A Reconfigurable 2D-Convolution Accelerator for DNNs Quantized with Mixed-Precision

# A Reconfigurable 2D-Convolution Accelerator for DNNs Quantized with Mixed-Precision

Luca Urbinati and Mario R. Casu

Department of Electronics and Telecommunications, Politecnico di Torino, Italy
{luca.urbinati,mario.casu}@polito.it

**Abstract.** Mixed-precision uses in each layer of a Deep Neural Network the minimum bit-width that preserves accuracy. In this context, our new Reconfigurable 2D-Convolution Module (RCM) computes $N$=1, 2 or 4 Multiply-and-Accumulate operations in parallel with configurable precision from 1 to 16/$N$ bits. With our design-space exploration via high-level synthesis we found the best points in the latency vs area space, varying the size of the tensor tile handled by our RCM and its parallelism. A comparison with a standard non-configurable module on a 28-nm technology shows many reconfigurable Pareto points for low bit-width configurations, making our RCM a promising mixed-precision accelerator.

**Keywords:** 2D-Convolution, Reconfigurable Accelerator, Mixed-Precision.

## 1 Introduction

Low bit-width quantization is used in Deep Neural Networks (DNNs) to satisfy memory and latency constraints of embedded edge devices. Mixed-precision aims to quantize each DNN layer with the minimum bit-width [1] that preserves accuracy [2]. DNN accelerators started to support multiple precisions. For example, UNPU [3] uses serial multipliers and supports from 1 to 16 bits; DNPU [4] uses look-up table-based reconfigurable multipliers that support 4-/8-/16-bit multiplication; Bit Fusion [5] supports multiple input/weight pairs bit precisions (8/2, 4/4, 2/8 and 8/8) by composing and decomposing 2-bit multipliers.

In this context, our new Reconfigurable 2D-Convolution Module (RCM) uses Multiply-and-Accumulate (MAC) units with Sum Together (ST) multipliers [6] to process in one shot $N$ (activations, weights) pairs with up to 16/$N$ bits, where $N$ is 1, 2 or 4. Thus, $N$–1 MAC operations are saved compared to a non-configurable 16-bit multiplier. We define the supported configurations as 16x, 8x and 4x. The same precisions are used by both Envision [7] and our previous Reconfigurable Depth-wise Convolution accelerator [8]. However, Envision's Sum Separate (SS) multiplier requires an external partial product addition. Moreover, our RCM supports a variable kernel size to satisfy many different convolution layers, from the Point-wise layers of MobileNets to the first layer of ResNetV1.

We report the results of a Design-Space Exploration (DSE) of Latency vs Area. We quickly explored sixty-four RCM variants with High-Level Synthesis (HLS), varying the maximum number of supported input and output chan-

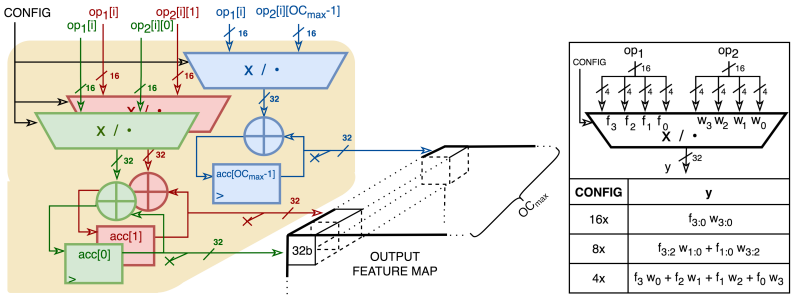


Fig. 1: RCM MAC Unit array with reconfigurable multipliers.

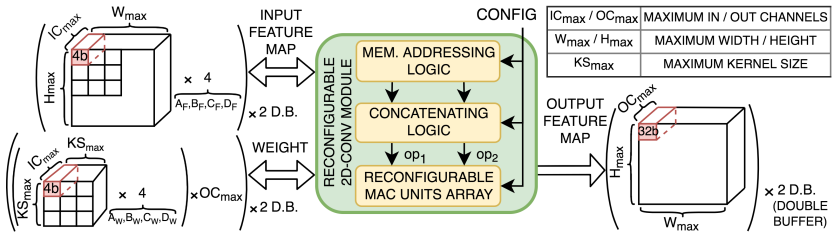


Fig. 2: Overview of the RCM.

nels, and compared them with the corresponding non-configurable Standard 2D-Convolution Module (SCM) designs in a CMOS FDSOI 28-nm technology.

## 2 Hardware Architecture

The RCM's MAC Unit array, shown in Fig. 1, has as many ST reconfigurable multipliers as the maximum number of output channels that the RCM can process in parallel, $\mathrm{OC_{max}}$. Each MAC receives $op_1$ and $op_2$ 16-bit operands from the input and weight buffers, respectively, unpacks them in four 4-bit values to match the internal reconfigurable multiplier arrangement, and accumulates partial results in a register. The table in Fig. 1 shows the three operations done by one reconfigurable multiplier according to the CONFIG signal. When a convolution between a kernel and an input receptive field is completed, the accumulated result is cast to 32-bit and stored in an output buffer for successive computations.

Fig. 2 is an overview of the RCM, which includes a memory with double buffers made of four 4-bit SRAMs for input features and weights, each with size $(\mathrm{W_{max}}\times\mathrm{H_{max}}\times\mathrm{IC_{max}})$ and $(\mathrm{KS_{max}}^2\times\mathrm{IC_{max}}\times\mathrm{OC_{max}})$, respectively. We refer to these as $A_F$, $B_F$, $C_F$ and $D_F$ for features, and $A_W$, $B_W$, $C_W$ and $D_W$ for weights. The output memory is a single 32-bit SRAM of size $(\mathrm{W_{max}}\times\mathrm{H_{max}}\times\mathrm{OC_{max}})$.

The size of the feature-map and weight tensors of a layer can exceed the size of the memory buffers of our RCM, which requires to iterate over multiple tiles. Therefore, a wise selection of the buffers size is essential to strike the right balance between latency, which decreases with larger buffer size, and area. Toward

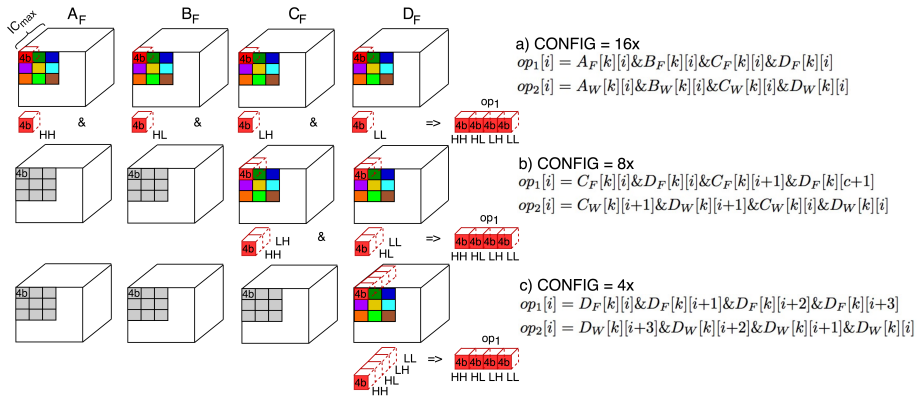


Fig. 3: Memory addressing and concatenation of the input feature-map buffer.

this goal, we analyzed the layers of the most popular DNNs for classification and object detection supported by multiple commercial edge devices and their development platforms. These include ResNetV1/V2, MobileNetV1/V2 (and the SSD and SSD-Lite versions), YOLO-V2/V3/V4/Tiny, and EfficientNet-B0. Based on our survey, we set $\mathrm{KS}_{\max} = 7$, because some networks require a 7x7 kernel (e.g., ResNetV1), and $\mathrm{W}_{\max} = 18$ ($= \mathrm{H}_{\max}$), to limit the RCM iterations when computing large layers and to limit the buffers area. For the parameters $\mathrm{IC}_{\max}$ and $\mathrm{OC}_{\max}$, we performed the DSE outlined in Sec. 3.

The RCM memories will be filled by an embedded processor or DMA engine as follows. In the 16x case, the input feature and weight tiles are split into 4-bit chunks and stored from the most to the least significant into $A_F$-$D_F$ and $A_W$-$D_W$, respectively. In the 8x case, the *two* 4-bit chunks and stored in $C_F$-$D_F$ and $C_W$-$D_W$. Finally, in the 4x case each element is stored in $D_F$ and $D_W$.

Our RCM requires a particular memory addressing and concatenating logic to feed all the MAC units in parallel. Let us refer to the toy example in Fig. 3. Here, filters with shape $3{\times}3{\times}\mathrm{IC}_{\max}$ create a $3{\times}3$ receptive field on a feature-map tile. Fig. 3 shows how a tile is read to obtain $op_1$. Similarly, a weight tile is read to get $op_2$. The example refers to only one of the filters and one MAC unit out of $\mathrm{OC}_{\max}$, but it applies of course to all filters and MAC units with different weights. Index $i \in \{0, \ldots, \mathrm{IC}_{\max} - 1\}$ denotes the input channel, and index $k \in \{0, \ldots, \mathrm{KS} \times \mathrm{KS} - 1\}$ the receptive field. The three configurations use a different addressing scheme as indicated in the same figure.

In general, the number of MAC cycles to get $\mathrm{OC}_{\max}$ output *pixels* is $\mathrm{IC}_{\max}/N$. The theoretical speedup achievable by the RCM would be $N$, but due to the control logic overhead the actual speedup $s(N)$ is lower than $N$, as shown in Fig. 4. OW ($\leqslant \mathrm{W}_{\max}$), IC ($\leqslant \mathrm{IC}_{\max}$) and KS ($\leqslant \mathrm{KS}_{\max}$) are three run-time, tile-dependent configuration parameters that correspond to the output width, the number of input channels and the kernel size of the tiles processed by the RCM, respectively; $\mathrm{IC}/N \times \mathrm{KS}^2$ are the useful clock cycles, while $o_1 = 2$ and $o_2 = 5$

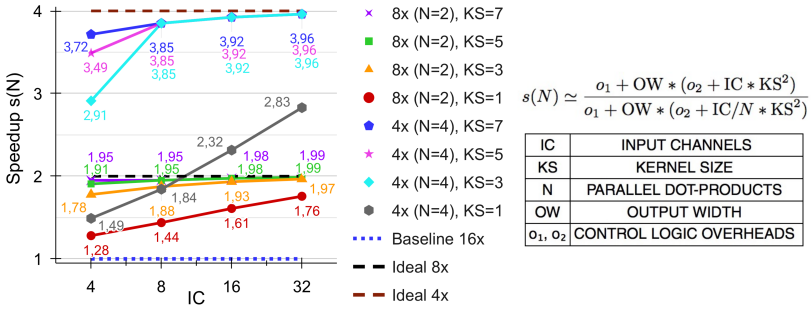


Fig. 4: RCM speedup for OW = 18.

are those responsible for the control logic overhead. The curves in Fig. 4 show that the speedup tends to saturate as IC approaches 32. KS = 1 is the worst case because of the dominant contribution of the overheads.

## 3 Experimental Results

More than 90% of the analyzed 2D-convolution layers have input and output channels multiple of 4, 8, 16 and 32. Therefore, we performed a DSE using Catapult HLS by sweeping $\mathrm{IC}_{\max}$ and $\mathrm{OC}_{\max}$ in $\{4, 8, 16, 32\}$ and the operating clock frequency $f_{clk}$ from 400 to 1000 MHz (200-MHz steps). To reduce latency, at the expense of area, we used the HLS unrolling directive applied to the output channels loop, and the memory interleave directive applied to the weight memories. We synthesized the RTL netlists generated by Catapult HLS with Synopsys Design Compiler. We compared our RCM with an SCM based on standard 16-bit multipliers, which extend the operands sign for low precision configurations.

We analyzed the performance of RCM and SCM over two different 2D-convolution layers. The first is the most frequent layer among the selected DNNs: (16×16×256) as feature-map tensor (padding included) and (3×3×256×256) as weight tensor. The second is the last point-wise layer of MobileNetV1: (7×7×1024) for inputs and (1×1×1024×1024) for weights. Since the results for both layers are similar, due to space limitations we report the results of the DSE of Latency vs Area for the first case only, in Fig. 5. The latency is the total number of clock cycles multiplied by the clock period. The table adjacent to each plot contains the sorted Pareto-points in ascending order of area and reports input and output channels for each point. From Fig. 5 we observe:

– In the 16x case (Fig. 5a), as expected, all Pareto point are of SCM type, because the RCM points suffer from the area overhead of a more complex memory addressing logic and of the ST multipliers. Thus, in the following we only consider the 8x (Fig. 5b) and 4x (Fig. 5c) cases.
– $\mathrm{OC}_{\max}$ significantly affects area and latency: as $\mathrm{OC}_{\max}$ increases, the size of weight and output memories increase, but the number of MAC units grow and more output channels can be computed in parallel. Indeed, all the points

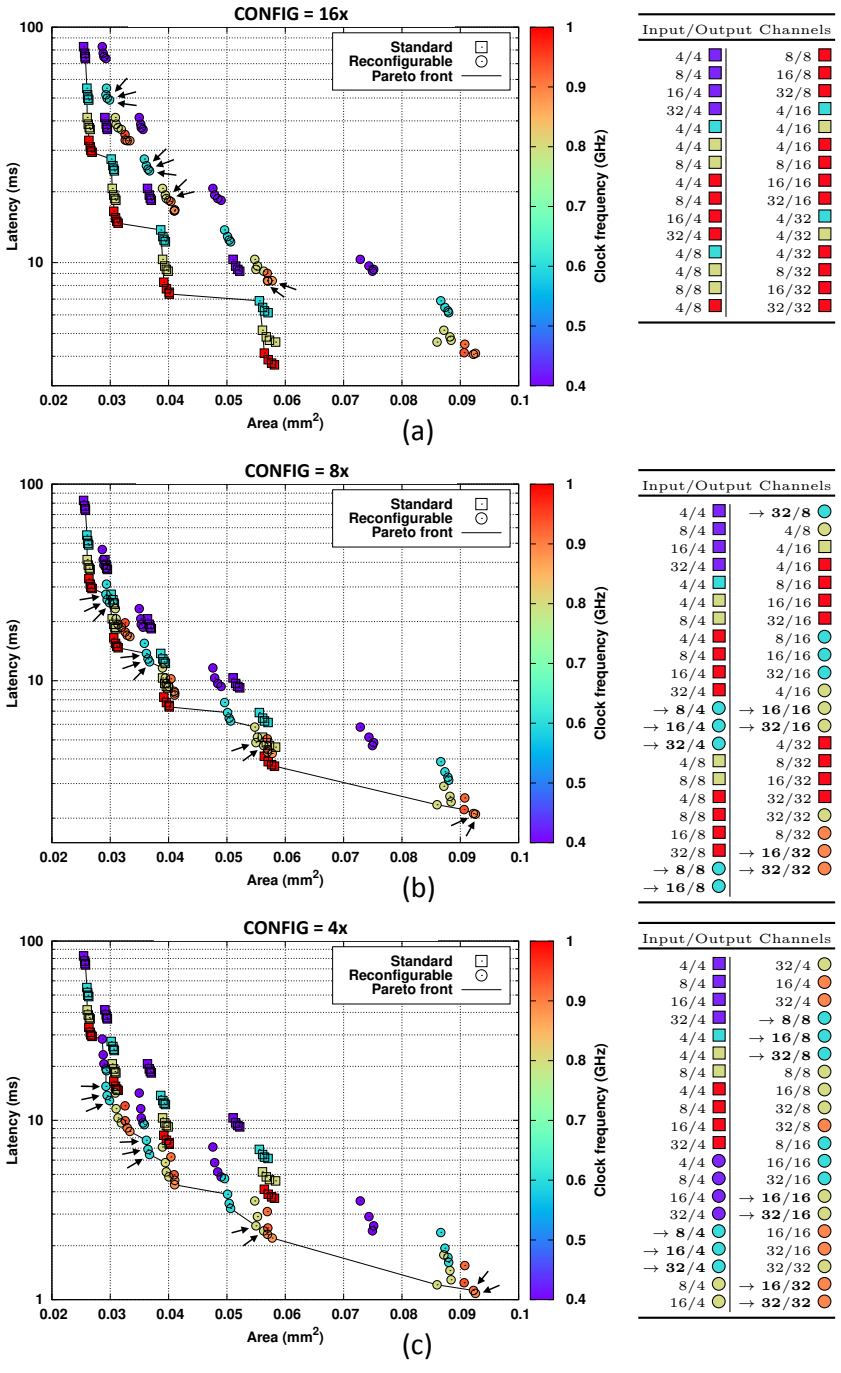


Fig. 5: Latency vs Area DSEs for CONFIG 16x (a), 8x (b) and 4x (c).

with larger area and lower latency have a high value of $OC_{max}$.
- For area $< 0.06\,mm^2$ increasing $IC_{max}$ reduces latency more than increasing $OC_{max}$ for a given area increase. However, to reduce latency below 2-3 ms (e.g., area $\geq 0.06\,mm^2$) $OC_{max}$ must increase up to 32 and $IC_{max}$ must saturate.
- The low frequency RCM solutions ($f_{clk} = 400\,MHz$) in the Pareto curve occur only for area $< 0.03\,mm^2$ in the 4x case. However, they are not worth when latency is the goal, as a higher clock frequency ($\geqslant 600\,MHz$) leads to a significant better latency for a marginal area increase.
- For 8x and 4x, 39% and 73% of Pareto points are reconfigurable, respectively.

Since 8-bit precision is enough for many DNNs and the trend is to go below 8-bit [9], a designer willing to use our RCM can choose among many Pareto points, as shown in Fig.5. For example, those marked with an arrow ($\rightarrow$) are optimal in the 8x and 4x case, and close enough to optimal in the 16x case, making them suitable to DNNs requiring variable precision in their layers.

## 4 Conclusion

We presented a Reconfigurable 2D-Convolution Module for Heterogeneously Quantized DNNs synthesized in a CMOS FDSOI 28-nm technology from a high-level description. The results of the design-space exploration show many Pareto points, especially for low-precision configurations, which dominate the non-reconfigurable counterparts. In the future, we plan to combine this accelerator and our previous Reconfigurable Depth-wise Module [8] into an SoC, hence providing a complete solution to accelerate mixed-precision DNNs in hardware.

## References

1. S. Anwar *et al.*, "Fixed point optimization of deep convolutional neural networks for object recognition," in *IEEE ICASSP*, 2015.
2. K. Vasquez *et al.*, "Activation density based mixed-precision quantization for energy efficient neural networks," *CoRR*, vol. abs/2101.04354, 2021.
3. J. Lee *et al.*, "Unpu: An energy-efficient deep neural network accelerator with fully variable weight bit precision," *IEEE J. Solid-State Circuits*, 2019.
4. D. Shin *et al.*, "14.2 dnpu: An 8.1tops/w reconfigurable cnn-rnn processor for general-purpose deep neural networks," in *IEEE ISSCC*, 2017.
5. H. Sharma *et al.*, "Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network," in *IEEE ISCA*, 2018.
6. L. Mei *et al.*, "Sub-word parallel precision-scalable mac engines for efficient embedded dnn inference," in *IEEE AICAS*, 2019.
7. B. Moons *et al.*, "14.5 envision: A 0.26-to-10tops/w subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28nm fdsoi," in *IEEE ISSCC*, 2017.
8. L. Urbinati and M. R. Casu, "A Reconfigurable Depth-Wise Convolution Module for Heterogeneously Quantized DNNs," in *Proc. IEEE ISCAS*, 2022.
9. J. Choi *et al.*, "Accurate and efficient 2-bit quantized neural networks," in *Proc. Machine Learning and Systems*, 2019.