TEMET: Truncated REconfigurable Multiplier with Error Tuning

# TEMET: Truncated REconfigurable Multiplier with Error Tuning

Flavia Guella (✉), Emanuele Valpreda, Michele Caon, Guido Masera, and Maurizio Martina (✉)

Dipartimento di Elettronica e Telecomunicazioni, Politecnico di Torino, Turin, Italy
{flavia.guella, maurizio.martina}@polito.it

**Abstract.** Approximate computing is a well-established technique to mitigate power consumption in error-tolerant domains such as image processing and machine learning. When paired with reconfigurable hardware, it enables dynamic adaptability to each specific task with improved power-accuracy trade-offs. In this work, we present a design methodology to enhance the energy and error metrics of a signed multiplier. This novel approach reduces the approximation error by leveraging a statistic-based truncation strategy. Our multiplier features 256 dynamically configurable approximation levels and run-time selection of the result precision. Our technique improves the mean-relative error by up to 34% compared to the zero truncation mechanism. Compared with an exact design, we achieve a maximum of 60.1% power saving for a PSNR of 10.3dB on a 5x5 Sobel filter. Moreover, we reduce the computation energy of LeNet by 31.5%, retaining 89.4% of the original accuracy on FashionMNIST.

**Keywords:** multiplier, approximate computing, reconfigurable computing, image processing, VLSI

## 1 Introduction

Multiplication is the fundamental building block in most computer vision, digital signal processing, and artificial intelligence applications. Such tasks could involve millions or billions of multiplications, which significantly contribute to the overall complexity and energy demand. These high power requirements could become unbearable for mobile and edge devices, raising the urge to find viable alternatives. Approximate computing exploits the intrinsic robustness of computer vision algorithms to numeric errors to lower power consumption while preserving acceptable results. Several inexact multipliers designs have been described in the literature, leveraging different techniques, with the main aim of enhancing power performance with negligible loss in accuracy [6–9]. However, few works deal with the ever-increasing need for an architecture capable of dynamically adjusting based on specific tasks and requirements [12]. Reconfigurability allows hardware to adapt run-time to the workload requisites, potentially increasing energy saving at the cost of additional area. We propose a 9-bit input, signed multiplier with dynamic truncation for the run-time selection of the precision

(i.e., bit-width) and approximation level of the final product. This is necessary to support mixed-precision and layer-wise quantization, as it is fundamental to reduce the computational cost, the memory footprint, and the overall data movement during the execution [2]. The contributions of this paper are summarized as follows: (i) enabling simultaneous run-time selection of precision and approximation of the operands and result, (ii) fine-grain regulation of the error, with the possibility of choosing among 256 available levels, including one generating the correct result, providing several energy-quality trade-offs, (iii) introduction of a simple error compensation technique to lessen the effect of bit truncation.

## 2 Related Works

Beyond dynamic voltage-scaling [1], which generally produces an error that is complex to control and requires additional hardware to tweak the voltage, and Cartesian Genetic Programming [3], many works explore parallel multipliers. Their architecture comprises three parts on which approximation can be performed: (i) partial product generation, (ii) reduction tree, (iii) two-operands adder. The first technique simplifies the logic required to generate partial products [4]. Approximating the tree structure means substituting exact compressors with inexact ones [5] or truncating the least significant part of the matrix. Approach (iii) is quite common since there are many available approximate adders designs ready to deploy. As approximate computing gained renewed appeal for several IoT applications, including Deep Neural Network (DNN) for monitoring people health, precision agriculture, and food quality, the need for reconfigurable and inexact hardware emerged [15]. Most studies on approximate multipliers do not include dynamic reconfigurability, as it generally comes at the expense of increased power and area. Works such as [6–8] exploit previously described techniques to implement inexact unsigned multipliers with some error trimming. None of the cited works supports run-time selection of the input precision, and the configuration capability is limited; these designs sacrifice the possibility of having the correct result, thus flexibility, in exchange for higher energy saving. Dynamic truncation is introduced in [9], implemented with a signal that can set to zero selected columns of the partial product matrix. This multiplier is designed for signed operands and can execute exact multiplication besides inexact ones. A similar underlying principle is used in our architecture. As a final remark, while most previous works proposed to implement either 8x8 or 16x16 unsigned multipliers, our work suggests a 9x9 signed multiplier to support all combinations of 8-bit dot products: signed-signed, unsigned-signed, signed-unsigned, and unsigned-unsigned.

## 3 Methodology

This section covers the design phase of the signed 9x9 multiplier. From [10], it is known that the Dadda tree has a lower delay and complexity than the Wallace one. Therefore, the former structure is selected to allocate the compressors. As our multiplier should support one level producing the correct result, a final exact

two-operands adder is instantiated and the tree-reduction uses only exact 3 to 2 or 2 to 1 compressors. The multiplication algorithm is implemented with the Modified Baugh-Wooley (MBW) due to its lower power consumption and area compared to Modified Booth Encoding [11]. Once the architectural structure of the exact multiplier has been fixed, the possibility of run-time configuration is achieved by using two masks, setting the precision and the approximation levels fed to the multiplier. Since reducing the dynamic power is the main objective of the optimization procedure, the choice is to perform data gating. The part of the partial product matrix (PPM) that is not involved in the operation is kept constant to reduce the switching activity and, thus, dynamic power consumption. A signal called `res_mask` is in charge of masking to zero the portion of the PPM not required for the computation, with a bitwise *and*. The `res_mask` signal covers the most significant fourteen bits of the result, as it is assumed that the minimum precision of the inputs is two bits. As the result has to be signed, additional logic is required to guarantee that its left part is correctly sign extended.

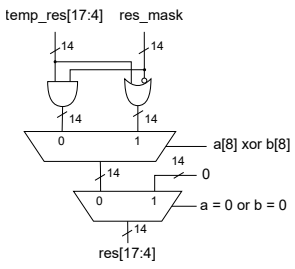


Fig. 1: Result sign extension

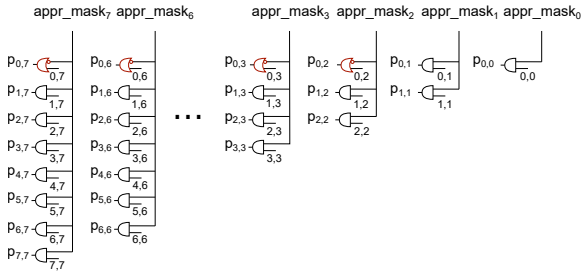


Fig. 2: Approximation mask

As depicted in Figure 1, the sign bit, obtained as the *xor* between the most significant bit of the multiplicand and that of the multiplier, is used as the selection signal of a multiplexer producing the leftmost part of the output. The precision mask is used to correctly sign-extend the product. Two zero comparators are implemented to cover the corner case where one of the two inputs is zero, and the other is negative. Although the two 9-bit zero comparators are an additional cost in terms of area and power, they enable the multiplier to produce the correct result whenever at least one operand is zero. Accurate zero multiplication is essential in DNN inference or image processing, so it is an acceptable trade-off. Moreover, the sensitivity to numeric errors of different DNNs layers has high variability [2]. Consequently, designing a multiplier with different approximation levels is crucial to ensure flexibility and adaptability. Run-time reconfigurability requires an area and power overhead; thus, the most convenient trade-off has to be evaluated. Following a holistic approach, a variation of truncation is identified as the target method for approximation, as it easily enables dynamic configuration without requiring the design of programmable approximate compressors and their insertion in specific points of the PPM. The selection of the level of approximation is achieved using a second mask signal, called `appr_mask`. While for the precision configuration the cut is on the most significant bits (MSBs)

of the matrix, here the truncation is on the right part. Once an approximation method has been established, the issue of the number of reconfigurability levels has to be investigated. A first consideration, also argued by other works such as [8] and [7], is that there is no advantage in pushing approximation to the most significant half of the result, as the error becomes unacceptable. The design choice is thus to limit the truncation to the least significant 8 bits of the multiplier. Given a configuration, for each bit at zero in the `appr_mask`, all the corresponding columns of bits of the PPM are set to a fixed value. Configurations with more zeros in the mask are more effective than others for data gating. Although some levels are Pareto-dominated in terms of power saving and error metrics, they might be the optimal choice for a specific layer in a DNN, as this strictly depends on its inputs and weights distribution. Consequently, the choice is to keep all 256 levels, even if some are less power efficient. Figure 2 shows the implemented mechanism to decrease the generated error. The first row has the bits from 2 to 7 fixed at one, while all others are data-gated to zero. An in-depth analysis is carried out to find optimal configurations of the data-gated matrix minimizing the error.

| $Bit_i$ | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| $P_{sum}$ | $\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{2}$ |
| $P_{C_o}$ | $\frac{119}{256}$ | $\frac{1}{2}$ | $\frac{9}{16}$ | $\frac{5}{8}$ | $\frac{5}{8}$ | $\frac{1}{2}$ | $\frac{1}{4}$ | 0 |
| $P_{C_{o+1}}$ | $\frac{41}{64}$ | $\frac{1}{2}$ | $\frac{11}{32}$ | $\frac{3}{16}$ | $\frac{1}{16}$ | 0 | 0 | 0 |
| $P_{O_i \geq 1}$ | $\frac{107}{128}$ | $\frac{421}{512}$ | $\frac{211}{256}$ | $\frac{13}{16}$ | $\frac{3}{4}$ | $\frac{5}{8}$ | $\frac{1}{2}$ | $\frac{1}{2}$ |

Table 1: Sum and carry probabilities

| Design | Area $[\mu m^2)]$ | Arrival Time $[ns)]$ | Approx Level | Power $[\mu W)]$ |
|---|---|---|---|---|
| Exact | 607.3 | 1.7 | - | 414.0 |
| Proposed | 822.6 | 1.8 | 0 | 240.6 |
| | | | 255 | 164.8 |
| Dyn Trunc | 816.5 | 1.8 | 0 | 242.4 |
| | | | 255 | 165.0 |

Table 2: Metrics of 9x9 signed exact and reconfigurable multipliers

Under the assumption of uniformly distributed inputs and of 9-bit full-precision inputs some considerations can be made. The purpose of the Dadda tree is to compress each of the columns of the PPM up to a single bit. Given a column, it is supposed that its bits can take all possible values with the same probability. Consequently, for each bit of the result we can evaluate its likelihood of being one or greater (i.e. generating a one in the position at its left). The probability of the sum or the carry produced by a column resulting in a one is thus evaluated. As the sum bit is an odd function, its probability of being one is always 0.5. All the probabilities for the carries are evaluated and reported in Table 1. The probability of the bit $i^{th}$ of the result, from position 0 to 7, being one or greater, $P_{O_i \geq 1}$ in Table 1, is so estimated:

$$P_{(O_i \geq 1)} = P_{sum_{(i)}} \cup P_{C_{o_{(i-1)}}} \cup P_{C_{(o+1)_{(i-2)}}} \tag{1}$$

Results in Table 1 demonstrate that from position 2, as expected, the probability of the output bit being greater than 0 is higher than 50%. This high likelihood justifies the logic shown in Figure 2.

## 4 Experimental Results

The synthesis is performed with the UMC 65nm technology library and a timing constraint of $2ns$, using Synopsys Design Compiler (DC). The golden model for the exact multiplier is a behavioral 9x9 bit signed multiplier whose architecture selection and optimization are left to Synopsys DC, with the library Synopsys DesignWare. Furthermore, an architecture similar to ours but with the traditional dynamic truncation to zero for the approximation logic, based on [9], is considered as a comparison. Power estimation is performed through back-annotation of the post-synthesis netlist using a testbench producing 1000 random stimuli. The procedure is repeated for every configuration of the precision and approximation masks of our multiplier. The main metrics are compared in Table 2.

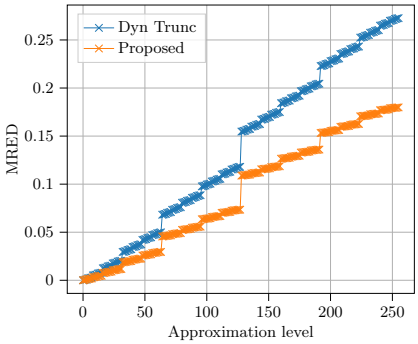


Fig. 3: MRED comparison between Dyn Trunc and Proposed, 9-bit

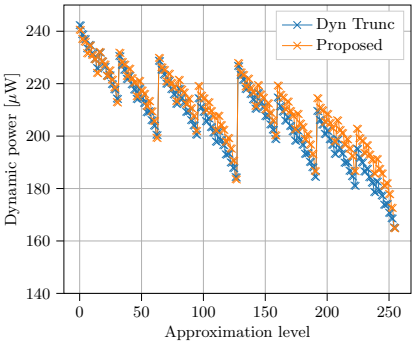


Fig. 4: Power comparison between Dyn Trunc and Proposed, 9-bit

The complexity of the exact architecture is lower as reconfigurability has some overhead on the occupied area and delay. Our multiplier, in the full-precision, exact configuration, saves 41.4% of power compared to the one by Synopsys, while the highest approximate level can save up to 60.1%. These results are remarkable and demonstrate the effectiveness of approximation in reducing power. Furthermore, they prove that the reconfigurability overhead is tolerable and does not significantly impact the critical part of the design. As a final remark, modifying the approximation logic does not impact the timing of the architecture and power increase is always below 5%. Figure 3 shows the mean-relative error distance (MRED) variation with the approximation level, comparing our multiplier with traditional dynamic truncation for the full-precision case. As can be observed, the proposed strategy allows an improvement of the MRED for every approximation level, with a maximum gain of 34%. Experiments are carried out on a typical image processing task for edge detection using a 5x5 Sobel filter, as in [5]. Figure 5 reports the PSNR values, evaluated on the gray-scale peppers benchmark image[1] for changing approximation levels for both traditional dynamic truncation and ours. The proposed truncation methodology improves

[1] https://sipi.usc.edu/database/database.php?volume=misc&image=11#top

PSNR by up to 81.6%. Considering an acceptable threshold of 25dB for the PSNR, 64 configurations of our multiplier, against 25 for traditional truncation, are usable. Moreover, we tested the 256 approximation levels with LeNet [13] on the FashionMNIST dataset [2]. We used PyTorch [3] and AdaPT [14] to implement the neural network, training the model for 32 epochs using stochastic gradient descent, a learning rate of $10^{-3}$, and a decay of $5 \cdot 10^{-4}$, with weights and activations quantized to 8-bit. Then we changed the approximation level of our multiplier and retrained the bias of each convolutional and dense layer for one epoch, without updating the other parameters. Figure 6 reports the results before and after the re-training. The accuracy loss can be recovered by updating the bias, accounting for the computation error introduced by the approximation. We evaluate the total computation energy as the product of a single multiplication and the number of operations required to execute the inference. With our approximate multiplier, it is possible to reduce by 31.5% the computation energy of LeNet, with an absolute top 1 accuracy degradation of 9.67%.

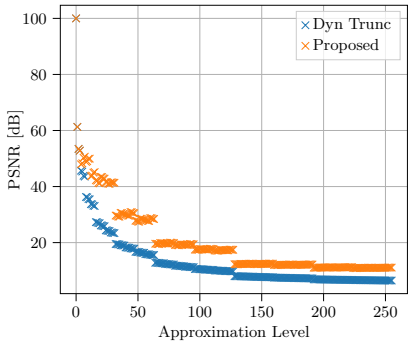


Fig. 5: PSNR comparison between Proposed and Dyn Trunc multiplier for 5x5 Sobel filter.

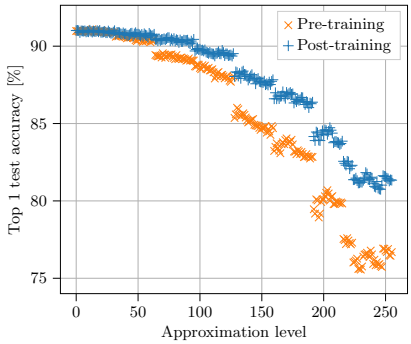


Fig. 6: Top 1% test accuracy variation with LeNet trained on FashionMNIST.

## 5 Conclusion and Future Works

This work demonstrates the effectiveness of approximate computing in reducing the power consumption in error-resilient computationally complex tasks. Furthermore, it points out the importance of reconfigurability for providing flexibility. In the future, we will investigate the effect of inexact computation on different applications, such as DNNs with layerwise approximation, and we will explore in depth the trade-offs that the concurrent adoption of approximation and reduced precision can yield. We will also consider approximate and resilient computing in different applications such as agritech [15][4].

[2] https://www.kaggle.com/datasets/zalando-research/fashionmnist

[3] https://pytorch.org/

[4] This work is part of the project NODES which has received funding from the MUR – M4C2 1.5 of PNRR with grant agreement no. ECS00000036

## References


1. Moons, B., Verhelst, M.: DVAS: Dynamic Voltage Accuracy Scaling for increased energy-efficiency in approximate computing. In: 2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED), pp. 237-242, 2015.
2. Fasfous, N., et al.: Hw-flowq: A multi-abstraction level hw-cnn co-design quantization methodology. In: ACM Transactions on Embedded Computing Systems (TECS), 20, 5s, Article 66, 2021.
3. Mrazek, V., Sekanina, L., Vasicek, Z.: Libraries of Approximate Circuits: Automated Design and Application in CNN Accelerators. In: IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 10, no. 4, pp. 406-418, 2020.
4. Yin, P., et al.: Designs of Approximate Floating-Point Multipliers with Variable Accuracy for Error-Tolerant Applications. In: Journal of Signal Processing Systems, vol. 90, no. 4, pp. 641-654, 2018.
5. Strollo, A. G. M., et al.: Comparison and Extension of Approximate 4-2 Compressors for Low-Power Approximate Multipliers. In: IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 67, no. 9, pp. 3021-3024, 2020.
6. Jiang, H., et al.: Low-Power Approximate Unsigned Multipliers With Configurable Error Recovery. In: IEEE Transactions on Circuits and Systems I: Regular Papers, vol.66, no.1, pp. 189-202, 2019.
7. Yang, T., Ukezono, T., Sato, T.: A low-power high-speed accuracy-controllable approximate multiplier design. In: 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC), pp.605-610, 2018.
8. Gu, F.-Y., Lin, I.-C. and Lin, J.-W.: A Low-Power and High-Accuracy Approximate Multiplier With Reconfigurable Truncation. In: IEEE Access, vol. 10, pp. 60447-60458, 2022.
9. de la Guia Solaz, M., Han, W., Conway, R.: A Flexible Low Power DSP With a Programmable Truncated Multiplier. In: IEEE Transactions on Circuits and Systems I: Regular Papers, vo. 59, no.11, pp. 2555-2568, 2012.
10. Parhami, B.: Computer Arithmetic - Algorithms and Hardware Designs (2nd Edition). Oxford University Press, New York (2010)
11. Sjalander, M., Larsson-Edefors, P.: High-speed and low-power multipliers using the Baugh-Wooley algorithm and HPM reduction tree. In: 2008 15th IEEE International Conference on Electronics, Circuits and Systems, pp. 33-36, 2008.
12. Fasfous, N., et al.: AnaCoNGA: Analytical HW-CNN Co-Design Using Nested Genetic Algorithms. In: 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 238-243, 2022.
13. Lecun, Y., et al.:Gradient-based learning applied to document recognition. In: Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, 1998.
14. Danopoulos, D., et al.: Adapt: Fast emulation of approximate dnn accelerators in pytorch. In: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 42, no. 6, pp. 2074-2078, 2022.
15. Barezzi, M., et al.: On the impact of the stem electrical impedance in neural network algorithms for plant monitoring applications. In: 2022 IEEE Workshop on Metrology for Agriculture and Forestry (MetroAgriFor), pp. 131-135, 2022.