Highly-Adaptive Mixed-Precision MAC Unit for Smart and Low-Power Edge Computing

Guillaume Devic, Maxime France-Pillois, Jérémie Salles, Gilles Sassatelli, Abdoulaye Gamatié

Highly-Adaptive Mixed-Precision MAC Unit for Smart and Low-Power Edge Computing

Guillaume Devic, Maxime France-Pillois, Jérémie Salles, Gilles Sassatelli, Abdoulaye Gamatié
LIRMM, Univ Montpellier, CNRS, Montpellier, France
first.last@lirmm.fr

***Abstract*—Machine learning algorithms are compute- and memory-intensive. Their execution at the edge on resource-constrained embedded systems is challenging. Data quantization, i.e. data bit-width reduction, contributes to reducing *de-facto* the memory bandwidth requirement. In order to best exploit this bit-width reduction, a prevailing approach consists of tailored hardware accelerators. Another approach relies on general-purpose compute units with Single Instruction Multiple Data (SIMD) support for reduced data bit-width precision, as in ARM Cortex-M [1] or RISC-V based RI5CY [2] processors. However, such processors only handle a few predefined bit-width ranges, e.g. 8-bit and 16-bit only for the ARM SIMD.**

This paper proposes a flexible architecture of Multiply-and-Accumulate (MAC) unit allowing asymmetric multiplication for operand sizes in powers of 2, up to 32 bits. The synthesis of this architecture in 28nm FD-SOI technology shows 10% and 25% reduction in area and dynamic power respectively, compared to the RI5CY MAC unit. From the energy-efficiency point of view, up to 50% improvements are achieved.

***Index Terms*—Multiply-and-Accumulate units, MAC, Machine Learning, Edge-Computing, Quantization**

I. Introduction

Internet-of-Things heavily relies on cloud resources that process sensor data acquired and transmitted by edge devices.

Given the ever-increasing volume of transmitted data, this centralized approach quickly leads to network bandwidth saturation, which is detrimental to response time [3]–[5]. An alternative approach lies in edge computing which advocates to bring compute resources closer to the edge devices. In particular, smart edge computing executes Machine Learning (ML) algorithms on mobile devices for faster decision making and control. These algorithms are often compute- and memory-intensive and further require large data storage. Therefore, their use at the edge is a real design challenge, which calls for combined hardware and software optimizations [6].

Among the popular techniques aiming at mitigating the impact of ML algorithm execution on mobile (or embedded) devices, data quantization is very popular. This technique consists in compressing the volume of data (e.g. the weights of an artificial neural network) by reducing the bit-width representation [7]. However, to preserve the initial properties and accuracy of a given ML algorithm, bit-width reduction must be applied on a case by case basis. In Deep Neural Networks (DNN) [8], [9] for instance quantization cannot be applied uniformly and must be decided on a per-layer basis. Some studies have shown the relevance of using mixed-precision representations [10], where different data types are described with their most appropriate bit-widths. The precision reduction can even go down to 1 bit. Note that to effectively exploit such a precision level, the hardware must be adapted (the decided precision requires resizing the network).

To fully benefit from quantization and mixed-quantization techniques, custom hardware support is required. This often takes the form of low-power hardware accelerators dedicated to ML workloads. Another solution consists in revisiting the general design of microprocessors w.r.t. the quantization requirements. ML algorithms heavily rely on the Multiply-and-Accumulate (MAC) operation. The basic multiply unit of microprocessors generally allows performing only one MAC iteration per clock cycle. The addition of a Single-Instruction Multiple-Data (SIMD) unit enables to increase the number of MAC iterations per clock cycle. Besides the additional hardware required for SIMD, the supported operations are usually restricted to bit-widths of 16-bit and/or 8-bit [1]. For embedded systems, this is inefficient both in terms of area overhead and power consumption. For instance, we estimated the MAC unit of the open-source RISC-V RI5CY core [2], which consumes about 40% of the total power [11] of the core.

In this work, we propose a flexible MAC unit architecture. Our solution is able to support data bit-width representations between 32 and 2 bits and enables to perform asymmetric bit-width operations. A comparison with the MAC unit of the RISC-V RI5CY core reveals reductions of 25% and 10% in power consumption and area respectively.

The remainder of the paper is organized as follows: Section II reviews related work and discusses the motivations of this work. Section III presents our MAC unit architecture. The experimental results are shown in Section IV. Section V concludes the paper.

II. Related work

Over the past decade, various approaches have been proposed for efficient execution of ML algorithms, and in particular DNNs at the edge. In the sequel, we first discuss hardware accelerator based approaches and then concentrate on the design of MAC units for DNNs.

Hardware accelerators for DNNs. The first family of approaches focuses on the design of specific circuits that exclusively fulfill a given function. For instance, such a function can represent either a layer of a DNN, or even an entire network [12], [13].

The second family of approaches rather promotes the realization of different DNN models on the same circuit [8], [9], [14], [15]. Here, the considered solution integrates several processing elements that enable to perform massively parallel MAC operations. It is similar to GPU architectures but is more energy efficient. The corresponding hardware accelerators can be implemented either on FPGAs or as ASICs.

The last family of approaches differs from the above two by higher flexibility. Indeed, the previous accelerators are restricted to DNN execution only. General-purpose approaches based on multicore architectures allow to execute any kind of workload. For instance, the SIMD feature commonly offered by modern CPUs allows parallelizing MAC operations for DNN execution. For instance, the ARM M-Cortex CPU provides SIMD support for 16 and 8-bit data [1]. The promising RISC-V RI5CY CPU [2] also supports SIMD operations from 16 to 2 bits via an ISA extension [11].

Figure 1 summarizes the aforementioned approaches for executing ML workloads in an edge computing context, according to power consumption and flexibility.

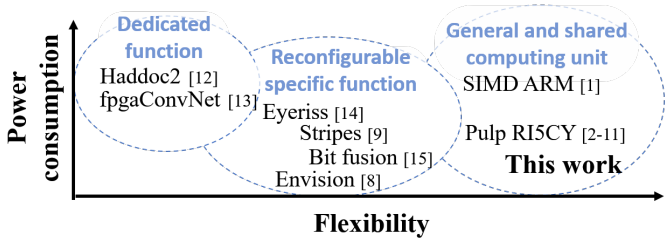

Fig. 1: Different hardware approaches for ML execution in edge computing context

Focus on MAC unit design. In a recent survey [16], four types of scalable precision MAC architectures dedicated to deep learning accelerators are presented: Bit-serial [17], Multibit-serial [18], Subword-Parallel [8] and Divide-and-Conquer [15], [19]. Among all these architectures, Divide-and-Conquer stands out for its higher flexibility. The reference hardware accelerator implementing such a MAC architecture is Bit Fusion [15]. It uses several 2-bit × 2-bit multipliers to handle variable bit-width multiplications up to 8-bit × in 1 clock cycle. It can support 16-bit multiplication at the cost of more clock cycles. Partial products from these 2-bit × 2-bit multipliers are accumulated before reaching the output.

How does our approach differ from literature? We leverage the Divide-and-Conquer strategy to devise flexible and low-power MAC unit. Unlike previous works, this unit is designed for a general-purpose CPU. It handles classical multiplications as well as MAC operations, for input data ranging from 2 to 32 bits. Moreover, the flexibility of our solution supports asymmetric bit-width inputs, imposed by mixed-precision quantization. For example, a mixed-precision DNN may require 4-bit × 8-bit MAC operations. When a classical CPU handles this asymmetric operation with its SIMD support, both operands are resized to the bit-width that is closer to the larger operand bit-width (i.e. 8-bits in the best case in our example). Unfortunately, this "resizing" imposes the use of more hardware than strictly necessary, inducing unnecessary power consumption. To address this issue, our MAC unit is able to activate the minimal required hardware for achieving a 4-bit × 8-bit MAC operation resulting in interesting energy savings.

III. Multiply-Accumulate Unit Architecture

The proposed MAC unit aims to be integrated into a 32-bit microprocessor. It handles multiplication and MAC operations for data up to 32 bits in 1 clock cycle. We first present the principle of the considered binary multiplication. Then we describe the implementation of the MAC unit.

A. General principle of binary multiplication

Figure 2 ① illustrates the binary multiplication principle. Here, there are two 4-bit operands A and B: the former is multiplied by each bit of the latter. The partial products resulting from this multiplication are appropriately shifted before being finally added together. The bit-by-bit products are performed using AND gates. Then, the results of this operation are added by a half/full adder to obtain the final result.

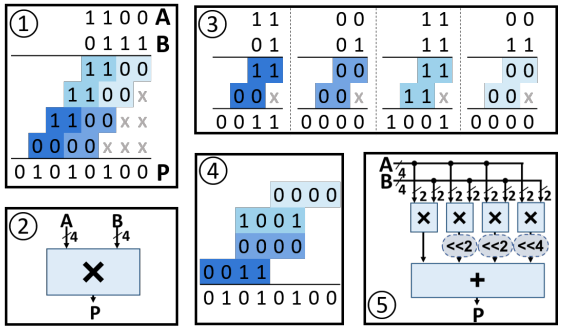

Fig. 2: Illustration of binary multiplication

Figure 2 ② is the schematic of a 4-bit × 4-bit multiplier. It is the most common form of a multiplier. However, other approaches are possible, like decomposition.

Indeed, instead of performing a 4-bit × 4-bit multiplication, one can perform the same multiplication, in four independent 2-bit × 2-bit multiplications as illustrated in Figure 2 ③ . The final result is obtained by adding with a proper shift the four results from the 2-bit × 2-bit multipliers as shown in Figure 2 ④ . Finally, Figure 2 ⑤ shows the diagram of the basic blocks required to perform the decomposed multiplication.

B. Multiply-Accumulate unit description

The proposed MAC unit architecture is composed of three distinct parts: the multipliers, the adders, and the accumulator. Figure 3 shows a simplified schematic of this architecture: a line of 2-bit × 2-bit multipliers on the upper part, the adders and shifts in the middle, the shift control on the left, the output multiplexer on the right, and the accumulator at the bottom.

To support 32-bit multiplication in 1 clock cycle, our MAC unit is composed of 256 independent 2-bit × 2-bit multipliers.

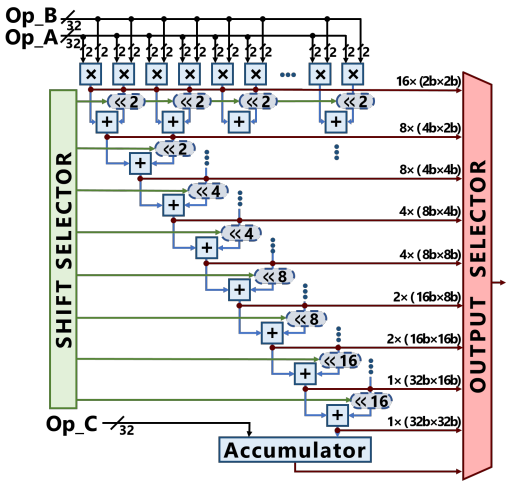

Fig. 3: Schematic representation of the multiplier

Indeed, in Section III-A we explained a 4-bit multiplication requires 4 independent 2-bit multiplication $((4/2)^2)$. Therefore, for a 32-bit multiplication, 256 independent 2-bit × 2-bit multipliers are required $((32/2)^2)$.

To facilitate the configurability and the adaptability of the multiplier, the addition of the partial products is performed by 2-input adders. As mentioned in Section III-A, during the addition step of the partial products, a bit shifting must be performed. To reduce the number of shifts, for each adder one of these inputs is shifted beforehand. As shown in Figure 3, a maximum of 8 adder levels is necessary to carry out the multiplications. The multiplication results are captured at the outputs of the 2-bit × 2-bit multipliers and each adder level.

Finally, the accumulator allows the MAC operation to be performed. A dedicated register stores the intermediate value of the previous accumulation. The intermediate value is returned to the accumulator via Op_C depicted in Figure 3.

32 bits															
16 bits								16 bits							
8 bits				8 bits				8 bits				8 bits			
4 bits	4 bits	4 bits	4 bits	4 bits	4 bits	4 bits	4 bits	4 bits	4 bits	4 bits	4 bits	4 bits	4 bits	4 bits	4 bits
2b	2b	2b	2b	2b	2b	2b	2b	2b	2b	2b	2b	2b	2b	2b	2b

(a) Representation of data contained in a 32-bit register for each data bit-width

Data bit-width	MAC operation
32 & 32×16	1
16×16 & 16×8	2
8×8 & 8×4	4
4×4 & 4×2	8
2×2	16

(b) Number of available operations for each data bit-width

TABLE I: Details on supported operands and operations.

The 32-bit op_A and op_B operands represented in Figure 3 can be used for SIMD operations. They could be loaded with 2-bit × 16-bit, or 4-bit × 8-bit, or 8-bit × 4-bit, or 16-bit × 2-bit operands as shown in Table Ia. When used in SIMD mode, the proposed multiplication/MAC module can perform parallel operations in 1 clock cycle as specified in Table Ib.

IV. Experimental Results

The first part of this section describes the experimental environment setup. Then, the second part exposes the gains achieved by our proposal compared to the MAC unit implemented in the open-source RI5CY core [2], [11].

A. Experimental setup

The proposed MAC unit is developed in SystemVerilog, as the RI5CY. The RI5CY MAC unit available on a dedicated Github page is based on the redundancy of hardware multiplier. It realizes the execution of two 16-bit × 16-bit and four 8-bit × 8-bit operations respectively by two 16-bit and four 8-bit independent multipliers It does not support 4-bit × 4-bit and 2-bit × 2-bit configurations. We add these matching the description provided in [11] and the required syntax.

The tools used to perform simulation and synthesis are respectively ModelSim and Synopsys Design Compiler. The simulation enables the verification of the functionality of the proposed MAC unit. Also, it allows the acquisition of the switching-activity for the two evaluated MAC architectures necessary for accurate power estimation. The synthesis provides area and power cost in the selected technology, which is the 28nm FD-SOI technology at 200MHz. To emulate mixed-precision quantized operands, the assessment of the MAC units is accomplished with different bit-widths data according to Table Ib. The operand values are randomly generated by python scripts. This data is then loaded into the 32-bit input registers of the MAC architectures with a SystemVerilog benchmark. For each tested bit-width, the input registers of the MAC units are loaded 1000 times with different values.

B. Power, Area, and Energy

The first information given by the architecture synthesis is the area. It is respectively $10830\mu m^2$ and $9930\mu m^2$ for the RI5CY MAC unit and for our proposed MAC architecture. Our proposal provides a reduction of 10%, compared to RI5CY MAC unit, which is about 40% of the total CPU area.

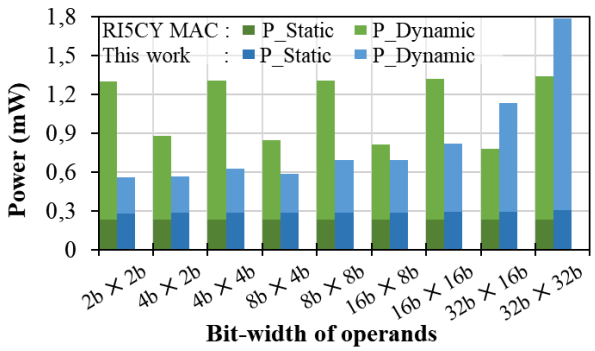

Fig. 4: Dynamic power provided by Synopsys Design Compiler including switching-activities from simulated testbench.

Figure 4 shows the evolution of the static and dynamic power consumption for both MAC architectures. The dynamic power of our solution increases progressively according to the data bit-width. Indeed, our solution is very fitted and only activates the hardware required for a specific bit-width. In fact, the Divider-and-Conquer principle required to perform a multiplication is proportional to the operands bit-width.

The dissipated power is directly related to the number of activated logical gates. This is why the 32-bit × 32-bit operations which use all the logic of the multiplier consume more power. Compared to RI5CY, the power consumption is higher for 32-bit × 16-bit and 32-bit × 32-bit operations. Our presumable explanation is a less optimized logical component placement during synthesis. Indeed, our MAC unit is composed of several connections distributed over the circuit, reducing the possible optimization space. It also explains the slightly higher static power compared to RI5CY despite a smaller area. As for the MAC of the RI5CY core, we notice two levels of dynamic power. The highest level corresponds to the values of operands with the same bit-width, and the lowest level corresponds to the values of operands with asymmetric bit-width. Explained by the presence of extra zeros to fit with the asymmetric bit-width. On average, the dynamic power dissipated by our MAC architecture is 25% less than that measured for the RI5CY.

To compare the energy-efficiency of both solutions, Figure 5 reports the number of operations per mW. For both MAC architectures, the best energy efficiency is at the lowest bit-widths. A drop in energy efficiency occurs from 16-bit data. On average the energy-efficiency is 50% higher for our proposed MAC unit compared to that of RI5CY CPU. In [10], the MobileNet V2 convolutional neural network (CNN) [20] where the activations and weights are respectively quantized to 8b and 4b, achieves a 71% top-1 accuracy. Considering the 300M MACS operations composing this CNN, our proposal enables to save 43% to the RI5CY MAC unit.

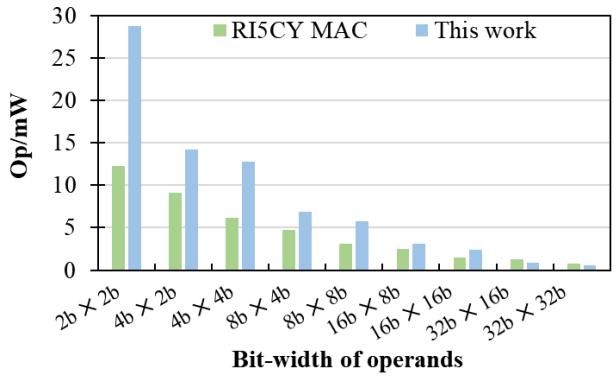

Fig. 5: Energy efficiency determined from dynamic power

V. Conclusion and Perspectives

Hardware limitations of embedded systems at the edge severely restricts the range of usable ML algorithms. Emerging solutions such as quantization allow a reduction in the impact of the execution of these algorithms. However, the efficiency of quantization depends on custom hardware support. Today, this support is provided by the use of SIMD compute units. However, SIMD units are limited by the natively supported bit-widths (usually 16-bit and 8-bit). In this paper, we propose a flexible MAC unit architecture that with support for every possible power of 2, from 2 to 32 bits. The MAC unit of the open-source RI5CY core is used as a reference for evaluating the proposed architecture. Our work shows the proposed architecture is more efficient in dynamic power by 25% and also 50% more energy efficient. The area of our solution is even slightly lower by 10% compared to the RI5CY core MAC unit. Future work include the integration of the proposed MAC unit to typical low-power cores, e.g. RI5CY [2] or Cortus APS25 [21].

References

[1] "Dsp for cortex-m," https://developer.arm.com/architectures/instruction-sets/dsp-extensions/dsp-for-cortex-m, accessed: 28/01/2021.

[2] M. Gautschi, P. D. Schiavone, A. Traber, I. Loi, A. Pullini, D. Rossi, E. Flamand, F. K. Gürkaynak, and L. Benini, "A near-threshold RISC-V core with DSP extensions for scalable iot endpoint devices," *CoRR*, vol. abs/1608.08376, 2016.

[3] S. Verma, Y. Kawamoto, Z. M. Fadlullah, H. Nishiyama, and N. Kato, "A survey on network methodologies for real-time analytics of massive iot data and open research issues," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1457–1477, 2017.

[4] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. M. Barnaghi, and A. P. Sheth, "Machine learning for internet of things data analysis: A survey," *CoRR*, vol. abs/1802.06305, 2018.

[5] D. Georgakopoulos, P. P. Jayaraman, M. Fazia, M. Villari, and R. Ranjan, "Internet of things and edge cloud computing roadmap for manufacturing," *IEEE Cloud Computing*, vol. 3, no. 4, pp. 66–73, July 2016.

[6] H. Li, K. Ota, and M. Dong, "Learning iot in edge: Deep learning for the internet of things with edge computing," *IEEE Network*, vol. 32, no. 1, pp. 96–101, 2018.

[7] B. Moons, K. Goetschalckx, N. Van Berckelaer, and M. Verhelst, "Minimum energy quantized neural networks," in *2017 51st Asilomar Conference on Signals, Systems, and Computers*, 2017, pp. 1921–1925.

[8] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "DVAFS: trading computational accuracy for energy through dynamic-voltage-accuracy-frequency-scaling," in *DATE*. IEEE, 2017, pp. 488–493.

[9] P. Judd, J. Albericio, and A. Moshovos, "Stripes: Bit-serial deep neural network computing," vol. 16, no. 1, 2017, pp. 80–83.

[10] Q. Jin, L. Yang, and Z. Liao, "Adabits: Neural network quantization with adaptive bit-widths," *CoRR*, vol. abs/1912.09666, 2019.

[11] A. Garofalo, G. Tagliavini, F. Conti, D. Rossi, and L. Benini, "Xpulpnn: Accelerating quantized neural networks on RISC-V processors through ISA extensions," in *DATE*. IEEE, 2020, pp. 186–191.

[12] K. Abdelouahab, M. Pelcat, C. Bourrasset, F. Berry, and J. Sérot, "Tactics to directly map CNN graphs on embedded fpgas," *CoRR*, vol. abs/1712.04322, 2017.

[13] S. I. Venieris and C. Bouganis, "fpgaconvnet: A framework for mapping convolutional neural networks on fpgas," in *FCCM*. IEEE Computer Society, 2016, pp. 40–47.

[14] Y. Chen, J. S. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," in *ISCA*. IEEE Computer Society, 2016, pp. 367–379.

[15] H. Sharma, J. Park, N. Suda, L. Lai, B. Chau, V. Chandra, and H. Esmaeilzadeh, "Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural networks," vol. abs/1712.01507, 2017.

[16] V. Camus, L. Mei, C. Enz, and M. Verhelst, "Review and benchmarking of precision-scalable multiply-accumulate unit architectures for embedded neural-network processing," *IEEE Journal on Emerging and Selected Topics in Cir. and Sys*, vol. 9, no. 4, pp. 697–711, 2019.

[17] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H. Yoo, "UNPU: A 50.6tops/w unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision," in *ISSCC*. IEEE, 2018, pp. 218–220.

[18] S. Sharify, A. D. Lascorz, P. Judd, and A. Moshovos, "Loom: Exploiting weight and activation precisions to accelerate convolutional neural networks," *CoRR*, vol. abs/1706.07853, 2017.

[19] D. Shin, J. Lee, J. Lee, and H. Yoo, "14.2 DNPU: an 8.1tops/w reconfigurable CNN-RNN processor for general-purpose deep neural networks," in *ISSCC*. IEEE, 2017, pp. 240–241.

[20] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation," *CoRR*, vol. abs/1801.04381, 2018.

[21] A. Gamatié, G. Devic, G. Sassatelli, S. Bernabovi, P. Naudin, and M. Chapman, "Towards energy-efficient heterogeneous multicore architectures for edge computing," *IEEE Access*, vol. 7, 2019.