A Reconfigurable Multiplier/Dot-Product Unit for Precision-Scalable Deep Learning Applications

# A Reconfigurable Multiplier/Dot-Product Unit for Precision-Scalable Deep Learning Applications

Luca Urbinati[0000−0001−5317−1960] and Mario R. Casu[0000−0002−1026−0178]

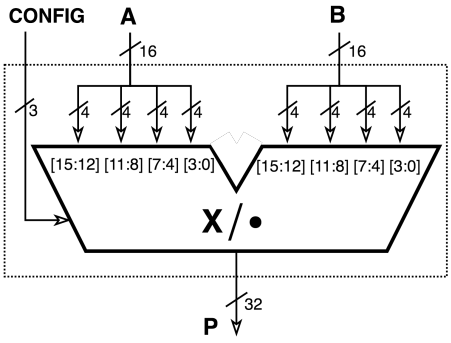
Dept. of Electronics and Telecommunications, Politecnico di Torino, Torino, Italy
{luca.urbinati,mario.casu}@polito.it

**Abstract.** Across different Deep Learning (DL) applications or within the same application but in different phases, bitwidth precision of activations and weights may vary. Moreover, energy and latency of MAC units have to be minimized, especially at the edge. Hence, various precision-scalable MAC units optimized for DL have recently emerged. Our contribution is a new precision-configurable multiplier/dot-product unit based on a modified Radix-4 Booth signed multiplier with Sum-Together (ST) mode. Besides 16-bit full precision multiplications, it can be reconfigured to perform dot products among two 8-bit or four 4-bit sub words of the input operands without requiring an external adder, thus reducing the number of cycles of MAC operations. The results of the synthesis in performance, power and area on a 28-nm technology show that our unit (1) is superior to other state of the art ST multipliers in area (≈35% less) in the clock frequency range between 100 and 1000 MHz and (2) reduces latency up to 4x when used to compute a convolutional layer, at the cost of limited overheads in area (+10%) and power (+13%) compared to a conventional 16-bit Booth multiplier. This unit can play an important role in designing variable-precision MAC units or DL accelerators for edge devices.

**Keywords:** Variable-precision Multiplier · Precision-Scalable MAC Unit · Deep Learning.

## 1 Introduction

At the basis of Deep Learning (DL) algorithms are convolutions and matrix multiplications, which require the computation of many dot products and simple scalar multiplications between features and weights. These operations are typically executed by multiply-and-accumulate (MAC) units. In particular, when running DL applications on edge devices, energy and latency of these MAC units have to be minimized. This requires reducing the data bit-width to the minimum, while keeping a satisfying level of accuracy. Such minimum data precision may vary across different applications, but also within the same application in different phases (e.g. mixed-precision quantization for convolutional layers).




(a)

| CONFIG | P |
|---|---|
| 16x16 (000b) | A[15:0] x B[15:0] |
| 16x8 (100b) | A[15:0] x B[7:0] |
| 8x8 (010b) | A[15:8] x B[7:0] + A[7:0] x B[15:8] |
| 8x4 (011b) | A[15:8] x B[3:0] + A[7:0] x B[11:8] |
| 4x4 (001b) | A[15:12] x B[3:0] + A[11:8] x B[7:4] + + A[7:4] x B[11:8] + A[3:0] x B[15:12] |

(b)

Fig. 1: Outside view of a Multiplier/dot-product unit (a) and its precision configurations (b).

Multipliers with Sum-Together (ST) mode [1] are good candidates to realize precision-scalable MAC units [2–5]. They are special sub-word parallel multipliers that can perform either a multiplication at full precision or a dot-product at lower precision. In particular, they can compute $N = 1, 2, 4$ multiplications/dot-products in parallel among input operands with precision inversely proportional to $N$ (e.g., $16/N$ bits). When used inside MAC units, they reduce the overall latency up to $1/N$ because they save $N$-1 MAC additions.

In the literature we find different proposals for the implementation of ST multipliers. The design of [2] uses four 16-bit Booth multipliers with a configurable partial products compression array and three configurable 33-bit adders, instead of the usual final adder. The two sub-word parallel dot-product units of [3] comprise of two 17-bit multipliers or four 9-bit multipliers, respectively, followed by a 32-bit adder. The reconfigurable parallel inner product of [4], the ancestor of [5], dynamically composes and decomposes 4 or 8-bit multipliers with a network of combinational logic.

In this context, we propose a new precision-scalable modified Radix-4 Booth signed multiplier with ST mode. Its configuration options are reported in Fig. 1. The main difference with respect to other state of the art (SoA) ST multipliers is that our design does not require a dedicated adder to sum the low-precision products together, but it exploits the normal alignment of partial products as in a standard multiplier (Fig. 2).

The comparison between the SoA ST multipliers and our design in performance, power and area (PPA) on a 28-nm technology shows that, at the cost of limited overhead in area and power compared to a conventional non-ST Booth design, our multiplier/dot-product unit (1) is superior to the other SoA units in the clock frequency range between 100 and 1000 MHz and (2) could reduce latency and energy of convolutional layers when used in MAC units or in variable-precision DL accelerators.

## 2 Hardware Design

We propose a new precision-configurable Radix-4 Booth signed multiplier with ST mode. The supported configurations are those of Fig. 1(b). The key feature of our multiplier/dot-product unit is the lack of a dedicated external adder to *sum together* the low-precision products during dot-product operations. Instead, such addition comes for free in our design. In fact, our unit exploits the normal alignment of partial products in a standard multiplier, enabling the computation of dot products when two or four scalar inputs are packed in each operand, as shown in Fig. 2. The bits of output $P$ (yellow circles) are obtained by vertically summing the full-colored circles representing the bits of the eight partial products (PP0-PP7). These full-colored bits are the result of the products of operands with the same color, while the half-colored bits are zeroed as explained below.

In the multiplier architecture shown in Fig. 3, the yellow blocks are the standard components of a Radix-4 Booth multiplier, while the green ones are for precision reconfiguration, for zeroing the half-colored bits of Fig. 2, and for sign extending the inputs in asymmetric configurations, like 16×8 and 8×4, in order to treat them as 16×16 and 8×8. We implement the reduction tree as a Wallace tree with 4:2 compressors, while the final adder is a Carry Propagate Adder with Prefix Network. Regarding the additional logic for reconfigurability, the configuration signal $CONFIG$ controls: 1) how the bits of operand $A$ are properly composed to form $X_0$-$X_7$ input triplets for the encoder; 2) how the sub-words of operand $B$ are arranged and presented to the $Y_0$-$Y_7$ inputs of the selector; 3) the number of positions to right-shift the output to the LSB position (Fig. 2).

## 3 Experimental Results

For a fair comparison, we re-implemented the ideas of the SoA multipliers introduced in Sec. 1, making these minimal adjustments:

- we standardized their configurations to match those presented in Figs. 1 and we removed all the unnecessary logic that was not necessary to implement the ST multiplier's behavior;
- since the authors of [3] implemented their ST multiplier with a behavioral RTL description, we made the same, but we forced the synthesizer to use a 16-bit Booth multiplier for 16×16/16×8 configurations;
- we right shifted the output of [4] to align it to the LSB because it produces the sum-of-products on higher bit positions, as it happens in our design;
- we added input and output registers to all the ST multipliers, ours included.

We synthesized the designs with Synopsys Design Compiler on a 28-nm technology, varying the clock frequency from 100 to 1000 MHz. The PPA results in the area vs clock period space and power vs clock period space are in Fig. 4(a)-(b), respectively. Our unit is Pareto optimal in area at all frequencies with ≈35% less area than other SoA competitors, while in power all the designs almost overlap each other. A close examination at 1000 MHz is reported in Tab. 1(a), where ST

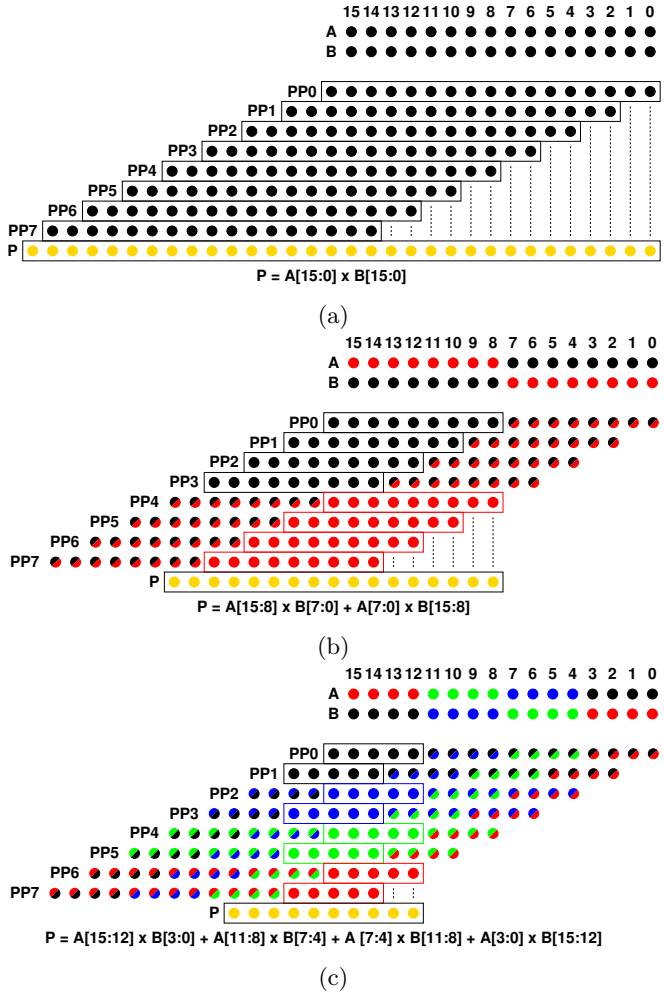



Fig. 2: Alignment of PPi partial products for $CONFIG$ 16×16/16×8 (a), 8×8/8×4 (b) and 4×4 (c).

multipliers are also compared with a conventional non-ST 16-bit Booth multiplier. From this table we find that our Booth multiplier with ST mode consumes +10% of area and +13% of power compared to the baseline version.

In Tab. 1(b) we show how an ST multiplier could reduce the number of MAC operations and the latency of a convolutional layer, in this case the first of MobileNetV1 and EfficientNet-B0. The theoretical reduction that is possible to achieve is $1/N$ and depends on the precision of activations and weights at which the layer is computed. Finally, it is important to note that we also expect a significant energy saving at lower precisions ($N = 2$ or 4) because energy scales like latency, while power overhead is constant.

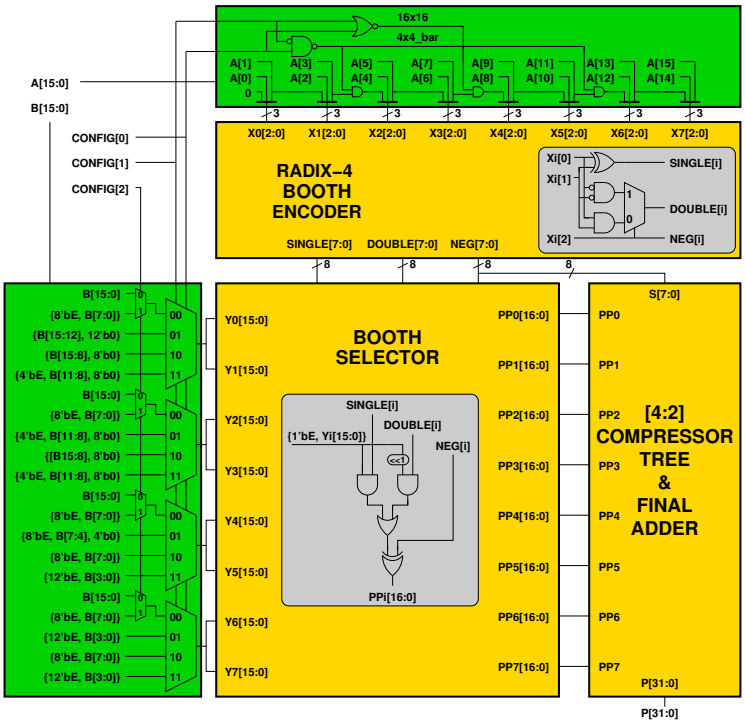



Fig. 3: The design of our Booth ST multiplier.

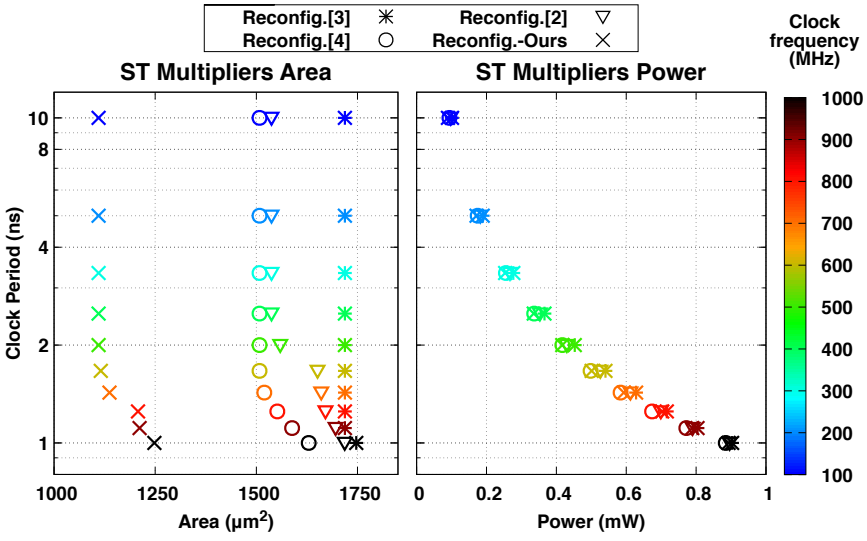



Fig. 4: PPA analysis of the analyzed ST multipliers.

(a)

| Multiplier | Area [$\mu m^2$] | Power [mW] |
|---|---|---|
| Non-reconfig. | 1133 | 0.791 |
| Reconfig.-ours | 1248 (+10%) | 0.893 (+13%) |
| Reconfig.[2] | 1747 (+54%) | 0.903 (+14%) |
| Reconfig.[3] | 1718 (+52%) | 0.896 (+13%) |
| Reconfig.[4] | 1629 (+44%) | 0.885 (+12%) |

(b)

| Multiplier | MAC Ops. | Latency |
|---|---|---|
| Non-reconfig. | 10.8 M | 1x |
| Reconfig. 16×16/16×8 | 10.8 M | 1x |
| Reconfig. 8×8/8×4 | 5.4 M | 0.5x |
| Reconfig. 4×4 | 2.7 M | 0.25x |

Table 1: ST multipliers vs baseline (non-ST 16-bit Booth multiplier) at 1000 MHz (a); theoretical reduction in MAC operations and latency for the first layer of MobileNetV1 and EfficientNet-B0 computed with an ST-multiplier (b).

## 4 Conclusion

This Booth multiplier with ST mode can play an important role inside precision-scalable MAC units or in variable-precision DL accelerators for edge devices [6] because it supports low-precision configurations which can reduce latency and energy. It also outperforms the SoA alternatives in area with limited reconfigurability overheads against a conventional non-configurable Booth multiplier.

## References

1. Camus, V. et al: Review and Benchmarking of Precision-Scalable Multiply-Accumulate Unit Architectures for Embedded Neural-Network Processing. IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JESTCS) **9**(4), 697–711 (2019)
2. Zhang, Z. Li, Zheng, Q.: Design of a configurable fixed-point multiplier for digital signal processor. In: Proceedings Asia Pacific Conference on Postgraduate Research in Microelectronics & Electronics (PrimeAsia), pp. 217–220. IEEE, Shanghai China (2009).
3. Gautschi, M. et al: Near-threshold RISC-V core with DSP extensions for scalable IoT endpoint devices. IEEE Transactions on Very Large Scale Integration (VLSI) Systems **25**(10), 2700–2713 (2017)
4. Lin, R.: Reconfigurable parallel inner product processor architectures. IEEE Transactions on Very Large Scale Integration (VLSI) Systems **9**(2), 261–272 (2001).
5. Sharma, H. et al.: Bit Fusion: Bit-Level Dynamically Composable Architecture for Accelerating Deep Neural Network. In: Proceedings 45th Annual International Symposium on Computer Architecture (ISCA), pp. 764–775, ACM/IEEE, (2018).
6. Urbinati, L., Casu, M. R.: A Reconfigurable Depth-Wise Convolution Module for Heterogeneously Quantized DNNs. In: Proceedings International Symposium on Circuits and Systems (ISCAS), pp. 128–132. IEEE, Austin Texas (2022).