` that introduces support for sub-byte (2-bit, 4-bit, 8-bit) symmetric precision dot-product instructions and a fused MAC&LOAD mechanism, relying on a dedicated Neural Network Register File (NN-RF), that enables the cores to achieve near-100% MAC unit utilization during the execution of linear algebra kernels. The architecture of the MARSELLUS cluster RISC-V cores is discussed in detail in Sec. II-A. The 16 cores share a hierarchical Instruction Cache (I$) composed of a 4 KiB of 4-way associative, 128-bit/line shared cache (L1.5) common between all cores, with the addition of smaller 512 B L1 private per-core caches [25]. The L1.5 shared I$ employs multi-port (MP) memories to remove direct critical paths between different L1 I$s and enable scaling the cluster to 16 cores. The L1 private per-core I$s minimize the path between the RISC-V core prefetchers and the L1.5 I$, enabling higher clock speed. Both I$ levels are realized with standard-cell memories (SCMs) to enable the MP architecture and improve the overall energy efficiency compared with regular SRAM cuts. The cores also share and 8 FPUs with support for IEEE 32-bit float, IEEE 16-bit float, and BF16 formats, for efficient support of floating-point DSP applications [26].

Together with the RISC-V cores, the main computational element of the CLUSTER is the RBE: an accelerator for DNN convolution layers with a unified datapath that can be runtime-configured in different modes (3×3 and 1×1 convolutions) and activation/weight precisions (asymmetric 2–8 bits). Other layers (e.g., fully-connected, 3×3 depth-wise convolution) can be implemented as corner cases of the two natively supported modes, whereas unsupported layers are executed on the CLUSTER RISC-V cores. A 64-bit/cycle read, 64-bit/cycle write DMA engine can be used to marshal data between the L2 memory in the SOC and the CLUSTER, through a CLUSTER-level AXI crossbar connected to the SOC through dual-clock AXI FIFOs. Cores, DMA engine and RBE share at L1 the same 128 KiB of SRAM Tightly Coupled Data Memory (TCDM). The TCDM is organized in 32 word-interleaved banks to provide high-bandwidth parallel access to all the traffic generators; access is delivered via a 0-wait-state, 928-bit/cycle aggregate bandwidth CLUSTER interconnect. The interconnect is organized hierarchically and split in two branches; the logarithmic interconnect (LIC) branch is a fully combinational crossbar to route & arbitrate accesses from the 16 cores, the DMA and a further 32-bit SOC port towards the TCDM banks; the RBE interconnect (RBE-IC) branch routes RBE accesses, which are always contiguous, towards TCDM banks with no bank-wise arbitration. A set of bank-level multiplexers are employed to grant access to the LIC or RBE-IC branch, utilizing a round-robin rotation scheme to avoid starvation.

The CLUSTER includes also an Event Unit to enable high-performance parallel programming synchronization primitives (barriers, critical sections, etc.) between the 16 cores, as well as for fast communication of end-of-job events from the DMA and the RBE. Finally, the cluster interconnect also includes a secondary 32-bit peripheral interconnect that is used for the configuration of RBE, the DMA, and the Event Unit. All components of the MARSELLUS CLUSTER are tightly coupled and can not be power-gated at a fine grain as, e.g., the modules in Jain *et al.* [17]. The specific bus widths and memory sizes utilized in the CLUSTER were chosen to enable the target applications of MARSELLUS to be run in a compute-bound scenario in most cases, supporting both RISC-V and RBE-accelerated computation.

A. `Xpulpnn` extensions and MAC&LOAD

1) Operating principle: As previously introduced, the main innovation of Marsellus that we present at the core level is the design, at architectural and micro-architectural level, of a set of ISA instructions, namely `XpulpNN`, aiming to boost the performance and efficiency of reduced-precision integer DSP and linear algebra kernels.



Xpulpnn instructions' format

```
pv.OP{sig/usig}.{vv/vs}.{DT} rD, rs1, rs2
```

s/u + vv/vs	rs2	rs1	DT	rd	OP
31..25	24..20	19..15	14..12	11..7	6..0

```
pv.(s)dot{u,us,s}p.{v/sc}.{h,b,n,c} rD, rs1, rs2
pv.add.{v/sc}.{h,b,n,c} rD, rs1, rs2
...
```

Mac&Load instruction's format

s/u	Imm	rs1	DT	rd	OP
31..25	24..20	19..15	14..12	11..7	6..0

```
pv.mlsdot{u,us,s}p.{h,b,n,c} rD, rs1, Imm
```

a)

8-bit MatMul kernel w/o M&L (innermost loop)

```
     lp.setup  l1,l2,end

     p.lw      w1,4(aw1!)
     p.lw      w2,4(aw2!)
     p.lw      w3,4(aw3!)
     p.lw      w4,4(aw4!)
     p.lw      x1,4(ax1!)
     p.lw      x2,4(ax2!)

     pv.sdotsp s1,x1,w1
     pv.sdotsp s2,x1,w2
     pv.sdotsp s3,x1,w3
     pv.sdotsp s4,x1,w4
     pv.sdotsp s5,x2,w1
     pv.sdotsp s6,x2,w2
     pv.sdotsp s7,x2,w3
end: pv.sdotsp s8,x2,w4
```

8 *sdotp, 6 load ops*

8-bit MatMul kernel w M&L (innermost loop)

```
      pv.ml_load  aw1,16
      pv.ml_load  aw2,18
      pv.ml_load  aw3,20
      pv.ml_load  aw4,22
      pv.ml_load  ax1,8

      lp.setup   l1,l2,end

      pv.ml_load  ax2,9

      pv.mlsdotsp s1,aw2,0
      pv.mlsdotsp s2,aw4,2
      pv.mlsdotsp s3,aw3,4
            . . .
            . . .
      pv.mlsdotsp s14,aw2,19
      pv.mlsdotsp s15,aw3,21
end:  pv.mlsdotsp s16,aw4,23
```

16 *sdotp, 1 load ops*

c)

Fig. 2. a) Encoding formats of the Xpulpnn instructions, including the MAC&LOAD. b) Modified RI5CY pipeline to implement the Xpulpnn extension with *i*) low-bitwidth dot-product units and *ii*) additional NN-RF for MAC&LOAD support. c) Assembly code snippets of innermost loops of 8-bit symmetric matrix multiplication kernels (MatMul), implemented without (left-sided code) and with MAC&LOAD.

We build such extensions on top of Xpulp extending the support in the ISA of the RISC-V processor to packed-SIMD operations performed on vectors of nibble and crumb data types, i.e. of 4-bits and 2-bits respectively. The core of Xpulpnn consists of dot-product (*dotp*) operations, including the MAC-equivalent sum-of-dot-product (*sdotp*), executed in various formats: the two inputs can be both vectors (*vv*) or one vector and the other a scalar value replicate in each vector element (*vs*); the vectors can be interpreted as both signed (*s*), both unsigned (*u*) or the first unsigned and the second signed (*us*) (and viceversa); the third scalar operand and the accumulator feature always 32-bit precision and can be either signed or unsigned. Xpulpnn includes also nibble and crumb packed-SIMD ALU operations, such as vector addition, subtraction, maximum, minimum, shuffling, and other instructions for bit-manipulation at the granularity of the vector elements.

To significantly improve the utilization of the processor's pipeline on regular kernels like the matrix-multiplication, we design a fused MAC&LOAD instruction, which applies to all the *dotp* SIMD formats supported by Xpulpnn, that collapses one packed-SIMD *dopt* and one post-modified load operations into the same instruction. Since the data-path activated by the *dotp* does not interfere with the Load-Store Unit (LSU), the two units can run in parallel without requiring complex logic to control the instruction flow.

2) Microarchitecture: As shown in Fig. 2 b), to implement the Xpulpnn instructions, we modify the micro-architecture of the baseline RI5CY core as follows: we extend the DOTP unit of RI5CY, which consists of two sets of multipliers supporting 16- and 8-bit packed-SIMD *dotp* operations, with two additional multipliers islands for nibble and crumb operands. We choose to replicate the multipliers islands for different precisions to minimize the impact of Xpulpnn on the critical path of the core: extra logic to manipulate and reshape the operands before feeding the single multi-precision island would increase the critical path of the core. This helps improving efficiency because the multi-branch datapath uses operand isolation to remove spurious switching activity. At the same time, we enhance the ALU with the support for operations on nibble and crumb operands and we integrate the new instructions into the decoder of the pipeline.

The design of the MAC&LOAD is more complex and it is the result of an architecture/micro-architecture codesign to maximize the efficiency of the instruction and minimize the hardware costs to implement it. In reference to the Fig. 2 a) and b) that shows the encoding of the instruction and the micro-architecture of the core, the MAC&LOAD works as follows: it fetches the operands of the *dotp* operation from a dedicated register file, namely NN-RF, containing 6 32-bit SIMD vector registers, addressed by a 5-bit immediate field of the instruction; the accumulator resides in the GP-RF of the core instead and it is updated once the *dopt* operation is completed, in the EX stage. One of the addressed NN-RF register can be refreshed with a new data from the memory. In such case, one of the two most significant bits of the immediate is set (note that, by construction, only one of the registers can be refreshed with one instruction, since the core's pipeline features a single 32-bit LSU data port towards the memory) and the LSU of the core receives from the GP-RF the pointer for the memory access. Such pointer is then incremented by one word in the ALU and stored back into the GP-RF in the EX stage, while the data fetched from the memory, available in the WB stage of the pipeline, is directly routed to the write port of the NN-RF.

The operating mechanism of the MAC&LOAD instruction, with the dedicated NN-RF, has three important advantages: first, it avoids to add a costly write port to the GP-RF (which features only two write ports in the baseline RI5CY core), otherwise necessary because the MAC&LOAD needs to store three results: the dopt result, the updated pointer and the new data fetched from the memory; second, we can directly control how long an operand will reside in the NN-RF without being constrained by the compiler scheduler, allowing more flexibility for data-reuse at register file level, particularly effective to reduce memory traffic and increase the energy efficiency of compute- and memory-intensive kernels (especially critical in the context of multi-core systems, where memory conflicts reduce performance and energy efficiency); third, fetching the *dotp* operands from the NN-RF leaves more room in the GP-RF to store more accumulators: exploited in combination with data reuse techniques, it increases the number of outputs produced in the innermost loops.

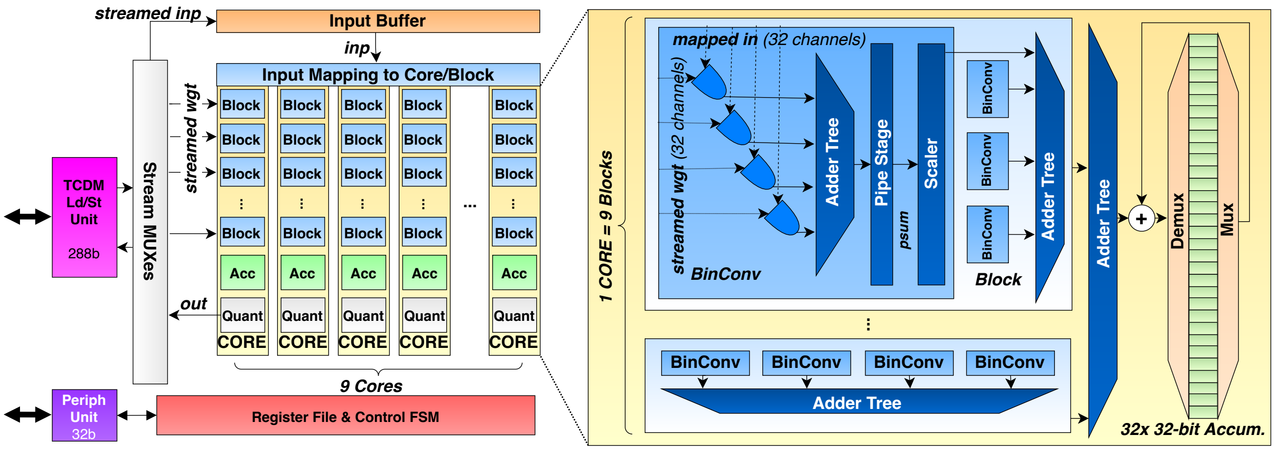


Fig. 3. RBE microarchitecture (left), showing the 9 Cores divided in 9 Blocks each; and detail of a RBE Core microarchitecture (right).

Each Xpulpnn RISC-V core in the cluster of Marsellus costs 78kGE, with a total overhead of 17.5% compared to the baseline RI5CY core, due to the additional multiplier islands in the DOTP unit of RI5CY, the extended ALU and the decoding stage. The extra hardware cost of the MAC&LOAD is for the NN-RF, the logic to distribute the operands to the DOTP unit from the NN-RF and to route the data from the LSU to the NN-RF write port. From a power consumption perspective, to avoid unnecessary switching activity when the MAC&LOAD and/or the classic *dotp* SIMD operations are not executed, we perform operand isolation on critical wires and apply clock-gating to the NN-RF and the DOTP unit, limiting the power overhead of the core on general-purpose applications to $\sim 3\%$.

3) Compiler support: To ease the exploitation of the proposed ISA extension, we integrate the machine-level description of the Xpulpnn operations into the GCC compiler, allowing the programmer to infer these instructions in the C application code through explicit invocation of built-in functions. Compared to inline assembly, this approach enables optimization passes by the compiler back-end that maximizes reuse of operands and efficiently schedules the instruction flow. We provide also a set of optimized C routines, based on the Xpulpnn ISA, for QNN and linear algebra kernels, publicly available under open-source permissive licence[1].

B. Reconfigurable Binary Engine (RBE)

1) Operating principle: As previously introduced, the RBE is a hardware accelerator targeting DNN convolution layers with a unified datapath that can be reconfigured at runtime in two different modes of operation (3×3 and 1×1 convolutional layers) and activation/weight precisions (asymmetric between 2 and 8 bits, including non-power-of-two bitwidths). Considering W-bit weights, I-bit inputs, and O-bit outputs, RBE splits each $W \times I$-bit product into $W \times I$ distinct single-bit contributions, which are then allocated partially in parallel on different 1-bit MAC units, and partially serialized in time.

Considering for simplicity a 1×1 convolution, in RBE weights and inputs are decomposed in binary contributions, **wgt** and **inp**, respectively, and the convolution operation is performed in the binary domain accumulating scaled partial sum in a 32-bit accumulator:

$$\mathbf{acc}_{[h,w,k_{out}]} = \sum_{i=0}^{W}\sum_{j=0}^{I}\sum_{k_{in}} 2^{i+j} \cdot \mathbf{wgt}_{[k_{out},k_{in},i]} \wedge \mathbf{inp}_{[h,w,k_{in},j]} \tag{1}$$

where $\wedge$ is a logical AND operation and the multiplication by 2^{i+j} is a left-shift. The 3×3 case is analogous, with a further summation over 9 filter contributions.

After complete accumulation, the value of **acc** is normalized and right-shifted:

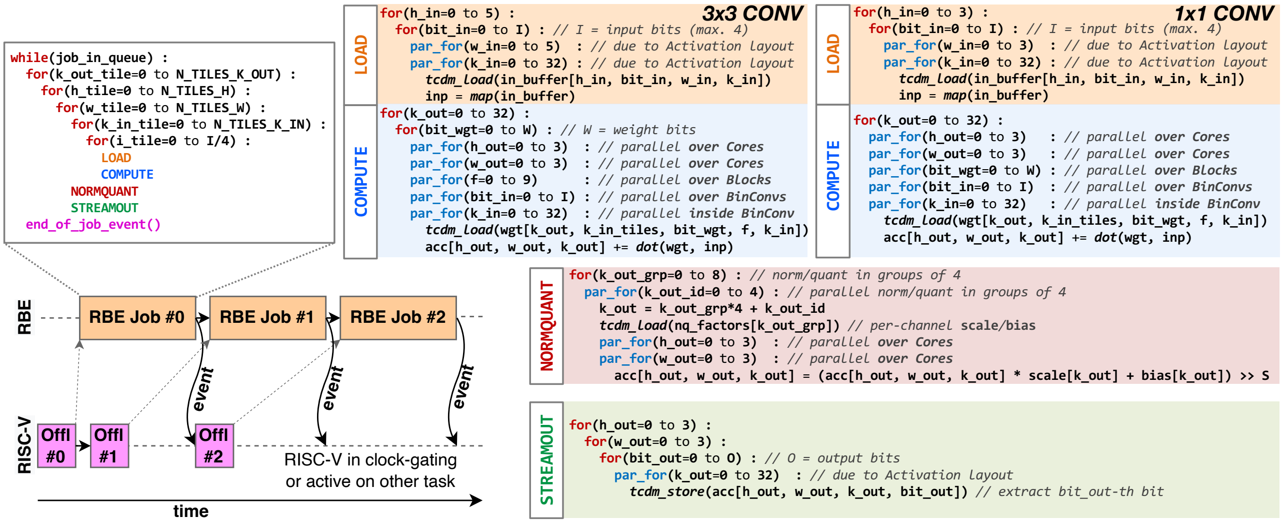
$$\mathbf{out}_{[h,w,k_{out}]} = \left(\mathbf{scale}_{[k_{out}]} \cdot \mathbf{acc}_{[h,w,k_{out}]} + \mathbf{bias}_{[k_{out}]}\right) \gg \mathbf{S} \tag{2}$$

Implementing convolutions by means of AND, left-shift, and add operations enables finely-controlled mixed precision computation on the same underlying hardware. For example, a convolution with $W = 3$, $I = 4$, and $O = 2$ can be implemented on the same hardware of one using only 8-bit data representation by tuning shift factors and loop iterations. The RBE microarchitecture implements Eqs. 1 and 2 by tiling the inner accumulation and outer loops, and executing part of the inner tile loops on a hierarchical architecture that is discussed in the following.

2) Microarchitecture: Fig. 3 introduces the architecture of RBE, which is based on the open-source Hardware Processing Engine (HWPE)[2] architectural template and set of intellectual property (IP) elements. Following this template, RBE is divided into three parts: *controller*, *streamer*, and *datapath* (or engine). The RBE controller consists of a Peripheral Unit, connected as a target to the CLUSTER peripheral interconnect, a latch-based dual-context register file, and a hierarchical finite-state machine (FSM) that controls the overall behavior of a job offloaded to the RBE. To simplify the datapath control, in particular, the tiled loop nests that implement convolutional layers, part of the FSM is realized using a software (SW) configurable *uloop*, i.e., a tiny microcoded loop processor [27].

[1] https://github.com/pulp-platform/pulp-nn-mixed/tree/main/XpulpNN

[2] https://hwpe-doc.readthedocs.io



```
while(job_in_queue) :
  for(k_out_tile=0 to N_TILES_K_OUT) :
    for(h_tile=0 to N_TILES_H) :
      for(w_tile=0 to N_TILES_W) :
        for(k_in_tile=0 to N_TILES_K_IN) :
          for(i_tile=0 to I/4) :
            LOAD
            COMPUTE
        NORMQUANT
        STREAMOUT
  end_of_job_event()
```

3x3 CONV

```
for(h_in=0 to 5) :
  for(bit_in=0 to I) : // I = input bits (max. 4)
    par_for(w_in=0 to 5)  : // due to Activation layout
    par_for(k_in=0 to 32) : // due to Activation layout
      tcdm_load(in_buffer[h_in, bit_in, w_in, k_in])
      inp = map(in_buffer)
```

```
for(k_out=0 to 32) :
  for(bit_wgt=0 to W) : // W = weight bits
    par_for(h_out=0 to 3)  : // parallel over Cores
    par_for(w_out=0 to 3)  : // parallel over Cores
    par_for(f=0 to 9)      : // parallel over Blocks
    par_for(bit_in=0 to I) : // parallel over BinConvs
    par_for(k_in=0 to 32)  : // parallel inside BinConv
      tcdm_load(wgt[k_out, k_in_tiles, bit_wgt, f, k_in])
      acc[h_out, w_out, k_out] += dot(wgt, inp)
```

1x1 CONV

```
for(h_in=0 to 3) :
  for(bit_in=0 to I) : // I = input bits (max. 4)
    par_for(w_in=0 to 3)  : // due to Activation layout
    par_for(k_in=0 to 32) : // due to Activation layout
      tcdm_load(in_buffer[h_in, bit_in, w_in, k_in])
      inp = map(in_buffer)
```

```
for(k_out=0 to 32) :
  par_for(h_out=0 to 3)    : // parallel over Cores
  par_for(w_out=0 to 3)    : // parallel over Cores
  par_for(bit_wgt=0 to W) : // parallel over Blocks
  par_for(bit_in=0 to I)  : // parallel over BinConvs
  par_for(k_in=0 to 32)    : // parallel inside BinConv
    tcdm_load(wgt[k_out, k_in_tiles, bit_wgt, f, k_in])
    acc[h_out, w_out, k_out] += dot(wgt, inp)
```

```
for(k_out_grp=0 to 8) : // norm/quant in groups of 4
  par_for(k_out_id=0 to 4) : // parallel norm/quant in groups of 4
    k_out = k_out_grp*4 + k_out_id
    tcdm_load(nq_factors[k_out_grp]) // per-channel scale/bias
    par_for(h_out=0 to 3)  : // parallel over Cores
    par_for(w_out=0 to 3)  : // parallel over Cores
      acc[h_out, w_out, k_out] = (acc[h_out, w_out, k_out] * scale[k_out] + bias[k_out]) >> S
```

```
for(h_out=0 to 3) :
  for(w_out=0 to 3) :
    for(bit_out=0 to O) : // O = output bits
      par_for(k_out=0 to 32)  : // due to Activation layout
        tcdm_store(acc[h_out, w_out, k_out, bit_out]) // extract bit_out-th bit
```

Fig. 4. RBE execution flow, with multiple jobs offloaded and detail of the dataflow loop nests implemented in the accelerator for 3×3 and 1×1 convolutions (LOAD and COMPUTE phases), normalization/quantization (NORMQUANT) and stream-out of outputs (STREAMOUT).

The streamer is composed of a 288-bit wide TCDM Load/Store Unit, realized using standard open-source HWPE IPs; this unit accesses data in the TCDM memory and converts it into an internal streaming representation using a simple ready/valid handshake protocol to enable latency insensitiveness of the inner datapath. The streamer uses a three-dimensional strided address generator and is capable of linearizing any 3D strided memory access pattern into a stream. A set of multiplexers and de-multiplexers helps the streamer to address the incoming memory stream towards the correct consumer in the *datapath* (or vice versa for outgoing streams).

The RBE datapath exploits an output-stationary and partially bit-serial dataflow with reuse over the spatial dimensions mapped onto a set of 9 Cores, each one working on the receptive field of one pixel on 32 channels in the output space. RBE is organized hierarchically into a grid of 9×9 = 81 Blocks, arranged in 9 Cores of 9 Blocks each. Each Block includes 4× Binary Convolution Engines (BinConvs), and each BinConv performs a 32×32 1-bit dot-product per cycle (using AND gates, and achieving 32 binary MAC/cycle). The result of the dot-product reduction is then scaled by the required power-of-two value (according to the operating mode) with the help of small dynamic shifters. The scaled results of the four BinConvs are accumulated at Block level, and the cumulative results of all Blocks within a single Core are accumulated and stored in one of 32× 32-bit latch-based Accumulator Banks (Accums) in each core. After the accumulation is complete (i.e., all reduction dimensions in the convolutional layer have been fully computed), a Quantizer module in each Core is used to perform ReLU activation and reduce the 32-bit accumulators into O bits. In this way, RBE supports mixed-precision DNNs with I, O, W arbitrarily set to any value in 2-8 bits.

3) Data layout: To implement the principle of Eqs. 1 and 2 on top of the RBE microarchitecture, we introduce a specialized data layout in TCDM for both weights and activations, which exposes the same parallelism that is used inside the accelerator, by swapping the "bit index" dimension with a (tiled) channel dimension. Specifically, input activation bitstreams are stored in memory in the *(H, W, K/32, I, 32)* format, to align with the BinConv parallelism of 32 channels. Similarly, output bitstreams are stored in the *(H, W, K/32, O, 32)* format. Weights are stored in such a way that they can be directly streamed from memory into RBE without marshaling. For 3×3 convolutions, we employ a *(Kout, Kin/32, W, 9, 32)* format, where the two innermost dimensions are aligned with the number of Blocks per Core and with the parallelism of the BinConvs, respectively. For 1×1 convolutions, we use *(Kout, Kin/32, W, 32)*.

4) Execution flow: RISC-V cores can enqueue up to 2 jobs in the RBE register file; whenever the accelerator is free, it will start executing the oldest offloaded job and emit an event to synchronize with the core at the end of each job execution. Each job performs a complete convolutional layer implementing Eqs. 1 and 2. Dimensions not aligned with the size of the RBE accelerator are tiled and executed sequentially by the control FSM, using the embedded *uloop* unit to implement deeply nested loops with minimal overhead. Fig. 4 specifies in detail the complete execution flow of RBE in the form of nested temporal loops (**for**) and parallel execution (**par_for**); the core of the execution is composed by the **LOAD** and **COMPUTE** states, which operate differently in the two 1×1 and 3×3 operating modes. After accumulation is complete, the RBE performs normalization (**NORMQUANT**) and stream-out of outputs (**STREAMOUT**).

For the 3×3 convolutional mode, RBE loads an input patch of 4-bits (less if $I < 4$) of 32 channels of 5×5 pixels into the input buffer. Each of the 9 Cores works on the receptive field of a single pixel on 32 channels in the output space. The 32 channels of the 3×3 filters are unrolled over the 9 Blocks of each Core, broadcasted to all 9 Cores, and bit-serialized in time. With the output-stationary flow, the partial output results are stored in the latch-based Accums while new

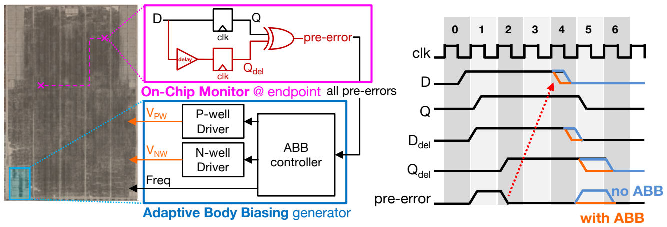


Fig. 5. (left) Adaptive Body Biasing (ABB) mechanism implemented in MARSELLUS with On-Chip Monitors detecting pre-errors at 1% of timing critical endpoints and ABB generator; (right) an example of pre-error detection and ABB generation.

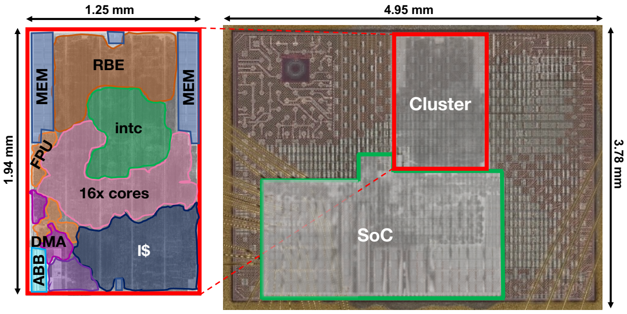


Fig. 6. Microphotograph of the MARSELLUS prototype fabricated in GlobalFoundries 22FDX technology.

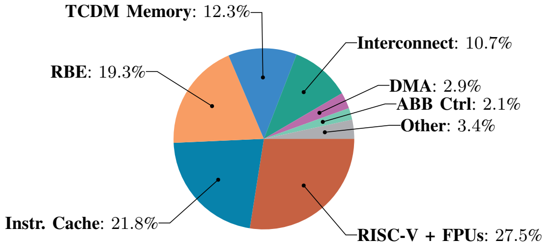


Fig. 7. Area distribution of CLUSTER.

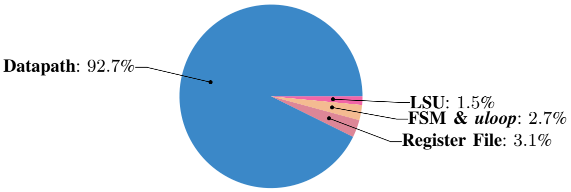


Fig. 8. Post-synthesis area distribution of RBE.

input patches are loaded if $I > 4$, or the $K_{in} > 32$. For the 1×1 convolutional mode, the streamers load a smaller patch of up to 4-bits (less if $I < 4$) of 32 channels of 5×5 pixels into the input buffer. The individual W bits of the weight are now mapped in a bit-parallel fashion on the Blocks of a Core while still being broadcasted to all 9 Cores. The last Block of each Core remains, therefore, unused and clock-gated in this operation mode. Overall, running jobs with lower W results in a faster computation time for the 3×3 operation mode and a reduced acceleration utilization for 1×1 operation mode. The RBE, with its total of 10368 AND gates used as single-bit multipliers, achieves peak throughput (1610 operations/cycle in the **COMPUTE** state) in the 3×3 mode with W=2, I=2 or 4, O=2 or 4. We refer to Section III-C2 for a complete analysis of the RBE performance in several operating modes.

C. On-Chip Monitors and Adaptive Body Biasing

To further improve energy efficiency in computationally demanding applications beyond the described architectural innovation, MARSELLUS introduces a dynamic Adaptive Body Biasing (ABB) mechanism, based on the circuit introduced in Moursy et al. [20], whose operation is shown in Fig. 5. The cluster ABB generator incurs in a very small area (0.039 mm^2) and power (+0.4%) overhead in exchange for significant performance and energy efficiency gains, that are discussed in the following. The ABB mechanism is based on the empirical observation that after timing closure, a minority of combinational register-to-register paths in a system will remain near-critical, i.e., have a very small positive slack, due to their length (number of gates, wire length). Reducing the supply voltage without scaling frequency (or, conversely, increasing frequency but not voltage) will result in timing failures that, with a high probability, will hit one or more of these near-critical paths.

At MARSELLUS' signoff, the 1% of the register-to-register path endpoints with smallest positive slack (i.e., nearest to being critical) were selected and augmented with On-Chip Monitors. OCMs work by pairing the endpoint register with a shadow copy, which is fed with a delayed version of the endpoint register input. XOR-ing the outputs of the functional and shadow registers, OCMs detect whether an endpoint is close to becoming timing-critical (e.g., when the SoC is undervolted or over-clocked), and raise a *pre-error* signal that is propagated to the ABB generator (Fig. 5).

The ABB generator collects all pre-error signals and, depending on its configuration, its internal hardware loop can react by directly tuning the SoC's N-well and P-well biasing voltages to increase forward body biasing (FBB), which in turn reduces the logic's voltage threshold and hence, all propagation delays. Conversely, if it does not detect any pre-error the generator will progressively reduce the body biasing voltage, thereby raising the devices' thresholds, to save power. By properly calibrating the pre-error delay margins detected by OCMs, the ABB effectively dynamically trims the setup margins of flip-flops, enabling a higher frequency operation at the same voltage.

Fig. 5 (right) shows an example. In the absence of ABB (blue line), a pre-error condition is ignored, and after a few cycles, a real error arises due to a setup time violation. Conversely, with ABB, the pre-error condition is detected, and the generator reacts by increasing forward body biasing, lowering thresholds and preventing the error condition thanks to the consequent speed-up.

III. SoC MEASUREMENTS

A. Area, frequency and power

The MARSELLUS prototype was fabricated in GlobalFoundries 22 nm FDX using the flip-well flavor of the tech-

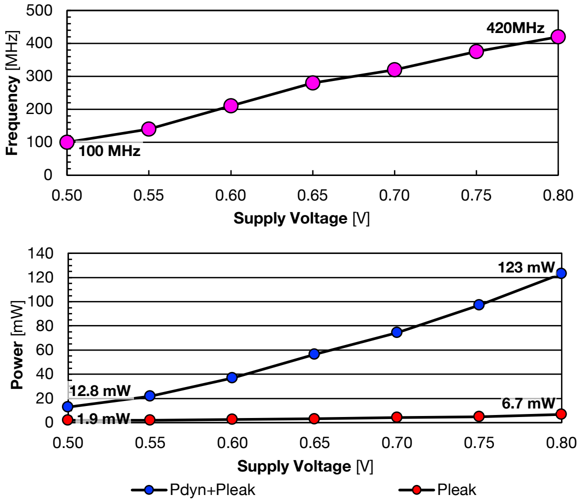


Fig. 9. MARSELLUS measured frequency and power sweep while varying V_{DD} (with no ABB).

nology, which enables low voltage threshold (LVT) and super-low voltage threshold (SLVT) cells. A microphotograph of MARSELLUS is shown in Fig. 6. The prototype is operating with a nominal supply voltage of 0.8 V with a signoff frequency of 400 MHz for the cluster. The total die area is 18.7 mm^2, including the full architecture described in Sec. II as well as other IPs out of the scope of this work. We focus our analysis and all power measurements on the compute CLUSTER, which occupies an area of 2.42 mm^2, divided as detailed in Fig. 7. The RISC-V cores, together with the shared instruction cache, occupy almost half of the CLUSTER area, while the RBE accelerator takes one fifth. For further insight, Fig. 8 shows in detail the post-synthesis area breakdown of the 652 kGE large RBE; the datapath is with 605 kGE (92.7%) the largest part of RBE.

Fig. 9 shows the results of a supply voltage sweep between 0.5 V and 0.8 V on the fabricated prototype, without applying the automatic Adaptive Body Biasing described in Section II-C. The maximum frequency achieved at 0.8 V is 420 MHz (5% more than the signoff frequency), which scales down to 100 MHz at 0.5 V. We characterized the SoC's power consumption in this sweep by using an INT8 matrix-multiplication kernel exploiting the MAC&LOAD functionality discussed in Section II-A. At the nominal 0.8 V supply voltage, this corresponds to a total power consumption of 123 mW (94.6% dynamic, 5.4% leakage); dynamic power is reduced by a factor of 10.7× and leakage by 3.5× when moving to the lowest-voltage operating point explored (0.5 V).

B. Adaptive Body Biasing

The adoption of flip-well transistors in the MARSELLUS prototype enables using the technique described in Section II-C to boost the energy-efficiency of the MARSELLUS SoC by applying FBB. Fig. 10 shows how ABB can be exploited to aggressively undervolt the SoC supply voltage without hitting any performance penalty. In this experiment, we set the initial operating point at 0.8 V targeting the signoff frequency of 400 MHz, and then progressively down-scale the voltage in several power measurements performed on a baseline `Xpulp` INT8 matrix multiplication kernel. Without applying ABB, the minimum operating voltage is 0.74 V, beyond which the SoC stops working due to timing violations. ABB enables to adaptively compensate the increased path delay by detecting pre-errors and correcting them with FBB, while retaining the efficiency advantage given by the lower supply voltage. In this way, it is possible to reduce the supply to 0.65 V without scaling frequency, with a power reduction of 30% with respect to the nominal operating point and of 16% with respect to the 0.74 V one.

Fig. 11 shows an example of ABB operation over a synthetic benchmark that alternates three phases of operation: RBE-centric and hardware accelerated; low-intensity data marshaling from RISC-V cores; RISC-V based high-intensity computation. This kind of pattern arises, for example, when RBE-supported operators are mixed with non-supported ones,

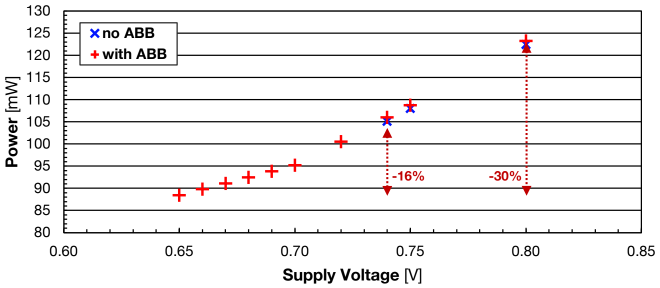


Fig. 10. MARSELLUS power measurement with and without applying ABB for a fixed operating frequency of 400 MHz. Only operating points without timing violations are plotted.

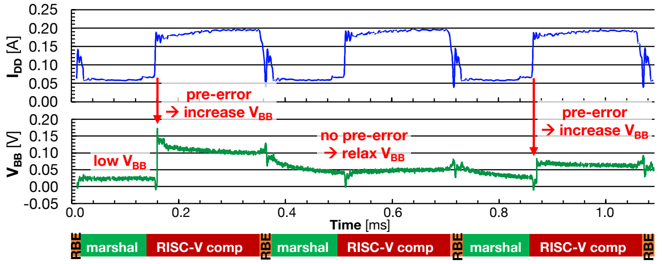


Fig. 11. Example of ABB operation measured on the MARSELLUS prototype, with over-clocking at 470 MHz in the nominal 0.8 V operating point.

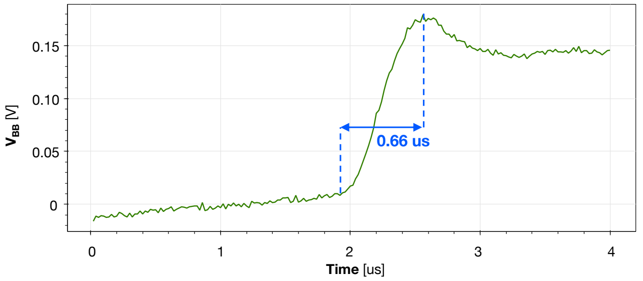


Fig. 12. Detail of 4 µs of ABB transition measured on the MARSELLUS prototype, with over-clocking at 470 MHz in the nominal 0.8 V operating point.

requiring data marshaling to align the RBE-centric data layout to a software-centric one. In the experiment of Fig. 11, the MARSELLUS CLUSTER was clocked at 470 MHz, a 17.5% boost with respect to the signoff frequency. We observe that during the 1 ms of operation of the benchmark, the ABB mechanism is triggered twice to boost FBB and enable errorless operation, both times during the higher compute intensity phases. This is not suprising, as during these phases more near-critical paths are exercised, hence the probability of a pre-error is larger. Fig. 12 shows the detail of one such ABB transition, which has a duration of ∼0.66 µs (∼310 clock cycles) after the pre-error is triggered. When no pre-error is detected in a given time window, the body biasing voltage is progressively relaxed to improve energy efficiency by increasing the effective threshold voltage of flip-well transistors.

C. Performance & Energy Efficiency

1) RISC-V Performance: By exploiting `Xpulpnn`, symmetric 2-bit or 4-bit matrix-vector or matrix-matrix multiplications can be executed in 6× and 9× less instructions compared to the baseline `Xpulp`; thanks to the native support for nibble and crumb SIMD operations, `Xpulpnn` eliminates the overhead to manipulate data to match the lowest precision of the operations available in the ISA of the baseline RI5CY, i.e. 8-bit. MAC&LOAD accelerated matrix-multiplication kernels further boost the performance by up to 67%, achieving a DOTP unit utilization as high as 94%. As also visible in Fig. 2 c), after some instructions to initialize the NN-RF, which anyway happens outside the innermost loop, the MAC&LOAD is able to mask all the explicit loads except one; in combination with data reuse strategy, the innermost loop of the so built kernel outputs 16 accumulators at the cost of a single explicit load operation.

Combining the MAC&LOAD with the ABB mechanism, MARSELLUS achieves area efficiency on integer linear algebra kernels comparable to that of some application-specific integrated circuits (ASICs): 9.63 Gop/s/mm^2 in 2×2-bit operation; 4.81 Gop/s/mm^2 in 4×4-bit, 2.54 Gop/s/mm^2 in 8×8-bit. At the same time, the RISC-V cores of MARSELLUS retain the full flexibility of floating-point execution when highest precision is needed. The design is similar to that of Vega [13], but the ABB mechanism and the doubled number of FPUs result in 2.1× better absolute performance and slightly better area efficiency (0.37 Gop/s/mm^2 against 0.33 Gop/s/mm^2 for Vega).

To assess the performance of the RISC-V cluster on non-Machine Learning (ML) applications, we targeted a typical kernel used in DSP: Fast Fourier Transform (FFT). FFTs can constitute a significant fraction of the overall compute workload of an AI-IoT application [10]. We exploited the implementation proposed by Mazzoni *et al.* [28], which, measured on a window of 2048 samples, achieves a peak throughput of 4.69 FLOp/cycle when parallelized on 16 cores, leading to a peak performance of 1.97 GFLOPS in the 0.8 V,420 MHz operating point, and a top efficiency 36 GFLOPS/W in the 0.5 V,100 MHz one.

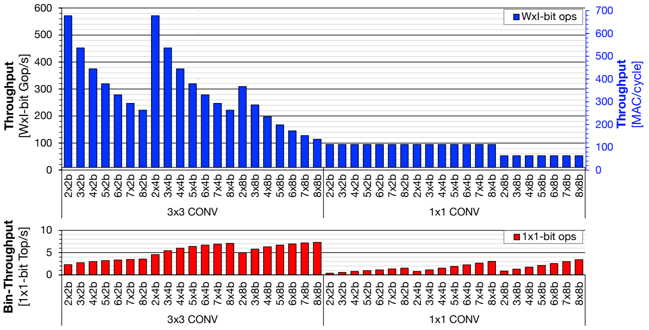


Fig. 13. Main `LOAD`-`COMPUTE` loop throughput for 3×3 and 1×1 convolutions in terms of $W\times I$-bit (blue) and 1×1-bit (red) operations running on RBE at 0.8V and a nominal frequency of 420 MHz, computing a layer with $K_{in} = 64$, $K_{out} = 64$, $H = W = 3$. The right axis reports performance scaled in $W\times I$-bit MAC/cycle.

2) RBE Performance: As detailed in Section II-B, the RBE can be used in several different configurations in terms of operation and precision, which correspond to different throughput. Fig. 13 analyses the performance of RBE in many supported configurations[3] when computing a convolutional layer with 64 output channels (K_{out}), 64 input channels (K_{in}), and output spatial size 3×3. We focus on two different metrics: actual throughput at the $W\times I$-bit precision, which is the target application-aware performance metric, and "raw" throughput at 1×1-bit, which instead measures the raw utilization of the compute resources over the full computation loop (considering both the `LOAD` and `COMPUTE` phases).

Several effects are visible. First of all, the 1×1-bit throughput is higher when activations are 4-bit or larger, as in that case the RBE all BinConvs in a Block are utilized. Configuration with I=8-bit, however, result in a ∼50% actual throughput reduction because their contributions are split in consecutive iterations. 3×3 convolutions suffer from little overhead introduced by the `LOAD` phase, while 1×1 convolutions are hit more heavily, due to the fact that the duration of their `COMPUTE` phase is much shorter and comparable with `LOAD`. Finally, in 3×3 mode actual throughput gets higher when reducing the W size (although the binary throughput is decreased): this is due to the fact that the weight-bit dimension is serialized in this mode. Changing W does not impact performance in 1×1 convolutions because this dimension is parallelized across the Blocks in each RBE Core and hence only affects the Core's utilization. Overall, the highest 1×1-bit throughput achieved is ∼7100 1×1-bit Top/s in the W=8, I=4 configuration of the 3×3 convolution mode. The highest throughput, on the other hand, is of 571 Gop/s in the W=2, I=4 configuration of 3×3 convolution.

3) RISC-V & RBE Performance/Efficiency: Fig. 14 collects overall speedups for four tasks: floating-point 2048-point FFT, 8-bit 1×1 pointwise and 3×3 convolutional layers including batch-normalization on a 9 × 9 × 64 output space with 64 input channels, tensor addition of two 9 × 9 × 64 tensors. Results are shown for the baseline `XpulpV2` core in the SOC, for a single CLUSTER core, for the full CLUSTER and, where

[3] Configurations with non-power-of-two I are omitted due to space reasons, but follow the same trends highlighted in Fig. 13 and in the discussion.

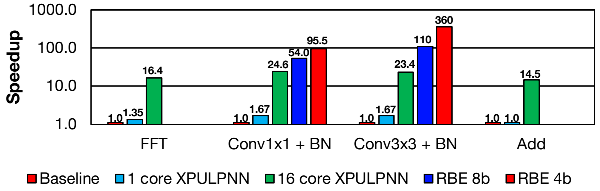


Fig. 14. Speedup of AI and non-AI tasks on MARSELLUS CLUSTER vs execution on MARSELLUS SOC.

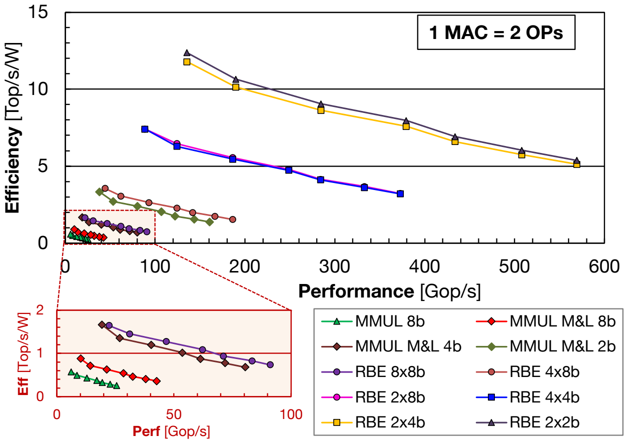


Fig. 15. Energy efficiency versus performance for 3×3 convolutions on RBE and MMUL on RISC-V cores measured on MARSELLUS.

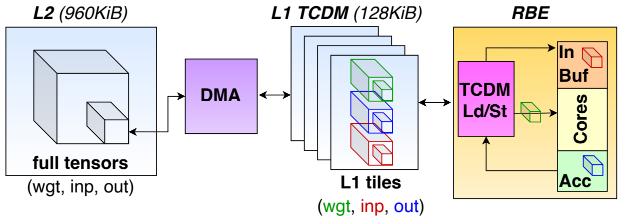


Fig. 16. DNN execution with data tiling on MARSELLUS.

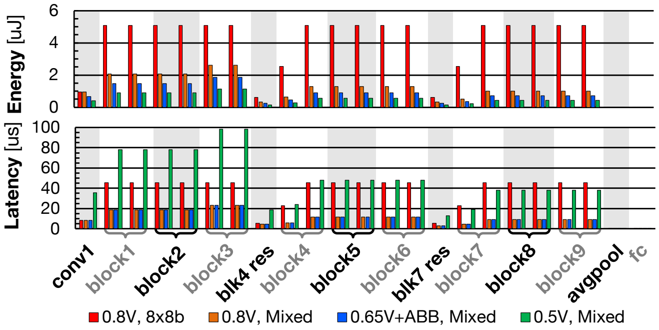


Fig. 17. Layer-wise breakdown of latency and energy consumption of an end-to-end ResNet-20 network on the CIFAR-10 dataset running on MARSELLUS with 8-bit and mixed precision quantization for different operating points.

possible, for execution with RBE (8- and 4-bit). We note that this speedup can be reduced in pathological conditions; for example, a Conv1×1 on a single input channel will be ∼2× faster on the RISC-V cluster than on RBE.

Fig. 15 gives an overview of the CLUSTER-level measured energy efficiency versus achieved performance without applying ABB, while sweeping frequency and supply voltage as indicated in Fig. 9. Each curve reflects measurements at different operating points on one of three different benchmarks: parallel RISC-V matrix multiplication (MMUL); parallel RISC-V MMUL exploiting the `Xpulpnn` extensions (MMUL M&L); and RBE-based 3×3 convolution kernels. All measurements include the full CLUSTER power. The baseline MMUL kernel achieves a performance of 25.45 Gop/s and efficiency of 250 Gop/s/W in the nominal operating point (0.8 V), which scales to 6.06 Gop/s and 580 Gop/s/W, respectively, when downscaling V_{DD} to 0.5 V. The architectural improvements introduced with `Xpulpnn` and the MAC&LOAD mechanism improve performance and efficiency by 67% and 51%, respectively. Introducing aggressive quantization at 4-bit and 2-bit leads to more savings: 3.2× and 2.9× in performance and efficiency, respectively, comparing the MAC&LOAD 4-bit versus the MMUL 8-bit baseline; 6.3× and 5.7×, respectively, when considering the 2-bit MAC&LOAD version.

The highest-precision (8×8-bit) RBE configuration has a throughput of 91 Gop/s and an efficiency of 740 Gop/s/W in the nominal operating point. Voltage downscaling boosts efficiency up to 1.64 Top/s/W at the expense of a significant loss of performance (down to 22 Gop/s). In the RBE case, aggressive quantization can be used with considerable freedom; Fig. 15 restricts analysis to power-of-two configurations where $W \leq I$, which are the most common in quantized CNNs. When scaling W and I to the minimum of 2 bits, performance and efficiency are maximized, yielding 569 Gop/s and 5.37 Top/s/W, respectively, in the 0.8 V operating point and 136 Gop/s and 12.36 Top/s/W, respectively, in the 0.5 V operating point.

IV. MAPPING DNNS ON MARSELLUS

To deploy DNNs on MARSELLUS, we exploit a modified version of the QuantLab open-source library[4] built on PyTorch for quantization. QuantLab exports a fully quantized Open Neural Network Exchange (ONNX) graph which can then be mapped on MARSELLUS by the back-end DORY [29] tool. DNN tensors need to be tiled between the various levels of the memory hierarchy, which are explicitly managed [29],

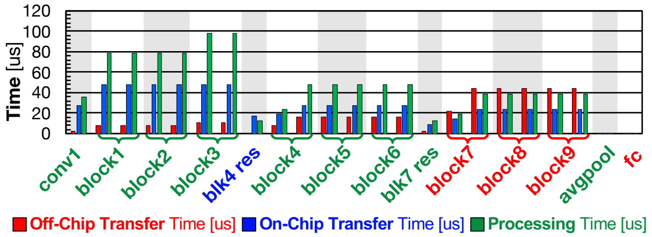


Fig. 18. Detail of ResNet-20/CIFAR-10 in the 0.5 V mixed precision configuration, showing latency of off-chip and on-chip transfers and processing (compute + tiling overheads). Latencies are fully overlapped and thus the tallest bar in each group defines the latency of a layer. Red/blue/green-labeled groups of layers are off-chip/on-chip/compute dominated, respectively.

[4] https://github.com/pulp-platform/quantlab

implementing the mechanism shown in Fig. 16. With data tiling, the execution of the full layers is split in smaller chunks that can fit the available L1 TCDM memory budget (128 KiB). The CLUSTER DMA loads each weight and input activation tile from the L2 memory into the L1 TCDM memory where the RISC-V cores and RBE can work on them. Similarly, the final output activations are stored by the DMA from the L1 into the L2 memory. A double-buffering mechanism is used to perform DMA transfers autonomously, while the cores and/or RBE are computing.

We deployed layer-by-layer an end-to-end ResNet-20 trained on the CIFAR-10 dataset on MARSELLUS, focusing on exploiting the RBE accelerator. Off-chip memory accesses are modeled using an analytical model of I/O obtained from data of a previous prototype using the same technology node [13]. Exploiting the HAWK [30] quantization scheme, it is possible to quantize the network weights at arbitrary precisions (with 2-, 3-, 6-, 8-bits) and activations at 4- and 8-bit activations; as reported by Dong *et al.*, this quantization-aware training scheme leads to a negligible accuracy loss (from 92.4% to 92.2%). Exploiting the bit-flexible DNN support from RBE, this aggressive quantization scheme yields significant energy efficiency improvement, saving 68% of the execution energy, down to ∼28 µJ.

The aggressive voltage scaling and body biasing capabilities of MARSELLUS enable further substantial energy savings by either lowering the supply voltage to 0.65 V and activating ABB (down to ∼21 µJ), or exploiting aggressive voltage scaling to 0.5 V without ABB (down to 12 µJ). The former technique has the advantage of inducing no performance penalty, whereas aggressive voltage scaling yields better energy efficiency but with 4× higher execution time. Fig. 17 details the layer-level performance and efficiency for this network in four operating points/precision configurations, while Fig. 18 shows the detailed latency of off-chip (L3/L2), on-chip (L2/L1) and execution (RBE compute + tiling overheads). Depending on the arithmetic intensity of each layer (number of operations performed per byte of data transferred), DMA transfers can be fully overlapped with execution (green labels) or bound by on-chip L2-L1 traffic (blue labels) or external L3-L2 traffic (red labels). In the latter two cases, the overhead paid is simply the difference between the DMA transfer time and the execution time. Overall, architectural heterogeneity and dynamic ABB enable opportunistically exploiting different key techniques for efficiency depending on the application needs.

V. STATE-OF-THE-ART COMPARISON

A. ABB methods in the State-of-the-Art

Table I compares the ABB strategy employed in the MARSELLUS prototype with other works in the SoA. Most techniques achieve power savings in the order of 20–30%; however, many of these works [20], [33], [34] target very simple prototypes where ABB is employed to regulate a simple digital core. Rossi et al. [31] focus on a more complex architecture comprising a 4-core OpenRISC cluster and 64 KiB of memory (which can be seen as a much simpler iteration of MARSELLUS's CLUSTER). Finally, SleepRunner [32] is a complete micro-controller unit (MCU) with a Cortex-M0 core, focusing on ultra-low-voltage and power execution. Among these, MARSELLUS stands out as is it is arguably the most complex system on which ABB is applied.

Moreover, all comparable ABB methods, except for SleepRunner, apply their techniques due to offline analysis or after empirical trial-and-error. SleepRunner and MARSELLUS are unique in featuring on-chip techniques to automatically tune ABB according to runtime requirements. In the case of SleepRunner, this happens with a technique caled Unified Frequency/Biasing Regulation (UFBR), which is based on a centralized ring oscillator within the regulator to track the actual operating frequency with respect to the set point, therefore compensating process-voltage-temperature (PVT) variations. MARSELLUS's technique is unique as, thanks to OCMs, it can track the operating frequency in an application-specific way.

B. SoCs targeted at AI-IoT

The MARSELLUS SoC is positioned in the SoA as part of an industry trend toward architectural heterogeneity to cope with AI-IoT workloads. Commercial devices following this pattern include, for example, Analog Devices MAX78000 (which combines a Cortex-M4 MCU with a custom Neural Processing Unit (NPU))[5], the Syntiant NDP120 (Cortex-M0 MCU + Syntiant Core 2 NPU)[6], the upcoming STMicroelectronics STM32N6 (Cortex-M class MCU + Neural-Art NPU)[7], the Alif Semiconductor Ensemble family (Cortex-M55 + Ethos-U55 NPU[8]), and the GreenWaves Technologies GAP9 (RISC-V MCU + 9× RISC-V DSP cluster + NE16 Neural Engine)[9]. Except for the latter, all of these devices dedicate most silicon area to DNN acceleration, leaving non-ML workloads to execution on the microcontroller. Both MARSELLUS and the commercial GAP9 SoC take a different approach, employing efficient (but inflexible) hardware acceleration engines (RBE and NE16, respectively) and more flexible (but less efficient) parallel acceleration engines (the RISC-V clusters). The reason for this choice is that, by Amdahl's law, even if a small percentage of the original non-accelerated workload is non-ML, with hardware acceleration this can become a major performance bottleneck.

We focus quantitative comparisons on other non-commercial research prototypes. Table II compares MARSELLUS with four recently presented SoA SoCs targeted at emerging AI-IoT applications: Vega [13], an AI-IoT 10-core SoC (precursor to the GAP9 SoC) with legacy Convolutional Neural Network (CNN) acceleration capabilities; SamurAI [14], a single-core RISC-V microcontroller with a powerful embedded DNN accelerator; DIANA [15], a SoC with hybrid Analog in-memory compute (AiMC) and digital hardware acceleration capabilities; and QNAP [16], an 8-bit DNN-dedicated ASIC with zero-skipping capabilities. All SoCs are fabricated in 22 nm or 28 nm technology nodes, reducing the impact of pure technology scaling on the comparison.

[5] https://www.analog.com/en/products/max78000.html

[6] https://www.syntiant.com/ndp120

[7] https://blog.st.com/stm32n6/

[8] https://alifsemi.com/ensemble/

[9] https://greenwaves-technologies.com/gap9_processor/

TABLE I
ABB METHODS IN THE STATE-OF-THE-ART (SOA).

	Technology Node	**Prototype**	**Area**	**Best power gain**	**Automatic tuning method**
Moursy et al. [20]	22nm FDX	Cortex-M4F (core+memory)	2 mm	-19.9%	On-Chip Monitors + ABB-generator
Rossi et al. [31]	28nm FD-SOI	4-core PULP cluster	3 mm^2	-43% (sleep)	None
SleepRunner [32]	28nm FD-SOI	Cortex-M0 MCU	0.6 mm^2	-	Unified Frequency/Bias Regulators
Akgul et al. [33]	28nm FD-SOI	32-bit VLIW DSP	-	-17%	Offline software
Quelen et al. [34]	28nm FD-SOI	0.1-2mm2 digital core	2 mm^2	-32%	On-Chip Monitors + ABB-generator
Marsellus (our work)	22nm FDX	17 RISC-V + RBE	2.42 mm^2	-30%	On-Chip Monitors + ABB-generator

TABLE II
COMPARISON OF MARSELLUS WITH RELATED WORK.

	Vega [13]	SAMURAI [14]	DIANA [15]	QNAP [16]	MARSELLUS (this work)
Technology	22nm FDX	28nm FD-SOI	22nm FDX+AIMC	28nm	22nm FDX
Die Area	10mm^2	4.5mm^2	10.24mm^2	1.9mm^2	18.7mm^2 (2.42mm^2) [a]
Applications	IoT GP+DNN	IoT GP+DNN	AI-IoT	DNN ASIC	IoT GP+DNN + AI-IoT
SRAM	128KiB L1 + 1.6MiB L2	464KiB	896KiB	206KiB	128KiB L1 + 1MiB L2
Cores	10×RV32IMCFXpulp + HWCE	1×RV32IMCFXpulp + digital Accel	1×RV32IMCFXpulp + digital Accel + AIMC	digital Accel	16×RV32IMCFXpulpnn + 1× RV32IMCFXpulp + RBE
INT precisions	8,16,32	8,16,32	2,4,8,16,32	8	2,4,8,16,32, RBE: 2-8
FP precisions	FP32, FP16, BF16	-	-	-	FP32, FP16, BF16
Supply Voltage	0.5-0.8V	0.45-0.9V	0.5-0.9V	0.6-0.9V	0.5-0.8V
Max. Frequency	450MHz	350MHz	320MHz	470MHz	420MHz
Power Range	1.7μW-49.4mW	6.4μW-96mW	10mW-129mW	19.4-131mW	12.8mW-123mW
Best SW (INT) Perf	15.6 Gop/s [b]	1.5 Gop/s [c]	-	-	**180 Gop/s** [d] (2×2b, 0.8V+ABB)
Best SW (INT) Area Eff	1.56 Gop/s/mm^2 [b]	0.33 Gop/s/mm^2 [c]	-	-	**9.63 Gop/s/mm^2** [d] (2×2b, 0.8V+ABB)
Best SW (INT) Energy Eff	614 Gop/s/W @ 7.6 Gop/s [b]	230 Gop/s/W @ 110 MOp/s [c]	-	-	**3.32 Top/s/W** @ 19 Gop/s [d] (2×2b, 0.5V)
Best SW (FP16) Perf	3.3 Gflop/s [b]	-	-	-	**6.9 Gflop/s** [d] (0.8V+ABB)
Best SW (FP16) Area Eff	0.33 Gflop/s/mm^2 [b]	-	-	-	**0.37 Gflop/s/mm^2** [d] (0.8V+ABB)
Best SW (FP16) Energy Eff	129 Gflop/s/W @ 1.7 Gflop/s [b]	-	-	-	**207 Gflop/s/W** @ 3.1 Gflop/s [d]
Best HW-Accel Perf	32.2 Gop/s	36.0 Gop/s	digital: 180 Gop/s AIMC: **29.5 Top/s**	140 Gop/s	**637 Gop/s** (2×2b, 0.8V+ABB)
Best HW-Accel Area Eff	3.22 Gop/s/mm^2	8.0 Gop/s/mm^2	digital: 17.6 Gop/s/mm^2 AIMC: **2.9 Top/s/mm^2**	**73.7 Gop/s/mm^2**	34.1 Gop/s/mm^2 (2×2b, 0.8V+ABB)
Best HW-Accel Energy Eff	1.3 Top/s/W @ 15.6 Gop/s	1.3 Top/s/W @ 2.8 Gop/s	digital: 4.1 Top/s/W AIMC: **600 Top/s/W**	**12.6 Top/s/W** @ 140 Gop/s (8b)	**12.4 Top/s/W** @ 136 Gop/s (2×2b, 0.5V)
ResNet-20/CIFAR Eff	-	-	AIMC: **14.4 Top/s/W**	-	6.38 Top/s/W (RBE mixed)
ResNet-20/CIFAR Lat [g]	-	-	1.26ms	-	**1.05ms**
ResNet-18/ImageNet Eff	-	-	**19 Top/s/W**	12.1 Top/s/W [f]	5.83 Top/s/W (RBE 4×4b)
ResNet-18/ImageNet Lat [g]	-	-	**6.15ms**	24.8ms	48ms

[a] CLUSTER area in brackets [b] architecture with 8×RISC-V cores sharing 4×FPUs [c] architecture with 1×RISC-V core w/o FPUs [d] architecture with 16×RISC-V cores sharing 8× FPUs [f] zero-skipping [g] at best efficiency operating point.

The contribution of MARSELLUS stands out in several dimensions. First, MARSELLUS provides significantly larger software performance while keeping within a power range comparable with that of the other SoCs. This is due to the combined effects of the 16-core architecture, aggressive 2-bit quantization with `Xpulpnn`, and ABB, which are features available only in MARSELLUS. Second, while most of the SoCs in the table employ heterogeneous architectures, MARSELLUS is more completely pushing in the direction of combining high-performance software with flexible acceleration, thanks to the fine-grain precision support in RBE. Combined, these effects yield SoA-leading results in terms of performance, area efficiency, and energy efficiency for software (INT and FP); hardware-accelerated execution is leading or equivalent in terms of both performance and energy efficiency compared to the other digital hardware-accelerated SoCs, and it is second in terms of area efficiency after QNAP, which is not surprising as the latter is a dedicated ASIC. DIANA's AiMC accelerator provides (as could be expected) peak performance and efficiency metrics that are ∼100× better than digital accelerators; however, compared with digital accelerators, AiMC units are significantly harder to utilize efficiently, as also noted by the architects of DIANA [35]: this means that such metrics are difficult to compare fairly with those of digital accelerators.

To deepen the perspective of the comparison between the proposed architectures, Table II also includes performance and efficiency achieved by the various SoCs on two common benchmarks for CNN accelerators, namely, ResNet-20/CIFAR (using the scheme discussed in Section IV) and ResNet-18/ImageNet (targeting a full HAWQ-quantized 4-bit network, achieving 68.5% on ImageNet[10]). Interestingly, these results confirm that the difference in efficiency between digital and AiMC accelerators is significantly reduced, from the theoretical ∼50–100× to a 2.3–5× advantage in practice. Similar considerations apply to peak performance. This is due to limited utilization of the AiMC module in practical layers, as well as overheads from digital periphery. In terms of performance, in the case of ResNet-20/CIFAR inference, MARSELLUS can actually deliver 20% faster runtime due to the same effects. On ResNet-18/ImageNet the results of DIANA are significantly better (likely due to higher utilization of the AiMC array compared to the ResNet-20/CIFAR case); QNAP delivers the second-best result. Overall, our work is very competitive even when considering architectures with aggressive approximation from AiMC and micro-optimizations such as zero-skipping, which are not used in MARSELLUS – while at the same time offering more flexibility as non-hardware-accelerated tasks can be executed in software at SoA-leading performance and efficiency on the ISA-enhanced RISC-V cores.

VI. Conclusion

This work presented MARSELLUS, an advanced AI-IoT SoC fabricated in 22 nm FDX technology combining a heterogeneous architecture with a cluster of 16 RISC-V cores with advanced DSP and AI ISA extensions, fully integrated with a flexible-precision partially bit-serial DNN accelerator. The MARSELLUS SoC can be aggressively voltage- and frequency-scaled to improve energy efficiency, and the dynamic ABB mechanism introduced in this prototype enables fine-grained tuning of performance, and efficiency optimization even without scaling frequency. Fig. 19 summarizes all the efficiency optimization techniques discussed in this work in terms of energy per elementary operation. The combination of architecture improvements, data quantization, and voltage scaling with body biasing or frequency scaling yield a plethora of different options for energy vs flexibility/accuracy trade-offs on the same SoC. MARSELLUS responds to the demand for architectural flexibility and capability to adapt the same computing fabric to diverse tasks, as required by current and future AI-IoT applications.

The baseline Register Transfer Level (RTL) code of the CLUSTER[11] and of the RBE accelerator[12] are released as open-source under a liberal license to foster future research in the area of AI-IoT computing devices.

Acknowledgment

We thank Dolphin Design, France, for collaborating with the implementation and fabrication of the MARSELLUS silicon prototype and all measurements.

[10] https://github.com/Zhen-Dong/HAWQ
[11] https://github.com/pulp-platform/pulp
[12] https://github.com/pulp-platform/rbe

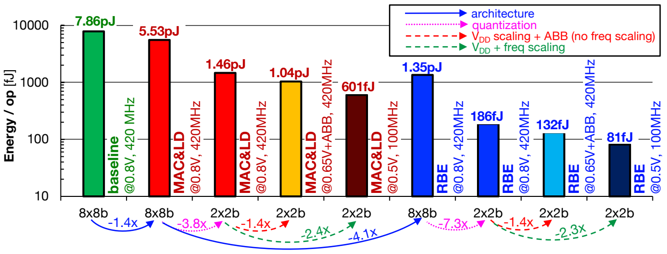


Fig. 19. Summary of energy efficiency optimization techniques available in MARSELLUS.

References

[1] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: analysis, applications, and prospects," *IEEE transactions on neural networks and learning systems*, 2021.

[2] C. Liu, S. Chen, T.-H. Tsai, B. de Salvo, and J. Gomez, "Augmented Reality - The Next Frontier of Image Sensors and Compute Systems," in *2022 IEEE International Solid- State Circuits Conference (ISSCC)*, vol. 65, Feb. 2022, pp. 426–428.

[3] X. Dong, B. De Salvo, M. Li, C. Liu, Z. Qu, H. T. Kung, and Z. Li, "SplitNets: Designing Neural Architectures for Efficient Distributed Computing on Head-Mounted Systems," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 559–12 569.

[4] M. Abrash, "Creating the Future: Augmented Reality, the next Human-Machine Interface," in *2021 IEEE International Electron Devices Meeting (IEDM)*, Dec. 2021, pp. 1.2.1–1.2.11.

[5] P. Tsinganos, B. Cornelis, J. Cornelis, B. Jansen, and A. Skodras, "Deep Learning in EMG-based Gesture Recognition," in *5th International Conference on Physiological Computing Systems*, Apr. 2023, pp. 107–114.

[6] A. Burrello, F. B. Morghet, M. Scherer, S. Benatti, L. Benini, E. Macii, M. Poncino, and D. J. Pagliari, "Bioformers: Embedding Transformers for Ultra-Low Power sEMG-based Gesture Recognition," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Mar. 2022, pp. 1443–1448.

[7] A. Vitale, A. Renner, C. Nauer, D. Scaramuzza, and Y. Sandamirskaya, "Event-driven Vision and Control for UAVs on a Neuromorphic Chip," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. Xi'an, China: IEEE, May 2021, pp. 103–109.

[8] M. O'Connell, G. Shi, X. Shi, and S.-J. Chung, "Meta-Learning-Based Robust Adaptive Flight Control Under Uncertain Wind Conditions," *arXiv:2103.01932 [cs, eess]*, Mar. 2021.

[9] V. Niculescu, L. Lamberti, F. Conti, L. Benini, and D. Palossi, "Improving Autonomous Nano-Drones Performance via Automated End-to-End Optimization and Deployment of DNNs," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 11, no. 4, pp. 548–562, Dec. 2021.

[10] M. Fariselli, M. Rusci, J. Cambonie, and E. Flamand, "Integer-Only Approximated MFCC for Ultra-Low Power Audio NN Processing on Multi-Core MCUs," in *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, Jun. 2021, pp. 1–4.

[11] M. Nagel, M. v. Baalen, T. Blankevoort, and M. Welling, "Data-Free Quantization Through Weight Equalization and Bias Correction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1325–1334.

[12] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental network quantization: Towards lossless cnns with low-precision weights," *arXiv preprint arXiv:1702.03044*, 2017.

[13] D. Rossi, F. Conti, M. Eggiman, A. D. Mauro, G. Tagliavini, S. Mach, M. Guermandi, A. Pullini, I. Loi, J. Chen, E. Flamand, and L. Benini, "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*, vol. 57, no. 1, pp. 127–139, Jan. 2022.

[14] I. Miro-Panades, B. Tain, J.-F. Christmann, D. Coriat, R. Lemaire, C. Jany, B. Martineau, F. Chaix, G. Waltener, E. Pluchart, J.-P. Noel, A. Makosiej, M. Montoya, S. Bacles-Min, D. Briand, J.-M. Philippe, Y. Thonnart, A. Valentian, F. Heitzmann, and F. Clermidy, "SamurAI: A Versatile IoT Node With Event-Driven Wake-Up and Embedded ML Acceleration," *IEEE Journal of Solid-State Circuits*, pp. 1–0, 2022.

[15] P. Houshmand, G. M. Sarda, V. Jain, K. Ueyoshi, I. A. Papistas, M. Shi, Q. Zheng, D. Bhattacharjee, A. Mallik, P. Debacker, D. Verkest, and M. Verhelst, "DIANA: An End-to-End Hybrid DIgital and ANAlog Neural Network SoC for the Edge," *IEEE Journal of Solid-State Circuits*, vol. 58, no. 1, pp. 203–215, Jan. 2023.

[16] H. Mo, W. Zhu, W. Hu, Q. Li, A. Li, S. Yin, S. Wei, and L. Liu, "A 12.1 TOPS/W Quantized Network Acceleration Processor With Effective-Weight-Based Convolution and Error-Compensation-Based Prediction," *IEEE Journal of Solid-State Circuits*, vol. 57, no. 5, pp. 1542–1557, May 2022.

[17] V. Jain, S. Giraldo, J. D. Roose, L. Mei, B. Boons, and M. Verhelst, "TinyVers: A Tiny Versatile System-on-Chip With State-Retentive eM-RAM for ML Inference at the Extreme Edge," *IEEE Journal of Solid-State Circuits*, pp. 1–12, 2023.

[18] M. Blagojević, M. Cochet, B. Keller, P. Flatresse, A. Vladimirescu, and B. Nikolić, "A fast, flexible, positive and negative adaptive body-bias generator in 28nm FDSOI," in *2016 IEEE Symposium on VLSI Circuits (VLSI-Circuits)*, Jun. 2016, pp. 1–2.

[19] R. Gomez, C. Dutto, V. Huard, S. Clerc, E. Bano, and P. Flatresse, "Design methodology with body bias: From circuit to engineering," in *2017 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*, Oct. 2017, pp. 1–4.

[20] Y. Moursy, T. R. Da Rosa, L. Jure, A. Quelen, S. Genevey, L. Pierrefeu, E. Grand, J. Winkler, J. Park, G. Pillonnet, V. Huard, A. Bonzo, and P. Flatresse, "35.2 A 0.021mm2 PVT-Aware Digital-Flow-Compatible Adaptive Back-Biasing Regulator with Scalable Drivers Achieving 450% Frequency Boosting and 30% Power Reduction in 22nm FDSOI Technology," in *2021 IEEE International Solid- State Circuits Conference (ISSCC)*, vol. 64, Feb. 2021, pp. 492–494.

[21] F. Conti, D. Rossi, G. Paulin, A. Garofalo, A. Di Mauro, G. Rutishauer, G. M. Ottavi, M. Eggimann, H. Okuhara, V. Huard, O. Montfort, L. Jure, N. Exibard, P. Gouedo, M. Louvat, E. Botte, and L. Benini, "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)*, 2023, pp. 21–23.

[22] M. Gautschi, P. D. Schiavone, A. Traber, I. Loi, A. Pullini, D. Rossi, E. Flamand, F. K. Gürkaynak, and L. Benini, "Near-Threshold RISC-V Core With DSP Extensions for Scalable IoT Endpoint Devices," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 10, pp. 2700–2713, 2017.

[23] A. Pullini, D. Rossi, G. Haugou, and L. Benini, "μDMA: An autonomous I/O subsystem for IoT end-nodes," in *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*. IEEE, 2017, pp. 1–8.

[24] C. Li, L. Longinotti, F. Corradi, and T. Delbruck, "A 132 by 104 10μm-Pixel 250μW 1kefps Dynamic Vision Sensor with Pixel-Parallel Noise and Spatial Redundancy Suppression," in *2019 Symposium on VLSI Circuits*, Jun. 2019, pp. C216–C217.

[25] C. Jie, I. Loi, L. Benini, and D. Rossi, "Energy-efficient two-level instruction cache design for an Ultra-Low-Power multi-core cluster," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2020, pp. 1734–1739.

[26] F. Montagna, S. Mach, S. Benatti, A. Garofalo, G. Ottavi, L. Benini, D. Rossi, and G. Tagliavini, "A Low-Power Transprecision Floating-Point Cluster for Efficient Near-Sensor Data Analytics," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 5, pp. 1038–1053, May 2022.

[27] F. Conti, P. D. Schiavone, and L. Benini, "XNOR Neural Engine: A Hardware Accelerator IP for 21.6-fJ/op Binary Neural Network Inference," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2940–2951, mar 2018.

[28] B. Mazzoni, S. Benatti, L. Benini, and G. Tagliavini, "Efficient Transform Algorithms for Parallel Ultra-Low-Power IoT End Nodes," *IEEE Embedded Systems Letters*, vol. 13, no. 4, pp. 210–213, Dec. 2021.

[29] 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 Transactions on Computers*, vol. 70, no. 8, pp. 1253–1268, Aug. 2021.

[30] Z. Dong, Z. Yao, A. Gholami, M. W. Mahoney, and K. Keutzer, "HAWQ: Hessian AWare Quantization of Neural Networks With Mixed-Precision," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 293–302.

[31] D. Rossi, A. Pullini, I. Loi, M. Gautschi, F. K. Gürkaynak, A. Bartolini, P. Flatresse, and L. Benini, "A 60 GOPS/W, -1.8V to 0.9V body bias ULP cluster in 28nm UTBB FD-SOI technology," *Solid-State Electronics*, vol. 117, pp. 170–184, Mar. 2016.

[32] D. Bol, M. Schramme, L. Moreau, P. Xu, R. Dekimpe, R. Saeidi, T. Haine, C. Frenkel, and D. Flandre, "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 Journal of Solid-State Circuits*, vol. 56, no. 7, pp. 2256–2269, Jul. 2021.

[33] Y. Akgul, D. Puschini, S. Lesecq, E. Beigné, I. Miro-Panades, P. Benoit, and L. Torres, "Power management through DVFS and dynamic body biasing in FD-SOI circuits," in *Proceedings of the 51st Annual Design Automation Conference*, ser. DAC '14. New York, NY, USA: Association for Computing Machinery, Jun. 2014, pp. 1–6.

[34] A. Quelen, G. Pillonnet, P. Flatresse, and E. Beigné, "A 2.5μW 0.0067mm2 automatic back-biasing compensation unit achieving 50% leakage reduction in FDSOI 28nm over 0.35-to-1V VDD range," in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, Feb. 2018, pp. 304–306.

[35] J. Van Delm, M. Vandersteegen, A. Burrello, G. M. Sarda, F. Conti, D. Jahier Pagliari, L. Benini, and M. Verhelst, "HTVM: Efficient Neural Network Deployment On Heterogeneous TinyML Platforms," in *Proceedings of the 60th Annual Design Automation Conference (DAC'23), to appear.*, San Francisco, 2023.

Francesco Conti (Member, IEEE) received the Ph.D. degree in electronic engineering from the University of Bologna, Italy, in 2016. He is currently a Tenure-Track Assistant Professor with the DEI Department, University of Bologna. From 2016 to 2020, he held a research grant with the University of Bologna and a Post-Doctoral Researcher with ETH Zürich. His research is centered on hardware acceleration in ultra-low power and highly energy efficient platforms, with a particular focus on System-on-Chips for Artificial Intelligence applications. His research work has resulted in more than 70 publications in international conferences and journals and was awarded several times, including the 2020 IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I: REGULAR PAPERS Darlington Best Paper Award.

Gianna Paulin received her BSc and MSc in "Electrical Engineering and Information Technology" from the Swiss Federal Institute of Technology Zürich (ETHZ), Switzerland, in 2017 and 2019, respectively. In 2019 she joined the Integrated Systems Laboratory of ETH Zürich as a PhD candidate. Her research interests include computer architecture and hardware acceleration of deep learning applications targeting both, low power embedded systems and high-performance computing systems.

Angelo Garofalo (Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees in electronic engineering from the University of Bologna, Italy, in 2016 and 2018, and 2021, respectively. He is currently an Assistant Professor with the Department of Electrical, Electronic and Information Engineering (DEI). His main research topic is hardware–software design of ultra-low-power multiprocessor systems on chip for edge AI. His research interests include quantized neural networks, hardware efficient machine learning, in-memory computing, heterogeneous architectures, and fully programmable embedded architectures.

Davide Rossi (Member, IEEE) received the Ph.D. degree from the University of Bologna, Bologna, Italy, in 2012. He has been a Post-Doctoral Researcher with the Department of Electrical, Electronic and Information Engineering "Guglielmo Marconi," University of Bologna, since 2015, where he is currently an Associate 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. He was a recipient of the Donald O. Pederson Best Paper Award 2018, the 2020 IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS Darlington Best Paper Award, and the 2020 IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS Prize Paper Award.

Alfio Di Mauro (Member, IEEE) received the M.Sc. degree in electronic engineering from the Electronics and Telecommunications Department (DET), Politecnico di Torino, in 2016, and the Ph.D. degree with the Integrated System Laboratory (IIS), Swiss Federal Institute of Technology, Zürich, in 2021. His research focuses on the design of digital ultra-low power (ULP) system-on-chip (SoC) for event-driven edge computing.

Georg Rutishauser (Graduate Student Member, IEEE) received the B.Sc. and M.Sc. degrees in electrical engineering and information technology from ETH Zürich, Zürich, Switzerland, in 2015 and 2018, respectively, where he is currently pursuing the Ph.D. degree with the Integrated Systems Laboratory. His research interests include algorithms and hardware for reduced-precision deep learning and their application in computer vision and embedded systems.

Gianmarco Ottavi received the M.Sc. degree in 2019. He is currently pursuing the Ph.D. degree in electronics engineering with the University of Bologna, Italy. He was a Research Fellow with the Department of Electrical, Electronic and Information Engineering (DEI), Bologna, for two years. His research is focused on hardware design for efficient inference in low-power systems, where he developed specialized ISA extensions for RISC-V and system-level implementation of in-memory computing accelerators.

Manuel Eggimann (Member, IEEE) received the M.Sc. degree in electrical engineering and information technology from ETH Zürich, Zürich, Switzerland, in 2018, where he is currently pursuing the Ph.D. degree with the ETH Zürich Integrated Systems Laboratory. His research interests include low-power hardware design, edge computing, and very-large-scale integration (VLSI). Mr. Eggimann was a recipient of the Best Paper Award at the 2019 IEEE 8th International Workshop on Advances in Sensors and Interfaces.

Hayate Okuhara (Member, IEEE) received the Ph.D. degree from Keio University, Kanagawa, Japan, in 2018. He has been a Postdoctoral Researcher with the Department of Electrical, Electronic and Information Engineering "Guglielmo Marconi," University of Bologna, Bologna, Italy till 2021 and is currently with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore. His research interest includes low-power VLSI system design.

Luca Benini (Fellow, IEEE) holds the chair of digital Circuits and systems at ETHZ and is Full Professor at the Università di Bologna. He received a PhD from Stanford University. Dr. Benini's research interests are in energy-efficient parallel computing systems, smart sensing micro-systems and machine learning hardware. He is a Fellow of the ACM and a member of the Academia Europaea. He is the recipient of the 2016 IEEE CAS Mac Van Valkenburg award, the 2020 EDAA achievement Award, the 2020 ACM/IEEE A. Richard Newton Award and the 2023 IEEE CS E.J. McCluskey Award.