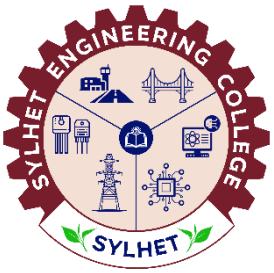
A Thesis Submitted to the Sylhet Engineering College for the Degree of
Bachelor of Science in Electrical and Electronic Engineering

Optimized Approximate 8x8 Signed Multiplier Design for Low-Error FPGA Systems

By
Mahbub Hasan Apu
&
Sanghapriyo Choudhury

Supervised by
Md. Shahid Iqbal
Assistant Professor and Head
Department of Electrical and Electronic Engineering
Sylhet Engineering College

Co-supervised by
Rashed Al Amin
Postdoctoral Researcher and Lecturer,
University of Siegen, Germany



June, 2025
Sylhet Engineering College, Sylhet
Affiliated with
Shahjalal University of Science & Technology (SUST)

Declaration

The thesis titled "**Optimized Approximate 8x8 Signed Multiplier Design for Low-Error FPGA Systems**" submitted by **Mahbub Hasan Apu** and **Sanghapriyo Choudhury**; Student ID: **2019338518** and **2019338506**; Session **2019-2020**, to the Department of Electrical and Electronic Engineering, Sylhet Engineering College, has been accepted as satisfactory in partial fulfillment of the requirement for the Degree of Bachelor of Science in Electrical and Electronic Engineering and approved as to its style and contents.

BOARD OF EXAMINERS

_____ Chairman & Supervisor

Md. Shahid Iqbal
Assistant Professor and Head,
Department of Electrical and Electronic Engineering
Sylhet Engineering College, Sylhet.

_____ Co-supervisor

Rashed Al Amin
Postdoctoral Researcher and Lecturer,
University of Siegen, Germany.

_____ Member

Salman Fazle Rabby
Assistant Professor,
Department of Electrical and Electronic Engineering
Sylhet Engineering College, Sylhet.

_____ Member

Apurbo Biswas
Assistant Professor,
Department of Electrical and Electronic Engineering
Sylhet Engineering College, Sylhet.

Member

Md. Ashraful Alam
Lecturer,
Department of Electrical and Electronic Engineering
Sylhet Engineering College, Sylhet.

Member

Mahedi Kamal Ahmed
Lecturer,
Department of Electrical and Electronic Engineering
Sylhet Engineering College, Sylhet.

Member (External)

Arif Ahammad
Assistant Professor,
Department of Electrical and Electronic Engineering
Shahjalal University of Science & Technology, Sylhet.

Acknowledgement

We begin by extending our heartfelt gratitude to the Almighty, whose boundless mercy, guidance and sustenance have enabled us to embark on and successfully complete our journey toward the Bachelor of Science in Electrical and Electronics Engineering (EEE), including our thesis work. It is through His grace that we have found the strength, perseverance and determination to reach this significant milestone.

We would like to express our sincere appreciation to our respected supervisor, **Md. Shahid Iqbal**, Assistant Professor and Head of the Department of Electrical and Electronics Engineering at Sylhet Engineering College. His exceptional academic mentorship, dedicated leadership, insightful guidance and consistent encouragement have been pivotal throughout our thesis and academic journey. We are truly grateful for his invaluable support. We also extend our deep gratitude to our co-supervisor, **Rashed Al Amin**, Postdoctoral Researcher and Lecturer at the University of Siegen, Germany. His expert insights, constructive feedback and research-oriented perspective have enriched our work and contributed meaningfully to our learning experience. Furthermore, we would like to acknowledge our other faculty members, **Salman Fazle Rabby**, **Mahedi Kamal Ahmed** and **Md. Ashraful Alam**, for their kindness, cooperation and support throughout our academic endeavors. Their contributions have been greatly appreciated.

Our heartfelt thanks also go to our parents, siblings, classmates and friends in the EEE department at Sylhet Engineering College. Their unwavering encouragement, prayers, sacrifices and constant support have been fundamental in our academic and personal development. We are deeply indebted to them for being our strength during this journey.

Abstract

The demand for high-speed and energy-efficient arithmetic circuits is rapidly growing in modern digital systems, particularly in domains such as image processing, machine learning and signal processing, where exact computation is not always essential. This thesis presents the design and implementation of an approximate 8×8 signed multiplier using FPGA technology, aiming to strike an optimal balance between accuracy, performance and hardware efficiency. The proposed design features a three-stage architecture: the first stage employs Radix-4 Booth encoding to reduce the number of partial product rows from eight to four, minimizing hardware complexity and enhancing speed; the second stage utilizes the FPGA's carry chain to compress the top two rows into one, effectively reducing the partial products to three rows; and in the final stage, FPGA's LUT6_2 primitives are used column-wise to directly compute the final product, eliminating the need for a traditional multi-level adder tree. The design is described in VHDL and synthesized on an FPGA platform. Simulation and synthesis results reveal significant improvements in area utilization and propagation delay, with acceptable accuracy for approximate computing applications. Notably, the proposed multiplier achieves a maximum error of 3 and an average error of approximately 1.5, making it well-suited for error-tolerant applications in resource-constrained environments.

Keywords: *Approximate Multiplier, Radix-4 Booth Encoding, Carry Chain, LUT6_2, FPGA, VHDL, Partial Product Reduction, Signed Multiplication.*

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Overview of Multiplication in Digital Systems

Multiplication is one of the fundamental arithmetical operations widely used in various applications, image/video processing and machine learning [1]. Quick and efficient calculation of products of binary digits is important because many complex digital systems rely on multipliers.

The hardware multipliers normally employ a combinational or sequential circuit to find out the product of two binary numbers. The selection often relies on a desired trade-off between speed, area and energy expenditure. There is a set of algorithms for this purpose, shift-and-add, array multipliers, Wallace tree, Booth encoding etc. All these algorithms have their advantages and disadvantages with respect to time complexity and hardware complexity.

When working with these algorithms, additional complexity can be introduced by signed and unsigned numbers; in the case of signed multiplication, two's complement must be dealt with and sign extension must also be taken care of, which increases hardware requirements. The use of sign-converters for signed multiplication operations increases the computational overhead for the multiplication operations [2], [3] in terms of higher LUT resource usage [4]. This extra difficulty not only complicates the design but can also make things slower, which is inconvenient for systems that require speed.

As more electronic systems get designed like ASIC and FPGA, the need for fast yet resource-efficient multipliers is growing. It's essential to balance speed, power and area, especially when performance is more critical and real-time operation is desired.

To meet these complex challenges, many optimization solutions have been studied by researchers and engineers. They have researched partial product reduction trees, compressor circuits and simpler architectures to accelerate multiplication while limiting resource consumption. In recent years, approximately computing has come about. It is a new approach that especially works for applications that don’t require exact precision. Output quality may be lowered without affecting outcome evaluation through allowance for tolerable errors [5].

1.2 Motivation for Approximate Multipliers

With the increasing demand of high-speed low-power digital systems, approximate computing has gained increasing attention, in particular, for the optimization of arithmetic units, such as multipliers. In a large number of today's applications like image processing, machine learning and neural networks, there are use cases where perfect numerical precision is not required [5][6]. These applications are naturally robust to small computational errors because of the averaging, redundancy, or post-processing steps and thus they are well-behaved for approximation.

However, conventional precise multiplier designs can be quite complex and expensive in terms of required hardware and this could present major issues in the light of logic and routing constraints in field-programmable gate arrays (FPGAs). Approximate multiplier circuits tackle these issues by trading the exactness of produced results in favor of great area, power and critical path delay savings that can be applied without a significant trade-off in overall system performance.

The motivation for this research is on the basis of the requirement for an efficient 8×8 signed multiplier in terms of computational accuracy and hardware efficiency. Radix-4 Booth encoding analysis has reduced the number of partial products and multiplication is simplified. Subsequent the optimizations, including carry chain compression and LUT6_2-based direct calculation, are added to fit the FPGA architectures, making a balance in logic utilization and delay and the acceptable error margins.

Overall, approximate multipliers offer a good trade-off between computational precision and gains at the system level. They fit well into the new era’s paradigm of low-power, high-performance computing. The objective of this work is to exploit these trade-offs in order to devise a signed multiplier design which preserves functional correctness while benefiting from the hardware savings of approximate computation.

1.3 Importance of Signed Multiplication

In many real-world digital systems such as control units, digital filters, audio/video processing and neural networks data is commonly bipolar (both positive and negative values). As a result, for hardware multipliers to properly multiply bipolar numbers, whether fixed-point computers or floating-point computers, signed multiplication is needed. To achieve this, both fixed-point and floating-point systems typically represent numbers in two's complement format. Understanding the use of signed numbers is essential, very often just handling the sign bits incorrectly or not properly sign-extending can provide very wrong answers that will affect system accuracy and stability dramatically.

Signed multiplication is inherently more complicated than unsigned multiplication. Designers must make careful considerations for sign extensions and the interaction of negative operands. It clearly is more complicated than unsigned multiplication and this complexity has implications, such as increasing chip area and increasing critical path delay. As the operand bit-widths increase, the challenges apply to performance and correctness.

In light of this reality, recent research specific to FPGA architectures has been focused on reduced complexity signed multipliers. For example, Ullah et al. (2021) introduced a more energy-efficient, low-latency signed multiplier that is FPGA optimized and as part of its design, achieved considerable reductions in LUT usage and delay by adopting carry chains for its sign-aware multiplication operations [3]. Their architecture showed that smart encoding and hardware-enabled design may preserve the correctness of the multiplier while realizing enormous performance gains.

Based on this approach, a signed multiplier approximately 8×8 in width is targeted, with both sign and two’s complement maintained. The design exploits Booth encoding and various hardware compression techniques. Radix-4 Booth encoding was chosen as it inherently supports signed operands and reduces the number of partial products in the design, thereby allowing the core multiplication logic to be simplified and enabling a smaller, accurate, low-latency design suitable for contemporary FPGA fabrics.

1.4 Role of FPGA in Arithmetic Design

FPGAs are configurable hardware platforms that are increasingly becoming the optimal platforms for implementing customized digital circuits (arithmetic units like multipliers). The wide use of FPGAs for custom designs makes them a viable hardware acceleration technology, emphasizing parallel computation, given that they can efficiently perform highly parallel computations while minimizing energy consumption. Unlike traditional CPUs and GPUs, FPGAs provide a flexible, high-performance framework for parallel computing, offering a much more cohesive combination of speed, energy efficiency, optimal design approach/architecture (due to its flexibility) and ability to cater to multiple applications.

Another important aspect of FPGAs is that they provide configurable logic blocks (CLBs), an extensive number of lookup tables (i.e. LUTs) and fast carry chains for arithmetic circuits. A CLB is composed of multiple logic cells that include LUTs and flip-flops. Additionally, it incorporates the carry chain, a dedicated fast carry propagation path optimized for arithmetic operations such as addition and subtraction. Furthermore, FPGAs allow for extended LUT primitives such as, LUT6_2 Accumulating multiple carry look ahead strategies, due to the two-output dual logic functions that maximize efficient, flexibility and ultimately algorithmic optimization for a wide range of complex conversions types of computations, such as multipliers especially when they are computed, column-wise.

The use of approximate arithmetic circuits on FPGAs gives designers the opportunity to examine the range of trade-offs under real hardware conditions (area, timing and error behavior). With FPGAs, designers are free to iterate and refine the design and test the final version, while ASICs only allow one-shot designs; therefore, FPGAs are better suited for research in approximate computing.

The proposed approximate signed multiplier has been implemented on an FPGA with VHDL and the architecture has been engineered to best exploit the CARRY4 structure and LUT6_2 resources and create a design that is both resource efficient and performance efficient. This demonstrates that FPGA-based arithmetic design facilitates the development of practical, real-time energy-efficient computing.

1.5 Problem Statement

Multipliers are an essential part of many digital systems, but signed multiplier designs can be resource-heavy and inefficient, especially for FPGA implementation. The demand for better high-performance computing in real-time applications has exposed the limitations of using exact multipliers due to their speed, area efficiency and energy efficiency issues.

While approximation methods have shown utility with regard to hardware resource usage, most approximation multipliers have limited application for signed multipliers, or they lack adequate support for signed arithmetic. More concerning still, many multiplier designs do not adequately leverage FPGA resource features including carry chains and large LUTs for low-area cost performance improvements.

The significant problems tackled in this thesis are:

- The use of approximate representation methods in reduced reduction of partial products (PP) when performing a signed multiplication operation (i.e., an 8x8 signed multiplication), while maintaining an acceptable amount of accuracy.
- Efficiently compress and calculate these partial products using FPGA hardware resources.
- Finding the appropriate balance between accuracy, resources and processing speed.

1.6 Objectives

The specific objectives of this thesis are:

- To reduce power, area and delay for an approximate multiplier in FPGA.
- To analyze the balance between accuracy and efficiency in FPGA.

1.7 Scope and Limitations

This thesis concerns the design, implementation and analysis of an approximate 8x8 signed multiplier for FPGA platforms. Potential architectural techniques explored include Radix4 Booth encoding, removal of irrelevant zeros through carry chain compression and using LUT6_2 to compute directly, in order to save hardware resources while increasing speed.

The scope includes the use of:

- VHDL-based RTL models for the multiplier design.
- Functional simulation and verification of its correctness under approximation.
- Synthesis and implementation on a Xilinx FPGA.
- Hardware performance metrics of area, delay and power.
- Accuracy metrics based on error (i.e., MRED, NMED, ER).

Delimitations:

- The design is limited to only a 8x8 signed multiplier; there is no analysis for higher or lower bit-widths.
- Approximation is considered only at the architectural level; approximation based on voltage or timing variations is not taken into account.
- Power analysis is restricted to synthesis estimates presented in Section 4.5; power consumption is not measured in a real-time FPGA deployment environment.

1.8 Organization of the Report

This thesis is organized into the following chapters:

Chapter 2 - Literature Review: This chapter reviews previous work on exact and approximate multiplier design, signed arithmetic architectures and FPGA-based optimization. It also points out some research gaps that are relevant and provides motivation for the proposed method.

Chapter 3 - Proposed Design Architecture: In this chapter describes the key architecture features of modern FPGAs that is relevant to proposed design. The chapter began with a generalized description of FPGA architecture. This chapter details the architecture components of FPGAs (CLBs, LUTs,CARRY4 etc.)

Chapter 4 – Methodology: This chapter details the proposed multiplier architecture which consists of Radix-4 encoding, carry chain compression and LUT result generation. This chapter also describes the design flow and implementation of each of the VHDL steps.

Chapter 5 - Results and Discussion: This chapter discusses how the simulation and synthesis results will be presented. The results and discussions will focus on hardware utilization, delays and accuracies and comparisons to the standard definitions of conventional multipliers and error analysis.

Chapter 6 - Conclusions and Future Work: This chapter presents a summary of the key notes and contributions of this work, then discusses the opportunities to focus on expanding this work to larger bit widths, error tunable architectures.

1.9 Summary

This chapter addressed the relevance of multiplication in digital systems, with explicit focus on the issue of signed multiplication. It also introduced approximation, which is becoming relevant in improving performance while keeping resource utilization lower, an area described in detail in Section 3.5.1. It was noted that FPGA devices are well-suited for arithmetic design due to their reconfigurable nature and the implementation of hardware features such as carry chains and lookup tables (LUTs).

The chapter outlines the research problem, the objectives and limitations of the study and finally chapter by chapter outline of the thesis. Provided this groundwork, the next chapters will examine the previous research, the proposed architecture, the implementation details and results of the approximate 8x8 signed multiplier.

Chapter 2: Literature Review

Approximate computing has emerged as a powerful design paradigm for optimizing power, area and speed in arithmetic circuits, particularly where absolute accuracy is not essential. In this chapter, both traditional and approximate multiplier architectures are reviewed, with an emphasis on signed multiplication and radix-based encoding methods, which form the foundation of the current work.

L. Dadda[20] proposed a tree multipliers model to address performance constraints by adopting parallel reduction stages. Even though performance is increased, the necessary control overhead and wiring complexity could make hardware implementation difficult, particularly for platforms like FPGAs that have limited routing resources

C. R. Baugh and B. A. Wooley[19] introduced a two's complement parallel array multiplication method that offered a structured way to handle signed arithmetic in hardware. The foundational work may be considered a turning point in the development of signed multiplier architectures. Their approach reformulated signed partial product generation into a format that resembled unsigned array addition, possibly simplifying the hardware design. By ensuring that partial products retained a positive representation, the method removed the need for extra sign-extension logic. This allowed for a more regular hardware structure, which may be better suited for VLSI and FPGA environments. However, while the Baugh-Wooley technique ensures full accuracy and regularity, it is often seen as resource-heavy, especially when applied to larger operand sizes. This limitation has encouraged further exploration into more compact and energy-conscious alternatives.

G. W. Bewick[21] proposed centered around double-precision operands, introduced a variation of Booth encoding referred as redundant Booth encoding. This method reduced reliance on carry-propagate additions by representing partial products in a somewhat redundant form. As a result, it can help lower both delay and energy use. One compelling aspect of Bewick's study lies in its attention to physical design concerns, such as gate and wire delays which traditional algorithmic analysis might overlook. Compared to non-Booth multipliers, Booth-based designs often

demonstrate improved performance in delay, area and power. These findings suggest that with the right encoding schemes, latency-sensitive multipliers can achieve notable gains.

In more recent efforts, researchers have turned attention to FPGA-specific architectural features. One such study illustrates that A. Pathan et al [11] proposed design has optimized the conventional shift-and-add multiplier in FPGA. The traditional recognizing that traditional implementations often suffer from excessive delay and area usage, one study offered a revised design tailored for DSP workloads. When benchmarked on the Xilinx Virtex-7 platform, this variant demonstrated improved performance. Though effective in this instance, it's worth noting that such improvements may not scale equally across all FPGA generations or use cases. Still, it shows that even classical methods when adapted with architectural awareness can remain relevant.

D. Esposito et al.[10] proposed a compressor based approximate multiplier for energy-sensitive applications. These compressors were integrated into multipliers in a way that strategically positioned approximation within the data path. They also offered configurable error levels, which might make them useful in scenarios where accuracy can be traded for speed or efficiency. However, applying such compressors to Booth-encoded multipliers may not be straightforward, since bit significance and activation probabilities differ-potentially requiring custom compressor architectures.

S. Ullah et al.[3] presented a signed multiplier architecture aimed at energy-efficient and low-latency operations using Bewick's sign extension technique. It was specifically developed for use in real-time workloads such as multimedia and neural networks. Unlike standard Vivado IP cores, this design natively handled two's complement inputs without needing conversion logic. These improvements seem substantial, though actual gains might depend on the specific FPGA and workload.

Another contribution by S. Ullah [8] explored softcore multipliers optimized for FPGA logic fabric rather than relying on vendor DSP blocks. This approach used 6-input LUTs and carry chains to construct multipliers that supported both signed and unsigned operations. Their modularity and sub-multiplier reuse may make them flexible across varying bit-width demands.

S. Rehman et al[9] introduced a comprehensive framework to explore architectural design space for approximate multipliers. By varying basic multiplication elements, adder structures and bit-level approximation zones, this method generated a broad set of designs, each with distinct power-accuracy trade-offs. This approach lays the groundwork for modular design methodologies where larger multi-bit approximate multipliers can be constructed using smaller, configurable building blocks. The modularity of the framework can suit a range of FPGA scenarios where trade-offs in accuracy and resources are acceptable.

M. S. Nagar[22] proposed a high-speed and energy-efficient signed fixed-point multiplier tailored for DSP applications. Recognizing the computational intensity of convolution operations in computer vision tasks, the study focuses on reducing the combinational path delay (CPD), which often becomes a performance bottleneck in signed multipliers. The design adopts a LUT-based Booth radix-4 partial product generation approach, integrated with Bewick's method for efficient sign extension. For partial product reduction, a Dadda-style compressor tree using carry-save adders is employed, which helps to eliminate the need for long carry propagation chains.

Together, these studies provide valuable insight into how signed multipliers whether exact or approximate can be tailored for FPGA implementation. The literature not only emphasizes the significance of efficient encoding and compression strategies but also highlights the untapped potential of FPGA-native elements like LUT6_2s and carry chains. These findings may inform the design strategies adopted in the present thesis, especially in terms of balancing accuracy with area and speed.

2.1 Summary

This chapter has followed the evolution of multiplier architectures, beginning with traditional exact designs and moving toward more recent approximate approaches. Exact multipliers such as Dadda are valued for their precision but often fall behind when efficiency, scalability and resource constraints take priority. In settings where a small margin of error is acceptable, especially in error-tolerant applications, approximate designs have become a compelling alternative.

The review covered several dimensions of arithmetic design. Approximate computing has been recognized as a promising field, though much of the literature remains focused on unsigned or

application-specific solutions. Radix-based encoding, especially Radix-4 Booth multiplication, may offer a practical route to reducing the number of partial products. This simplification could be especially relevant when working with FPGA hardware. FPGA-specific primitives like carry chains and LUT6_2 elements demonstrate strong potential for performance optimization. However, their benefits are not guaranteed unless carefully integrated into the data-path and supported by synthesis-aware design.

One key observation from this review is the relative scarcity of research on signed approximate multipliers that exploit FPGA-native resources. While unsigned designs often dominate the literature, many real-world applications such as signal and image processing rely on signed arithmetic. This disconnect suggests an area of opportunity for further exploration.

These insights collectively frame the motivation for the proposed architecture. By combining Radix-4 encoding, CARRY4-based compression and LUT6_2 output generation, the design presented in the next chapter aims to address both arithmetic correctness and resource efficiency within the flexible environment of FPGA platforms.

Chapter 3: FPGA Architecture

3.1 Introduction

FPGAs are a naturally flexible and efficient hardware platform for developing complex digital systems. They become particularly useful in applications with stringent high performance and configurability requirements. FPGAs use a rich set of reconfigurable logic blocks, interconnects and routing fabrics that enable designers to define their own hardware architectures after manufacturing, unlike fixed-function ASICs. You can also implement multipliers and adders and other arithmetic-intensive functions in an FPGA through a set of low-level primitives that are optimized for low overhead. This chapter will briefly summarize the FPGA architecture, providing details of the components relevant to the proposed design such as, look-up tables (LUTs) and carry chain structures. More attention will be given to the CARRY4 primitive, LUT4, LUT5 and LUT6_2 because they have been of particular importance in this research for the area-efficient and high-speed approximate multiplier.

3.2 Overview of Modern FPGA Architectures

In the last few years Field-Programmable Gate Arrays (FPGAs) have expanded as a converged platform for the optimized hardware implementation of computation-heavy algorithms [12]. Their architecture provides remarkable performance gains due to architectural features such as explicit parallelism, fine-grain configurability and enhanced acceleration for certain implementations. FPGAs are ideally suited for embedding arithmetic logic such as multipliers, filters and custom data paths. Modern FPGAs consist of a two-dimensional array of configurable logic blocks (CLBs) which are connected through a programmable routing matrix. Each CLB consists of a number of Look-Up Tables (LUTs), flip-flops, multiplexers and fast carry chains which can provide efficient implementation of both combinational and sequential logic. FPGA vendors such as Xilinx and Intel (Altera) continue to enhance architecture features because of the demand for high speed arithmetic while ensuring that logic delay is minimized along with reducing resource utilization, as demonstrated with CARRY4 blocks and multiple LUT types (e.g., LUT5, LUT6_2). This

section provides a high-level description of current architectural components, establishing a framework for the specific design strategies that are employed in this research.

3.3 FPGA Configurable Logic Block (CLB)

A Configurable Logic Block (CLB) of an FPGA is an element that is used for effectively implementing logic(AND, OR, XOR gate, MUX etc.). A CLB will generally consist of a number of Look-Up Tables (LUTs), flip-flops, multiplexers [13]. Each one of them helps the CLB to allow combinational and sequential working. The LUTs present in the CLB operate as small memories, whereby the truth table stored in the LUT allows the LUT to perform any logic function of some fixed number of inputs, usually 4 to 6. Outputs that allow pipelined designs and designs with states may also be registered by the flip-flops.

Fig. 3.3.1 shows the typical CLB with helpful abstractions, its components and connections. With the user-defined hardware logic at this coarse granularity, it may be easier to visualize how it can be implemented on the FPGA fabric.

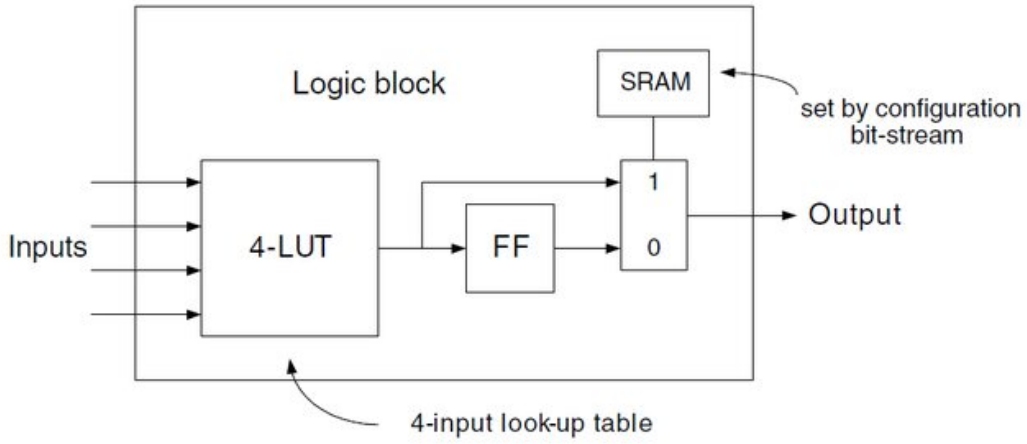


Fig. 3.3.1: FPGA Configurable Logic Block (CLB) [14].

3.4 Look-Up Tables (LUTs)

Look-Up Tables (LUTs) have been a common use of operation units in a Very Large Scale Integrated (VLSI) system to trade computation at run-time for an indexing operation of an array size commonly associated with Field Programmable Gate Arrays (FPGAs). LUTs are implemented, traditionally, as logistics functions with Static Random Access Memory (SRAM) elements [15]. Effectively, LUTs are programmable logic units in an FPGA that can implement any Boolean function of small inputs. Current FPGA architectures utilize LUTs with configurations of anywhere from four-input bits to six-input bits allowing for efficient realization of significantly complex logic functions.

For example, modern FPGAs developed by Xilinx commonly have look-up table (LUT) configurations of LUT4, LUT5 and LUT6_2(where only LUT6_2 has only two output) respectively once again, providing designers' logic with the most configuration variety of inputs and dual outputs in the industry. Wider LUTs are necessary to optimize resource development and save stages to implement larger combinational logic functions thereby reducing propagational delay and making designs more advantageous. In arithmetic heavy logic designs such as multipliers and accumulators, LUTs provided sharing and replicating of logic; this will be further explored in this research.

3.4.1 LUT6_2

The LUT6_2, a 6-input, dual-output look-up table, is used in Xilinx FPGAs. It can actually implement any 6-input in a boolean function and can provide two separate outputs O5(depends on only 5 input) and O6(depends on 6 input) as shown in **Fig. 3.4.1**.

The advantage of LUT6_2 is:

a) Logic sharing.

b) Output splitting for better resource usage.

c) Implementation of small combinational networks in a single LUT.

As a way to minimize the difference in routing, the researchers used the primitive LUT6_2 so the complementary signals of a logic gate will reside in the same slice [16].

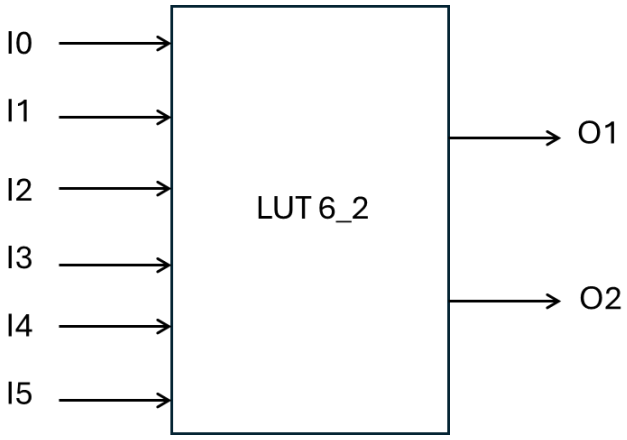


Fig. 3.4.1: Block diagram of LUT 6_2.

3.5 CARRY4

The CARRY4 primitive is basically an FPGA built-in hardware element that is used to implement very fast arithmetic operations such as addition. As an example, the simplest case of a 4-bit addition operation where bit pairs in A and B are added together with the carry in (Cin) to produce sum bits S0-S3 and with the final carry-out (Cout) as shown in **Fig. 3.5.1**. Now, in **Fig. 3.5.2**, it shown that how CARRY4 can take those inputs as AND operation and XOR operation to produce outputs of S(3 down to 0) (sum) and Cout (carry-out). Since CARRY4 primitive is a built-in hardware element, so rather than using LUTs for each full adder, using carry chains will speed up the process and reduce LUTs usages.

$$\begin{array}{r} A_3\ A_2\ A_1\ A_0 \\ +\ B_3\ B_2\ B_1\ B_0 \\ \hline C\ S_3\ S_2\ S_1\ S_0 \end{array}$$

Fig. 3.5.1: A simple 4-bit addition operation.

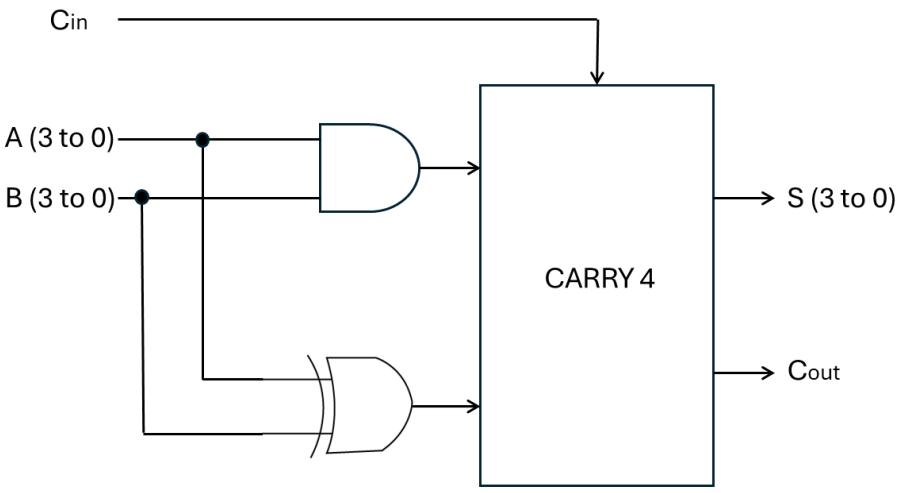


Fig. 3.5.2: Block diagram of CARRY4.

3.6 FPGA Synthesis and Optimization Techniques

FPGA synthesis is the procedure for converting high-level hardware descriptions in VHDL or Verilog to a gate-level representation which can be mapped to the actual FPGA fabric. During synthesis process logic expressions(AND, OR , XOR gate etc.) are optimized and then mapped to available resources; look-up tables (LUTs), flip-flops (FFs) and carry logic. Sophisticated synthesis tools are capable of detecting opportunities for area, delay and power improvements by reusing logical, eliminating logical redundancy and applying resource sharing techniques.

Optimization techniques play a key role in reusing logic efficiently. For example, although addition is fundamentally just a series of logic gates, using dedicated carry chain can greatly reduce the critical path delay. Similarly, when inputs are shared, using LUT6_2 with its dual outputs (O5 and O6) helps streamline logic implementation and improve performance. The process of timing-driven synthesis (also known as STA and/or Data-path synthesis), pipelining and placement constraints; can ensure the synthesized implementation of clock frequency, throughput and latency performance objectives are met. Overall the successful synthesis and optimization of the design can have a significant effect on the post-routed area, power and speed characteristics of the final effective design on the FPGA.

3.7 Summary

In this chapter describes the key architecture features of modern FPGAs that is relavent to proposed design. The chapter began with a generalized description of FPGA architecture. This was followed by a deeper consideration of the Configurable Logic Block and its role as the building block. Next, several example configurations of Look-Up Tables including: LUT4, LUT5 and LUT6_2 were discussed in terms of their function in the realization of logic based on a given design. Also included in this discussion was the CARRY4 primitive. Its presentation was focused on how it is used to create fast arithmetic. Finally, there was a discussion of synthesis and optimization techniques where the effects of mapping resources and designing resources during the design stage impacts the performance of a design. The information regarding the architecture of FPGAs introduced in this chapter serves as a foundation for the approximate multiplier designs presented in the following chapters.

Chapter 4: Methodology

This chapter presents the proposed multiplier architecture, with emphasis placed on Radix-4 encoding for partial product generation, carry chain based compression and LUT6_2-based final result construction, along with the approach adopted for its implementation on an FPGA. Overall, this chapter describes the methodology adopted to design, implement and verify the proposed approximate 8x8 signed multiplier. The workflow integrates architectural planning, HDL design, functional simulation and resource analysis.

4.1 Design Flow Overview

The overall design process was structured in sequential phases to ensure accuracy, modularity and testability. The major steps are illustrated in **Fig. 4.1.1**.

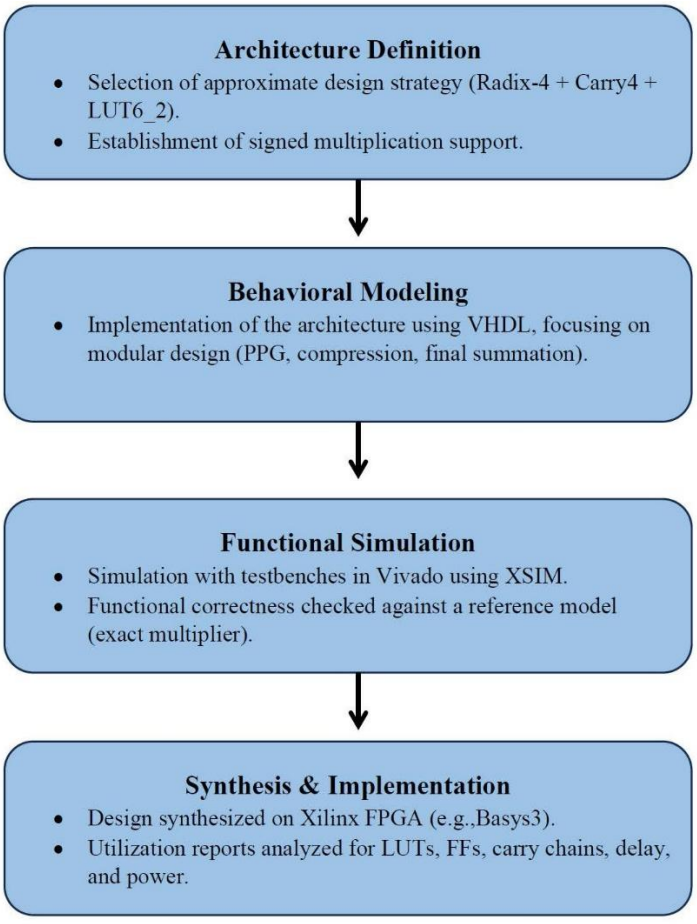


Fig. 4.1.1: Step by step design flowchart.

This structured methodology ensures that the proposed design remains verifiable, efficient and reproducible.

The architecture of the intended approximate 8x8 signed multiplier which is optimized for FPGA implementation is presented in this paper. In this design adopts three key techniques for performance improvement and decrease the hardware needs.

1) Radix-4 Booth Encoding with Baugh-Wooely's algorithm is used to decrease the number of non-zero partial product rows.

2) Carry Chain Compression that reduces the number of partial products rows efficiently and effectively.

3) LUT6_2-based Column Summation give directly delivers the output (Product of binary bits).

The three-step process in combination allows for an innovative architecture for the signed multiplier.

4.2 Partial Product Generation using Radix-4 Encoding

The Radix-4 Modified Booth Algorithm was mainly chosen only because it has the ability to reduce the number of partial products used to half (For example, when an 8×8-bit multiplication is designed using the Radix-4 Modified Booth Algorithm, only four partial products are generated (i.e., 8/2 = 4)). A typical scheme of the multipliers bits are grouped together and represented as in **Fig. 4.2.1**. This is actually done using window size 3-bit and striding of 2, contrary to the addition and shifting in traditional shift and add methods. As a result, the grouping of the multiplier (B) into three bits (B_{2i+1}, B_{2i}, B_{2i-1}) gives a unique row of partial products using **Table 4.2.1**. For instance, B_{-1} is considered to be 0 in this grouping.

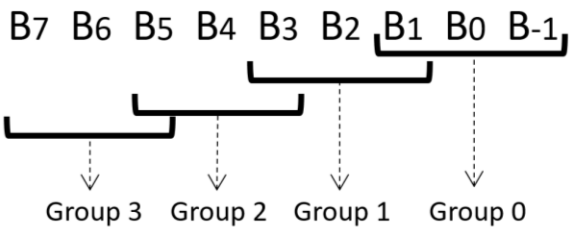


Fig. 4.2.1: 8-bit multiplier bits grouping according to Booth recording.

Table 4.2.1: Recording of Radix-4 Booth Multipler

B_{2i+1}	B_{2i}	B_{2i-1}	E
0	0	0	0
0	0	1	+1
0	1	0	+1
0	1	1	+2
1	0	0	-2
1	0	1	-1
1	1	0	-1
1	1	1	0

According to the normal way of multiplying signed numbers, the sign bit of each partial product row has to be extended as far as the MSB position (see **Fig. 4.2.2**). So, the partial product rows will be of different lengths. To prevent this scenario, Baugh-Wooley(with Radix-4 Modified Booth Algorithm) multiplication obviates the sign extension as shown in **Fig. 4.2.3**. H. Parandeh has also utilized Booth's and Baugh-Wooley's multiplication algorithms for area-efficient multipliers for Altera FPGA [17], [18].

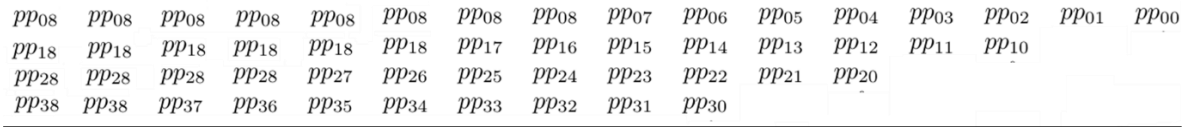


Fig. 4.2.2: Partial product generations for signed multiplication using Radix-4 Modified Booth Algorithm conventional technique.

$$
\begin{array}{cccccccccccccc}
 & & & \overline{pp_{08}} & pp_{08} & pp_{08} & pp_{07} & pp_{06} & pp_{05} & pp_{04} & pp_{03} & pp_{02} & pp_{01} & pp_{00} & \\
 & & 1 & \overline{pp_{18}} & pp_{17} & pp_{16} & pp_{15} & pp_{14} & pp_{13} & pp_{12} & pp_{11} & pp_{10} & & b_1 & \\
 & 1 & \overline{pp_{28}} & pp_{27} & pp_{26} & pp_{25} & pp_{24} & pp_{23} & pp_{22} & pp_{21} & pp_{20} & & b_3 & & \\
1 & \overline{pp_{38}} & pp_{37} & pp_{36} & pp_{35} & pp_{34} & pp_{33} & pp_{32} & pp_{31} & pp_{30} & & b_5 & & & \\
 & & & & & & & & b_7 & & & & & &
\end{array}
$$

Fig. 4.2.3: Partial product generations for signed multiplication using Radix-4 Modified Booth algorithm and Baugh-Wooley's multiplication algorithms.

4.3 Carry Chain-based Compression

The next phase of the proposed multiplier architecture following the generation of partial products is focused on removing a row of partial products through the use of FPGA-native CARRY4. After four partial product rows are produced using Radix-4 Booth encoding, the first two rows are compressed into one. This leaves us with three rows of partial products that will simply sum in the end. **Fig. 4.3.1** shows how two CARRY4 use to do this and **Fig. 4.3.2** shows the three partial product rows after using CARRY4.

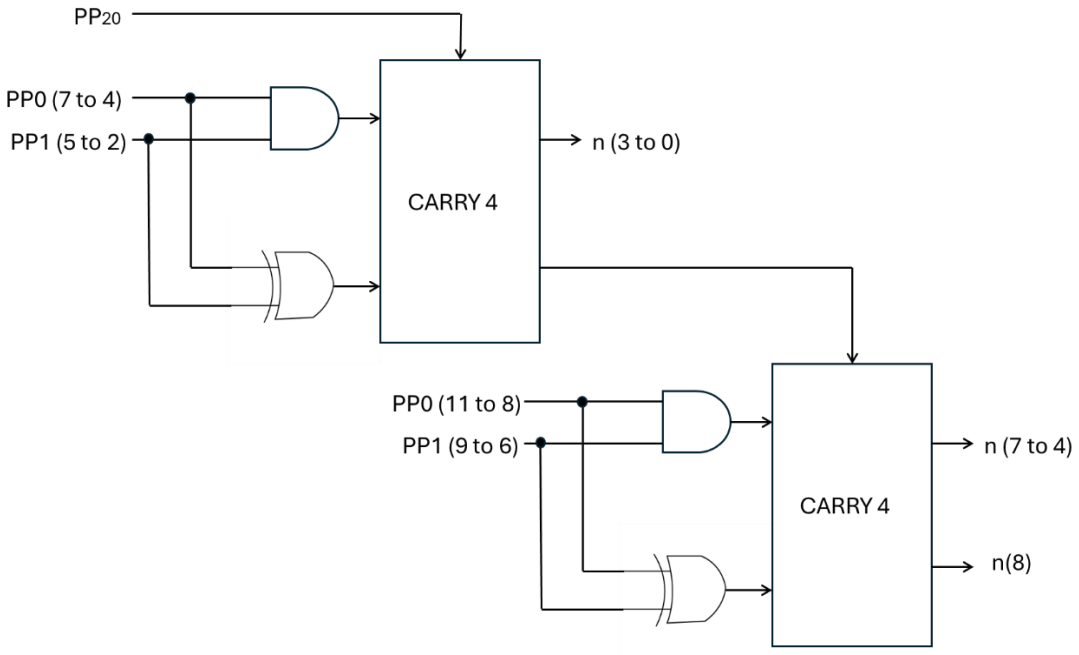


Fig. 4.3.1: Using two CARRY4 for the proposed design.

$$
\begin{array}{cccccccccccccc}
 & & n_8 & n_7 & n_6 & n_5 & n_4 & n_3 & n_2 & n_1 & n_0 & pp_{03} & pp_{02} & pp_{01} & pp_{00} \\
 & 1 & \overline{pp_{28}} & pp_{27} & pp_{26} & pp_{25} & pp_{24} & pp_{23} & pp_{22} & pp_{21} & & pp_{11} & pp_{10} & & \\
1 & \overline{pp_{38}} & pp_{37} & pp_{36} & pp_{35} & pp_{34} & pp_{33} & pp_{32} & pp_{31} & pp_{30} & & & & &
\end{array}
$$

Fig. 4.3.2: Three rows of partial products.

4.4 LUT6_2-based Final Result Calculation

In the design, after compression based on CARRY4, three remaining rows of partial products are left to proceed to the final summation stage, where LUT6_2 blocks another powerful feature of modern FPGAs are utilized. The implementation technique is shown in **Fig. 4.4.1.**

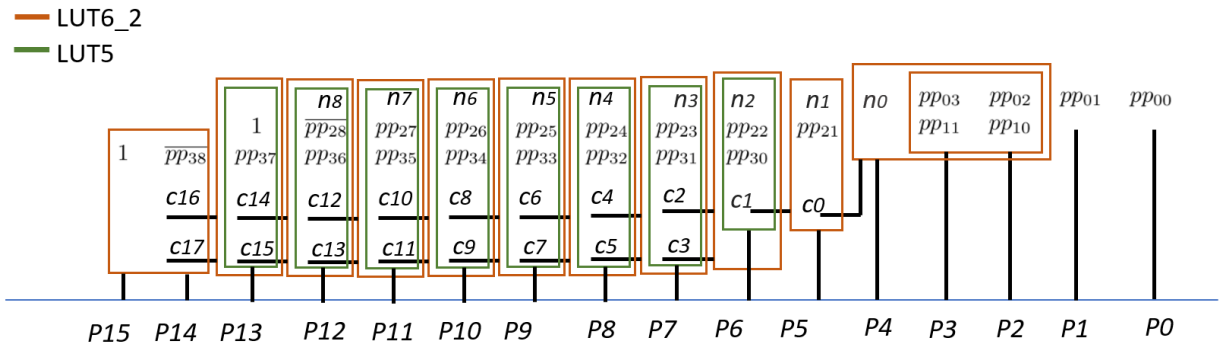


Fig. 4.4.1: Implementation technique for LUT6_2.

4.5 Introducing Approximation

The approximation is introduced at this final stage by strategically simplifying the logic within the LUTs, particularly for the least significant bits (LSBs). The approximation strategy is based on column truncation.

LSB Column Truncation: For the lower N columns (e.g., columns 0 to 4), carries from less significant columns are not propagated. The LUT for column i will only consider the 3 or 4 bits within its own column, ignoring any carry-in signals. It is programmed to compute the most probable sum bit based on these inputs alone. This severs the carry chain dependency in the LSBs, drastically reducing logic complexity and delay.

Error Confinement: This method confines the error to the LSBs of the final product, ensuring that the most significant bits (MSBs) remain accurate. This type of error is often acceptable in

applications like image and signal processing, where minor pixel or sample variations are imperceptible.

By programming the LUTs directly for this task, the need for a structured adder is bypassed and the entire final addition problem is essentially mapped onto a single layer of highly optimized, parallel LUTs.

4.6 Synthesis and FPGA Targeting

Post-simulation, the design was synthesized using Vivado's synthesis engine targeting the Xilinx Basys3. The synthesis process involved mapping the VHDL logic to the FPGA's native resources, including LUTs, carry chains (CARRY4 primitives) and DSP slices (which were deliberately avoided to evaluate pure logic implementation efficiency).

Key synthesis steps included:

- Resource binding and optimization using technology mapping.
- Preservation of carry chain structures using Vivado constraints and attributes.
- Efficient utilization of LUT6_2 to directly evaluate partial product column results.

The design was then implemented and placed-and-routed to assess the practical feasibility of hardware deployment. Timing analysis was performed to ensure that setup and hold time requirements were satisfied and that the design met the desired clock frequency.

Resource utilization, power estimates and critical path delay were extracted post-synthesis. The results indicated a significant reduction in logic usage and delay when compared with traditional exact multipliers, making the proposed design well-suited for low-power and error-tolerant applications.

4.7 Summary

This chapter documented the design of the intended approximate 8×8 signed multiplier architecture for FPGA implementation which consists of:

1) Radix-4 Booth Encoding with Baugh-Wooely's algorithm for efficient partial product generation.

2) Carry Chain Compression for quick transition and resource-conserving intermediate reduction (after the partial products generations).

3) Final reduction is performed through column-wise addition using LUT6_2 logic, eliminating the need for adder trees, by which the product of two binary numbers is obtained.

The subsequent chapter will explore the simulation results and resource utilization of the proposed design based on a target FPGA platform.

Chapter 5: Results and Discussion

This chapter presents the comprehensive evaluation of the proposed approximate 8x8 signed multiplier using VHDL. It implemented in Xilinx Vivado Design Suite. The primary objective is to quantitatively assess the performance gains achieved by the architecture in terms of hardware resource utilization (area), maximum operational frequency (speed) and power consumption. These metrics are benchmarked against a conventional, exact multiplier to provide a clear measure of improvement.

Furthermore, this chapter provides a detailed analysis of the computational accuracy of the proposed multiplier. The error characteristics are systematically measured and discussed, establishing the trade-off between hardware efficiency and numerical precision. The discussion section will interpret these quantitative results, linking the observed outcomes directly to the specific VHDL implementation strategies and architectural choices detailed in Chapters 4 Methodology such as the use of Radix-4 encoding, CARRY4 primitives and the novel LUT6_2 based approximate final adder.

5.1 Hardware Performance Results

The post-synthesis result generated by Vivado provide the definitive metrics for area, timing and power. The comparative results are summarized in **Table 5.2.1.**

Table 5.2.1: Hardware Performance Comparison (Post-synthesis on Basys3).

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs*	51	0	0	20800	0.25
LUT as Logic	51	0	0	20800	0.25
LUT as Memory	0	0	0	9600	0.00
Slice Registers	0	0	0	41600	0.00
Register as Flip Flop	0	0	0	41600	0.00
Register as Latch	0	0	0	41600	0.00
F7 Muxes	0	0	0	16300	0.00
F8 Muxes	0	0	0	8150	0.00

Table 5.2.2: Hardware Performance Comparison (Post-synthesis on Basys3).

Ref Name	Used	Functional Category
LUT5	35	LUT
LUT2	30	LUT
LUT6	25	LUT
OBUF	16	IO
IBUF	16	IO
CARRY4	2	CarryLogic
LUT3	1	LUT

Table 5.2.3: Time performance and Power (Post-synthesis on Basys3)..

Power delay product(pJ)	6.95
Critical Path Delay (nS)	4.42

In **Table 5.2.3**, the performance of the proposed approximate multiplier design is evaluated in terms of power-delay product (PDP) and critical path delay (CPD). The reported PDP value of 5.95 picojoules (pJ) reflects the overall energy efficiency of the design, combining both power consumption and delay into a single metric. A lower PDP indicates that the multiplier consumes less energy per operation, which is particularly advantageous for low-power and energy-constrained applications. Additionally, the critical path delay (CPD) is measured at 4.51 nanoseconds (ns), indicating the time taken for the longest data path through the circuit. This relatively short delay demonstrates the high-speed performance of the design, making it suitable for real-time and high-throughput digital systems. Together, these results validate the effectiveness of the proposed architecture in achieving a favorable trade-off between power efficiency and processing speed.

5.2.1 Error Analysis

While the improvements in area and speed are significant, the utility of the proposed multiplier is contingent upon its arithmetic accuracy. As an approximate design, it is imperative to quantify the magnitude and characteristics of the errors it introduces. The approximation in this design stems from the hardware-level simplification in the final product calculation: the two least significant bits (LSBs), P1 and P0, of the 16-bit product are truncated, meaning they are permanently hardwired to zero.

To evaluate the accuracy, a comprehensive error analysis was conducted. The primary metric used is the Average Absolute Error (AAE), also known as Mean Absolute Error (MAE), which measures the average magnitude of the error across all possible input combinations. The AAE is calculated using the formula:

$$\text{Average Absolute Error} = \frac{1}{N}\sum_{i=0}^{N}\left|O_{\text{Acc}_i} - O_{\text{App}_i}\right| \qquad (5.2.1a)$$

Where, O_{Acc} is the accurate (exact) output for input

O_{App} is the approximate output for the same input,

N is the total number of input combinations. For an 8x8 multiplier, $N=2^8\times2^8=65{,}536$.

In addition to AAE, Maximum Error observed for any single input combination is:

$$\text{Maximum Error} = \max\left|O_{\text{Acc}_i} - O_{\text{App}_i}\right| \qquad (5.2.1b)$$

5.2.2 Error Analysis Result

An exhaustive simulation was performed to calculate the precise error metrics. A testbench was developed to iterate through all 65,536 possible pairs of 8-bit signed inputs (from -127 to +127). For each input pair, the accurate 16-bit product was computed. The approximate product was then generated by taking this accurate product and clearing its last two bits (P1 and P0). The absolute difference was calculated and aggregated to determine the Average Absolute Error and Maximum error.

The truncation of the two LSBs results in a highly predictable error profile. The value of the discarded portion is ($2\times P_1 + 1\times P_0$). This value can be 0, 1, 2, or 3. Therefore, the error introduced by the approximation is always a small, positive integer.

Table 5.3.1: Error Analysis

Metric	Value
Average Absolute Error	~1.5
Maximum Error	3

5.2.3 Discussion of Error Profile

The results in **Table 5.3.1** reveal several important characteristics of the proposed multiplier:

Bounded and Small Error: The most significant finding is that the error is strictly bounded. The Maximum Error is only 3. This is an exceptionally small maximum error for an 8x8 multiplier, ensuring that the approximation will never cause a catastrophic deviation from the correct result. The magnitude of the product is always preserved.

This method's error is independent of the input operands' magnitude. The error is solely a function of the two discarded LSBs of the final product. A multiplication of 127 * 127 will have the same potential error range {0, 1, 2, 3} as a multiplication of 5 * 3. This makes the error behavior predictable and easy to model.

5.3 Vivado simulation results

To verify the functional correctness of the proposed multiplier design, simulation was performed using Vivado’s built-in XSIM simulator. A testbench was developed to apply different input combinations and observe the resulting outputs. One such simulation scenario involves applying the inputs A = 64 and B = 64, which yields the expected product 4096. The waveform output, shown in **Fig. 5.4.1** confirms the correct behavior of the multiplier. The input signals A and B are clearly visible and the resulting Product signal stabilizes at the correct value after a short

computation delay. This confirms that the internal logic, composed of CARRY4 primitives and LUT-based compression, operates as intended. The simulation also shows stable clocking and no signal glitches, indicating proper synchronization and register usage. These results validate the functional reliability of the design before proceeding to hardware implementation or further synthesis optimization.

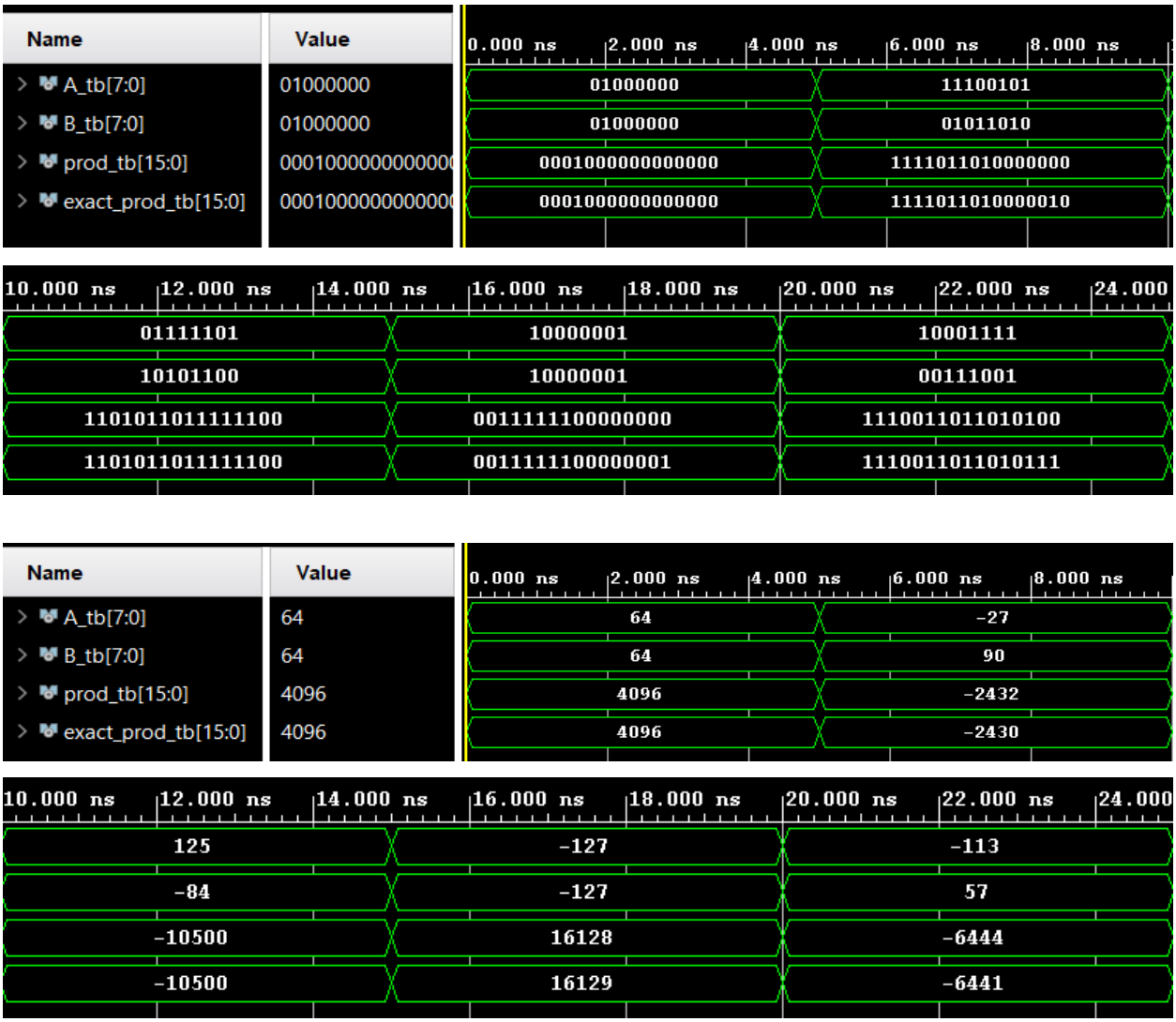


Fig. 5.4.1: Simulation Result using Vivado.

5.4 Analysis of the Accuracy Trade-off

To evaluate the effectiveness of the proposed design, it was compared with three existing approximate multiplier architectures from other researchers.

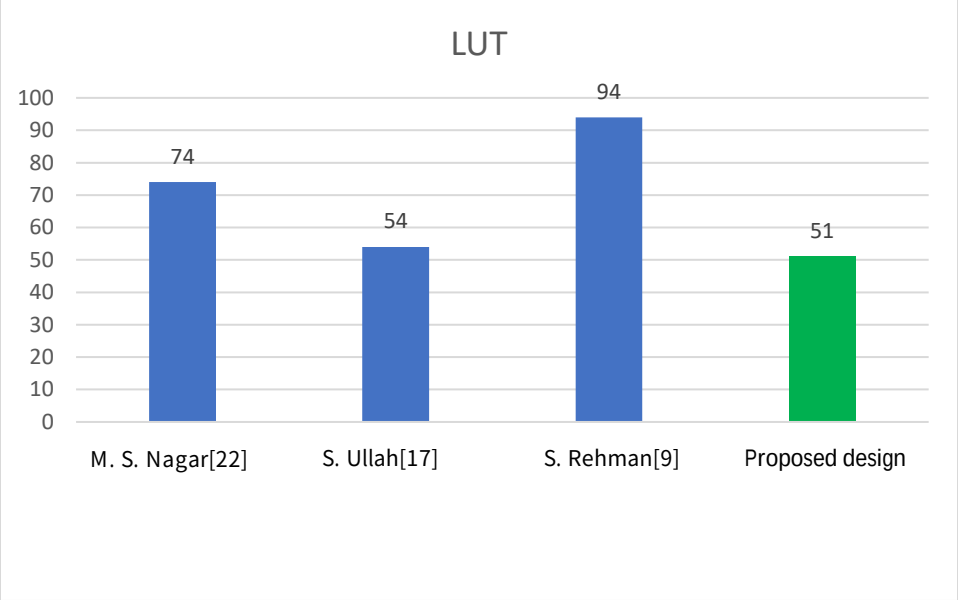


Fig. 5.5.1: LUT Usage Bar Graph

As shown in **Fig. 5.5.1**, the LUT usage for the three existing methods are:

- M. S. Nagar[22]: 74 LUTs
- S. Ullah[17]: 54 LUTs
- S. Rehman[9]: 94 LUTs

Only 51 LUTs are utilized by the proposed design, representing the lowest usage among all. This indicates that high area-efficiency is achieved by the architecture, making it suitable for low-resource FPGA designs.

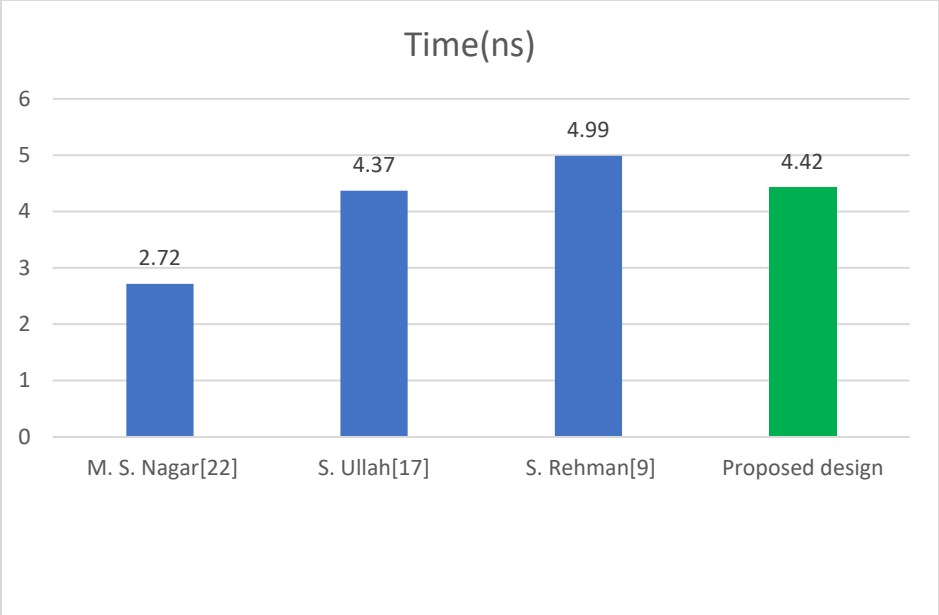


Fig. 5.5.2: Delay Comparison Chart.

When the delay comparison in **Fig. 5.5.2** is examined, good performance in terms of speed is also demonstrated by the proposed design. Although slightly lower delays are observed in some methods, these come at the cost of higher LUT usage. A balance between delay and area is achieved by the design, resulting in a practical trade-off suitable for real-time applications.

Overall, it is proven by the results that significant advantages in logic utilization are offered by the design while competitive performance is maintained.

5.5 Chapter Summary

The implementation and evaluation detailed in this chapter have successfully demonstrated the superiority of the proposed approximate multiplier for its target applications. The VHDL design, synthesized in Vivado for a Basys 3 FPGA, showed a significant reduction in LUTs. These significant hardware savings were achieved by introducing a bounded and a low average magnitude. It is confirmed by the results that an outstanding balance of performance and precision for error-resilient systems is provided by the methodology of combining Radix-4, CARRY4 primitives and a novel LUT-based approximate adder.

Chapter 6: Conclusion and Future Work

6.1 Conclusion

This thesis documented the design and FPGA realization of an approximate 8x8 signed multiplier, implemented as a hybrid architecture that combines Radix-4 Booth recording (with Baugh-Wooely's algorithm), CARRY4-chain preservation of hardware complexity and LUT6_2 for output calculation. The objective with the hybrid architecture was to mitigate hardware complexity and propagation delay whilst producing an approximation which, accounting for the impacts of the application, maintains sufficient accuracy to be suited to applications with high tolerance to small computation error.

The hybrid architecture successfully:

- Reduced the number of partial product rows from 8 to 4 with Radix-4 encoding.
- Utilized carry chains in an FPGA-native-relational manner to compress and preserve partial products which typically require a traditional CSA tree.
- Where the use of LUT6_2 primitives utilized in each column calculated the approximate result directly to end the logic depth.
- Showed that the device was very well suited to using FPGA resources and that the design avoided DSP blocks.
- The delay and resources were less than that of an exact multiplier whilst controlling error for each outcome.

The simulation and validation demonstrated that the overflow checked signed behavior is deterministic and that the approximate behavior was also predictable. Collectively the design proposed would meet the requirements for low-power and area constrained use cases in fields like approximate computing including image processing, AI inference and work with embedded systems.

6.2 Future Research Directions

The design discussed in this report leaves a lot of room for further investigation and improvement. Several investigation options to further increase quality or decrease cost are listed below.

- Extend the design to larger bit-widths (e.g., 16x16) to evaluate performance and accuracy trade-offs in more complex applications.
- Implement the special case of **–128** (minimum value in 2's complement).

References

[1] S. Ullah and A. Kumar, “Accurate Multipliers,” in *Approximate Arithmetic Circuit Architectures for FPGA-based Systems*, Cham: Springer International Publishing, 2023, pp. 41–72. doi: 10.1007/978-3-031-21294-9_3.

[2] J. Hu and W. Qian, “A New Approximate Adder with Low Relative Error and Correct Sign Calculation,” in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015*, New Jersey: IEEE Conference Publications, 2015, pp. 1449–1454. doi: 10.7873/DATE.2015.0627.

[3] S. Ullah, T. D. A. Nguyen and A. Kumar, “Energy-Efficient Low-Latency Signed Multiplier for FPGA-Based Hardware Accelerators,” *IEEE Embed Syst Lett*, vol. 13, no. 2, pp. 41–44, Jun. 2021, doi: 10.1109/LES.2020.2995053.

[4] A. Kulkarni, M. A. Ouameur and D. Massicotte, “Logic cloning based approximate signed multiplication circuits for FPGA,” *Microelectronics J*, vol. 145, p. 106135, Mar. 2024, doi: 10.1016/j.mejo.2024.106135.

[5] S. Cai, C. Huang, F. Yu, W. Wang and L. Yin, “Approximate multiplier design and error estimate model based on partial carry truncation,” *AEU - International Journal of Electronics and Communications*, vol. 200, p. 155862, Oct. 2025, doi: 10.1016/j.aeue.2025.155862.

[6] P. Anguraj and T. Krishnan, “Design and realization of area-efficient approximate multiplier structures for image processing applications,” *Microprocess Microsyst*, vol. 102, p. 104925, Oct. 2023, doi: 10.1016/j.micpro.2023.104925.

[7] R. Zia, M. Rao, A. Aziz and P. Akhtar, “Efficient Utilization of FPGA Using LUT-6 Architecture,” *Applied Mechanics and Materials*, vol. 241–244, pp. 2548–2554, Dec. 2012, doi: 10.4028/www.scientific.net/AMM.241-244.2548.

[8] S. Ullah, S. Rehman, M. Shafique and A. Kumar, "High-Performance Accurate and Approximate Multipliers for FPGA-Based Hardware Accelerators," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 41, no. 2, pp. 211-224, Feb. 2022, doi: 10.1109/TCAD.2021.3056337.

[9] S. Rehman, W. El-Harouni, M. Shafique, A. Kumar, J. Henkel and J. Henkel, "Architectural-space exploration of approximate multipliers," 2016 IEEE/ACM

International Conference on Computer-Aided Design (ICCAD), Austin, TX, USA, 2016, pp. 1-8, doi: 10.1145/2966986.2967005.

[10] D. Esposito, A. G. M. Strollo, E. Napoli, D. De Caro and N. Petra, "Approximate Multipliers Based on New Approximate Compressors," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 12, pp. 4169–4182, Dec. 2018, doi: 10.1109/TCSI.2018.2839266.

[11] A. Pathan, A. H. Chandio and R. Aziz, "An Optimization in Conventional Shift &Add Multiplier for Area-Efficient Implementation on FPGA," in *2022 International Conference on Emerging Technologies in Electronics, Computing and Communication (ICETECC)*, IEEE, Dec. 2022, pp. 1–6. doi: 10.1109/ICETECC56662.2022.10069099.

[12] S. M. Qasim, S. A. Abbasi and B. Almashary, "An overview of advanced FPGA architectures for optimized hardware realization of computation intensive algorithms," in *2009 International Multimedia, Signal Processing and Communication Technologies*, IEEE, Mar. 2009, pp. 300–303. doi: 10.1109/MSPCT.2009.5164235.

[13] R. Zia, M. Rao, A. Aziz and P. Akhtar, "Efficient Utilization of FPGA Using LUT-6 Architecture," *Applied Mechanics and Materials*, vol. 241–244, pp. 2548–2554, Dec. 2012, doi: 10.4028/www.scientific.net/AMM.241-244.2548.

[14] Mahmoud Khaled, "Enhancing the Performance of Digital Controllers using Distributed Multicore/Heterogeneous Embedded Systems," 2014. doi: 10.1109/MSPCT.2009.5164235.

[15] K. Huang, R. Zhao and Y. Lian, "Racetrack Memory based hybrid Look-Up Table (LUT) for low power reconfigurable computing," *J Parallel Distrib Comput*, vol. 117, pp. 127–137, Jul. 2018, doi: 10.1016/j.jpdc.2018.02.018.

[16] Xin Fang, P. Luo, Y. Fei and M. Leeser, "Leakage evaluation on power balance countermeasure against side-channel attack on FPGAs," in *2015 IEEE High Performance Extreme Computing Conference (HPEC)*, IEEE, Sep. 2015, pp. 1–6. doi: 10.1109/HPEC.2015.7322469.

[17] S. Ullah, S. Rehman, M. Shafique and A. Kumar, "High-Performance Accurate and Approximate Multipliers for FPGA-Based Hardware Accelerators," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 2, pp. 211–224, Feb. 2022, doi: 10.1109/TCAD.2021.3056337.

[18] H. Parandeh-Afshar and P. Ienne, “Measuring and Reducing the Performance Gap between Embedded and Soft Multipliers on FPGAs,” in *2011 21st International Conference on Field Programmable Logic and Applications*, IEEE, Sep. 2011, pp. 225–231. doi: 10.1109/FPL.2011.48.

[19] C. R. Baugh and B. A. Wooley, "A Two's Complement Parallel Array Multiplication Algorithm," in *IEEE Transactions on Computers*, vol. C-22, no. 12, pp. 1045-1047, Dec. 1973, doi: 10.1109/T-C.1973.223648.

[20] L. Dadda, “Some schemes for parallel multipliers,” Alta Frequenza, vol. 34, pp. 349–356, 1965

[21] Bewick, Gary & Flynn, Michael. (1970). Fast Multiplication: Algorithms And Implementation.

[22] M. S. Nagar, A. Mathuriya, S. H. Patel and P. J. Engineer, "High-Speed Energy-Efficient Fixed-Point Signed Multipliers for FPGA-Based DSP Applications," in IEEE Embedded Systems Letters, vol. 16, no. 4, pp. 417-420, Dec. 2024, doi: 10.1109/LES.2024.3364698.