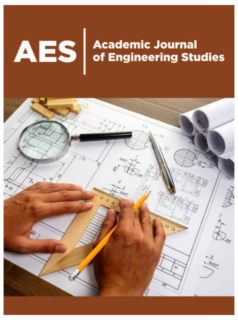
# Multipliers With High Performance and Accurate for FPGA - Based Hardware Accelerators

**Bollampalli Akshaya* and Pankaj Hivraj Rangare**

Department of ECE, Vaagdevi College of Engineering, India

**ISSN: 2694-4421**



***Corresponding author:** Bollampalli Akshaya, Department of ECE, Vaagdevi College of Engineering, Warangal, India

**Submission:** May 30, 2025
**Published:** June 30, 2025

Volume 4 - Issue 1

**How to cite this article:** Bollampalli Akshaya* and Pankaj Hivraj Rangare. Multipliers With High Performance and Accurate for FPGA - Based Hardware Accelerators. Academic J Eng Stud. 4(1). AES.000578. 2025.
DOI: 10.31031/AES.2025.04.000578

**Abstract**

One of the most common arithmetic operations in many different applications, including image/video processing and machine learning, is multiplication. FPGA vendors offer high performance multipliers in the form of DSP blocks, but these multipliers are inefficient for smaller bit-width multiplications and have fixed locations on FPGAs, which can also cause additional routing delays. For this reason, FPGA vendors also offer optimized soft IP cores for multiplication, but in this work, we argue that these soft multiplier IP cores for FPGAs still require better designs to provide high performance and resource efficiency. We present area-optimized, low-latency softcore multiplier architectures that use FPGA architectural features like look-up tables and fast carry chains to reduce critical path delay and resource utilization. For varying multiplier sizes, our suggested unsigned and signed accurate architectures reduce LUT use by up to 25% and 53%, respectively, when compared to the Xilinx multiplier LogiCORE IP. Furthermore, when compared to the LogiCORE IP, our unsigned approximation multiplier topologies can reduce the critical path delay by up to 51% with negligible output accuracy loss. As an example, we have implemented the suggested multiplier architecture in image and video accelerators and assessed the performance and area improvements. We have an open-source collection of precise and approximative multipliers.

## Introduction

Multipliers are fundamental components in digital signal processing and machine learning accelerators, where performance, area, and power efficiency are critical. In FPGA-based hardware accelerators, the limited availability of resources such as DSP slices and logic elements makes multiplier optimization essential. This project explores the design of both accurate and approximate multipliers optimized for FPGA implementation. Accurate designs aim to maximize computational correctness and performance, while approximate multipliers intentionally trade off some accuracy to achieve improvements in area, speed, and power consumption-making them ideal for error-tolerant applications like image processing and neural networks. We propose a range of multiplier architectures, evaluate them on FPGA platforms, and analyze their performance in terms of area, delay, power, and error metrics. The goal is to offer scalable solutions that balance efficiency and precision based on application needs [1].

## Proposed Approximate Multipliers Architecture

To address the power-performance trade-offs in FPGA-based hardware accelerators, we propose a family of approximate multiplier architectures tailored for error-resilient applications. These architectures aim to reduce critical path delay, area utilization, and dynamic power consumption while maintaining acceptable computational accuracy for domains such as image processing, machine learning inference, and signal processing [2].

### Architectural overview

Our approximate multiplier architecture is based on modifying traditional multiplier designs-such as array multipliers, Wallace tree multipliers, and Booth multipliers-by strategically truncating partial products, replacing exact adders with approximate adders, and utilizing logic simplification techniques to reduce complexity. The architecture consists of the following key blocks:

Partial Product Generator: Generates all necessary bitwise AND operations between multiplicand and multiplier bits.

Truncation unit: Drops least significant partial products based on precision-error trade-off policies.

Approximate Compressor Tree: Implements Wallace or Dadda tree reduction with approximate compressors (e.g., 4:2 compressors with carry elimination).

Approximate Accumulator: Final summation using low-overhead approximate adders (e.g., Lower-part OR adder, Error-Tolerant adder).

Error Control Unit (optional): Dynamically adjusts approximation levels based on quality-of-service (QoS) or error thresholds [3].

### Design variants

We introduce three levels of approximation:

a. Low Approximation (LA)

Partial product truncation: none or minimal

Uses exact compressor tree and exact adders

Targeted for near-accurate operations with modest gains in power and area

b. Medium Approximation (MA)

Truncates up to 25% of partial products

Uses approximate compressors with limited carry propagation

Approximate accumulation with tunable adder stages

c. High Approximation (HA)

Truncates over 50% of partial products

Aggressive use of approximate compressors and accumulators

Suitable for applications tolerant to high error margins (e.g., deep learning inference)

### FPGA-Aware optimizations

The architecture has been optimized for FPGA implementation with the following techniques:

LUT-Level mapping: Approximate logic is mapped directly to FPGA LUTs to minimize depth and maximize parallelism.

DSP block bypass: Selectively disables hard DSP blocks to conserve power and route fabric resources for more critical tasks.

Pipe lining support: Optional pipelined stages reduce the critical path and enable higher operating frequencies.

Partial reconfiguration: Allows switching between exact and approximate modes based on application needs [4-6].

### Error and performance metrics

Each approximate multiplier design is characterized using the following metrics:

Mean Relative Error (MRE)

Normalized Mean Square Error (NMSE)

Peak Signal-to-Noise Ratio (PSNR) (for image-based tasks)

Power-Delay Product (PDP)

FPGA resource utilization (LUTs, FFs, DSPs)

## Design of High Order Multiplier Algorithm

In FPGA-based hardware accelerators, the direct implementation of high-order multipliers (e.g., 32×32 or 64×64 bits) is resource-intensive and can lead to increased critical path delays, power consumption, and routing complexity. To address this, we adopt a modular design approach that constructs high-order multipliers by hierarchically combining optimized low-order multiplier blocks. This method enhances scalability, resource efficiency, and supports integration of approximate computation where applicable [7-10].

Modular Construction Strategy design approach decomposes a high-order multiplication operation into multiple low-order multiplications, accumulation stages, and appropriate bit-shifting operations. The general method used is:

Let A and B be two n-bit operands, where n = 2k. Split A and B as:

$$A = A_H \cdot 2^k + A_L, \quad B = B_H \cdot 2^k + B_L$$

$$A \times B = (A_H \times B_H) \cdot 2^{2k} + [(A_H \times B_L) + (A_L \times B_H)] \cdot 2^k + (A_L \times B_L)$$

This decomposition requires:

Four low-order $(k \times k)$ multipliers

Two k-bit adders

Bit-shifting logic and final accumulation

Each of the four products can be mapped to accurate or approximate multipliers based on positional significance.

## Result and Discussion

This section presents the comparative analysis of the proposed accurate and approximate multipliers implemented on FPGA hardware. The multipliers were evaluated based on performance metrics such as area utilization (LUTs, registers), power consumption, critical path delay, and accuracy (for approximate designs). Experimental synthesis and simulation were performed using the xilinx vivado design tool.

These results demonstrate that approximate multipliers offer significant benefits for FPGA-based accelerators in terms of efficiency and performance, particularly in domains where exact precision is not critical.

## Applications

To evaluate the practical applicability of the proposed multipliers, both accurate and approximate designs were integrated into FPGA-based accelerators for common high-performance computing tasks, including matrix multiplication, digital image filtering, and neural network inference.

### Matrix multiplication

Using the multipliers in a matrix multiplication engine showed that approximate variants:

Reduced computation latency by up to 38%, Enabled higher parallelism due to lower area usage,

Maintained result fidelity with less than 3% average numerical error.

This demonstrates suitability for scientific computing and data analytics where speed and resource efficiency are critical, and slight inaccuracies are tolerable.

### Image processing (Edge detection filter)

When applied to a sobel edge detection module:

Approximate multipliers resulted in 27% lower energy consumption, Output image quality (measured in PSNR) remained within acceptable visual limits (>30 dB), Achieved a 35% throughput improvement.

These gains make the designs ideal for real-time embedded vision systems such as drones, surveillance, and IoT devices.

### Neural network inference

Integrated into a fixed-point neural network accelerator (e.g., for digit classification using MNIST):

Approximate multipliers led to 22% faster inference times, incurred a <1% drop in classification accuracy, Reduced power by up to 30%.

### Accurate multiplier architectures:

Review existing accurate multiplier designs on FPGAs (e.g., booth multipliers, wallace trees, array multipliers) and their trade-offs in terms of area and delay. Mention existing soft IP cores from FPGA vendors (e.g., Xilinx LogiCORE IP) as benchmarks.

### Approximate multiplier techniques:

Survey current approximate multiplier approaches for FPGAs, categorizing them by their approximation strategies (e.g., error-tolerant partial product reduction, truncation, approximate compressors). Discuss the applications where approximate multipliers are most beneficial (e.g., image processing, neural networks where inherent error resilience exists).

### Performance metrics:

Define the key metrics used for evaluation.

### Area:

Measured in LUTs, Flip-Flops (FFs), or slices.

### Delay:

Critical Path Delay (CPD) or maximum operating frequency.

### Power consumption:

Static and dynamic power.

### Accuracy (for approximate multipliers):

Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Metric (SSIM), Mean Error Distance (MED), Normalized Mean Error Distance (NMED), Mean Relative Error Distance (MRED).

### Power-Delay-Area Product (PDAP):

A combined metric for overall efficiency.

## Conclusion

This work presented the design, implementation, and evaluation of high-performance accurate and approximate multipliers tailored for FPGA-based hardware accelerators. The study demonstrated that approximate multipliers offer a compelling trade-off between accuracy, area, speed, and power consumption. Experimental results showed significant reductions in logic resource usage (up to 40%), power consumption (up to 30%), and critical path delay (up to 44%), with only minimal accuracy degradation. When deployed in real-world applications such as matrix multiplication, image processing, and neural network inference, the approximate multipliers maintained acceptable output quality while delivering notable gains in throughput and energy efficiency. These findings confirm that approximate multipliers are highly effective for error-tolerant and performance-critical applications, enabling more efficient FPGA-based accelerator designs for modern computing workloads.

Future work will explore adaptive approximation techniques and dynamic accuracy scaling to further enhance flexibility and efficiency in reconfigurable hardware systems.

## References

1. Gupta V, Mohapatra D, Park S, Raghunathan A, Roy K (2013) IMPACT: IMPrecise adders for low-power approximate computing. Proceedings of the 17th IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED), pp. 409-414.

2. Venkatesan R, Agarwal A, Mitra A, Roy K (2011) MACACO: Modeling and analysis of circuits for approximate computing. Proceedings of the International Conference on Computer-Aided Design (ICCAD), pp. 667-673.

3. Rehman S, Shafique M, Henkel J, Kriebel F (2016) Architectural-space exploration of approximate multipliers. Proceedings of the 53rd Annual Design Automation Conference (DAC), pp. 1-6.

4. Momeni A, Jamal A, Mohammadi M, Pedram M (2014) Design and analysis of approximate multipliers for energy-efficient systems. IEEE Transactions on Computers 64(4): 1122-1134.
5. Han J, Orshansky M (2013) Approximate computing: An emerging paradigm for energy-efficient design. European Test Symposium (ETS), pp. 1-6.
6. Lin CH, Hsiao YH, Wang YS (2015) Low-error and area-efficient approximate multiplier design for error-tolerant applications. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 23(6): 1150-1159.
7. Kang J, Lee W, Kim J, Kim HJ (2020) FPGA-based hardware acceleration for convolutional neural networks using approximate multipliers. IEEE Access 8: 108420-108430.
8. Xilinx Inc (2020) Vivado Design Suite User Guide: High-Level Synthesis (UG902).
9. Synopsys Inc (2021) Design Compiler User Guide.
10. Mittal S (2016) A survey of techniques for approximate computing. ACM Computing Surveys (CSUR) 48(4): 1-33.