Scalable and Power Efficient Approximate Multipliers for Error-Resilient Applications in 22 nm

Victor Åberg[†], Sergio Castillo Mohedano[†], Kristoffer Westring[†], Alex Allfjord[†],
Per Andersson[†], Joachim Rodrigues[†‡], and Masoud Nouripayam[†]
[†] Department of Electrical and Information Technology, Lund University, Lund, Sweden
[‡]Department of EEI, System-on-Chip, Friedrich-Alexander-Universität, Erlangen-Nürnberg, Germany

***Abstract*—Approximate multipliers (AMs) show great potential to achieve substantial area and power savings when replacing their exact counterparts in error-resilient applications. In this paper, we propose an AM architecture, scalable across operand widths. Our 8-bit AM achieves a 1.91 % Mean Relative Error Distance (MRED) and a power-area-delay product (PADP) gain of >78.2 % over a Wallace multiplier. We also introduce an evaluation methodology focused on post-implementation hardware savings. Beyond standalone benchmarking, measurements of a 22 nm chip confirm substantial power reductions of 32.9 % when replacing exact multipliers with our AM.**

***Index Terms*—Approximate multiplier, energy-efficiency, error resilience, scalable architecture.**

I. Introduction

With the rapid growth in AI, the demand for area- and energy-efficient processing is soaring. Neural networks (NNs) allow for substantial savings through shortening wordlengths, as their inherent error-resilience compensates for approximation impacts during training [1]. This error-resilience also makes these algorithms suitable candidates for approximate computing techniques to realize further hardware savings [2]. The multiplier is identified as a suitable candidate due to its central role in computation and its high resource demand.

A standard multiplication is composed of partial product (PP) generation, PP accumulation, and the final addition. The binary compression of the PP array is commonly realized using a carry-save adder (CSA), using full-adders (FAs) and half-adders (HAs). However, this implementation results in a long critical path, having a significant impact on power and area. Many approximate multipliers (AMs) address this bottleneck by exploiting probabilistic error analysis, structural simplifications, truncated multiplication, or static PP perforation [3]–[5]. Shortening the depth of the accumulation tree may bring further power and area gains [6]–[8]. However, scalability of these works is often limited; fixed wordlength, rigid approximation structures, or over-dimensioned recursive construction all contribute to this inflexibility.

In this work, we introduce a scalable hardware-efficient AM architecture based on a novel priority-select PP removal mechanism. Our approach follows a deterministic algorithm, eliminating low-impact computations through row-pairing. We also introduce a new hardware evaluation approach, centered around actual hardware savings. Finally, we also demonstrate the hardware gain when using AMs in an actual application.

The remainder of the paper is organized as follows: Section II introduces the proposed AM algorithm. Section III presents a unit level analysis and compares the proposed AM against other state-of-the-art multipliers. In Section IV, the AM is evaluated and measured at system level to verify its applicability and measure the performance gains. Finally, the paper is concluded in Section V.

This work was financially supported by Vetenskapsrådet, the SSF-financed classIC Research Center, BCDC by the Bavarian Ministry of Economic Affairs (Regional Development and Energy, RMF-SG20-3410-2-17-14), and the Chip JU REBECCA project (101097224).

II. Approximation algorithm

Throughout this paper, the multiplicand and the multiplier operands are denoted A and B, respectively. For an *n*-bit multiplication, four distinct PP-regions may be formed by splitting each operand into two *n/2*-bit words, identified as low (L) and high (H), giving us $A_L \times B_L$, $A_L \times B_H$, $A_H \times B_L$, and $A_H \times B_H$. The cumulative property allows us to align the PPs in $A_H \times B_H$ with $A_L \times B_L$, as shown in Fig. 1a. This reveals that PPs in the cross regions ($A_L \times B_H$ and $A_H \times B_L$) are responsible for the accumulation bottleneck, which dominates the critical path increasing both delay and power. To address this accumulation bottleneck, we propose an approximation strategy that generalizes across wordlengths and selectively removes low-impact PPs. Our algorithm is guided by these two key factors:

- **Safe Discard:** All '0' bits in B produce null PPs, removable without accuracy loss.
- **Significance-Aware Approximation:** When two bits of B are '1', the PPs corresponding to the less significant bit have lower impact and are prioritized for removal.

With each bit in B directly mapping to a row in the PP tree, our proposed approach reduces the tree-depth by forming bit-pairs (B_{Li}, B_{Hj}), where *i* and *j* are indices within the B_L and B_H subwords, respectively. From our algorithm, we introduce a priority-based approximation (PRAX) logic

$$PRAX = A_{Li} \cdot B_{Hj} + A_{Hj} \cdot B_{Li} \cdot \overline{B_{Hj}} \tag{1}$$

that embeds the approximation logic directly into the PP generation, eliminating the need for a separate compression stage. This logic can be simplified into a single OR-gate

$$PRAX - OR = A_{Li} \cdot B_{Hj} + A_{Hj} \cdot B_{Li}, \tag{2}$$

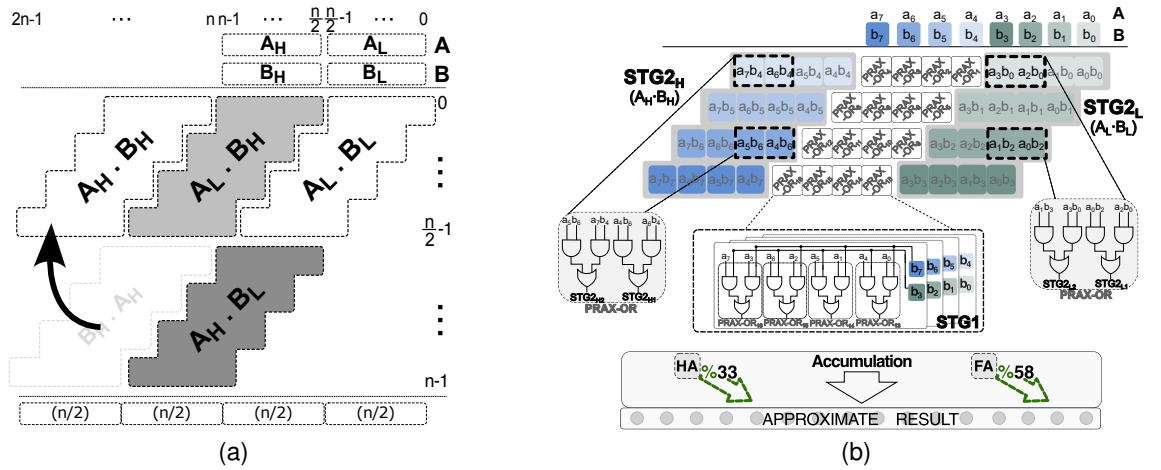

Fig. 1. (a) Accumulation bottlenecks in $A_L \times B_H$ and $A_H \times B_L$ due to densely stacked PPs. Proposed 8×8 AM architecture. Both STG1 and STG2 use PRAX-OR for PP reduction. HA/FA reductions compared to the corresponding Wallace tree [9].

without changing the significance-driven pairing. Figure 1b illustrates how PRAX-OR approximates 16 PPs from the $A_L \times B_H$ and $A_H \times B_L$ regions (STG1) in the 8-bit case. In the general case, for an *n*-bit multiplier, STG1 will eliminate $(n/2)^2$ PPs without modifying the PP tree structure or reassessing the operand ranges, making the approach both scalable and broadly applicable while preserving hardware efficiency. The PRAX-OR logic yields one incorrect output result, which happens when both PPs are one, causing the carry information to be lost.

To further eliminate PPs, a second approximation stage (STG2) is introduced. While STG1 solely operates on the cross regions ($A_L \times B_H$ and $A_H \times B_L$), STG2 operates on the remaining non-cross regions ($A_L \times B_L$ and $A_H \times B_H$), treating them as independent *n/2* multiplications. This allows us to apply the same row-pairing strategy used in STG1 in a smaller, localized context. The row-pairs targeted for STG2 compression are highlighted in Fig. 1b, labeled $STG2_H$ and $STG2_L$ for the high and low operand sub-regions, respectively. To minimize accuracy loss, the compression in STG2 is limited to half of that applied in STG1.

III. Multiplier Unit Level Evaluation

The proposed AM architecture (2-stage configuration) is evaluated here against two different exact multiplier architectures, namely a Wallace-tree and the "*". The former is commonly used as a reference implementation when evaluating the AM in the literature, and is the topology that best matches our approximate multiplier. The "*" resembles the most efficient exact multiplier inferred by the synthesis tool (Cadence Genus), given the provided constraints, and is commonly used for inferring a multiplier on RTL.

A. Accuracy Analysis

The accuracy of our proposed AM is analyzed for both 8-bit and 16-bit operands, which demonstrates the architectures' scalability. We have chosen the 2-stage configuration for its balance between accuracy loss, and power and area gain. The accuracy analysis is performed across all possible combinations of the input operands. The results are quantified using the following metrics: Error Distance (ED) $R_{Accu.} - R_{Appx.}$, max absolute ED (MAE), error rate (ER), relative ED (RED) $|R_{Accu.} - R_{Appx.}|/R_{Accu.}$, max relative ED (MRE), mean RED (MRED), and normalized mean RED (NRED) $\frac{1}{2^{2N}(2^N-1)^2}\sum_{i=1}^{2^{2N}} ED_i$. As shown in Table I, errors remain bounded and predictable across operand width.

TABLE I
Error Analysis for Different Operand Sizes.

Word length (n)	#PPs Discarded	MAE	MRE (%)	ER (%)	NMED ($\times 10^{-3}$)	MRED (%)
8	20	6.68 k	19.77	52.61	6.42	1.91
16	96	29.3 M	19.86	84.98	0.48	0.29

B. Hardware Analysis

To study the hardware efficiency, the 2-stage AM and the exact baselines, Wallace-tree and "*", were realized in RTL, with interface registers at inputs and outputs excluded from reported area and power metrics. Each multiplier is then synthesized for stricter timing constraints, ranging from unconstrained timing up to the point where timing closure is no longer possible. Synthesis is carried out on GlobalFoundries 22FDX standard-cell libraries (TT, 0.8 V, 25 °C). Power consumption is estimated using post-synthesis toggling information based on the switching statistics for 10^5 uniform random numbers, scaled to the full operand range, gathered using annotated propagation delays using the target timing constraint used in synthesis.

Figure 2a shows area and power for all multipliers across a timing sweep for 8 and 16-bit wordlengths. Each multiplier demonstrates two clear regimes: a steady increase under relaxed constraints, followed by sharp escalation as timing becomes critical. Unlike in the literature, where

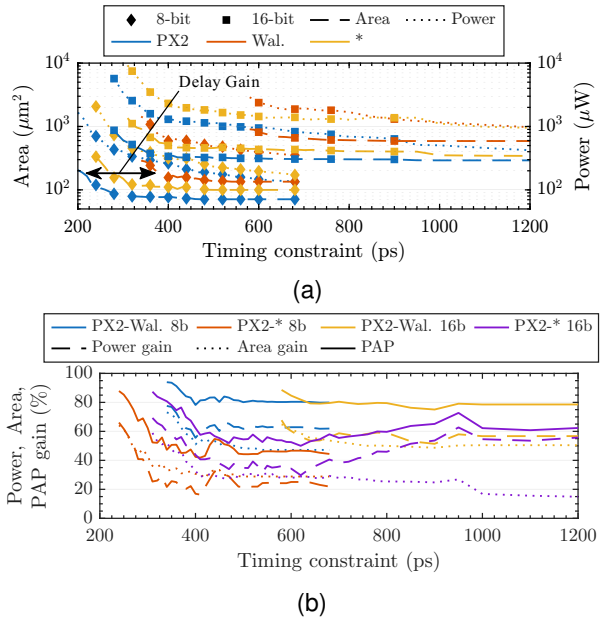

Fig. 2. (a) Area and power consumption versus propagation delay for our AM (PX2), Wallace (Wal.), and "*". (b) Power, Area, and PAP gain for our AM (PX2) AM relative to both Wallace (Wal.) and "*".

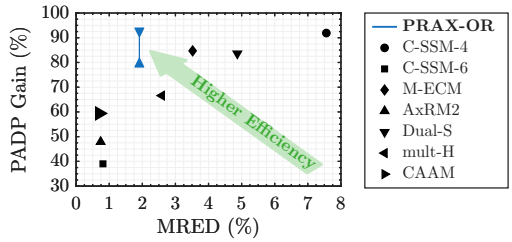

Fig. 3. Comparison of our proposed AM with state-of-the-art in terms of PADP gain and MRED. The blue arrow shows the achievable range of PADP gain for our proposed AM.

common "single-point" approaches decouple achieved area at minimum delay from simulated power at an arbitrary and unrelated frequency, our multi-point strategy captures their interdependencies at equal timing pressure, providing a realistic view of gains in actual hardware systems.

The power, area, and power-area product (PAP) gains relative to each of the exact multipliers are plotted in Fig. 2b, at equal timing conditions. Consequently, the power-delay product (PDP), area-delay product (ADP), and power-area-delay product (PADP) gains all scale directly with the power, area, and PAP gains. The proposed AM achieves area and power gains exceeding 47.2 % and 51.4 % against the Wallace baseline, and remains mostly above 20 % against "*"; aligning with observations from measurements. These improvements persist across the entire constraint sweep, validating the architecture's efficiency.

Because power and area strongly depend on the target constraint (Fig. 2b), Fig. 3 illustrates the accuracy–hardware

TABLE II
COMPARISON WITH STATE-OF-THE-ART 8-BIT UNSIGNED AMS. ALL RESULTS ARE BASED ON SIMULATIONS.

AM	Tech.	Power gain (%)	Area gain (%)	Delay gain (%)	NRED ($\times 10^{-3}$)	MRED (%)
PRAX-OR[1]	GF 22 nm	55.5[2]	47.2[2]	44.1[3]	6.42	1.91
[3]C-SSM-4	TSMC 28 nm	69.2	63.7	27.3	7.6	10.2
[3]C-SSM-6	TSMC 28 nm	16.5	21.5	6.9	0.8	1.6
[4]M-ECM	ASAP7	57.6	51.1	26.1	3.5	1.7
[5]AxRM2	28 nm	15	14.4	28.4	0.7	3.4
[6]Dual-S	45 nm	49.5	38	47.6	4.9	8.8
[7]mult-H	65 nm	36.2	32.4	22.7	2.6	1.1
[8]CAAM	TSMC 90 nm	40	26.6	7.8	0.7	0.4

[1] Manufactured and corroborated with measurements. [2] Minimum observed gain compared to Wallace. [3] Shortest delay fulfilling timing after synthesis.

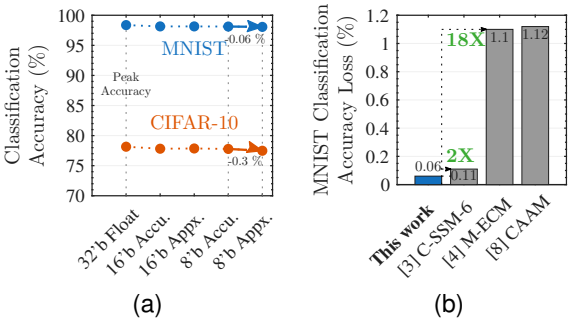

Fig. 4. Simulation Results: (a) Classification accuracy on MNIST and CIFAR-10 with exact and approximate multipliers across word lengths. (b) MNIST accuracy drop of the proposed AM vs. state-of-the-art.

trade-off by plotting our PADP gain range (vertical line) against the single-point results of state-of-the-art AMs. While constraint-relaxed, our AM reaches a PADP gain of 78.2 %, whereas reaching a peak gain of 93.9 % when the Wallace is strained. A further comparison against other 8-bit AMs is shown in Table II.

IV. System Level Impact

To further demonstrate the benefits of the proposed AM, the impact on a larger system's performance has been studied, both in terms of application accuracy and hardware cost.

A. Accuracy in Error-Tolerant Workloads

To evaluate the impact of the AM on an application, we have studied the impact in classification accuracy for two convolutional neural networks (CNNs), using a custom CNN for MNIST (10 k parameters) and AlexNet for CIFAR-10 (7.5 M parameters). Using 8-bit wordlengths, our proposed AM shows an accuracy loss of 0.06 % and 0.3 % for MNIST and CIFAR-10, respectively, compared to using exact multipliers. Similar results are also seen when using 16-bit wordlength, see Fig. 4a. This accuracy loss is negligible, outperforming all other state-of-the-art methods, see Fig. 4b.

B. Silicon Validation

Silicon validation of our proposed AM is realized by embedding it into a near-memory computing (NMC)

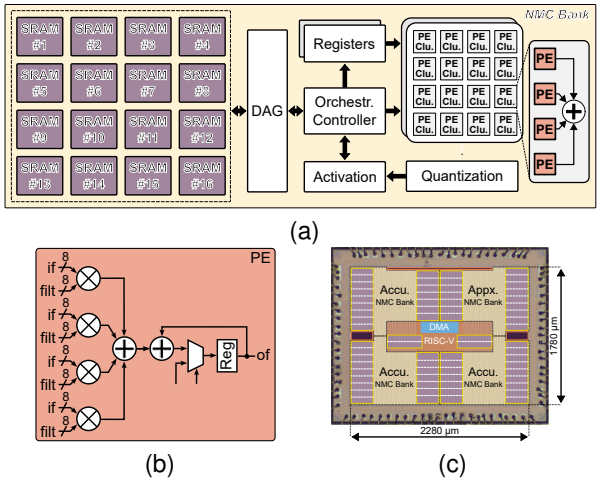

Fig. 5. (a) Block diagram for a single NMC bank. (b) PE with the four multipliers. (c) Chip photo highlighting the placement of the bank using AMs.

TABLE III
GATE AREA AND SIMULATED POWER CONSUMPTION AT DIFFERENT HIERARCHICAL LEVELS OF THE NMC WHEN USING EXACT AND APPROXIMATE MULTIPLIERS.

	Accu.		**Appx.**		**Gain**	
	Area (μm^2)	Power (mW)	Area (μm^2)	Power (mW)	Area (%)	Power (%)
Bank logic	139.3 k	57.2	133.6 k	48.9	4.1	12.8
PE array	30.0 k	14.0	27.0 k	9.4	10.2	32.9
PE	817	0.88	748	0.59	8.4	32.9

co-processor [10]. The NMC (see Fig. 5a) is built around an array of PEs and a surrounding data path that minimizes data movement to and from memory, thereby reducing power consumption. The PEs, shown in Fig. 5b, consist of four 8-bit multipliers followed by adders, making them a suitable candidate for replacing the exact multiplier with our proposed AM. An evaluation chip containing four NMC banks and a RISC-V processor for high-level management was manufactured using GlobalFoundries 22FDX, the die photo is shown in Fig. 5c. This chip contains three NMC banks using exact multipliers ("*") and a fourth bank where the exact multiplier is replaced by our approximate one; all other blocks in the NMC are kept unchanged.

The power savings brought by the introduction of AM is measured when running a tile of the first layer of AlexNet, using random input data. At 400 MHz@0.8 V, a single NMC bank consumes 53.5 mW using exact multipliers, and 46.6 mW using AM, showcasing a reduction of 12.9 % at the NMC bank level. We also performed post-layout simulations (TT, 0.8 V, 25 °C) using the same clock frequency and workload. Simulated NMC-bank power closely matches measurements, with minor differences due to the voltage reference plane being located on the PCB. Table III reports also power and area for both the PE array and a single PE (the lowest hierarchical level kept after placement), for each of the two multipliers. We here observe 8.4 % area gain and 32.9 % power gain when using AMs. The larger gains reported here are likely a result of differences in the optimizations performed on the standalone multiplier and the more complex data path we have in the PE. With the PE array making up 20 % of the NMC bank logic, the 12.9 % reduction in power consumption, measured at the NMC bank-level, as well as a reduced area cost, is significant.

V. CONCLUSION

In this paper, we present an AM architecture, scalable across operand widths, that shows a good balance between accuracy and efficiency. Using 8-bit operands, we achieve a MRED of 1.91 % and a minimum PADP gain of 78.2 %. The evaluation is performed using an approach that highlights the actual hardware gains across varying timing constraints to show the underlying behavior, rather than focusing on a single number. To showcase that the gains remain in an actual application, we replaced the exact multipliers in the PEs of a NMC co-processor with our proposed AM, gaining 12.9 % in power consumption measured across a complete NMC bank. To the best of our knowledge, this is the first time the benefits of using AMs in a larger system is demonstrated using a fabricated chip.

ACKNOWLEDGEMENT

The authors are grateful for financial support from Vetenskapsrådet, the SSF financed classIC Research Center, and the Chip JU REBECCA project (101097224), LUNARC at Lund University for the computational support, and GlobalFoundries for silicon fabrication.

REFERENCES

[1] W. G. Hatcher and W. Yu, "A survey of deep learning: Platforms, applications and emerging research trends," *IEEE Access*, vol. 6, pp. 24 411–24 432, 2018.
[2] S. S. Sarwar *et al.*, "Energy efficient neural computing: A study of cross-layer approximations," *IEEE Trans. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 4, pp. 796–809, 2018.
[3] A. G. M. Strollo, E. Napoli, D. De Caro, N. Petra, G. Saggese, and G. Di Meo, "Approximate multipliers using static segmentation: Error analysis and improvements," *IEEE Trans. Circuits Syst. I*, vol. 69, no. 6, pp. 2449–2462, 2022.
[4] F. Sabetzadeh, M. H. Moaiyeri, and M. Ahmadinejad, "An ultra-efficient approximate multiplier with error compensation for error-resilient applications," *IEEE Trans. Circuits Syst. II*, vol. 70, no. 2, pp. 776–780, 2023.
[5] H. Waris, C. Wang, C. Xu, and W. Liu, "AxRMs: Approximate recursive multipliers using high-performance building blocks," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 2, pp. 1229–1235, 2022.
[6] P. J. Edavoor, S. Raveendran, and A. D. Rahulkar, "Approximate multiplier design using novel dual-stage 4:2 compressors," *IEEE Access*, vol. 8, pp. 48 337–48 351, 2020.
[7] M. Zhang, S. Nishizawa, and S. Kimura, "Area efficient approximate 4–2 compressor and probability-based error adjustment for approximate multiplier," *IEEE Trans. Circuits Syst. II*, vol. 70, no. 5, pp. 1714–1718, 2023.
[8] U. Anil Kumar, S. V. Bharadwaj, A. B. Pattaje, S. Nambi, and S. E. Ahmed, "CAAM: Compressor-based adaptive approximate multiplier for neural network applications," *IEEE Embedded Syst. Lett.*, vol. 15, no. 3, pp. 117–120, 2023.
[9] G. Zervakis, K. Tsoumanis, S. Xydis, D. Soudris, and K. Pekmestzi, "Design-efficient approximate multiplication circuits through partial product perforation," *IEEE Trans. VLSI Syst.*, vol. 24, no. 10, pp. 3105–3117, 2016.
[10] A. Prieto *et al.*, "LUCIA: a scalable and versatile 0.82 TOPS/W/mm^2 chiplet architecture for CNN in 22 nm FDSOI," in *ESSERC*, 2026.