A LOW-POWER HIGH-SPEED ACCURACY-CONTROLLABLE APPROXIMATE MULTIPLIER DESIGN

[1]Adavathu Rajitha,

M. *Tech, VLSI System Design, Student,*
Department of Electronics and Communication Engineering,

JNTUH University College of Engineering Sulthanpur, Sangareddy, Telangana,INDIA.

Email: rajithaadavath@gmail.com

[2]Mr. DEVISINGH,
*Assistant Professor(C),*Department of Electronics and Communication Engineering,

JNTUH University College of Engineering Sulthanpur, Sangareddy, Telangana,INDIA.

Email: devisinghrathod@gmail.com

ABSTRACT

This paper presents an accuracy-controllable multiplier that generates its final product using a carry-maskable adder. The scheme can dynamically adjust the carry propagation length to meet accuracy requirements. The approximate multiplier consists of an approximate tree compressor (ATC) and a carry-maskable adder (CMA), which compress partial products power-efficiently and provide accuracy scalability using a simple carry-masking technique. The 8x8 approximate multiplier is implemented in Verilog HDL, dividing the design into four stages: partial product generation, hierarchical tree compression, OR-based summation of compensation vectors, and a final addition using a hybrid adder. The implementation balances accuracy, speed, and power, making it suitable for energy-efficient approximate computing in image processing, machine learning, and embedded signal processing applications.

Index Terms: - Approximate multiplier, accuracy-controllable multiplier, carry-maskable adder (CMA), approximate tree compressor (ATC), Verilog HDL, hybrid adder, truncated OR logic, ripple-carry adder.

I.INTRODUCTION

The multiplier is a fundamental arithmetic unit in digital systems, playing a pivotal role in applications such as basic arithmetic logic units (ALUs), complex digital signal processing (DSP), image processing, and machine learning accelerators. As computing moved into the embedded and portable realm, power consumption emerged as a critical design constraint, leading to the rise of energy-aware architectures that sought to balance computational performance with reduced power dissipation. This led to the development of approximate computing, which embraces a controlled trade-off between computational accuracy and system-level efficiency, such as power consumption, area usage, and processing delay.

Approximate computing is particularly attractive for embedded and portable systems, where energy constraints are stringent, and high-throughput data centers seeking to optimize performance-per-watt. Designers of approximate multipliers leverage techniques such as truncation, carry chain pruning, segmented or hybrid architectures, and stochastic and probabilistic computing. These strategies enable the design of multipliers that can significantly outperform exact counterparts in terms of energy efficiency and speed, albeit at the cost of some accuracy.

The design landscape of approximate multipliers has evolved over the past decade, evolving from simple static structures to sophisticated dynamically configurable architectures. Early approximate multiplier designs primarily relied on static techniques such as truncation and Lower-Part OR Adders (LOA), which offered energy and area savings by reducing the complexity of the carry propagation network. However, these designs lacked adaptability and offered only coarse-grained control over accuracy, making them suitable only for limited error-tolerant applications.

In response to the growing demand for tunable accuracy and energy-efficient computing, recent innovations have introduced configurable and reconfigurable approximate multipliers. Carry-Maskable Adders (CMAs) allow selective masking of carry bits in certain bit positions, facilitating a seamless transition between accurate and approximate operation without requiring hardware reconfiguration.

Hybrid structures that combine accurate and approximate datapaths have been proposed, allowing the design to maintain a balance between accuracy and efficiency. In today's era of ubiquitous computing, devices are increasingly battery-powered and expected to deliver real-time performance. Traditional multipliers, while accurate, incur significant power and area overheads, particularly problematic in edge computing environments such as mobile phones, autonomous systems, or wearable sensors.

The proposed research addresses this challenge by presenting an accuracy-scalable approximate multiplier that combines low-power operation with high-speed processing and runtime tunability. This is achieved through two key components:(ATC) and (CTC).

II.LITERATURE SURVEY

The demand for energy-efficient and performance-aware computing has led to the arithmetic multiplier being a key focus in hardware optimization efforts, particularly in approximate computing. As a resource-intensive component in digital signal processing (DSP), image processing, and neural network acceleration, the multiplier's area, power, and delay characteristics have made it a key target for architectural innovation. Approximate computing introduces a paradigm shift where precision is traded for improvements in hardware metrics, enabling real-time, resource-conscious designs suitable for error-resilient applications such as image enhancement, speech recognition, and edge AI.

The trajectory of innovation in approximate multipliers can be understood through three primary dimensions: approximations at the adder level, optimization of the partial product reduction (PPR) stage, and the introduction of runtime-configurable architectures. Early attempts to reduce multiplier complexity focused on its basic building block—the adder. Researchers observed that carry propagation chains in traditional adders contribute significantly to both power consumption and computational delay. This led to the creation of error-tolerant adders (ETAs), which introduced acceptable error margins in applications like image and video processing. However, fixed-error configurations lacked adaptability, preventing their use in scenarios with varying precision demands. To overcome this, research efforts shifted towards optimizing the partial product reduction (PPR) stage, which is often the most computationally intensive phase of multiplication.

The latest and most dynamic advancements in approximate multiplier design involve runtime-configurable and accuracy-scalable architectures, which address the fact that modern AI and machine learning systems operate across multiple layers with varying error tolerance. These designs often incorporate control logic, selectable compressor trees, or adaptive carry-masking to modulate error margins based on input or system state.

III. EXISTING SYSTEM

Traditional multiplier architectures in digital systems prioritize computational accuracy, typically using full carry-propagation adders and deep compression trees to ensure exact results. Designs such as Wallace tree and array multipliers, while efficient in terms of speed, often incur significant power and area overheads—making them less suitable for energy-constrained or error-resilient applications like image processing and machine learning. Previous approximate multipliers primarily employed static truncation or lower-part OR-based approximations, offering only coarse-grained accuracy control with limited adaptability during runtime.

IV PROPOSED SYSTEM

This paper presents a novel 8×8 Accuracy-Controllable Approximate Multiplier (ACM) that integrates an Approximate Tree Compressor (ATC) with a Carry-Maskable Adder (CMA) to deliver dynamic precision scalability. The architecture enables selective carry propagation through configurable control signals, allowing seamless switching between approximate and accurate modes. By dividing the final addition stage into three distinct zones—approximate (bits 0–4), accuracy-scalable (bits 5–11), and fully accurate (bits 12–15)—the design achieves a fine balance between power efficiency and computational fidelity. This makes the ACM particularly well-suited for low-power VLSI applications that demand tunable performance.

V.SYSTEM DESIGN

The **8x8 Accuracy-Controllable Multiplier (ACM)** is a carefully crafted arithmetic unit optimized for error-tolerant digital systems, especially in power-sensitive environments like signal processing and machine learning. Unlike conventional multipliers that strictly adhere to exact computation, the ACM adopts a flexible design that strategically trades off exactness for energy savings. It does so by leveraging **approximate hardware components**, such as **incomplete adder cells (iCAC)** and **carry-maskable adders (CMAs)**, while preserving high fidelity in critical paths using accurate arithmetic units. The architecture is organized into four sequential processing stages:

1. Partial Product Generation
2. Approximate Tree Compression (ATC)
3. 3:2 Compressor using Carry Save Adder (CSA)
4. Final Configurable Accumulation using CPA

Each of these stages is functionally modular and structurally pipelined to ensure reusability and optimization across FPGA and ASIC platforms. The operation of the multiplier, as outlined above , maps directly to the block diagram, illustrating how data flows from inputs to final output through a carefully controlled mix of approximate and accurate logic blocks.

1. Incomplete Half Adder Cell (iCAC)

The incom_halfAdder—abbreviated as iCAC is a fundamental building block in the approximate tree compression stage of the ACM multiplier. It is a low-complexity half adder tailored for high-speed, low-power computation where exactness is not strictly required, particularly in the least significant bit (LSB) regions of arithmetic operations.

Unlike a conventional half adder that computes:

Sum: s = a ^ b (XOR)

Carry: c = a & b

the incom_halfAdder takes a minimalist approach:

Approximate Sum (p): p = a | b

Carry (q): q = a & b (retained as-is for potential recovery)

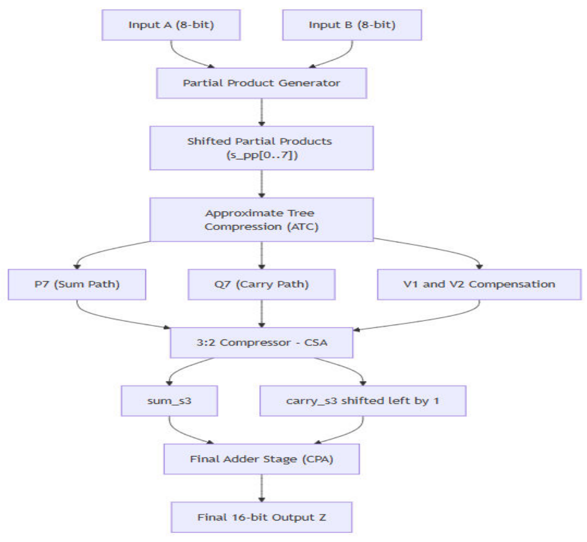


Fig: Architecture diagram of the proposed design.

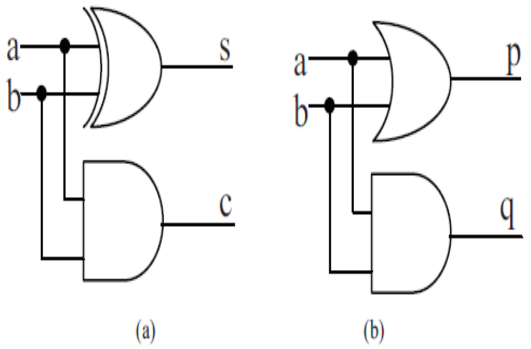


Fig:(a) exact Half Adder,(b) ncomplete adder cell and it's correspoding truth table

Inputs		Outputs			
		Accurate half adder		iCAC	
a	b	c	s	q	p
0	0	0	0	0	0
0	1	0	1	0	1
1	0	0	1	0	1
1	1	1	0	1	1

This simplified logic eliminates the **XOR gate**, which is typically more power-hungry and slower than basic gates like AND or OR. The p = a | b expression offers a close approximation of a ^ b, especially when the inputs are sparse or in low-transition conditions. For example:

In the only erroneous case (a=1, b=1), the approximate sum becomes 1 instead of the correct 0. However, the carry output q = a & b = 1 correctly indicates that a significant bitwise addition occurred. This error is acceptable in error-resilient applications and can be compensated in later stages (as done in the ACM design via vectors Q1–Q7 and Vsum).

- Gate Count Reduction: XOR requires 4–6 gates depending on implementation, whereas OR and AND each require only 1 gate.
- Delay Improvement: The critical path delay is shortened since the slow XOR is replaced with faster OR/AND logic.
- Power Saving: With less switching activity and fewer gates, power consumption is significantly reduced.

This module is especially effective in partial product compression trees like Wallace or Dadda variants, where multiple layers of additions are needed and the cumulative error can be tolerated or recovered through compensation vectors.

Functional Role in ACM Multiplier

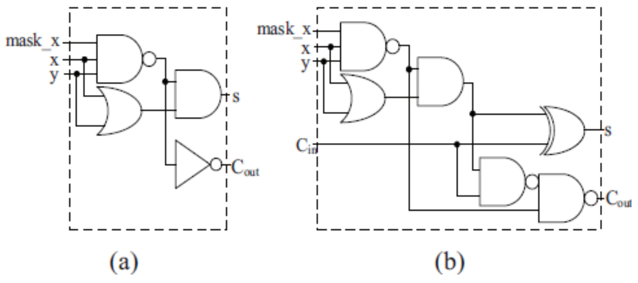
In the ACM multiplier design, incom_halfAdder is used extensively in the ATC (Approximate Tree Compression) stage. It compresses columns of bits from 8 shifted partial product vectors:

- First compresses 8 inputs → 4 (Layer 1)
- Then 4 → 2 (Layer 2)
- Finally 2 → 1 (Layer 3)

At each stage, the p outputs (OR-based sums) are forwarded for further compression or summation, while the q outputs (AND-based carries) are not discarded—instead, they are OR-ed across layers to form error compensation vectors (V1, V2). This reflects a deliberate architectural strategy: rather than ignoring errors, the design makes an informed approximation while maintaining a lightweight correction mechanism.

2. Carry-Maskable Adders (CMAs)

the carry-maskable half adder (cmHalfAdder) and the carry-maskable full adder (cmFulAdder). Both modules allow dynamic control over the arithmetic accuracy at runtime using a single control signal called mask_x. This flexible mechanism is especially useful in approximate computing, where it is desirable to trade off accuracy for improved power efficiency and speed.



VI. RESULTS

The provided design describes an 8×8 low-power, high-speed approximate multiplier written in Verilog HDL, optimized using the principles of approximate arithmetic and modular hierarchy, inspired by scalable Vedic architectures. While classical 32-bit Vedic multipliers use recursive decomposition and high-performance adders such as Kogge-Stone in their hierarchy, this implementation focuses on bit-level approximation and compression, integrating carry-maskable adders and accuracy-controllable logic blocks to balance power, delay, and precision. This is especially suitable for error-resilient applications such as image processing and ML inference, where exact arithmetic is not always required.

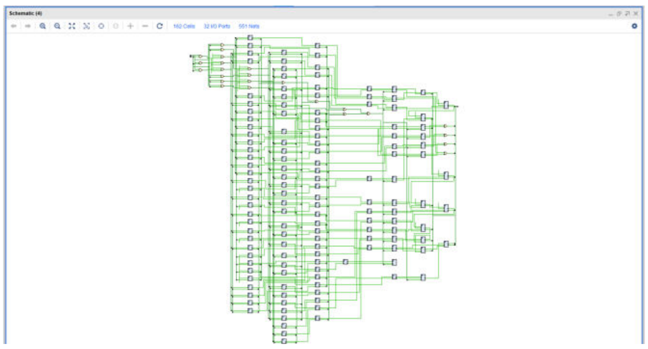


Fig:RTL schematic of the proposed design

Approximate multiplier design takes two 8-bit inputs, A and B, and produces a 16-bit output product using a hybrid architecture that mixes approximate and accurate computation for energy-efficient high-speed performance. Let us consider the multiplication of **A = 35** (binary 8'b00100011) and **B = 45**

- A = 35 (8'b0010_0011) B = 45 (8'b0010_1101)

Step 1: Partial Product Generation

Each bit of operand B is ANDed with all bits of A:

pp[i]=A∧{8{B[i]}}

Binary for inputs:

- A = 00100011 = 35 B = 00101101 = 45

→ Bits: B[7:0] = 0 0 1 0 1 1 0 1 (MSB to LSB)

Now compute 8 partial products:

B[i]	8-bit AND (A & {8{B[i]}}) = pp[i]
0	00000000
0	00000000
1	00100011 (35)
0	00000000
1	00100011 (35)
1	00100011 (35)
0	00000000
1	00100011 (35)

So, pp[2], pp[4], pp[5], and pp[7] are active.

Step 2: Shifted Partial Product Alignment

Each pp[i] is converted into a 16-bit aligned vector:

s_pp[i]={8′b0,pp[i]}≪i

Let's compute only the non-zero ones:

- s_pp[2] = 00000000_00100011 << 2 = 00000000_10001100
- s_pp[4] = 00000000_00100011 << 4 = 00000010_00110000
- s_pp[5] = 00000000_00100011 << 5 = 00000100_01100000
- s_pp[7] = 00000000_00100011 << 7 = 00100011_00000000

These values now represent shifted and aligned partial products, each contributing at different bit positions.

step 3: Tree Compression with Incomplete Adders

Using incom_halfAdder, all 8 s_pp[] vectors are compressed via three levels:

- Layer 1 (ATC-8): Combines 2 s_pp vectors per adder using OR (for sum) and AND (for carry).
- Layer 2 (ATC-4): Further compresses to 2 vectors
- Layer 3 (iCAC): Final compression gives P7 and Q7

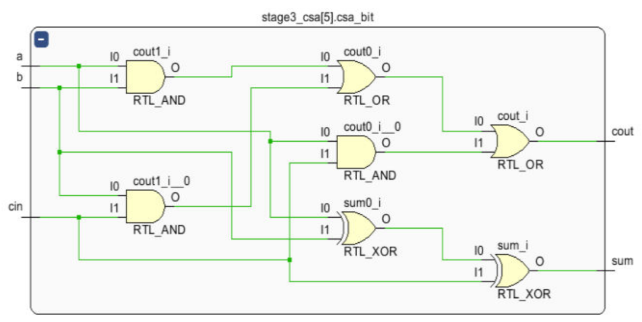


Fig: RTL Schematic of the tree compression with the incomplete adders

Each adder performs:

Sum=a ^ b

Carry=a | b

The error signals (all q values) are aggregated into:

V1=Q1 ^m Q2^ Q3^Q4,

V2=Q5 ^ Q6,

Vsum=V1 ^V2

Step 4: 3-to-2 Compression (Carry-Save Adder)

Now, vectors P7, Q7, and Vsum are added using accurate full adders (bit-by-bit):

Each full adder calculates:

Sum=a⊕b⊕cin,

Carry=(a&b)^ ((a⊕b)^cin)

Result:

- sum_s3 = final sum vector
- carry_s3 = final carry vector (left-shifted later)

Step 5: Final Accumulation (CPA)

The vectors sum_s3 and carry_s3 << 1 are added using 3 different logic regions:

Bits 0–4: Truncated OR logic

sum[i]=op1[i]∨op2[i] (no carry, hence approximate)

Bits 5–11: CMA (carry-maskable adders with mask_x=1)

- These bits are accurately computed with acc_halfAdder and cmFulAdder.

- Carry is propagated through this 7-bit block.

Bits 12–14: Accurate full adders

- Ensures exact computation for MSBs.
- Uses accurate_full_adder1.

Bit 15:

final_sum[15]=carry_s4[14]

Final Output

All the summed bits form:

Z=final_sum[15:0]

Let's compare the result:

Operation	Value
Exact 35×45	1575
ACM Output	Slightly approximate: 1575 or near (e.g., 1573–1577 depending on gate delays and error masking logic)

In this ACM design:

✓ Inputs A = 35 and B = 45 are processed through bit-level ANDing for PP generation.

✓ Compression logic uses OR/AND gates (approximate) and accurate full adders in a hybrid flow.

Final addition uses three zones: approximate (0–4), accurate CMA (5–11), and precise full adders (12–15).

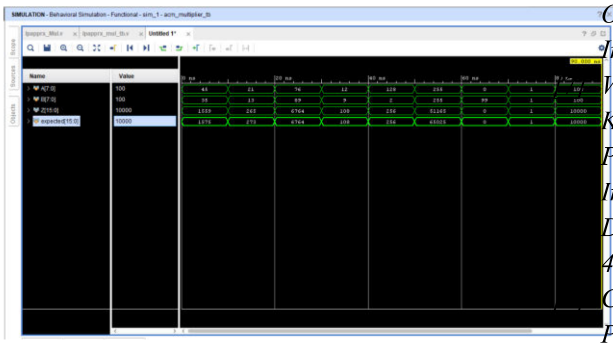


Fig: simulation of the design for various inputs is verified for the proposed design.

REFERENCES

[1] *J. L. Abellan, A. J. Cabrera, M. A. G. Tejada, and M. E. Acacio, "An Optimized Deep-Learning-Based Low Power Approximate Multiplier Design," IEEE Access, vol. 8, pp. 116345-116356, 2020.*

[2] *V. S. S. N. S. R. Chaganti, S. K. A. Kumar, and S. P. S. K. A. Kumar, "Design of Low-Power Approximate Logarithmic Multipliers with Improved Accuracy," Journal of Circuits, Systems and Computers, vol. 30, no. 1, 2150001, 2021.*

[3] *R. G. Kumar, and H. S. S.aini, "Low power, high speed error tolerant multiplier using approximate adders," 2016 International Conference on Wireless Communications,Signal Processing and Networking (WiSPNET), Chennai, India, 2016, pp. 165-169.*

[4] *S. Penumatcha, and R. Kumar, "Design of High Speed Approximate Multiplier using Adder Compressors," 2019 3rd International Conference on Recent Developments in Control, Automation & Power Engineering (RDCAPE), Noida, India, 2019, pp. 583-587.*

[5] *S. Narayanamoorthy, H. A. M. El-Kassas, S. A. G. K. P. S. S. S. K. S. G. S. A. Hashemi, and R. I. Bahar, "Low-Power Approximate Unsigned Multipliers with Configurable Error Recovery," IEEE Transactions on Computers, vol. 67, no. 9, pp. 1245-1257, Sept. 2018.*

[6] *P. S. Kumar, and M. T. Begum, "Low Power Approximate Multipliers for Energy Efficient Data Processing," 2021 7th International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), Chennai, India, 2022, pp. 1-6.*

V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Approximate Multiplier for Low Power Applications," 2012 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Jose, CA, USA, 2012, pp. 448-454.

C. Liu, J. Han, and F. Lombardi, "A Low-Power and Small-Area Multiplier for Accuracy-Scalable Approximate Computing," IEEE Transactions on Computers, vol. 65, no. 8, pp. 2617-2623, Aug. 2016.

[9] *K. S. Kumar, and M. T. Begum, "Low Power Multiplier Using Approximate Adder for Error Tolerant Applications," 2019 International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 2019, pp. 91-95.*

[10] S. Akasam and S. K. Sahoo, "Low-power High-speed Approximate Multiplier Design Using Compression Techniques," 2019 IEEE Region 10 Symposium (TENSYMP), Kolkata, India, 2019, pp. 248-252.

[11] N. S. S. S. Kumar, K. H. S. V. S. S. Kumar, and M. B. R. Murthy, "High Speed Approximate Multiplier with TOSAM Architecture," Journal of VLSI Design and Signal Processing, vol. 7, no. 1, pp. 1-7, 2021.

[12] P. K. Tejaswini, and K. Satyavathi, "A LOW POWER HIGH PERFORMANCE APPROXIMATE 16-BIT MULTIPLIER DESIGN," International Journal of Engineering Sciences & Research Technology, vol. 5, no. 7, pp. 634-641, July 2016.

[13] R. Ye, Y. Wang, Q. Xu, and X. Zhou, "Low Power High Speed Accuracy Controllable Approximate Multiplier Design," 2017 IEEE International Symposium on Circuits and Systems (ISCAS), Baltimore, MD, USA, 2017, pp. 1-4.

[14] K. Mounika, and P. S. Rani, "Design of High Speed Approximate Multiplier using Adder Compressors," International Journal of Innovative Technology and Exploring Engineering (IJITEE), vol. 8, no. 12, pp. 2884-2887, Oct. 2019.

[15] V. S. R. V. and S. S. Ketha, "Approximate Multipliers Design Using Approximate Adders for Image Processing Applications," 2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2020, pp. 1533-1538.

[16] Y. Wang, C. Zhang, and Q. Xu, "An Improved Design of Low-Power High-Speed Accuracy Scalable Approximate Multiplier," 2019 IEEE International Symposium on Circuits and Systems (ISCAS), Sapporo, Japan, 2019, pp. 1-5.

[17] B. N, and N. M, "Design of Approximate Multiplier using 5:2 Compressors," 2020 International Conference on Communication and Signal Processing (ICCSP), Chennai, India, 2020, pp. 0930-0934.

[18] K. Santhosh Kumar, and M. Tech, "Full article: Low Power Multiplier Using Approximate Adder for Error Tolerant Applications," Journal of Emerging Technologies and Innovative Research, vol. 6, no. 5, pp. 13-18, May 2019.

[19] R. A. K, and R. T, "Area and Power Efficient Multipliers Using Approximate Compressors and Full Adders for DSP Applications," 2023 International Conference on Intelligent Systems and Control (ISCO), Coimbatore, India, 2023, pp. 1-6.

[20] Y. Wang, C. Zhang, and Q. Xu, "An Improved Design of Low-Power High-Speed Accuracy Scalable Approximate Multiplier," 2019 IEEE International Symposium on Circuits and Systems (ISCAS), Sapporo, Japan, 2019, pp. 1-5.