Running Efficiently CNNs on the Edge Thanks to Hybrid SRAM-RRAM In-Memory Computing

Marco Rios, Flavio Ponzina, Giovanni Ansaloni, Alexandre Levisse and David Atienza
Embedded Systems Laboratory (ESL), EPFL, Switzerland
{marco.rios, flavio.ponzina, giovanni.ansaloni, alexandre.levisse, david.atienza}@epfl.ch

***Abstract*—The increasing size of Convolutional Neural Networks (CNNs) and the high computational workload required for inference pose major challenges for their deployment on resource-constrained edge devices. In this paper, we address them by proposing a novel In-Memory Computing (IMC) architecture. Our IMC strategy allows us to efficiently perform arithmetic operations based on bitline computing, enabling a high degree of parallelism while reducing energy-costly data transfers. Moreover, it features a hybrid memory structure, where a portion of each subarray, dedicated to storing CNN weights, is implemented as high-density, zero-standby-power Resistive RAM. Finally, it exploits an innovative method for storing quantized weights based on their value, named Weight Data Mapping (WDM), which further increases efficiency. Compared to state-of-the-art IMC alternatives, our solution provides up to 93% improvements in energy efficiency and up to 6x less run-time when performing inference on Mobilenet and AlexNet neural networks.**

***Index Terms*—SRAM, RRAM, In-Memory Computing, CNN, Edge computing**

I. Introduction

The unprecedented success of deep learning algorithms is fostering a revolution in Artificial Intelligence (AI). Nowadays, many cloud-based applications, ranging from social media recommendations to business informatics, are AI-driven. Edge Intelligence is predicted to parallel this success by enabling AI applications on smart Internet-of-things (IoT) devices, paving the way for novel solutions in many exciting domains, such as, enhanced reality and personalized assistance [1].

Such applications (at least their inference phase) are not amenable to be executed in the cloud, because (a) they must present a high level of responsiveness, (b) they rely on a huge volume of locally-acquired data, whose streaming from device to cloud is extremely energy costly, and (c) they often process personal data, for which security and privacy are major concerns. Hence, the vast majority of Edge Intelligence solutions postulate that algorithms are executed locally. In turn, the high workloads characterizing state-of-the-art deep learning methods, such as CNNs, calls for innovative solutions at the hardware and architectural level to boost energy efficiency on constrained edge devices [2].

Among such strategies, In-Memory Computing (IMC) has recently attracted considerable interest in the research community. IMC effectively exploits data locality and regular computational patterns exposed by CNNs to compute arithmetic operations in-situ, thereby bypassing processor pipelines and drastically reducing memory transfers without altering memory hierarchies [3]. In IMC architectures, bit-wise operations are performed by activating more than one word during the same memory access, a technique referred to as Bitline (BL) computing [4]. Additional logic is then employed to implement arithmetic operations, such as additions and multiplications. At the hardware level, BL computing has been studied both considering SRAM [5]–[7] and RRAM [8], [9] memories.

As opposed to these efforts, we here present for the first time, a hybrid IMC architecture, in which RRAM and SRAM memory partitions are integrated in each memory subarray. Our design choice is motivated by the different run-time access patterns that characterize the data representing weights and activations of CNNs. On one hand, since the same kernels are re-used many times, memory locations storing weights are seldomly written to. Thus, as in this case the small write endurance and high write energy of RRAM [10] is less of a concern, we implement weight memories as RRAM cells to exploit their zero leakage power to reap important energy benefits. On the other hand, we employ SRAM for storing activations, as they frequently change values. We validate such a concept on a modified version of the BLADE architecture [6], taking advantage of the Local Group computing illustrated therein to enable SRAM-RRAM IMC operations in BLs.

Moreover, we observe that the number of weights of CNN models can number in the millions, hence vastly exceeding the RRAM size that can be implemented in practice inside memory arrays. One solution is to continuously load the required weights in the RRAM, but such an approach would rapidly deplete the RRAM write endurance. Consequently, instead we introduce a novel strategy, that we name Weight Data Mapping (WDM), in which all the possible representations of quantized weights are stored and mapped. Hence, in our implementations, weights are directly accessed by their value, reducing data transfers while at the same time increasing energy efficiency and performance. Overall, compared to an homogeneous SRAM-based equivalent architecture, our hybrid SRAM-RRAM design results in a performance gains of 6.21x and 6.89x when executing inference on Alexnet [11] and Mobilenet [12], respectively, with an increase in the energy efficiency of 93% in both cases. In summary, the main contributions of this paper are:

This work has been partially supported by the EC H2020 WiPLASH project (GA No. 863337), the ERC Consolidator grant COMPUSAPIEN (GA No. 725657), and by the Swiss NSF ML-Edge Project (GA No. 182009).

978-3-9819263-5-4/DATE21/©2021 EDAA

- We propose a new hybrid IMC architecture, which integrates RRAM and SRAM memory cells. We validate the resulting circuit-level design, showcasing the compatibility between the two technologies.
- We introduce Weight Data Mapping (WDM) as a novel strategy to map, access and perform IMC operations in-situ, drastically reducing data transfers.
- We evaluate the efficiency of our IMC solution, showing that a 128-subarrays design can process up to 22 images/second, while requiring only 4.73 mJ/image for MobileNet. For Alexnet, a 128-subarrays design requires 9.7 mJ/image and processes up to 11 images/second.

The remainder of the paper is organized as follows. Section II provides the necessary background on IMC architectures, RRAM technologies, and CNNs. Section III presents the proposed architecture integrating SRAM and RRAM bitcells, including layout considerations, as well as the WDM technique. Section IV presents the electrical validation of the architecture, while Section V details the methodology used to benchmark applications. Next, Section VI presents experiments highlighting the benefits deriving from our hybrid IMC design. Finally, section VII summarizes the main conclusions of the paper.

II. Background

A. In-Memory Computing Architectures

IMC architectures are extremely promising for supporting edge AI applications, as they enable highly efficient Single Instruction Multiple Data (SIMD) parallelism directly inside the memory hierarchy, while requiring a low area compared to computation-specific accelerators [3]. Research is vibrant in this field: notable works include Jeloka et al. [4], which introduced a novel content addressable memory able to perform bitwise operations, and Akyel et al. [5], whose DRC2 architecture also executes arithmetic operations by enriching the memory peripheral circuit.

The above-mentioned works consider SRAM memory cells. Implementations based on emerging Resistive Random Access Memories (RRAM) are instead proposed by Jain et al. [13] adopting magnetic memory. Haj-Ali et al. [8] and Bocquet et al. [9], instead present architectures based on Resistive RAM to implement parallel multiplications and binarized neural networks, respectively.

Conversely to this work, all these previous designs employ a homogeneous memory structures, either based on SRAMs or RRAMs. We instead show that hybrid structures, tailored to data access patterns, can leverage the benefits of both (e.g.: the high endurance of SRAM and the absence of leakage of RRAMs), while minimizing their downsides, and that they can be effectively co-integrated. Our solution builds on the previous work of Simon et al. [6], [7]. Their BLADE architecture presents an organization based on Local Groups (LGs), homogeneously realized in SRAM, that enables high-performance operations (up to 2.2GHz@1V). It performs SIMD bitwise operations (OR, AND, XOR), additions and shifts, which can be chained to realize multiplications. With respect to a state-of-the-art hardware accelerator (ARM's NEON), BLADE achieves performance/energy gains up to 6x when running edge AI benchmarks [6].

B. Emerging Non-Volatile Memories

RRAM exploits material resistivity control to implement persistent storage, i.e.: memories that retain their state even in the absence of any applied voltage. Different families of RRAM technologies have been introduced in recent years, such as Magnetic Random Access Memory (MRAMs), Phase Change Memory (PCMs), and filamentary-based RRAMs (ReRAMs). The absence of leakage power would make these technologies ideal candidates for replacing traditional SRAMs and DRAMs at different levels of the memory hierarchy [10], [14]. Nonetheless, their high write currents (compared to SRAM) and low endurance are, at present, limiting their adoption in a general-purpose context. These drawbacks are less of a concern in our case, as we dedicate RRAM to storing CNN weights, and we propose a new strategy that we call WDM to only rarely re-write the weight memory at run-time.

The conventional way to co-integrate RRAM and CMOS technologies is by creating arrays of 1-Transistor 1-Resistance (1T1R) bitcells, which enable 3x to 4x higher integration density than 6T SRAM memories [15], [16]. RRAM memories can be programmed between several states called Low Resistance State (LRS) and High Resistance State (HRS) through programming phases which are energy hungry ($>$ pJ/bit). On the other hand, read operations consist into sensing the resistance value and are in the same order of magnitude than SRAM reads. Due to all these reasons, added to their relative low-integration cost compared to conventional embedded non-volatile memories [10] RRAM technologies are very appealing when realizing resource-constrained edge AI systems.

C. CNNs at the edge

CNNs are composed by a succession of layers, each abstracting higher-level features from a data source in order to interpret it [17]. Convolutional layers slide three-dimensional filters on the feature maps at their input to compute sets of three-dimensional outputs. The convolution operation is typically followed by a non-linear activation function. Pooling layers are used to realize sub-sampling operations. Finally, fully connected layers combine all the data at their input, and are usually employed in the last stages of a CNN.

CNNs are being increasingly successful in various AI tasks, from image classification to natural language processing [18]. At the same time, they are also becoming deeper (i.e., presenting an increasing number of layers), and more complex, with recent implementations such as ResNet-50 presenting over 23 million trainable parameters [19]. While more compact alternatives, such as MobileNet [12], have been proposed, their execution, even when only considering the inference phase, is still very computing-intensive, making their deployment on edge devices an open research topic [20].

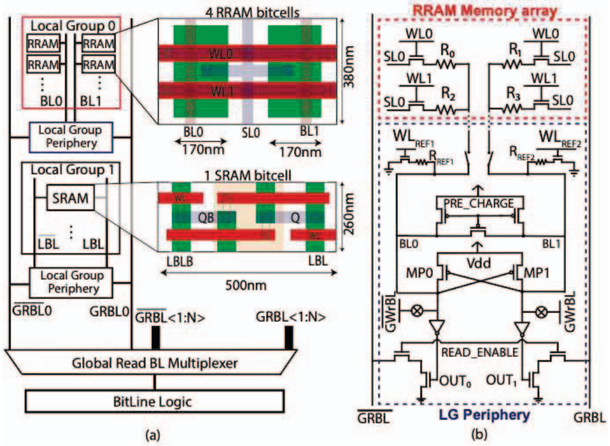


Fig. 1. (a) Block diagram of the proposed Hybrid SRAM-RRAM IMC architecture. It presents our proposed memory organization, the bitcells layout and the pitch matching between SRAM and RRAM-based Local Groups (LGs). (b) Architecture of a RRAM-based LG.

A well-known approach to address this challenge is that of quantization. Indeed, representing weights and features in the fixed point domain with a reduced number of bits only marginally impacts accuracy in most cases [21], [22], while dramatically decreasing storage and computational requirements. Interestingly, quantized CNNs are very well suited for being executed in a IMC architecture due to two reasons. First, techniques such as SIMD parallelization and weight sharing can be easily implemented by adopting multiple memory subarrays. Second, the regular computations characterizing CNN layers only requires a small amount of logic, which can be implemented at the memory periphery without incurring in huge overheads.

III. Proposed Hybrid SRAM-RRAM IMC

A. RRAM-based LG

Bitcell Design. Our hybrid memory architecture is composed by a varying number of heterogeneous subarrays, organized, similarly to [6], in Local Groups (LG). However, conversely to this previous work, we consider two LG types, either SRAM-based or RRAM-based. Irrespectively of their implementations, LGs must be physically aligned in order to share the same set of Global Read Bit Lines (GRBLs), as shown on Figure 1(a). Hence, considering a 28nm CMOS technology node and high density SRAM rules ($0.127\mu m^2$) as in [6], RRAM LGs must be pitched in a 500nm width, i.e. the width of SRAM cells. Furthermore, RRAM bitcells must present the same poly-silicon gate orientation of SRAM ones.

As shown in Figure 1(a), we abide to these constraints by pitch-matching 4 RRAM bitcells in a 2x2 configuration to a single SRAM one, since the minimum width for a RRAM bitcell in the considered technology is 180nm. We employ the additional available space, as determined by pitch matching, by increasing the access transistor width beyond the minimum size to 170nm, thus increasing the read margin and easing programming operations. By merging the access transistor sources together, we achieve a 380nm height for 2 bitcells. Therefore, the surface of RRAM bitcells is 0.0475 μm^2, 2.74x denser than the SRAM ones.

Local Group Design. Figure 1(b) presents a circuit-level view of our proposed RRAM-based LG, including its periphery circuit. Besides the memory array, it contains a resistance reference for read operations, a pre-charge circuit as well as a Sense Amplifier (SA), two read ports for IMC operations, and two transmission gates to connect the BLs to the write amplifier.

Write operations are performed by activating one WL and biasing both the SL and one BL to ground. The other BL is biased to the programming voltage (VPROG). The opposite is done for reset operation by grounding the selected BL and biasing the other BL and the SL to VPROG. Current control is managed on the write amplifier located inside the array periphery.

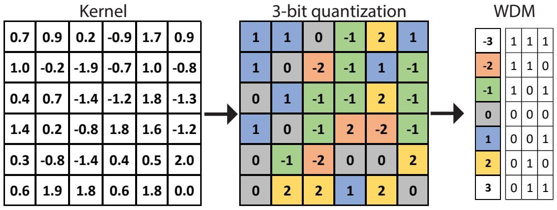
Both read and IMC operations are performed by comparing one RRAM bitcell to one R_{ref}. Two bitcells inside the same LG share the same WordLine (WL), thereby, in Figure 1-b, R0 and R2 are compared to R_{ref2} while R1 and R3 to R_{ref1}. To do so, only the selected BL is connected to one side of the SA, while the other one is connected to the opposite R_{ref} using two NMOS transistors (represented as switches in Figure 1-b. To discriminate between the two sides of a RRAM array during read operation, we inversely encode the data stored in one of the two BLs. Such approach enables compatibility between the single-ended RRAM LGs and the differential SRAM LGs.

The SA is composed of two PMOS transistors and two inverters. The transistors MP0 and MP1 are both deactivated after the pre-charge process. Once the WL is activated, the nodes BL0 and BL1 subsequently discharge. The side which the BL discharges faster opens the other side's PMOS, connecting it to VDD and forcing the BL to stay activated, latching the SA.

IMC operations: IMC operations are performed by accessing at the same time two WLs in two different LGs. Read ports connect local to global bitlines, hence computing bitwise AND and NOR operations among the accessed words on the BL and $\overline{BL}$ wires, respectively. IMC operations can be performed between one RRAM LG and one SRAM LG, as they present compatible read scheme and performance. Additional logic under the array is used to derive arithmetic operations (such as additions and shifts) from bitwise ones. Finally, repeated add-and-shifts are employed to perform multiplications. The number of clock cycles required for a multiplication equals 2x the word size, as one cycle is employed for the shift-and-add operation, and another one to write the result back in memory.

B. Weight Data Mapping

Storing all weights of a CNN model in RRAM-based LGs would require a huge memory capacity, and it may be unfeasible in practice within the resource constraints of edge devices. Therefore, to program and access the learned CNN



Kernel

0.7	0.9	0.2	-0.9	1.7	0.9
1.0	-0.2	-1.9	-0.7	1.0	-0.8
0.4	0.7	-1.4	-1.2	1.8	-1.3
1.4	0.2	-0.8	1.8	1.6	-1.2
0.3	-0.8	-1.4	0.4	0.5	2.0
0.6	1.9	1.8	0.6	1.8	0.0

3-bit quantization

1	1	0	-1	2	1
1	0	-2	-1	1	-1
0	1	-1	-1	2	-1
1	0	-1	2	-2	-1
0	-1	-2	0	0	2
0	2	2	1	2	0

WDM

-3	1	1	1
-2	1	1	0
-1	1	0	1
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1

Fig. 2. Encoding a 6x6 Kernel using 3-bit quantization and Weight Data Mapping.

parameters we instead adopt a novel Weight Data Mapping strategy (WDM), which greatly reduces memory requirements. We observe that, in quantized CNN models, quantization reduces the admissible weight values to a small set of size 2^q, where q is the post-quantization bitwidth. Therefore, we directly store admissible values in RRAM, and read them when required for IMC operations.

Figure 2 exemplifies our approach. Therein, the 6x6 kernel, where data is represented in floating point, is first quantized to a three-bits signed representation. Hence, the elements are cast to integers in the range [-3, 3]. Then, only the resulting post-quantization values are stored in RRAM. In the Figure 2, it can be notices that the values {-3, 3} are unused, thus the WDM scheme requires to reprogram only the remaining 5 values. This behaviour is common also when considering larger values of q. For $q = 8$ (the quantization level we adopt in the rest of the paper), AlexNet trained on the CIFAR-10 data set employs only 90 of 256 possible values, while in the case of $q = 16$ such ratio further drops to 1097 values out of 65536. Additional energy savings are therefore achieved by only programming in RRAMs the weight values used for inference.

IV. Electrical Validation

We validated the functionality and integration of the hybrid design described in Section III-A by implementing and simulating it using the 28nm bulk CMOS PDK from TSMC.

RRAM LG Simulation. We simulated the RRAM LG from Figure 1 using ideal resistance models connected to access NMOS transistors. Such an approach provides technology agnosticity as RRAM technologies usually exhibit an almost ideal ohmic behavior while biased under a given critical voltage. Then, we introduced a parasitic capacitance for the BLs (as proposed by [6]) and the RRAM ($10aF$ per bitcell). Also, we considered the resistance reference (R_{ref}) required to perform pseudo-differential read operations as a poly-silicon resistance connected to a regular access transistor. We set $R_{ref} = 30K\Omega$ for our simulations as it enables a BL discharge time intermediate between $R_{LRS} = 10K\Omega$ and $R_{HRS} = 100K\Omega$, which are commonly reported mean values for RRAM LRS and HRS resistance states distributions [10]. To cover for CMOS process variability as well as RRAM process and cycle-to-cycle variability, we swept the LRS (respectively HRS) values from $30K\Omega$ to $10K\Omega$ (respectively $100K\Omega$), and for each value we ran 1000 Monte-Carlo simulations. With this approach, the proposed circuit has no read failures down to $10K\Omega$ around the reference value at 1V, thus demonstrating that read operations from a RRAM LG can be reliably performed.

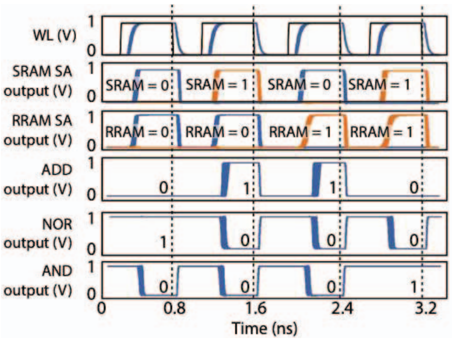


Fig. 3. Transient simulation of 256x64 memory array featuring two LG, one RRAM-based with 32 WLs and one SRAM-based with 32 WLs. The simulation shows the SRAM and RRAM sense amplifier output envelope.

Hybrid Sub-Array Simulation. At the subarray level, we adopted a methodology that relies on simulating only the critical paths (including the WL drivers, equivalent WLs and both local and global BL) to optimize simulation time. The propagation times of signals was modeled through equivalent circuits containing the extracted RC network and the corresponding gates. Finally, to simulate a realistic worst-case condition, we considered the last BL and WL to account for the longest propagation time along the metal lines. We implemented and simulated a memory array of 256BL×64WL, containing two distinct LGs of 32WL (1 of RRAM and one of SRAM).

Figure 3 shows 1000 Monte-Carlo transient simulations of the proposed hybrid architecture running IMC operations between SRAM and RRAM LGs at 1V. The first waveform shows the WL signals, the input signal being depicted in black and the propagated signal that arrives in the last bitcell in blue. SRAM and RRAM SA outputs are shown in the second and third plots from the top, respectively. The orange and blue lines show the signals transmitted to GRBL and $\overline{GRBL}$, respectively. The last three plots depict the output of the operations ADD, NOR and AND performed on the two read bits, showing that the RRAM and SRAM LGs provide compatible performance ranges, and can therefore be co-integrated inside the proposed Hybrid SRAM-RRAM subarray to perform IMC operations reliably. Our simulations report that the energy required for an IMC operation among a value stored in RRAM and one in SRAM is 16.5 fj/bit. Other system-level energy and performance values are derived from [6] which describes a detailed design space exploration of the BLADE architecture.

MacroMemory H-Tree. In order to perform a system-level exploration of a multi-array architecture, we also modelled the energy cost associated to the communication between each of the sub-arrays and a controller residing at the IMC array boundary[1]. As shown in Figure 4, we considered a

[1] The design of the memory controller is outside of the scope of this paper.

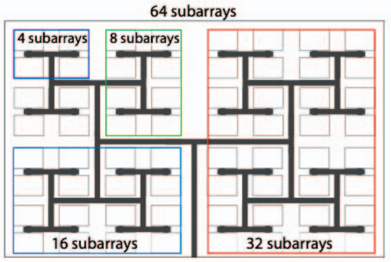


Fig. 4. Scalable H-tree interconnect from 4 subarrays to 64 subarrays.

TABLE I
EXPLORED HOMOGENEOUS AND HETEROGENEOUS IMC DESIGNS.

Architecture	**SRAM**	**RRAM**	**Area**	**Bitdensity**	**Leak/op**
	(bytes)	(bytes)	(μm^2)	(bit/μm^2)	(fj)
BLADE [6]	640	0	1318	3.88	27.8
SRAM_WDM	816	0	1973.6	3.31	35.4
HY_WDM_1	640	176	1556.4	4.19	27.8
HY_WDM_2	512	176	1318	4.18	22.2

scalable H-Tree interconnect, capable of operations pipelining. We assumed that communications over the H-tree would not be the frequency bottleneck of the system. We measured the length of each wire and performed a post-layout extraction of its parasitic capacitance considering 100nm pitch Metal 4 buses. We considered the equation $E = CV^2/2$ to determine the unitary energy cost of metal line charge. Each array is connected to 2 address buses, 1 bidirectional data bus, 5 control signals and the subarray decoder bus. In fact, IMC operations require the activation of the two address buses for the two operands, while normal read and write operations require the activation of 1 address bus and one data bus.

V. EXPERIMENTAL SETUP

A. Application-level Simulation

In order to assess the performance of applications running on our proposed IMC architecture, we developed a cycle-accurate simulator able to model the execution of entire CNN inferences (i.e. pooling, convolutional and fully connected layers). To this end, we assume a simple run-time behaviour in which the inputs to each CNN layer are streamed to the memory. Then IMC operations are performed to compute the outputs and, finally, these are read back.

Based on the geometry of the kernel and the subarray capacity, the simulator tiles the input data and distributes the tiles to different subarrays. Moreover, if the number of tiles exceeds the number of subarrays (a common occurrence for large feature maps), multiple rounds are performed for a single layer. Similarly, large convolutions exceeding the available memory capacity are decomposed in partial ones.

We assume a data transfer bandwidth of one word per cycle. IMC operations require instead two cycles, one to perform the bit-wise operation and the other to write the result, but the same operation can be performed in parallel on each subarray on different data.

TABLE II
RUNTIME PERFORMANCE (FRAME/SECOND) AND EFFICIENCY (ENERGY/FRAME) OF HOMOGENEOUS AND HETEROGENEOUS DESIGNS.

		Subarrays	4	32	128
Energy (mJ)	Alexnet	*BLADE*	10.22	14.00	18.73
		HY_WDM_2	9.34	9.99	9.70
	Mobilenet	*BLADE*	5.06	6.88	9.15
		HY_WDM_2	4.63	4.98	4.73
FPS	Alexnet	*BLADE*	0.35	1.26	1.73
		HY_WDM_2	0.43	3.34	11.93
	Mobilenet	*BLADE*	0.72	2.58	3.58
		HY_WDM_2	0.86	6.55	22.29

B. IMC Subarrays Evaluations

In the following, we investigate the benefits of our hybrid SRAM-RRAM IMC architecture from a performance and energy efficiency standpoints, when executing two different CNN benchmarks (Mobilenet [12] and Alexnet [11]).

We comparatively evaluate four different memory subarray implementations, which incrementally embody the novel features introduced in Section III. Multiple subarrays are integrated by considering the H-tree interconnect described in Section IV. The subarray characteristics are summarized below and in Table I:

- We adopt as a baseline an SRAM-based *BLADE* subarray [6]. It embeds 5 LGs, each containing 32 WLs (i.e., 64 words).
- The *SRAM_WDM* design adds further LGs, still implemented in SRAMs, dedicated to storing weights data using WDM.
- *HY_WDM_1* has the same organization than *SRAM_WDM*, but employs RRAM instead of SRAM for storing weights.
- *HY_WDM_2* is a further hybrid implementation which, by only embedding four SRAM LGs, has the same area of the *BLADE* baseline.

As reported in Table I, *SRAM_WDM* presents the highest leakage, since it features the highest amount of SRAM memory cells. Conversely, *HY_WDM_2* has the smallest SRAM capacity among the investigated design points, and therefore the smaller leakage energy per operation.

VI. EXPERIMENTAL RESULTS

A. Computation-to-communication analysis

While IMC architectures can greatly reduce the amount of required memory operations, data transfers still account for an important part of run-time when considering designs supporting a high degree of parallelism. Highlighting this effect, Figure 5 shows in blue the clock cycles devoted to memory transfers in the *BLADE* design, and in green the cycles required for in-memory computation. A cross-over point is reached for a 16-subarrays memory, after which data transfers dominate the overall workload. By instead adopting our WDM approach, IMC operations constitute the majority of run-time even for large memories of 128 subarrays, thanks to a large reduction in the amount of data transfers (60X and 27X for the AlexNet and Mobilenet, respectively).

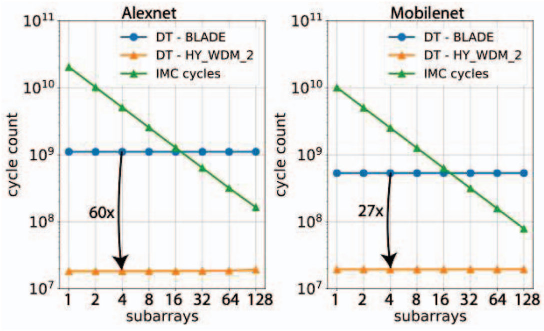


Fig. 5. Number of Data Transfer (DT) and In Memory Computing (IMC) clock cycles required for an inference in (a) Alexnet and (b) Mobilenet.

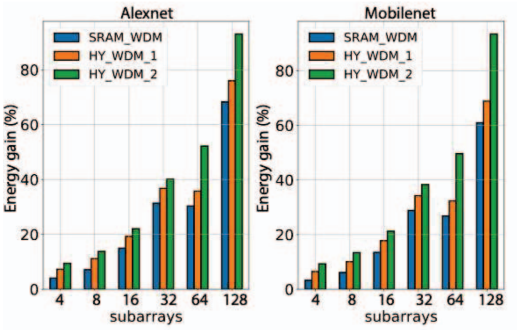


Fig. 6. Energy gain over baseline *BLADE* implementations, varying the number of subarrays. (a) Alexnet, (b) Mobilenet.

B. Energy Evaluation

Speedups achieved thanks to WDM positively impact energy efficiency, as showcased in Figure 6. This figure reports the energy gains achieved by WDM designs with respect to equivalent *BLADE* ones[2]. Moreover, energy benefits become more relevant with increasing memory sizes, because run-time (and, hence, leakage energy) becomes increasingly dominated by data transfers.

In addition, Figure 6 shows that hybrid architectures (*HY_WDM_1* and *2*) are even more energy-efficient, because weights are stored in zero-leakage RRAMs. The importance of minimizing leakage currents is highlighted by the high efficiency of the *HY_WDM_2*, which presents the least number of SRAM cells, achieving up to 93% energy gains (for both Alexnet and in the Mobilenet) with respect to a baseline *BLADE* implementation.

Finally, our findings are summarized in Table II, which compares the energy-per-inference and the frame-per-second (FPS) of homogeneous (*BLADE*) and heterogeneous (*HY_WDM_2*) designs. Energy and FPS are quite close in the two cases for simple memory organizations (e.g.: when only four subarrays are employed), but highly favour hybrid architectures for more complex cases. Indeed, a 128-subarrays *HY_WDM_2* architecture can execute Mobilenet at more than 22 FPS, while only requiring 4.73 mJ per frame. *HY_WDM_1*, which presents a larger SRAM capacity, achieves a slightly higher permormance (23 FPS), at the cost of a decreased efficiency.

VII. Conclusion

Energy-efficient computing is key in order to unlock the potential of AI at the edge. Therefore, we herein have proposed a hybrid IMC architecture, tailored to the workload characteristics of deep neural networks. Our IMC architecture embeds both volatile and non-volatile bitcells, conforming to the characteristics of data accesses at run-time. We have demonstrated that our proposed integration of SRAM and RRAM technologies can be effectively managed from a layout and electrical perspective. Moreover, we show that WDM and hybrid IMC drastically cuts down (up to 60x) the amount of data transfer required to process a full AlexNet or MobileNet CNN inference. Consequently, we show up to 93% energy efficiency and 6x performance improvement for these workloads.

[2] Gains are defined as $(E_{BLADE}/E_{\#} - 1)$, where E_{BLADE} is the energy-per-inference of *BLADE* and $E_{\#}$ the one of the case under study.

References

[1] Z. Zhou *et al.*, “Edge intelligence: Paving the last mile of artificial intelligence with edge computing,” *Proceedings of the IEEE*, 2019.
[2] D. Rossi *et al.*, “Pulp: A parallel ultra low power platform for next generation iot applications,” *IEEE HCS*, 2015.
[3] A. Sebastian *et al.*, “Memory devices and applications for in-memory computing,” *Nature Nanotechnology*, 2020.
[4] S. Jeloka *et al.*, “A 28 nm configurable memory (TCAM/BCAM/SRAM) using push-rule 6t bit cell enabling logic-in-memory,” *IEEE JSSC*, 2016.
[5] K. C. Akyel *et al.*, “DRC2: Dynamically reconfigurable computing circuit based on memory architecture,” *IEEE ICRC*, 2016.
[6] W. A. Simon *et al.*, “Blade: An in-cache computing architecture for edge devices,” *IEEE Transactions on Computers*, 2020.
[7] M. Rios *et al.*, “An associativity-agnostic in-cache computing architecture optimized for multiplication,” *IEEE VLSI-SoC*, 2019.
[8] A. Haj-Ali *et al.*, “Efficient algorithms for in-memory fixed point multiplication using magic,” *IEEE ISCAS*, 2018.
[9] M. Bocquet *et al.*, “In-memory and error-immune differential rram implementation of binarized deep neural net,” *IEEE IEDM*, 2018.
[10] E. Vianello *et al.*, “Resistive memories for ultra-low-power embedded computing design,” *IEEE IEDM*, 2014.
[11] A. Krizhevsky *et al.*, “Imagenet classification with deep convolutional neural networks,” *Communications of the ACM*, 2017.
[12] A. G. Howard *et al.*, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” *arXiv preprint arXiv:1704.04861*, 2017.
[13] S. Jain *et al.*, “Computing in memory with spin-transfer torque magnetic ram,” *IEEE Trans. VLSI*, 2017.
[14] D. Apalkov *et al.*, “Magnetoresistive random access memory,” *Proc. IEEE*, 2016.
[15] M. Chang *et al.*, “19.4 embedded 1mb reram in 28nm cmos with 0.27-to-1v read using swing-sample-and-couple sense amplifier and self-boost-write-termination scheme,” *IEEE ISSCC*, 2014.
[16] L. Wei *et al.*, “13.3 a 7mb stt-mram in 22ffl finfet technology with 4ns read sensing time at 0.9v using write-verify-write scheme and offset-cancellation sensing technique,” *IEEE ISSCC*, 2019.
[17] A. Khan *et al.*, “A survey of the recent architectures of deep convolutional neural networks,” *Artificial Intelligence Review*, 2020.
[18] Y. LeCun *et al.*, “Deep learning,” *Nature*, 2015.
[19] K. He *et al.*, “Deep residual learning for image recognition,” *CVPR*, 2016.
[20] X. Xu *et al.*, “Scaling for edge inference of deep neural networks,” *Nature Electronics*, 2018.
[21] M. Courbariaux *et al.*, “Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1,” *NIPS*, 2016.
[22] S. Han *et al.*, “Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding,” *ICLR*, 2016.